μ MOTOROLA

MC68360

QUad Integrated Communications Controller User's Manual

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola 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 Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola 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 Motorola was negligent regarding the design or manufacture of the part. Motorola and the Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

PREFACE

The complete documentation package for the MC68360 consists of the MC68360UM/AD, *MC68360 QUad Integrated Communications Controller User's Manual*, M68000PM/AD, *MC68000 Family Programmer's Reference Manual,* and the MC68360/D, *MC68360 QUad Integrated Communications Controller Product Brief*.

The *MC68360 QUad Integrated Communications Controller User's Manual* describes the programming, capabilities, registers, and operation of the MC68360 and the MC68EN360; the *MC68000 Family Programmer's Reference Manual* provides instruction details for the MC68360; and the *MC68360 QUad Integrated Communications Controller Product Brief* provides a brief description of the MC68360 capabilities.

This user's manual is organized as follows:

Section 1	Introduction
Section 2	Signal Descriptions
Section 3	Memory Map
Section 4	Bus Operation
Section 5	CPU32+
Section 6	System Integration Module (SIM60)
Section 7	Communication Processor Module (CPM)
Section 8	IEEE 1149.1 Test Access Port
Section 9	Applications
Section 10	Electrical Characteristics
Section 11	Ordering Information and Mechanical Data
Appendix A	Serial Performance
Appendix B	Development Tools and Support
Appendix C	RISC Microcode from RAM
Appendix D	MC68MH360 Product Brief

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

Section 1
Introduction

1.1 QUICC Key Features 1-1
1.2 QUICC Architecture Overview 1-4
1.2.1 CPU32+ Core 1-5
1.2.2 System Integration Module (SIM60) 1-5
1.2.3 Communications Processor Module (CPM) 1-6
1.3 Upgrading Designs from the MC68302 1-6
1.3.1 Architectural Approach 1-6
1.3.2 Hardware Compatibility Issues 1-7
1.3.3 Software Compatibility Issues 1-7
1.4 QUICC Glueless System Design 1-8
1.5 QUICC Serial Configurations 1-9
1.6 QUICC Serial Configuration Examples 1-16
1.7 QUICC System Bus Configurations 1-17

Section 2
Signal Descriptions

2.1 System Bus Signal Index 2-1
2.1.1 Address Bus 2-1
2.1.1.1 Address Bus (A27–A0). 2-1
2.1.1.2 Address Bus (A31–A28) 2-1
2.1.2 Function Codes (FC3–FC0) 2-5
2.1.3 Data Bus 2-5
2.1.3.1 Data Bus (D31–D16). 2-5
2.1.3.2 Data Bus (D15–D0). 2-6
2.1.4 Parity 2-6
2.1.4.1 Parity (PRTY0). 2-6
2.1.4.2 Parity (PRTY1). 2-6
2.1.4.3 Parity (PRTY2). 2-6
2.1.4.4 Parity (PRTY3). 2-6
2.1.5 Memory Controller 2-6
2.1.5.1 Chip Select/Row Address Select (CS6–CS0/RAS6–RAS0) 2-6
2.1.5.2 Chip Select/Row Address Select/Interrupt Acknowledge (CS7/RAS7/IACK7). 2-6
2.1.5.3 Column Address Select/Interrupt Acknowledge (CAS3–CAS0/IACK6, 3, 2, 1). 2-7
2.1.5.4 Address Multiplex (AMUX). 2-7
2.1.6 Interrupt Request Level (IRQ7–IRQ1) 2-7
2.1.7 Bus Control Signals 2-7
2.1.7.1 Data and Size Acknowledge (DSACK1–DSACK0). 2-8
2.1.7.2 Autovector/Interrupt Acknowledge (AVEC/IACK5). 2-8
2.1.7.3 Address Strobe (AS). 2-8
2.1.7.4 Data Strobe (DS). 2-8

Paragraph Number	Title	Page Number
2.1.7.5	Transfer Size (SIZ1, SIZ0).	2-8
2.1.7.6	Read/Write (R/W).	2-8
2.1.7.7	Output Enable/Address Multiplex (OE/AMUX)	2-9
2.1.7.8	Byte Write Enable (WE3–WE0).	2-9
2.1.8	Bus Arbitration Signals	2-9
2.1.8.1	Bus Request (BR).	2-9
2.1.8.2	Bus Grant (BG).	2-9
2.1.8.3	Bus Grant Acknowledge (BGACK).	2-9
2.1.8.4	Read-Modify-Write Cycle/Initial Configuration (RMC/CONFIG0).	2-9
2.1.8.5	Bus Clear Out/Initial Configuration/Row Address Select Double-Drive (BCLRO/CONFIG1/RAS2DD).	2-9
2.1.9	System Control Signals	2-10
2.1.9.1	Soft Reset (RESETS).	2-10
2.1.9.2	Hard Reset (RESETH)	2-10
2.1.9.3	Halt (HALT).	2-10
2.1.9.4	Bus Error (BERR).	2-10
2.1.10	Clock Signals	2-10
2.1.10.1	System Clock Outputs (CLKO2–CLKO1).	2-10
2.1.10.2	Crystal Oscillator (EXTAL, XTAL).	2-11
2.1.10.3	External Filter Capacitor (XFC).	2-11
2.1.10.4	Clock Mode Select (MODCK1–MODCK0).	2-11
2.1.11	Instrumentation and Emulation Signals	2-11
2.1.11.1	Instruction Fetch/Development Serial Input (IFETCH/DSI).	2-11
2.1.11.2	Instruction Pipe/Development Serial Output (**$\overline{\text{IPIPE0/DSO}}$**).	2-11
2.1.11.3	Instruction Pipe/Row Address Select Double-Drive (**$\overline{\text{IPIPE1/RAS1DD}}$**).	2-11
2.1.11.4	Breakpoint/Development Serial clock (BKPT/DSCLK).	2-11
2.1.11.5	Freeze/Initial Configuration (FREEZE/CONFIG2).	2-12
2.1.12	Test Signals	2-12
2.1.12.1	TRI-State Signal (TRIS).	2-12
2.1.12.2	Test Reset (TRST).	2-12
2.1.12.3	Test Clock (TCK).	2-12
2.1.12.4	Test Mode Select (TMS).	2-12
2.1.12.5	Test Data In (TDI).	2-12
2.1.12.6	Test Data Out (TDO).	2-12
2.1.13	Initial Configuration Pins (CONFIG)	2-12
2.1.14	Power Signals	2-13
2.1.14.1	VCCSYN and GNDSYN.	2-13
2.1.14.2	VCCCLK and GNDCLK.	2-13
2.1.14.3	GNDS1 and GNDS2.	2-13
2.1.14.4	VCC and GND.	2-13
2.1.14.5	NC4–NC1.	2-13
2.2	System Bus Signal Index in Slave Mode	2-14
2.3	On-Chip Peripherals Signal Index	2-15

Section 3

Paragraph Number	Title	Page Number
	QUICC Memory Map	
3.1	Dual-Port RAM Memory Map	3-2
3.2	CPM Sub-Module Base Addresses	3-3
3.3	Internal Registers Memory Map	3-4
3.3.1	SIM Registers Memory Map	3-4
3.3.2	CPM Registers Memory Map	3-6
	Section 4 Bus Operation	
4.1	Bus Transfer Signals	4-2
4.1.1	Bus Control Signals	4-3
4.1.2	Function Codes (FC3–FC0)	4-3
4.1.3	Address Bus (A31–A0)	4-4
4.1.4	Address Strobe (AS)	4-4
4.1.5	Data Bus (D31-D0)	4-4
4.1.6	Data Strobe (DS)	4-4
4.1.7	Output Enable (OE)	4-4
4.1.8	Byte Write Enable (WE0, WE1, WE2, WE3)	4-4
4.1.9	Bus Cycle Termination Signals	4-5
4.1.9.1	Data transfer and size acknowledge (DSACK1 and DSACK0).	4-5
4.1.9.2	Bus Error (BERR).	4-5
4.1.9.3	Autovector (AVEC).	4-6
4.2	Data Transfer Mechanism	4-6
4.2.1	Dynamic Bus Sizing	4-6
4.2.2	Misaligned Operands	4-11
4.2.3	Effects of Dynamic Bus Sizing and Operand Misalignment	4-19
4.2.4	Bus Operation	4-20
4.2.5	Synchronous Operation with DSACKx	4-21
4.2.6	Fast Termination Cycles	4-21
4.3	Data Transfer Cycles	4-22
4.3.1	Read Cycle	4-23
4.3.2	Write Cycle	4-26
4.3.3	Read-Modify-Write Cycle	4-28
4.4	CPU Space Cycles	4-31
4.4.1	Breakpoint Acknowledge Cycle	4-31
4.4.2	LPSTOP Broadcast Cycle	4-35
4.4.3	Module Base Address Register (MBAR) Access	4-36
4.4.4	Interrupt Acknowledge Bus Cycles	4-36
4.4.4.1	Interrupt Acknowledge Cycle—Terminated Normally	4-36
4.4.4.2	Autovector Interrupt Acknowledge Cycle.	4-38
4.4.4.3	Spurious Interrupt Cycle.	4-40
4.5	Bus Exception Control Cycles	4-41
4.5.1	Bus Errors	4-42
4.5.2	Retry Operation	4-44
4.5.3	Halt Operation	4-46
4.5.4	Double Bus Fault	4-48

Paragraph Number	Title	Page Number
4.6	Bus Arbitration	4-49
4.6.1	Bus Request	4-52
4.6.2	Bus Grant	4-53
4.6.3	Bus Grant Acknowledge	4-53
4.6.4	Bus Arbitration Control	4-54
4.6.5	Slave (Disable CPU32+) Mode Bus Arbitration	4-55
4.6.6	Slave (Disable CPU32+) Mode Bus Exceptions	4-59
4.6.6.1	HALT.	4-59
4.6.6.2	RETRY.	4-59
4.6.7	Internal Accesses	4-59
4.6.8	Show Cycles	4-62
4.7	Reset Operation	4-63

Section 5
CPU32+

5.1	Overview	5-1
5.1.1	Features	5-2
5.1.2	Loop Mode Instruction Execution	5-3
5.1.3	Vector Base Register	5-4
5.1.4	Exception Handling	5-4
5.1.5	Addressing Modes	5-5
5.2	Architecture Summary	5-5
5.2.1	Programming Model	5-6
5.2.2	Registers	5-7
5.3	Instruction Set	5-8
5.3.1	M68000 Family Compatibility	5-10
5.3.1.1	New Instructions.	5-10
5.3.1.2	Low-Power Stop (LPSTOP).	5-10
5.3.1.3	Table Lookup and Interpolate (TBL).	5-10
5.3.1.4	Unimplemented Instructions.	5-10
5.3.2	Instruction Format and Notation	5-10
5.3.3	Instruction Summary	5-13
5.3.3.1	Condition Code Register	5-17
5.3.3.2	Data Movement Instructions	5-19
5.3.3.3	Integer Arithmetic Operations	5-19
5.3.3.4	Logic Instructions.	5-21
5.3.3.5	Shift and Rotate Instructions.	5-22
5.3.3.6	Bit Manipulation Instructions	5-23
5.3.3.7	Binary-Coded Decimal (BCD) Instructions	5-24
5.3.3.8	Program Control Instructions	5-24
5.3.3.9	System Control Instructions	5-25
5.3.3.10	Condition Tests	5-26
5.3.4	Using the TBL Instructions	5-27
5.3.4.1	Table Example 1: Standard Usage	5-28
5.3.4.2	Table Example 2: Compressed Table.	5-29

Paragraph Number	Title	Page Number
5.3.4.3	Table Example 3: 8-Bit Independent Variable	5-30
5.3.4.4	Table Example 4: Maintaining Precision	5-32
5.3.4.5	Table Example 5: Surface Interpolations	5-33
5.3.5	Nested Subroutine Calls	5-33
5.3.6	Pipeline Synchronization with the NOP Instruction	5-34
5.4	Processing States	5-34
5.4.1	State Transitions	5-34
5.4.2	Privilege Levels	5-34
5.4.2.1	Supervisor Privilege Level	5-35
5.4.2.2	User Privilege Level	5-35
5.4.2.3	Changing Privilege Level	5-35
5.5	Exception Processing	5-36
5.5.1	Exception Vectors	5-36
5.5.1.1	Types of Exceptions	5-36
5.5.1.2	Exception Processing Sequence	5-38
5.5.1.3	Exception Stack Frame	5-38
5.5.1.4	Multiple Exceptions	5-39
5.5.2	Processing of Specific Exceptions	5-40
5.5.2.1	Reset	5-40
5.5.2.2	Bus Error	5-40
5.5.2.3	Address Error	5-42
5.5.2.4	Instruction Traps	5-42
5.5.2.5	Software Breakpoints	5-43
5.5.2.6	Hardware Breakpoints	5-43
5.5.2.7	Format Error	5-43
5.5.2.8	Illegal or Unimplemented Instructions	5-44
5.5.2.9	Privilege Violations	5-44
5.5.2.10	Tracing	5-45
5.5.2.11	Interrupts	5-46
5.5.2.12	Return from Exception	5-47
5.5.3	Fault Recovery	5-48
5.5.3.1	Types of Faults	5-51
5.5.3.1.1	Type I—Released Write Faults	5-51
5.5.3.1.2	Type II—Prefetch, Operand, RMW, and MOVEP Faults	5-51
5.5.3.1.3	Type III—Faults During MOVEM Operand Transfer	5-52
5.5.3.1.4	Type IV—Faults During Exception Processing	5-52
5.5.3.2	Correcting a Fault	5-53
5.5.3.2.1	Type I—Completing Released Writes via Software	5-53
5.5.3.2.2	Type I—Completing Released Writes via RTE	5-53
5.5.3.2.3	Type II—Correcting Faults via RTE	5-54
5.5.3.2.4	Type III—Correcting Faults via Software	5-54
5.5.3.2.5	Type III—Correcting Faults by Conversion and Restart	5-55
5.5.3.2.6	Type III—Correcting Faults via RTE	5-55
5.5.3.2.7	Type IV—Correcting Faults via Software	5-55
5.5.4	CPU32+ Stack Frames	5-56

Paragraph Number	Title	Page Number
5.5.4.1	Four-Word Stack Frame	5-56
5.5.4.2	Six-Word Stack Frame	5-56
5.5.4.3	Bus Error Stack Frame	5-56
5.6	Development Support	5-59
5.6.1	CPU32+ Integrated Development Support	5-59
5.6.1.1	Background Debug Mode (BDM) Overview	5-59
5.6.1.2	Deterministic Opcode Tracking Overview	5-60
5.6.1.3	On-Chip Hardware Breakpoint Overview	5-60
5.6.2	Background Debug Mode	5-60
5.6.2.1	Enabling BDM	5-60
5.6.2.2	BDM Sources	5-61
5.6.2.2.1	External BKPT Signal	5-62
5.6.2.2.2	BGND Instruction	5-62
5.6.2.2.3	Double Bus Fault	5-62
5.6.2.3	Entering BDM	5-62
5.6.2.4	Command Execution	5-62
5.6.2.5	BDM Registers	5-63
5.6.2.5.1	Fault Address Register (FAR)	5-63
5.6.2.5.2	Return Program Counter (RPC)	5-63
5.6.2.5.3	Current Instruction Program Counter (PCC)	5-63
5.6.2.6	Returning from BDM	5-63
5.6.2.7	Serial Interface	5-63
5.6.2.7.1	CPU Serial Logic	5-65
5.6.2.7.2	Development System Serial Logic	5-66
5.6.2.8	Command Set	5-68
5.6.2.8.1	Command Format	5-68
5.6.2.8.2	Command Sequence Diagram	5-69
5.6.2.8.3	Command Set Summary	5-69
5.6.2.8.4	Read A/D Register (RAREG/RDREG)	5-71
5.6.2.8.5	Write A/D Register (WAREG/WDREG)	5-71
5.6.2.8.6	Read System Register (RSREG)	5-71
5.6.2.8.7	Write System Register (WSREG)	5-72
5.6.2.8.8	Read Memory Location (READ)	5-73
5.6.2.8.9	Write Memory Location (WRITE)	5-74
5.6.2.8.10	Dump Memory Block (DUMP)	5-75
5.6.2.8.11	Fill Memory Block (FILL)	5-76
5.6.2.8.12	Resume Execution (GO)	5-77
5.6.2.8.13	Call User Code (CALL)	5-77
5.6.2.8.14	Reset Peripherals (RST)	5-79
5.6.2.8.15	No Operation (NOP)	5-79
5.6.2.8.16	Future Commands	5-80
5.6.3	Deterministic Opcode Tracking	5-80
5.6.3.1	Instruction Fetch (IFETCH)	5-80
5.6.3.2	Instruction Pipe (IPIPE1–IPIPE0)	5-80
5.6.3.3	Opcode Tracking during Loop Mode	5-82

Paragraph Number	Title	Page Number
5.7	Instruction Execution Timing	5-82
5.7.1	Resource Scheduling	5-83
5.7.1.1	Microsequencer	5-83
5.7.1.2	Instruction Pipeline	5-83
5.7.1.3	Bus Controller Resources	5-83
5.7.1.3.1	Prefetch Controller	5-84
5.7.1.3.2	Write-Pending Buffer	5-84
5.7.1.3.3	Microbus Controller	5-85
5.7.1.4	Instruction Execution Overlap	5-85
5.7.1.5	Effects of Wait States	5-86
5.7.1.6	Instruction Execution Time Calculation	5-86
5.7.1.7	Effects of Negative Tails	5-87
5.7.2	Instruction Timing Tables	5-88
5.7.2.1	Fetch Effective Address	5-90
5.7.2.2	Calculate Effective Address	5-91
5.7.2.3	MOVE Instruction	5-92
5.7.2.4	Special-Purpose MOVE Instruction	5-92
5.7.2.5	Arithmetic/Logic Instructions	5-93
5.7.2.6	Immediate Arithmetic/Logic Instructions	5-95
5.7.2.7	Binary-Coded Decimal and Extended Instructions	5-95
5.7.2.8	Single Operand Instructions	5-96
5.7.2.9	Shift/Rotate Instructions	5-96
5.7.2.10	Bit Manipulation Instructions	5-97
5.7.2.11	Conditional Branch Instructions	5-98
5.7.2.12	Control Instructions	5-99
5.7.2.13	Exception-Related Instructions and Operations	5-100
5.7.2.14	Save and Restore Operations	5-101

Section 6
System Integration Module (SIM60)

6.1	Module Overview	6-1
6.2	Module Base Address Register (MBAR)	6-3
6.3	System Configuration and Protection	6-3
6.3.1	System Configuration	6-5
6.3.1.1	SIM60 Interrupt Generation	6-6
6.3.1.2	Simultaneous SIM60 Interrupt Sources	6-8
6.3.1.2.1	Bus Monitor	6-8
6.3.1.2.2	Spurious Interrupt Monitor	6-8
6.3.1.2.3	Double Bus Fault Monitor	6-9
6.3.1.2.4	Software Watchdog Timer (SWT)	6-9
6.3.2	Periodic Interrupt Timer (PIT)	6-10
6.3.2.1	PIT Period Calculation	6-10
6.3.2.2	Using the PIT as a Real-Time Clock	6-11
6.3.3	Freeze Support	6-11
6.3.4	Low-Power Stop Support	6-11
6.4	Low Power in Normal Operation	6-12

Paragraph Number	Title	Page Number
6.5	SIM60 System Clock Generation	6-12
6.5.1	Clock Generation Methods	6-12
6.5.2	Oscillator Prescaler (Divide by 128)	6-13
6.5.3	Phase-Locked Loop (PLL)	6-14
6.5.3.1	Frequency Multiplication	6-14
6.5.3.2	Skew Elimination	6-15
6.5.4	Low-Power Divider	6-15
6.5.5	QUICC Internal Clock Signals	6-15
6.5.5.1	SPCLK	6-16
6.5.5.2	General System Clock	6-16
6.5.5.3	BRGCLK	6-17
6.5.5.4	SyncCLK	6-17
6.5.5.5	SIMCLK	6-18
6.5.5.6	CLKO1	6-18
6.5.5.7	CLKO2	6-18
6.5.6	PLL Power Pins	6-19
6.5.6.1	VCCSYN	6-19
6.5.6.2	GNDSYN	6-19
6.5.6.3	XFC	6-19
6.5.7	CLKO Power Pins	6-19
6.5.7.1	VCCCLK	6-19
6.5.7.2	GNDCLK	6-19
6.5.8	Configuration Pins (MODCK1–MODCK0)	6-19
6.6	Breakpoint Logic	6-20
6.7	External Bus Interface Control	6-21
6.7.1	Initial Configuration	6-22
6.7.2	Port D	6-22
6.7.3	Port E	6-23
6.8	Slave (Disable CPU32+) Mode	6-23
6.8.1	MBAR in a Multiple QUICC System	6-24
6.8.2	Global Chip Select (CS0) in Slave Mode	6-25
6.8.3	Bus Clear in Slave Mode	6-25
6.8.4	Interrupts in Slave Mode	6-26
6.8.5	Pin Differences in Slave Mode	6-26
6.8.6	Other Functionality in Slave Mode	6-27
6.9	Programmer’s Model	6-27
6.9.1	Module Base Address Register (MBAR)	6-27
6.9.2	Module Base Address Register Enable (MBARE)	6-29
6.9.3	System Configuration and Protection Registers	6-29
6.9.3.1	Module Configuration Register (MCR)	6-29
6.9.3.2	Autovector Register (AVR)	6-34
6.9.3.3	Reset Status Register (RSR)	6-34
6.9.3.4	Software Watchdog Interrupt Vector Register (SWIV)	6-35
6.9.3.5	System Protection Control Register (SYPCR)	6-35
6.9.3.6	Periodic Interrupt Control Register (PICR)	6-37

Paragraph Number	Title	Page Number
6.9.3.7	Periodic Interrupt Timer Register (PITR)	6-38
6.9.3.8	Software Service Register (SWSR)	6-39
6.9.3.9	CLKO Control Register (CLKOCR)	6-39
6.9.3.10	PLL Control Register (PLLCR)	6-40
6.9.3.11	Clock Divider Control Register (CDVCR)	6-42
6.9.3.12	Breakpoint Address Register (BKAR)	6-44
6.9.3.13	Breakpoint Control Register (BKCR)	6-44
6.9.4	Port E Pin Assignment Register (PEPAR)	6-48
6.10	Memory Controller	6-50
6.10.1	Memory Controller Key Features	6-50
6.10.2	Memory Controller Overview	6-51
6.11	General-Purpose Chip-Select Overview (SRAM Banks)	6-56
6.11.1	Associated Registers	6-56
6.11.2	8-, 16-, and 32-Bit Port Size Configuration	6-56
6.11.3	Write Protect Configuration	6-56
6.11.4	Programmable Wait State Configuration	6-56
6.11.5	Address and Address Space Checking	6-57
6.11.6	SRAM Bank Parity	6-57
6.11.7	External Master Support	6-57
6.11.8	Global (Boot) Chip-Select Operation	6-58
6.11.9	SRAM Bus Error	6-58
6.12	DRAM Controller Overview (DRAM Banks)	6-58
6.12.1	DRAM Normal Access Support	6-60
6.12.2	DRAM Page Mode Support	6-60
6.12.3	DRAM Burst Access Support	6-61
6.12.4	DRAM Bank Parity	6-62
6.12.5	Refresh Operation	6-62
6.12.6	DRAM Bank External Master Support	6-63
6.12.7	Double-Drive RAS Lines	6-63
6.12.8	DRAM Bus Error	6-63
6.13	Programming Model	6-64
6.13.1	Global Memory Register (GMR)	6-64
6.13.2	Memory Controller Status Register (MSTAT)	6-69
6.13.3	Base Register (BR)	6-70
6.13.4	Option Register (OR)	6-74
6.13.5	DRAM-SRAM Performance Summary;	6-78

Section 7
Communication Processor Module (CPM)

	Introduction	7-1
7.1	RISC Controller	7-3
7.1.1	RISC Controller Configuration Register (RCCR)	7-4
7.1.2	RISC Microcode Revision Number	7-5
7.2	Command Set	7-5
7.2.1	Command Register Examples	7-8
7.2.2	Command Execution Latency	7-8

Paragraph Number	Title	Page Number
7.3	Dual-Port RAM	7-8
7.3.1	Buffer Descriptors	7-10
7.3.2	Parameter RAM	7-10
7.4	RISC Timer Tables	7-11
7.4.1	RISC Timer Table Parameter RAM	7-12
7.4.2	RISC Timer Table Entries	7-14
7.4.3	RISC Timer Event Register (RTER)	7-14
7.4.4	RISC Timer Mask Register (RTMR)	7-14
7.4.5	SET TIMER Command	7-14
7.4.6	RISC Timer Initialization Sequence	7-14
7.4.7	RISC Timer Initialization Example	7-15
7.4.8	RISC Timer Interrupt Handling	7-16
7.4.9	RISC Timer Table Algorithm	7-16
7.4.10	RISC Timer Table Application: Track the RISC Loading	7-16
7.5	Timers	7-17
7.5.1	Timer Key Features	7-17
7.5.2	General-Purpose Timer Units	7-18
7.5.2.1	Cascaded Mode	7-19
7.5.2.2	Timer Global Configuration Register (TGCR)	7-20
7.5.2.3	Timer Mode Register (TMR1, TMR2, TMR3, TMR4)	7-21
7.5.2.4	Timer Reference Registers (TRR1, TRR2, TRR3, TRR4)	7-22
7.5.2.5	Timer Capture Registers (TCR1, TCR2, TCR3, TCR4)	7-22
7.5.2.6	Timer Counter (TCN1, TCN2, TCN3, TCN4)	7-22
7.5.2.7	Timer Event Registers (TER1, TER2, TER3, TER4)	7-22
7.5.3	Timer Examples	7-23
7.6	IDMA Channels	7-24
7.6.1	IDMA Key Features;	7-25
7.6.2	IDMA Registers	7-26
7.6.2.1	IDMA Channel Configuration Register (ICCR)	7-26
7.6.2.2	Channel Mode Register (CMR)	7-28
7.6.2.3	Source Address Pointer Register (SAPR)	7-30
7.6.2.4	Destination Address Pointer Register (DAPR)	7-31
7.6.2.5	Function Code Register (FCR)	7-31
7.6.2.6	Byte Count Register (BCR)	7-31
7.6.2.7	Channel Status Register (CSR)	7-32
7.6.2.8	Channel Mask Register (CMAR)	7-33
7.6.2.9	Data Holding Register (DHR)	7-33
7.6.3	Interface Signals	7-33
7.6.3.1	DREQ and DACK	7-33
7.6.3.2	DONEx	7-33
7.6.4	IDMA Operation	7-34
7.6.4.1	Single Buffer	7-34
7.6.4.2	Auto Buffer and Buffer Chaining	7-34
7.6.4.2.1	IDMA Parameter RAM	7-35
7.6.4.2.2	IDMA Buffer Descriptors (BDs)	7-36

Paragraph Number	Title	Page Number
7.6.4.2.3	IDMA Commands (INIT_IDMA)	7-38
7.6.4.3	Starting the IDMA	7-38
7.6.4.4	Requesting IDMA Transfers	7-39
7.6.4.4.1	Internal Maximum Rate	7-39
7.6.4.4.2	Internal Limited Rate	7-39
7.6.4.4.3	External Burst Mode	7-40
7.6.4.4.4	External Cycle Steal	7-42
7.6.4.5	IDMA Bus Arbitration	7-43
7.6.4.6	IDMA Operand Transfers	7-45
7.6.4.6.1	Dual Address Mode	7-45
7.6.4.6.2	Single Address Mode (Flyby Transfers)	7-48
7.6.4.6.3	Fast-Termination Option	7-50
7.6.4.6.4	Externally Recognizing IDMA Operand Transfers	7-51
7.6.4.7	Bus Exceptions	7-51
7.6.4.7.1	Reset	7-51
7.6.4.7.2	Bus Error	7-51
7.6.4.7.3	Retry	7-51
7.6.4.8	Ending the IDMA Transfer	7-52
7.6.4.8.1	Single Buffer Mode Termination	7-52
7.6.4.8.2	Auto Buffer Mode Termination.	7-53
7.6.4.8.3	Buffer Chaining Mode Termination	7-54
7.6.5	IDMA Examples	7-55
7.6.5.1	Single Buffer Examples	7-55
7.6.5.2	Buffer Chaining Example	7-55
7.6.5.3	Auto Buffer Example	7-56
7.7	SDMA Channels	7-57
7.7.1	SDMA Bus Arbitration and Bus Transfers	7-57
7.7.2	SDMA Registers	7-59
7.7.2.1	SDMA Configuration Register (SDCR)	7-59
7.7.2.2	SDMA Status Register (SDSR)	7-61
7.7.2.3	SDMA Address Register (SDAR)	7-61
7.8	Serial Interface with Time Slot Assigner	7-62
7.8.1	SI Key Features	7-62
7.8.2	TSA Overview	7-64
7.8.3	Enabling Connections to the TSA	7-67
7.8.4	SI RAM	7-68
7.8.4.1	One Multiplexed Channel with Static Frames	7-69
7.8.4.2	One Multiplexed Channel with Dynamic Frames	7-69
7.8.4.3	Two Multiplexed Channels with Static Frames	7-70
7.8.4.4	Two Multiplexed Channels with Dynamic Frames	7-71
7.8.4.5	Programming SI RAM Entries	7-72
7.8.4.6	SI RAM Programming Example	7-75
7.8.4.7	SI RAM Dynamic Changes	7-75
7.8.5	SI Registers	7-77
7.8.5.1	SI Global Mode Register (SIGMR)	7-77

Paragraph Number	Title	Page Number
7.8.5.2	SI Mode Register (SIMODE)	7-78
7.8.5.3	SI Clock Route Register (SICR)	7-86
7.8.5.4	SI Command Register (SICMR)	7-87
7.8.5.5	SI Status Register (SISTR)	7-87
7.8.5.6	SI RAM Pointers (SIRP)	7-88
7.8.5.6.1	SIRP When RDM = 00 (One Static TDM)	7-89
7.8.5.6.2	SIRP When RDM = 01 (One Dynamic TDM)	7-89
7.8.5.6.3	SIRP When RDM = 10 (Two Static TDMs)	7-90
7.8.5.6.4	SIRP When RDM = 11 (Two Dynamic TDMs)	7-90
7.8.6	SI IDL Interface Support	7-90
7.8.6.1	IDL Interface Example	7-91
7.8.6.2	IDL Interface Programming	7-95
7.8.7	SI GCI Support	7-96
7.8.7.1	SI GCI Activation/Deactivation Procedure	7-98
7.8.7.2	SI GCI Programming	7-98
7.8.7.2.1	Normal Mode GCI Programming	7-98
7.8.7.2.2	SCIT Programming	7-98
7.8.8	Serial Interface Synchronization	7-100
7.8.9	NMSI Configuration	7-100
7.9	Baud Rate Generators (BRGs)	7-103
7.9.1	Autobaud Support	7-105
7.9.2	BRG Configuration Register (BRGC)	7-106
7.9.3	UART Baud Rate Examples	7-108
7.10	Serial Communication Controllers (SCCs)	7-109
7.10.1	SCC Overview	7-110
7.10.2	General SCC Mode Register (GSMR)	7-111
7.10.3	SCC Protocol-Specific Mode Register (PSMR)	7-120
7.10.4	SCC Data Synchronization Register (DSR)	7-121
7.10.5	SCC Transmit on Demand Register (TODR)	7-121
7.10.6	SCC Buffer Descriptors	7-122
7.10.7	SCC Parameter RAM	7-124
7.10.7.1	BD Table Pointer (RBASE, TBASE)	7-125
7.10.7.2	SCC Function Code Registers (RFCR, TFCR)	7-125
7.10.7.3	Maximum Receive Buffer Length Register (MRBLR)	7-127
7.10.7.4	Receiver BD Pointer (RBPTR)	7-127
7.10.7.5	Transmitter BD Pointer (TBPTR)	7-127
7.10.7.6	Other General Parameters	7-128
7.10.8	Interrupts from the SCCs	7-128
7.10.8.1	SCC Event Register (SCCE)	7-128
7.10.8.2	SCC Mask Register (SCCM)	7-129
7.10.8.3	SCC Status Register (SCCS)	7-129
7.10.9	SCC Initialization	7-129
7.10.10	SCC Interrupt Handling	7-130
7.10.11	SCC Timing Control	7-130
7.10.11.1	Synchronous Protocols	7-130

Paragraph Number	Title	Page Number
7.10.11.2	Asynchronous Protocols	7-134
7.10.12	Digital Phase-Locked Loop (DPLL)	7-135
7.10.12.1	Data Encoding	7-135
7.10.12.2	DPLL Operation	7-136
7.10.13	Clock Glitch Detection	7-139
7.10.14	Disabling the SCCs on the Fly	7-139
7.10.14.1	SCC Transmitter Full Sequence	7-140
7.10.14.2	SCC Transmitter Shortcut SEQUENCE	7-140
7.10.14.3	SCC Receiver Full Sequence	7-140
7.10.14.4	SCC Receiver Shortcut Sequence	7-141
7.10.14.5	Switching Protocols	7-141
7.10.15	Saving Power	7-141
7.10.16	UART Controller	7-141
7.10.16.1	UART Key Features	7-143
7.10.16.2	Normal Asynchronous Mode	7-143
7.10.16.3	Synchronous Mode	7-144
7.10.16.4	UART Memory Map	7-145
7.10.16.5	UART Programming Model	7-147
7.10.16.6	UART Command Set	7-147
7.10.16.6.1	Transmit Commands	7-147
7.10.16.6.2	Receive Commands	7-148
7.10.16.7	UART Address Recognition (Receiver)	7-149
7.10.16.8	UART Control Characters (Receiver)	7-150
7.10.16.9	Wake-Up Timer (Receiver)	7-151
7.10.16.10	Break Support (Receiver)	7-151
7.10.16.11	Send Break (Transmitter)	7-153
7.10.16.12	Sending a Preamble (Transmitter)	7-153
7.10.16.13	Fractional Stop Bits (Transmitter)	7-153
7.10.16.14	UART Error-Handling Procedure	7-154
7.10.16.14.1	Transmission Error	7-155
7.10.16.14.2	Reception Errors	7-155
7.10.16.15	UART Mode Register (PSMR)	7-156
7.10.16.16	UART Receive Buffer Descriptor (Rx BD)	7-159
7.10.16.17	UART Transmit Buffer Descriptor (Tx BD).	7-163
7.10.16.18	UART Event Register (SCCE)	7-164
7.10.16.19	UART Mask Register (SCCM)	7-167
7.10.16.20	SCC Status Register (SCCS)	7-167
7.10.16.21	SCC UART Example	7-167
7.10.16.22	S-Records Programming Example	7-169
7.10.17	HDLC Controller	7-169
7.10.17.1	HDLC Controller Key Features	7-170
7.10.17.2	HDLC Channel Frame Transmission Processing	7-171
7.10.17.3	HDLC Channel Frame Reception Processing	7-172
7.10.17.4	HDLC Memory Map	7-172
7.10.17.5	HDLC Programming Model	7-174

Paragraph Number	Title	Page Number
7.10.17.6	HDLC Command Set	7-175
7.10.17.6.1	Transmit Commands	7-175
7.10.17.6.2	Receive Commands	7-176
7.10.17.7	HDLC Error-handling Procedure	7-176
7.10.17.7.1	Transmission Errors	7-176
7.10.17.7.2	Reception Errors	7-177
7.10.17.8	HDLC Mode Register (PSMR)	7-178
7.10.17.9	HDLC Receive Buffer Descriptor (Rx BD)	7-179
7.10.17.10	HDLC Transmit Buffer Descriptor (Tx BD)	7-183
7.10.17.11	HDLC Event Register (SCCE)	7-184
7.10.17.12	HDLC Mask Register (SCCM)	7-186
7.10.17.13	SCC Status Register (SCCS)	7-187
7.10.17.14	SCC HDLC Example #1	7-187
7.10.17.15	SCC HDLC Example #2	7-189
7.10.18	HDLC Bus Controller	7-189
7.10.18.1	HDLC Bus Key Features	7-192
7.10.18.2	HDLC Bus Operation	7-192
7.10.18.2.1	Accessing the HDLC Bus	7-192
7.10.18.2.2	More Performance	7-193
7.10.18.2.3	Delayed RTS Mode	7-194
7.10.18.2.4	Using the TSA	7-195
7.10.18.3	HDLC Bus Memory Map and Programming	7-196
7.10.18.3.1	GSMR Programming	7-196
7.10.18.3.2	PSMR Programming	7-196
7.10.18.3.3	HDLC Bus Controller Example	7-196
7.10.19	AppleTalk Controller	7-196
7.10.19.1	LocalTalk Bus Operation	7-197
7.10.19.2	Appletalk Controller Key Features	7-198
7.10.19.3	QUICC AppleTalk Hardware Connection	7-198
7.10.19.4	AppleTalk Memory Map and Programming Model	7-198
7.10.19.4.1	GSMR Programming	7-199
7.10.19.4.2	PSMR Programming	7-200
7.10.19.4.3	TODR Programming	7-200
7.10.19.4.4	AppleTalk Controller Example	7-200
7.10.20	BISYNC Controller	7-200
7.10.20.1	BISYNC Controller Features	7-201
7.10.20.2	BISYNC Channel Frame Transmission	7-201
7.10.20.3	BISYNC Channel Frame Reception	7-202
7.10.20.4	BISYNC Memory Map	7-203
7.10.20.5	BISYNC Command Set	7-204
7.10.20.5.1	Transmit Commands	7-204
7.10.20.5.2	Receive Commands	7-205
7.10.20.6	BISYNC Control Character Recognition	7-206
7.10.20.7	BSYNC-BISYNC SYNC Register	7-207
7.10.20.8	BDLE-BISYNC DLE Register	7-208

Paragraph Number	Title	Page Number
7.10.20.9	Transmitting and Receiving the Synchronization Sequence	7-208
7.10.20.10	BISYNC Error-Handling PROCEDURE	7-209
7.10.20.10.1	Transmission Errors	7-209
7.10.20.10.2	Reception Errors	7-209
7.10.20.11	BISYNC Mode Register (PSMR)	7-209
7.10.20.12	BISYNC Receive Buffer Descriptor (Rx BD)	7-211
7.10.20.13	BISYNC Transmit Buffer Descriptor (Tx BD)	7-213
7.10.20.14	BISYNC Event Register (SCCE)	7-216
7.10.20.15	BISYNC Mask Register (SCCM)	7-217
7.10.20.16	SCC Status Register (SCCS)	7-217
7.10.20.17	Programming the BISYNC Controller	7-217
7.10.20.18	SCC BISYNC Example	7-218
7.10.21	Transparent Controller	7-220
7.10.21.1	Transparent Controller Features	7-221
7.10.21.2	Transparent Channel Frame Transmission Processing	7-221
7.10.21.3	Transparent Channel Frame Reception Processing	7-222
7.10.21.4	Achieving Synchronization in Transparent Mode	7-223
7.10.21.4.1	In-Line Synchronization Pattern	7-223
7.10.21.4.2	Transparent Synchronization Example	7-224
7.10.21.5	Transparent Memory Map	7-225
7.10.21.6	Transparent Command Set	7-226
7.10.21.6.1	Transmit Commands	7-226
7.10.21.6.2	Receive Commands	7-227
7.10.21.7	Transparent Error-Handling Procedure	7-227
7.10.21.7.1	Transmission Errors	7-227
7.10.21.7.2	Reception Errors	7-228
7.10.21.8	Transparent Mode Register (PSMR)	7-228
7.10.21.9	Transparent Receive Buffer Descriptor (Rx BD)	7-228
7.10.21.10	Transparent Transmit Buffer Descriptor (Tx BD)	7-230
7.10.21.11	Transparent Event Register (SCCE)	7-232
7.10.21.12	Transparent Mask Register (SCCM)	7-233
7.10.21.13	SCC Status Register (SCCS)	7-233
7.10.21.14	SCC Transparent Example	7-233
7.10.22	RAM Microcodes	7-235
7.10.23	Ethernet Controller	7-235
7.10.23.1	Ethernet On QUICC—MC68EN360	7-236
7.10.23.2	Ethernet Key Features	7-237
7.10.23.3	Learning Ethernet on the QUICC	7-238
7.10.23.4	Connecting QUICC to Ethernet	7-239
7.10.23.5	Ethernet Channel Frame Transmission	7-241
7.10.23.6	Ethernet Channel Frame Reception	7-242
7.10.23.7	CAM Interface	7-243
7.10.23.8	Ethernet Memory Map	7-246
7.10.23.9	Ethernet Programming Model	7-250
7.10.23.10	Ethernet Command Set	7-250

Paragraph Number	Title	Page Number
7.10.23.10.1	Transmit Commands	7-250
7.10.23.10.2	Receive Commands	7-251
7.10.23.10.3	SET GROUP ADDRESS Command	7-251
7.10.23.11	Ethernet Address Recognition	7-252
7.10.23.12	Hash Table Algorithm	7-253
7.10.23.13	Interpacket Gap Time	7-254
7.10.23.14	Collision Handling	7-254
7.10.23.15	Internal and External Loopback	7-255
7.10.23.16	Ethernet Error-handling Procedure	7-255
7.10.23.16.1	Transmission Errors	7-255
7.10.23.16.2	Reception Errors	7-256
7.10.23.17	Ethernet Mode Register (PSMR)	7-256
7.10.23.18	Ethernet Receive Buffer Descriptor (Rx BD)	7-258
7.10.23.19	Ethernet Transmit Buffer Descriptor (Tx BD)	7-261
7.10.23.20	Ethernet Event Register (SCCE)	7-264
7.10.23.21	Ethernet Mask Register (SCCM)	7-265
7.10.23.22	Ethernet Status Register (SCCS)	7-265
7.10.23.23	SCC Ethernet Example.	7-266
7.11	Serial Management Controllers (SMCs)	7-268
7.11.1	SMC Overview	7-268
7.11.2	General SMC Mode Register (SMCMR)	7-270
7.11.3	SMC Buffer Descriptors	7-270
7.11.4	SMC Parameter RAM	7-270
7.11.4.1	BD Table Pointer (RBASE, TBASE)	7-271
7.11.4.2	SMC Function Code Registers (RFCR, TFCR)	7-272
7.11.4.3	Maximum Receive Buffer Length Register (MRBLR)	7-273
7.11.4.4	Receiver Buffer Descriptor Pointer (RBPTR)	7-273
7.11.4.5	Transmitter Buffer Descriptor Pointer (TBPTR)	7-274
7.11.4.6	Other General Parameters	7-274
7.11.5	Disabling the SMCs on the Fly	7-274
7.11.5.1	SMC Transmitter Full Sequence	7-275
7.11.5.2	SMC Transmitter Shortcut Sequence	7-275
7.11.5.3	SMC Receiver Full Sequence	7-275
7.11.5.4	SMC Receiver Shortcut Sequence	7-276
7.11.5.5	Switching Protocols	7-276
7.11.6	Saving Power	7-276
7.11.7	SMC as a UART	7-276
7.11.7.1	SMC UART Key Features	7-276
7.11.7.2	SMC UART Comparison	7-276
7.11.7.3	SMC UART Memory Map	7-277
7.11.7.4	SMC UART Transmission Processing	7-278
7.11.7.5	SMC UART Reception Processing	7-279
7.11.7.6	SMC UART Programming Model	7-279
7.11.7.7	SMC UART Command Set	7-279
7.11.7.7.1	Transmit Commands	7-279

Paragraph Number	Title	Page Number
7.11.7.7.2	Receive Commands	7-280
7.11.7.8	Send Break (Transmitter)	7-280
7.11.7.9	Sending a Preamble (Transmitter)	7-280
7.11.7.10	SMC UART Error-Handling Procedure	7-281
7.11.7.10.1	Overrun Error	7-281
7.11.7.10.2	Parity Error	7-281
7.11.7.10.3	Idle Sequence Receive	7-281
7.11.7.10.4	Framing Error	7-281
7.11.7.10.5	Break Sequence	7-281
7.11.7.11	SMC UART Mode Register (SMCMR)	7-281
7.11.7.12	SMC UART Receive Buffer Descriptor (Rx BD)	7-283
7.11.7.13	SMC UART Transmit Buffer Descriptor (Tx BD)	7-286
7.11.7.14	SMC UART Event Register (SMCE)	7-288
7.11.7.15	SMC UART Mask Register (SMCM)	7-290
7.11.8	SMC UART Example	7-290
7.11.9	SMC Interrupt Handling	7-291
7.11.10	SMC as a Transparent Controller	7-291
7.11.10.1	SMC Transparent Controller KEY Features	7-291
7.11.10.2	SMC Transparent Comparison	7-292
7.11.10.3	SMC Transparent Memory Map	7-292
7.11.10.4	SMC Transparent Transmission Processing	7-292
7.11.10.5	SMC Transparent Reception Processing	7-293
7.11.10.6	Using the SMSYNx Pin for Synchronization	7-293
7.11.10.7	Using the TSA for Synchronization	7-295
7.11.10.8	SMC Transparent Command Set	7-297
7.11.10.8.1	Transmit Commands	7-297
7.11.10.8.2	Receive Commands	7-297
7.11.10.9	SMC Transparent Error-Handling Procedure	7-298
7.11.10.9.1	Transmission Error (Underrun)	7-298
7.11.10.9.2	Reception Error (Overrun)	7-298
7.11.10.10	SMC Transparent Mode Register (SMCMR)	7-298
7.11.10.11	SMC Transparent Receive Buffer Descriptor (Rx BD)	7-299
7.11.10.12	SMC Transparent Transmit Buffer Descriptor (Tx BD)	7-300
7.11.10.13	SMC Transparent Event Register (SMCE)	7-302
7.11.10.14	SMC Transparent Mask Register (SMCM)	7-303
7.11.11	SMC Transparent NMSI Example	7-303
7.11.12	SMC Transparent TSA Example	7-304
7.11.13	SMC Interrupt Handling	7-305
7.11.14	SMC as a GCI Controller	7-305
7.11.14.1	SMC GCI Memory Map	7-306
7.11.14.1.1	SMC Monitor Channel Transmission	7-306
7.11.14.1.2	SMC Monitor Channel Reception	7-307
7.11.14.2	SMC C/I Channel Handling	7-307
7.11.14.2.1	SMC C/I Channel Transmission	7-307
7.11.14.2.2	SMC C/I Channel Reception	7-307

Paragraph Number	Title	Page Number
7.11.14.3	SMC Commands in GCI Mode	7-307
7.11.14.4	SMC GCI Mode Register (SMCMR)	7-308
7.11.14.5	SMC Monitor Channel Rx BD	7-309
7.11.14.6	SMC Monitor Channel Tx BD	7-310
7.11.14.7	SMC C/I Channel Receive Buffer Descriptor (Rx BD)	7-310
7.11.14.8	SMC C/I Channel Transmit Buffer Descriptor (Tx BD)	7-311
7.11.14.9	SMC Event Register (SMCE)	7-311
7.11.14.10	SMC Mask Register (SMCM)	7-312
7.12	Serial Peripheral Interface (SPI)	7-312
7.12.1	Overview	7-312
7.12.2	SPI Key Features	7-313
7.12.3	SPI Clocking and Pin Functions	7-314
7.12.4	SPI Transmit/Receive Process	7-315
7.12.4.1	SPI Master Mode	7-315
7.12.4.2	SPI Slave Mode	7-316
7.12.4.3	SPI Multi-Master Operation	7-316
7.12.5	SPI Programming Model	7-317
7.12.5.1	SPI Mode Register (SPMODE)	7-317
7.12.5.2	SPI Command Register (SPCOM)	7-319
7.12.5.3	SPI Parameter RAM Memory Map	7-320
7.12.5.3.1	BD Table Pointer (RBASE, TBASE)	7-320
7.12.5.3.2	SPI Function Code Registers (RFCR, TFCR)	7-321
7.12.5.3.3	Maximum Receive Buffer Length Register (MRBLR)	7-322
7.12.5.3.4	Receiver Buffer Descriptor Pointer (RBPTR)	7-322
7.12.5.3.5	Transmitter Buffer Descriptor Pointer (TBPTR)	7-323
7.12.5.3.6	Other General Parameters	7-323
7.12.5.4	SPI Commands	7-323
7.12.5.4.1	INIT TX PARAMETERS Command	7-323
7.12.5.4.2	CLOSE Rx BD Command	7-323
7.12.5.4.3	INIT RX PARAMETERS Command	7-323
7.12.5.5	SPI Buffer Descriptor Ring	7-324
7.12.5.5.1	SPI Receive Buffer Descriptor (Rx BD)	7-324
7.12.5.5.2	SPI Transmit Buffer Descriptor (Tx BD)	7-326
7.12.5.6	SPI Event Register (SPIE)	7-328
7.12.5.7	SPI Mask Register (SPIM)	7-329
7.12.6	SPI Master Example	7-329
7.12.7	SPI Slave Example	7-330
7.12.8	SPI Interrupt Handling	7-331
7.13	Parallel Interface Port (PIP)	7-331
7.13.1	PIP Key Features	7-331
7.13.2	PIP Overview	7-332
7.13.3	General-Purpose I/O Pins (Port B)	7-333
7.13.4	Interlocked Data Transfers	7-333
7.13.5	Pulsed Data Transfers	7-334
7.13.5.1	Busy Signal	7-335

Paragraph Number	Title	Page Number
7.13.5.2	Pulsed Handshake Timing	7-336
7.13.6	Transparent Data Transfers	7-338
7.13.7	Programming Model	7-338
7.13.7.1	Parameter RAM	7-338
7.13.7.2	PIP Configuration Register (PIPC)	7-339
7.13.7.3	PIP Timing Parameters Register (PTPR)	7-341
7.13.7.4	PIP Buffer Descriptors	7-341
7.13.7.5	PIP Event Register (PIPE)	7-341
7.13.7.6	PIP Mask Register (PIPM)	7-342
7.13.8	Centronics Controller Overview	7-342
7.13.8.1	Centronics Controller Key Features	7-344
7.13.8.2	Centronics Channel Transmission	7-345
7.13.8.3	Centronics Transmitter Memory Map	7-345
7.13.8.4	Buffer Descriptor Table Pointer (TBASE)	7-346
7.13.8.5	Status Mask Register (SMASK)	7-346
7.13.8.6	Centronics Function Code Register (CFCR)	7-346
7.13.8.7	Transmitter Buffer Descriptor Pointer (TBPTR)	7-347
7.13.8.8	Centronics Transmitter Programming Model	7-347
7.13.8.9	Centronics Transmitter Command Set	7-347
7.13.8.9.1	*STOP TRANSMIT* Command	7-347
7.13.8.9.2	*RESTART TRANSMIT* Command	7-347
7.13.8.9.3	*INIT TX PARAMETERS C*ommand	7-348
7.13.8.10	Transmission Errors	7-348
7.13.8.10.1	Buffer Descriptor Not Ready	7-348
7.13.8.10.2	Printer Off-Line Error	7-348
7.13.8.10.3	Printer Fault	7-348
7.13.8.10.4	Paper Error	7-348
7.13.8.10.5	Centronics Transmitter Buffer Descriptor	7-348
7.13.8.11	Centronics Transmitter Event Register (PIPE)	7-349
7.13.8.12	Centronics Channel Reception	7-350
7.13.8.13	Centronics Receiver Memory Map	7-350
7.13.8.14	Buffer Descriptor Table Pointer (RBASE)	7-351
7.13.8.15	Centronics Function Code Register (CFCR)	7-351
7.13.8.16	Receiver Buffer Descriptor Pointer (RBPTR)	7-352
7.13.8.17	Centronics Receiver Programming Model	7-352
7.13.8.18	Centronics Control Characters	7-352
7.13.8.19	Centronics Silence Period	7-354
7.13.8.20	Centronics Receiver Command Set	7-354
7.13.8.20.1	*INIT RX PARAMETERS* Command	7-354
7.13.8.20.2	*CLOSE RX BD* Command	7-354
7.13.8.21	Receiver Errors	7-354
7.13.8.21.1	Buffer Descriptor Busy	7-354
7.13.8.22	Centronics Receive Buffer Descriptor	7-354
7.13.8.23	Centronics Receiver Event Register (PIPE)	7-355
7.13.9	Port B Registers	7-356

Paragraph Number	Title	Page Number
7.13.9.1	Port B Assignment Registers (PBPAR)	7-356
7.13.9.2	Data Direction Register (PBDIR)	7-356
7.13.9.3	Data Register (PBDAT)	7-356
7.13.9.4	Open-Drain Register (PBODR)	7-356
7.14	Parallel I/O Ports	7-356
7.14.1	Parallel I/O Key Features	7-357
7.14.2	Parallel I/O Overview	7-357
7.14.3	Port A Pin Functions	7-357
7.14.4	Port A Registers	7-359
7.14.4.1	Port A Open-Drain Register (PAODR)	7-359
7.14.4.2	Port A Data Register (PADAT)	7-359
7.14.4.3	Port A Data Direction Register (PADIR)	7-359
7.14.4.4	Port A Pin Assignment Register (PAPAR)	7-359
7.14.5	Port A Examples	7-360
7.14.6	Port B Pin Functions	7-362
7.14.7	Port B Registers	7-363
7.14.7.1	Port B Open-Drain Register (PBODR)	7-363
7.14.7.2	Port B Data Register (PBDAT)	7-364
7.14.7.3	Port B Data Direction Register (PBDIR)	7-364
7.14.7.4	Port B Pin Assignment Register (PBPAR)	7-364
7.14.8	Port B Example	7-365
7.14.9	Port C Pin Functions	7-365
7.14.10	Port C Registers	7-367
7.14.10.1	Port C Data Register (PCDAT)	7-368
7.14.10.2	Port C Data Direction Register (PCDIR)	7-368
7.14.10.3	Port C Pin Assignment Register (PCPAR)	7-368
7.14.10.4	Port C Special Options (PCSO)	7-368
7.14.10.5	Port C Interrupt Control Register (PCINT)	7-369
7.15	CPM Interrupt Controller (CPIC)	7-369
7.15.1	Overview	7-370
7.15.2	CPM Interrupt Source Priorities	7-372
7.15.2.1	SCC Relative Priority	7-372
7.15.2.2	Highest Priority Interrupt	7-372
7.15.2.3	Nested Interrupts	7-373
7.15.3	Masking Interrupt Sources in the CPM	7-374
7.15.4	Interrupt Vector Generation and Calculation	7-375
7.15.5	CPIC Programming Model	7-377
7.15.5.1	CPM Interrupt Configuration Register (CICR)	7-377
7.15.5.2	CPM Interupt Pending Register (CIPR)	7-379
7.15.5.3	CPM Interrupt Mask Register (CIMR)	7-380
7.15.5.4	CPM Interrupt In-Service Register (CISR)	7-380
7.15.6	Interrupt Handler Examples	7-381
7.15.6.1	Example 1—PC6 Interrupt Handler	7-381
7.15.6.2	Example 2—SCC1 Interrupt Handler	7-381

Paragraph Number	Title	Page Number
	Section 8 Scan Chain Test Access Port	
8.1	Overview	8-1
8.2	TAP Controller	8-2
8.3	Boundary Scan Register	8-3
8.4	Instruction Register	8-10
8.4.1	EXTEST	8-10
8.4.2	SAMPLE/PRELOAD	8-10
8.4.3	BYPASS	8-11
8.4.4	CLAMP	8-11
8.4.5	HI-Z	8-11
8.5	QUICC Restrictions	8-11
8.6	Non-Scan Chain Operation	8-12
	Section 9 Applications	
9.1	Minimum System Configuration	9-1
9.1.1	QUICC Hardware Configuration	9-1
9.1.1.1	QUICC Basic Accesses	9-1
9.1.1.2	Clocking Strategy.	9-3
9.1.1.3	Resetting the QUICC	9-3
9.1.1.4	Interrupts.	9-3
9.1.1.5	Bus Arbitration.	9-3
9.1.1.6	Breakpoint Generation.	9-3
9.1.1.7	Bus Monitor Function.	9-3
9.1.1.8	Spurious Interrupt Monitor.	9-3
9.1.1.9	Software Watchdog.	9-3
9.1.1.10	Double Bus Fault.	9-4
9.1.1.11	JTAG and Three-State.	9-4
9.1.1.12	QUICC Serial Ports.	9-4
9.1.2	Memory Interfaces	9-4
9.1.2.1	QUICC Memory Interface Pins.	9-4
9.1.2.2	Regular EPROM	9-5
9.1.2.3	Flash EPROM.	9-5
9.1.2.4	SRAM	9-6
9.1.2.5	EEPROM.	9-7
9.1.2.6	DRAM SIMM.	9-8
9.1.2.7	DRAM Devices.	9-9
9.1.3	Software Configuration	9-10
9.1.3.1	Basic Initialization.	9-10
9.1.3.2	Configuring the Memory Controller.	9-11
9.1.3.3	Using the QUICC in 16-Bit Data Bus Mode	9-12
9.2	How to take A QUICC Software Test-Drive	9-13
	Step 1: Decide on Reset Stack Pointer and Initial Program Counter	9-13
	Step 2: Stay in Supervisor Mode	9-13
	Step 3: Write the VBR	9-14

Paragraph Number	Title	Page Number
	Step 4: Write the MBAR	9-14
	Step 5: Verify a Dual-Port RAM Location	9-14
	Step 6: Is This a Power-Up Reset?	9-14
	Step 7: Deal with the Clock Synthesizer	9-14
	Step 8: Initialize System Protection	9-15
	Step 9: Clear Entire Dual-Port RAM	9-15
	Step 10: Write the PEPAR	9-15
	Step 11: Remap Chip Select 0	9-15
	Step 12: Initialize the System RAM	9-15
	Step 13: Copy the EVT to System RAM	9-16
	Step 14: Initialize All Other Memory and Peripherals	9-16
	Step 15: Initialize the Rest of the SIM60	9-16
	Step 16: Generate a SIM60 Interrupt	9-16
	Step 17: Test the CPM	9-17
	Step 18: Generate Interrupts with the CPM	9-17
	Step 19: Enable External Interrupts	9-17
	Step 20: Enable External Bus Masters	9-18
	Step 21: Off to the Races	9-18
9.3	Porting MC68302 IMP Code to the MC68360 QUICC	9-18
9.3.1	CPU and Compilers	9-18
9.3.2	Differences/Similarities	9-18
9.3.3	Notes About Porting	9-19
9.3.4	How To Port MC68302 Functions	9-19
9.3.4.1	System Configuration Registers.	9-19
9.3.4.1.1	Base Address Register (BAR).	9-19
9.3.4.1.2	System Control Register (SCR).	9-20
9.3.4.2	System RAM.	9-21
9.3.4.2.1	Buffer Descriptors.	9-21
9.3.4.2.2	Protocol-Independent Parameter RAM Values.	9-21
9.3.4.2.3	Protocol-Dependent Parameter RAM Values.	9-22
9.3.4.3	Internal Registers (System Integration Block).	9-23
9.3.4.4	Internal Registers (Communication Processor).	9-26
9.4	Using the QUICC MC68040 Companion Mode	9-31
9.4.1	MC68EC040 to QUICC Interface	9-32
9.4.1.1	MC68EC040 Reads And Writes to QUICC.	9-32
9.4.1.2	Clocking Strategy.	9-34
9.4.1.3	Reset Strategy.	9-34
9.4.1.4	Interrupts.	9-34
9.4.2	Memory Interfaces	9-37
9.4.2.1	QUICC Memory Interface Pins.	9-37
9.4.2.2	Regular EPROM.	9-38
9.4.2.3	Burst EPROM.	9-38
9.4.2.4	Flash EPROM.	9-41
9.4.2.5	Regular SRAM.	9-41
9.4.2.6	Burst SRAM.	9-41

Paragraph Number	Title	Page Number
9.4.2.7	EEPROM.	9-45
9.4.2.8	DRAM SIMM	9-45
9.4.2.9	DRAM Devices.	9-46
9.4.3	Software Configuration	9-48
9.4.3.1	Basic Initialization.	9-49
9.4.3.2	Configuring the Memory Controller.	9-49
9.4.4	Interfacing Multiple QUICCs to an MC68EC040	9-51
9.5	Selecting Cache Modes on the MC68EC040	9-51
9.5.1	The Algorithm	9-52
9.5.2	Protection	9-52
9.5.3	MC68EC040 Cache Behavior	9-53
9.5.4	Enabling the Caching Modes	9-53
9.6	Interfacing the QUICC to the 53C90 scsi controller	9-54
9.6.1	SCSI General Overview	9-54
9.6.2	Physical Interface	9-54
9.6.3	Logical Interface	9-59
9.6.4	Functional Description	9-61
9.6.5	Hardware Configuration	9-62
9.6.5.1	Clocking Strategy.	9-62
9.6.5.2	Reset Strategy	9-62
9.6.5.3	Read/Write timing.	9-62
9.6.5.4	Interrupt Handling.	9-62
9.6.5.5	IDMA1 Setup and Timing.	9-64
9.6.5.6	QUICC I/O Ports.	9-65
9.6.6	Active SCSI Terminations	9-65
9.6.7	Software Configuration	9-65
9.6.7.1	Configuring IDMA1.	9-65
9.6.7.2	Configuring The Memory Controller.	9-66
9.7	Using the QUICC as a TAP Controller for Board Self-Test	9-66
9.7.1	Board Layout	9-67
9.7.2	Board Testing	9-68
9.7.3	Microcontroller Interface	9-70
9.7.4	Test Pattern Generation	9-72
9.8	Interfacing an MC68EC030 Master to the QUICC In Slave Mode	9-74
9.8.1	MC68EC030 to QUICC Interface	9-74
9.8.1.1	MC68EC030 Reads and Writes to QUICC.	9-75
9.8.1.2	Clocking Strategy.	9-75
9.8.1.3	Reset Strategy	9-77
9.8.1.4	Interrupts	9-77
9.8.1.5	Bus Arbitration	9-78
9.8.1.6	Breakpoint Generation	9-78
9.8.1.7	Bus Monitor Function	9-78
9.8.1.8	Spurious Interrupt Monitor	9-78
9.8.1.9	Software Watchdog	9-79
9.8.1.10	Periodic Interval Timer	9-79

Paragraph Number	Title	Page Number
9.8.1.11	MC68EC030 Caching Configuration	9-79
9.8.1.12	Double Bus Fault	9-79
9.8.1.13	JTAG and Three-State	9-79
9.8.1.14	QUICC Serial Ports	9-79
9.8.2	Memory Interfaces	9-79
9.8.2.1	QUICC Memory Interface Pins	9-80
9.8.2.2	Regular EPROM or Flash EPROM	9-80
9.8.2.3	Regular SRAM	9-82
9.8.2.4	EEPROM	9-84
9.8.2.5	DRAM SIMM	9-84
9.8.2.6	DRAM Devices	9-86
9.8.3	Software Configuration	9-86
9.8.3.1	Basic Initialization	9-86
9.8.3.2	Configuring the Memory Controller	9-87
9.8.4	Interfacing Multiple QUICCs to an MC68EC030	9-89
9.8.5	Using a Higher Speed MC68EC030 Master with the QUICC	9-89
9.9	Putting a Background Debug Mode Connector on a Target Board	9-90

Section 10
Electrical Characteristics

10.1	Maximum Ratings	10-1
10.2	Thermal Characteristics	10-2
10.3	Power Considerations	10-2
10.4	AC Electrical Specification Definitions	10-3
10.5	DC Electrical Specifications	10-5
10.6	AC Power Dissipation	10-6
10.7	AC Electrical Specifications Control Timing	10-7
10.8	External Capacitor for PLL	10-8
10.9	Bus Operation AC Timing Specifications	10-9
10.9	Bus Operation AC Timing Specifications (Continued)	10-10
10.9	Bus Operation AC Timing Specifications (Continued)	10-11
10.9	Bus Operation AC Timing Specifications (Continued	10-12
10.10	Bus Operation—DRAM Accesses AC Timing Specifications	10-28
10.11	030/QUICC Bus Type Slave Mode Bus Arbitration AC Electrical Specifications	10-33
10.12	030/QUICC Bus Type Slave Mode Internal Read/Write/IACK Asynchronous Cycles AC Electrical Specifications	10-36
10.14	030/QUICC Bus Type SRAM/DRAM Cycles AC Electrical Specifications	10-44
10.15	040 Bus Type Slave Mode Bus Arbitration AC Electrical Specifications	10-49
10.16	040 Bus Type Slave Mode Internal Read/write/IACK Cycles AC Electrical Specifications	10-51
10.17	040 Bus Type SRAM/DRAM Cycles Ac Electrical Specifications	10-56
10.18	IDMA AC Electrical Specifications	10-62
10.19	PIP/PIO AC Electrical Specifications	10-64

Paragraph Number	Title	Page Number
10.20	Interrupt Controller AC Electrical Specifications	10-66
10.21	Baud Rate Generator AC Electrical Specifications	10-67
10.22	Timer Electrical Specifications	10-68
10.23	SI Electrical Specifications	10-69
10.24	SCC in NMSI Mode—External Clock Electrical Specifications	10-75
10.25	SCC in NMSI MODE—Internal Clock Electrical Specifications	10-75
10.26	Ethernet Electrical Specifications	10-77
10.27	SMC Transparent Mode Electrical Specifications	10-80
10.28	SPI Master Electrical Specifications	10-82
10.29	SPI Slave Electrical Specifications	10-83
10.30	JTAG Electrical Specifications	10-85

Section 11
Ordering Information and Mechanical Data

11.1	Standard Ordering Information	11-1
11.2	Pin Assignment—240-Lead Quad Flat Pack (QFP)	11-2
11.3	Pin Assignment—241-Lead Pin Grid Array (PGA)	11-4
11.4	Pin Assignment—357-Lead BALL Grid Array (BGA)	11-5
11.5	Package Dimensions—CQFP (FE Suffix)	11-6
11.6	Package Dimensions—PGA (RC Suffix)	11-7
11.7	Package Dimensions—BGA (ZP Suffix)	11-8

Appendix A
Serial Performance

Appendix B
Development Tools and Support

B.1	Motorola Software Modules	B-1
B.2	Other protocol Software Support	B-5
B.3	Third-Party Software Support	B-6
B.4	M68360QUADS Development System	B-6
B.5	Other Development Boards	B-10
B.6	Direct Target Development	B-10

Appendix C
RISC Microcode from RAM

C.1	Signaling System #7 Controller	C-1
C.1.1	Performance	C-2
C.2	Multiple GCI Controller	C-3
C.2.1	Typical Application	C-3
C.2.2	MGCI Controller Key Features	C-3
C.2.3	Performance	C-4
C.3	ATOM1/ATM Controller	C-4
C.3.1	Key Features	C-4
C.3.2	Performance	C-5

Paragraph Number	Title	Page Number
C.4	Asynchronous HDLC for PPP	C-6
C.4.1	Key Features	C-6
C.4.2	Performance	C-7
C.5	PROFIBUS Controller	C-7
C.5.1	Key Features	C-7
C.6	Enhanced Ethernet Filtering	C-8
C.6.1	Key Features	C-8
C.6.2	Performance	C-8

Appendix D
MC68MH360 Product Brief

D.1	QUICC32 Key Features	D-1
D.1.1	General	D-1
D.1.2	Serial Interface	D-2
D.1.3	System Interface	D-2
D.2	QUICC Architecture Overview	D-2
D.2.1	CPU32+ Core	D-3
D.2.2	System Integration Module (SIM60)	D-4
D.2.3	Communications Processor Module (CPM)	D-4
4.2.3.1	QUICC32 Serial Configurations	D-5
D.2.4	The QMC Microcode	D-7
D.2.5	Data Flow	D-8
D.2.6	Data Management	D-8
D.2.7	Performance	D-9
D.2.8	Development Support	D-10
D.2.9	Ordering Information	D-10

SECTION 1
INTRODUCTION

The MC68360 QUad Integrated Communication Controller (QUICC™) is a versatile one-chip integrated microprocessor and peripheral combination that can be used in a variety of controller applications. It particularly excels in communications activities. The QUICC (pronounced “quick”) can be described as a next-generation MC68302 with higher performance in all areas of device operation, increased flexibility, major extensions in capability, and higher integration. The term "quad" comes from the fact that there are four serial communications controllers (SCCs) on the device; however, there are actually seven serial channels: four SCCs, two serial management controllers (SMCs), and one serial peripheral interface (SPI).

The purpose of this document is to describe the operation of all QUICC functionality. Although this document has an overview of the CPU32+, the M68000PM/AD *M68000 Family Programmer's Reference Manual* should be used in addition to this document. The CPU32RM/AD, *M68300 Family CPU32 Reference Manual,* also provides information on the CPU32.

1.1 QUICC KEY FEATURES

The following list summarizes the key MC68360 QUICC features:

- CPU32+ Processor (4.5 MIPS at 25 MHz)
  - —32-Bit Version of the CPU32 Core (Fully Compatible with the CPU32)
  - —Background Debug Mode
  - —Byte-Misaligned Addressing
- Up to 32-Bit Data Bus (Dynamic Bus Sizing for 8 and 16 Bits)
- Up to 32 Address Lines (At Least 28 Always Available)
- Complete Static Design (0–25-MHz Operation)
- Slave Mode To Disable CPU32+ (Allows Use with External Processors)
  - —Multiple QUICCs Can Share One System Bus (One Master)
  - —MC68040 Companion Mode Allows QUICC To Be an MC68040 Companion Chip and Intelligent Peripheral (22 MIPS at 25 MHz)
  - —Also Supports External MC68030-Type Bus Masters
  - —All QUICC Features Usable in Slave Mode
- Memory Controller (Eight Banks)
  - —Contains Complete Dynamic Random-Access Memory (DRAM) Controller
  - —Each Bank Can Be a Chip Select or Support a DRAM Bank
  - —Up to 15 Wait States

—Glueless Interface to DRAM Single In-Line Memory Modules (SIMMs), Static Random-Access Memory (SRAM), Electrically Programmable Read-Only Memory (EPROM), Flash EPROM, etc.
—Four CAS lines, Four WE lines, One OE line
—Boot Chip Select Available at Reset (Options for 8-, 16-, or 32-Bit Memory)
—Special Features for MC68040 Including Burst Mode Support

- Four General-Purpose Timers
  - —Superset of MC68302 Timers
  - —Four 16-Bit Timers or Two 32-Bit Timers
  - —Gate Mode Can Enable/Disable Counting
- Two Independent DMAs (IDMAs)
  - —Single Address Mode for Fastest Transfers
  - —Buffer Chaining and Auto Buffer Modes
  - —Automatically Performs Efficient Packing
  - —32-Bit Internal and External Transfers
- System Integration Module (SIM60)
  - —Bus Monitor
  - —Double Bus Fault Monitor
  - —Spurious Interrupt Monitor
  - —Software Watchdog
  - —Periodic Interrupt Timer
  - —Low Power Stop Mode
  - —Clock Synthesizer
  - —Breakpoint Logic Provides On-Chip Hardware Breakpoints
  - —External Masters May Use On-Chip Features Such As Chip Selects
  - —On-Chip Bus Arbitration with No Overhead for Internal Masters
  - —IJTAG Test Access Port
- Interrupts
  - —Seven External IRQ Lines
  - —12 Port Pins with Interrupt Capability
  - —16 Internal Interrupt Sources
  - —Programmable Priority Between SCCs
  - —Programmable Highest Priority Request
- Communications Processor Module (CPM)
  - —RISC Controller
  - —Many New Commands (e.g., Graceful Stop Transmit, Close RxBD)
  - —224 Buffer Descriptors
  - —Supports Continuous Mode Transmission and Reception on All Serial Channels
  - —2.5 Kbytes of Dual-Port RAM
  - —14 Serial DMA (SDMA) Channels
  - —Three Parallel I/O Registers with Open-Drain Capability
  - —Each Serial Channel Can Have Its Own Pins (NMSI Mode)
- Four Baud Rate Generators

—Independent (Can Be Connected to Any SCC or SMC)
—Allows Changes During Operation
—Autobaud Support Option

- Four SCCs

—Ethernet/IEEE 802.3 Optional on SCC1 (Full 10-Mbps Support)
—HDLC/SDLC[1] (All Four Channels Supported at 2 Mbps)
—HDLC Bus (Implements an HDLC-Based Local Area Network (LAN))
—AppleTalk[2]
—Signaling System #7
—Universal Asynchronous Receiver Transmitter (UART)
—Synchronous UART
—Binary Synchronous Communication (BISYNC)
—Totally Transparent (Bit Streams)
—Totally Transparent (Frame Based with Optional Cyclic Redundancy Check (CRC))
—Profibus (RAM Microcode Option)
—Asynchronous HDLC (RAM Microcode Option)
—DCMP[3] (RAM Microcode Option)
—V.14 (RAM Microcode Option)
—X.21 (RAM Microcode Option)

- Two SMCs

—UART
—Transparent
—General Circuit Interface (GCI) Controller
—Can Be Connected to the Time-Division Multiplexed (TDM) Channels

- One SPI

—Superset of the MC68302 SCP
—Supports Master and Slave Modes
—Supports Multimaster Operation on the Same Bus

- Time-Slot Assigner
- Supports Two TDM Channels

—Each TDM Channel Can Be T1, CEPT, PCM Highway, ISDN Basic Rate, ISDN Primary Rate, User Defined
—1- or 8-Bit Resolution
—Allows Independent Transmit and Receive Routing, Frame Syncs, Clocking
—Allows Dynamic Changes
—Can Be internally Connected to Six Serial Channels (Four SCCs and Two SMCs)

1. SDLC is a trademark of International Business Machines.
2. AppleTalk is a registered trademark of Apple Computer, Inc.
3. DDCMP is a trademark of Digital Equipment Corporation.

- Parallel Interface Port
  —Centronics[4] Interface Support
  —Supports Fast Connection Between QUICCs
- 240 Pins Defined: 241-Lead Pin Grid Array (PGA) and 240-Lead Plastic Quad Flat Pack (PQFP)

1.2 QUICC ARCHITECTURE OVERVIEW

The QUICC is 32-bit controller that is an extension of other members of the Motorola M68300 family. Like other members of the M68300 family, the QUICC incorporates the inter-module bus (IMB). (The MC68302 is an exception, having an M68000 bus on chip.) The IMB provides a common interface for all modules of the M68300 family, which allows Motorola to develop new devices more quickly by using the library of existing modules. Although the IMB definition always included an option for an on-chip 32-bit bus, the QUICC is the first device to implement this option.

The QUICC is comprised of three modules: the CPU32+ core, the SIM60, and the CPM. Each module utilizes the 32-bit IMB. The MC68360 QUICC block diagram is shown in Figure 1-1.

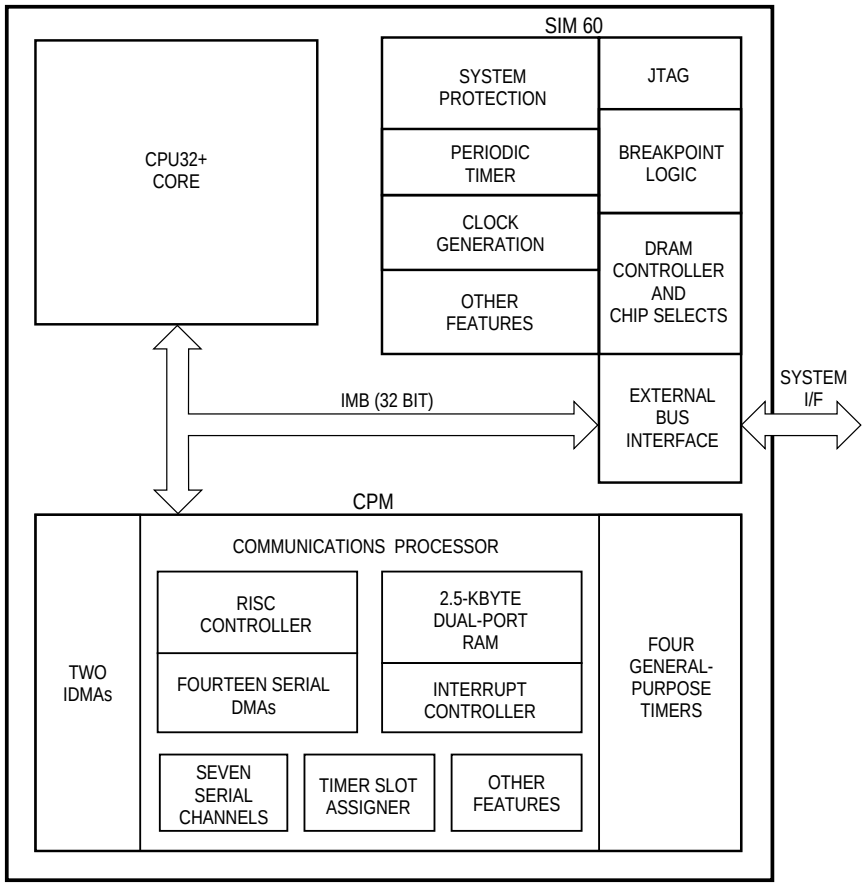


Figure 1-1. QUICC Block Diagram

4. Centronics is a trademark of Centronics, Inc.

1.2.1 CPU32+ Core

The CPU32+ core is a CPU32 that has been modified to connect directly to the 32-bit IMB and apply the larger bus width. Although the original CPU32 core had a 32-bit internal data path and 32-bit arithmetic hardware, its interface to the IMB was 16 bits. The CPU32+ core can operate on 32-bit external operands with one bus cycle. This allows the CPU32+ core to fetch a long-word instruction in one bus cycle and to fetch two word-length instructions in one bus cycle, filling the internal instruction queue more quickly. The CPU32+ core can also read and write 32-bits of data in one bus cycle.

Although the CPU32+ instruction timings are improved, its instruction set is identical to that of the CPU32. It will also execute the entire M68000 instruction set. It contains the same background debug mode (BDM) features as the CPU32. No new compilers, assemblers, or other software support tools need be implemented for the CPU32+; standard CPU32 tools can be used.

The CPU32+ delivers approximately 4.5 MIPS at 25 MHz, based on the standard (accepted) assumption that a 10-MHz M68000 delivers 1 VAX MIPS. If an application requires more performance, the CPU32+ can be disabled, allowing the rest of the QUICC to operate as an intelligent peripheral to a faster processor. The QUICC provides a special mode called MC68040 companion mode to allow it to conveniently interface to members of the M68040 family. This two-chip solution provides a 22-MIPS performance at 25 MHz.

The CPU32+ also offers automatic byte alignment features that are not offered on the CPU32. These features allow 16 or 32-bit data to be read or written at an odd address. The CPU32+ automatically performs the number of bus cycles required.

1.2.2 System Integration Module (SIM60)

The SIM60 integrates general-purpose features that would be useful in almost any 32-bit processor system. The term “SIM60” is derived from the QUICC part number, MC68360. The SIM60 is an enhanced version of the SIM40 that exists on the MC68340 and MC68330 devices.

First, new features, such as a DRAM controller and breakpoint logic, have been added. Second, the SIM40 was modified to support a 32-bit IMB as well as a 32-bit external system bus. Third, new configurations, such as slave mode and internal accesses by an external master, are supported.

Although the QUICC is always a 32-bit device internally, it may be configured to operate with a 16-bit data bus. Regardless of the choice of the system bus size, dynamic bus sizing is supported. Bus sizing allows 8-, 16-, and 32-bit peripherals and memory to exist in the 32-bit system bus mode and 8- and 16-bit peripherals and memory to exist in the 16-bit system bus mode.

1.2.3 Communications Processor Module (CPM)

The CPM contains features that allow the QUICC to excel in communications and control applications. These features may be divided into three sub-groups:

- Communications Processor (CP)
- Two IDMA Controllers
- Four General-Purpose Timers

The CP provides the communication features of the QUICC. Included are a RISC processor, four SCCs, two SMCs, one SPI, 2.5 Kbytes of dual-port RAM, an interrupt controller, a time slot assigner, three parallel ports, a parallel interface port, four independent baud rate generators, and fourteen serial DMA channels to support the SCCs, SMCs, and SPI.

The IDMAs provide two channels of general-purpose DMA capability. They offer high-speed transfers, 32-bit data movement, buffer chaining, and independent request and acknowledge logic. The RISC controller may access the IDMA registers directly in the buffer chaining modes. The QUICC IDMAs are similar to, yet enhancements of, the two DMA channels found on the MC68340 and the one IDMA channel found on the MC68302.

The four general-purpose timers on the QUICC are functionally similar to the two general-purpose timers found on the MC68302. However, they offer some minor enhancements, such as the internal cascading of two timers to form a 32-bit timer. The QUICC also contains a periodic interval timer in the SIM60, bringing the total to five on-chip timers.

1.3 UPGRADING DESIGNS FROM THE MC68302

Since the QUICC is a next-generation MC68302, many designers currently using the MC68302 may wish to use the QUICC in a follow-on design. The following paragraphs briefly discuss this endeavor in terms of architectural approach, hardware issues, and software issues. See Section 9 Applications for further information.

1.3.1 Architectural Approach

The QUICC is the logical extension of the MC68302, but the overall architecture and philosophy of the MC68302 design remains intact in the QUICC. The QUICC keeps the best features of the MC68302, while making the changes required to provide for the increased flexibility, integration, and performance requested by customers. Because the CPM is probably the most difficult module to learn, anyone who has used the MC68302 can easily become familiar with the QUICC since the CPM architectural approach remains intact.

The most significant architectural change made on the QUICC was the translation of the design into the standard M68300 family IMB architecture, resulting in a faster CPU and different system integration features.

Although the features of the SIM60 do not exactly correspond to those of the MC68302 SIM, they are very similar. The QUICC SIM60 combines the best MC68302 SIM features with the best MC68340 SIM features for improved performance.

Because of the similarity of the QUICC SIM60 and CPU to other members of the M68300 family, such as the MC68332 and the MC68340, previous users of these devices will be comfortable with these same features on the QUICC.

1.3.2 Hardware Compatibility Issues

The following list summarizes the hardware differences between the MC68302 and the QUICC:

- Pinout—The pinout is not the same. The QUICC has 240 pins; the MC68302 has 132 pins.
- Package—Both devices offer PGA and PQFP packages. However, the QUICC PQFP package has a 20-mil pitch; whereas, the MC68302 PQFP package has a 25-mil pitch.
- System Bus—The system bus signals now look like those of the MC68030 as opposed to those of the M68000. It is still possible to interface M68000 peripherals to the QUICC, utilizing the same techniques used to interface them to an MC68020 or MC68030.
- System Bus in Slave Mode—A number of QUICC pins take on new functionality in slave mode to support an external MC68EC040. On the MC68302, the pin names generally remained the same in slave mode.
- Peripheral Timing—The external timings of the peripherals (SCCs, timers, etc.) are very similar (if not identical) to corresponding peripherals on the MC68302.
- Pin Assignments—The assignment of peripheral functions to I/O pins is different in several ways. First, the QUICC contains more general-purpose parallel I/O pins than the MC68302. However, the QUICC offers many more functions than even a 240-pin package would normally allow, resulting in more multifunctional pins than the MC68302.

1.3.3 Software Compatibility Issues

The following list summarizes the major software differences between the MC68302 and the QUICC:

- Since the CPU32+ is a superset of the M68000 instruction set, all previously written code will run. However, if such code is accessing the MC68302 peripherals, it will require some modification.
- The QUICC contains an 8-Kbyte block of memory as opposed to a 4-Kbyte block on the MC68302. The register addresses within that memory map are different.
- The code used to initialize the system integration features of the MC68302 has to be modified to write the corresponding features on the QUICC SIM60. Code written for the MC68340 may be adapted in large part.
- As much as possible, QUICC CPM features were made identical to those of the MC68302 CP. The most important benefit is that the code flow (if not the code itself) will port easily from the MC68302 to the QUICC. The nuances learned from the MC68302 will still be useful in the QUICC.
- Although the registers used to initialize the QUICC CPM are new (for example, the SCM on the MC68302 is replaced with the GSMR and PSMR on the QUICC), most registers retain their original purpose such as the SCC event, SCC mask, SCC status, and com-

mand registers. The parameter RAM of the SCCs is very similar, and most parameter RAM register names and usage are retained. More importantly, the basic structure of a buffer descriptor (BD) on the QUICC is identical to that of the MC68302, except for a few new bit functions that were added. (In a few cases, a bit in a BD status word had to be shifted.)

- When porting code from the MC68302 CP to the QUICC CPM, the software writer may find that the QUICC has new options to simplify what used to be a more code-intensive process. For specific examples, see the INIT TX AND RX PARAMETERS, GRACEFUL STOP TRANSMIT, and CLOSE BD commands.

1.4 QUICC GLUELESS SYSTEM DESIGN

A fundamental design goal of the QUICC was ease of interface to other system components. An example of this goal is a minimal QUICC design using EPROM and DRAM, shown in Figure 1-2. This system interfaces gluelessly to an EPROM and a DRAM SIMM module. It also offers parity support for the DRAM.

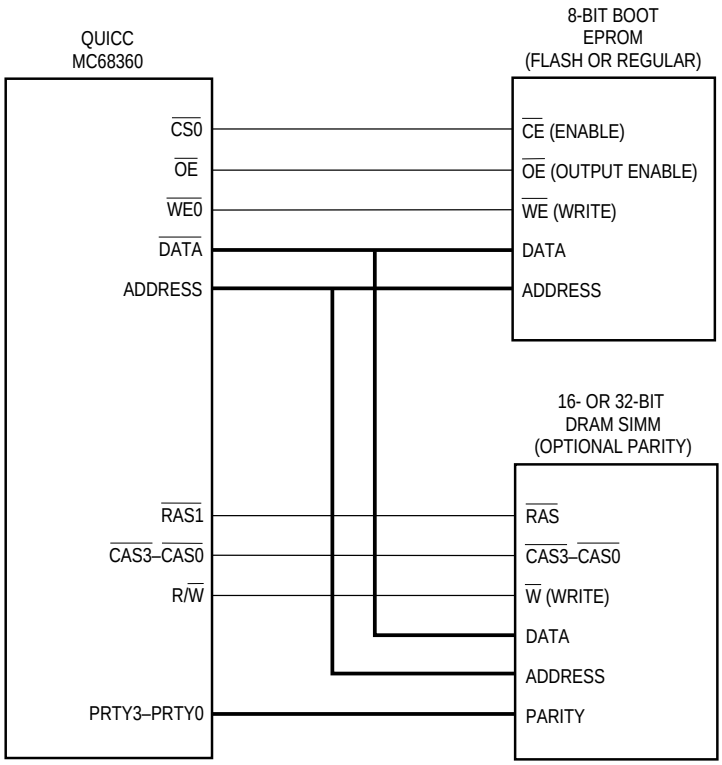


Figure 1-2. Minimum QUICC System Configuration

Figure 1-3 shows a larger system configuration. This system offers one EPROM, one flash EPROM, and supports two DRAM SIMMs. Depending on the capacitance on the system bus, external buffers may be required. From a logic standpoint, however, a glueless system is maintained.

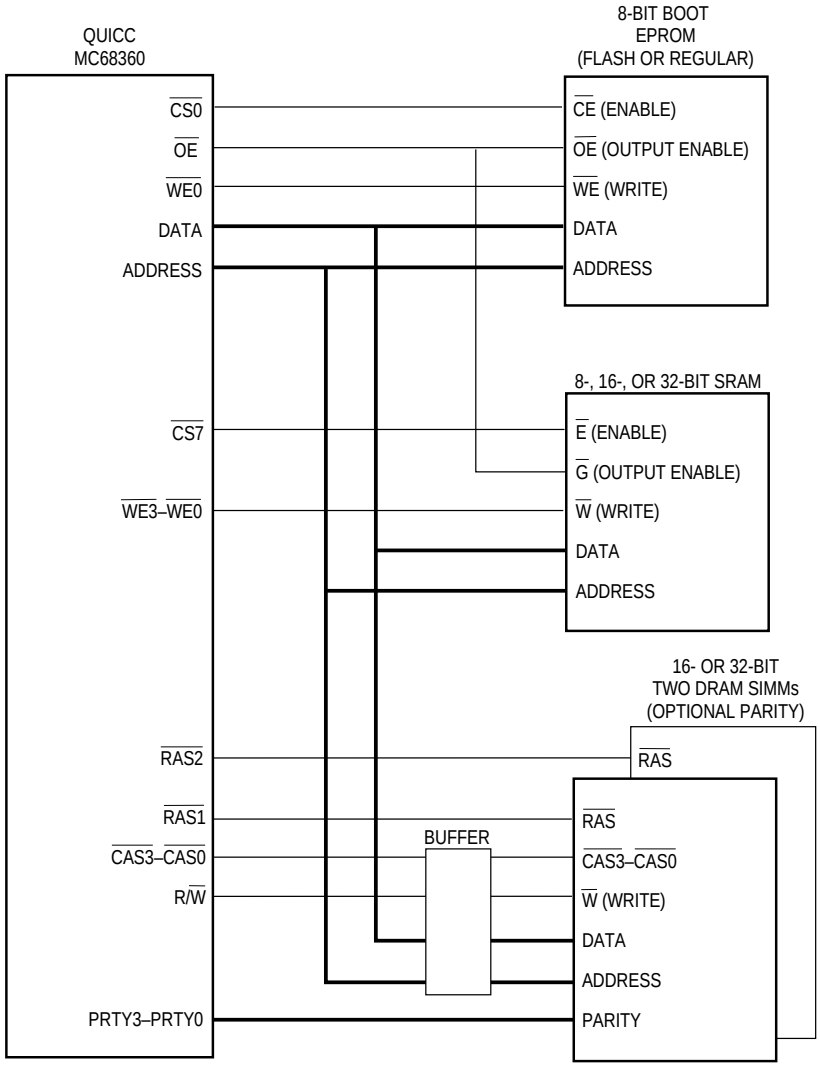


Figure 1-3. Larger QUICC System Configuration

1.5 QUICC SERIAL CONFIGURATIONS

The QUICC offers an extremely flexible set of communications capabilities. Although a full understanding of the possibilities requires reading the appropriate sections, some of the possibilities are shown in the following diagrams. They show possible connections between QUICC devices. In addition, connections are often shown between QUICCs and the MC68302 to show the compatibility between these devices.

For readability, transceivers are usually omitted in the following diagrams. For local on-board communications, however, transceivers are often optional and depend on the protocol used.

Figure 1-4 shows the Ethernet LAN capability of the QUICC. An external SIA transceiver is required to complete the interface to the media. This functionality is implemented in the MC68160 enhanced Ethernet serial transceiver (EEST™). The MC68160 EEST supports

connections to the attachment unit interface (AUI) or twisted-pair Ethernet formats and provides a glueless interface to the QUICC.

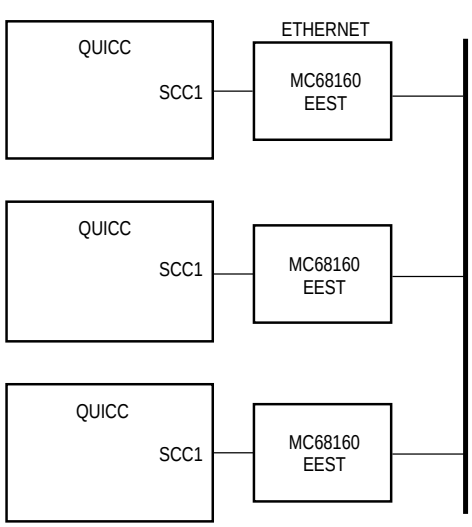
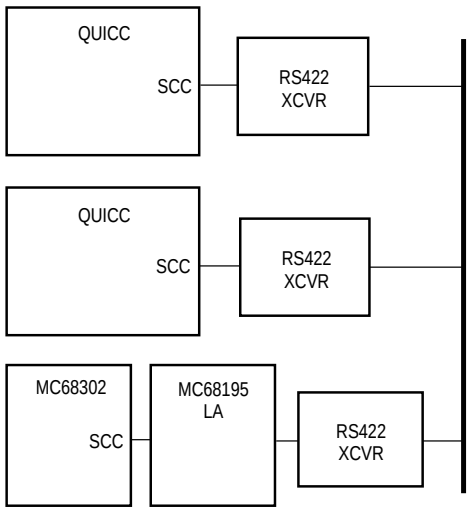


Figure 1-4. Ethernet LAN Capability

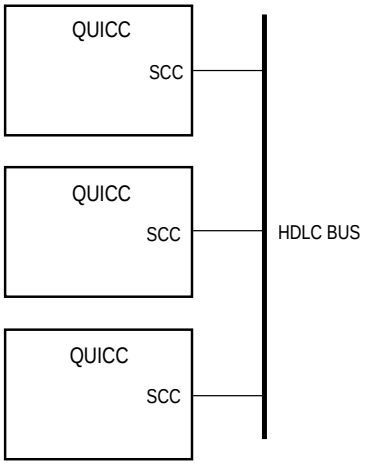
Figure 1-5 shows the AppleTalk LAN capability of the QUICC. Note that the MC68302 requires an extra device, the MC68195 LocalTalk adapter, to interface to AppleTalk.



NOTE: The QUICC implements the AppleTalk LAN protocol without the need for the MC68195.

Figure 1-5. AppleTalk LAN Capability

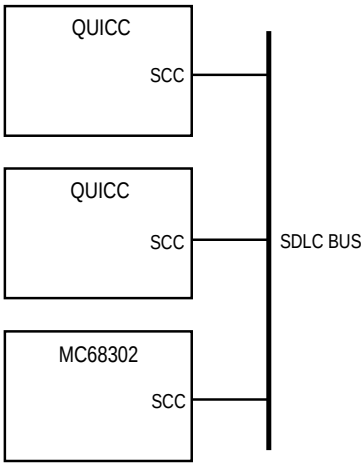
Figure 1-6 shows the implementation of a LAN structure of HDLC called HDLC bus. This protocol is the fastest, easiest way to interface multiple QUICCs in an HDLC-based protocol.



NOTES:
1. HDLC bus—any node can obtain mastership.
2. The QUICC handles collisions without external glue.

Figure 1-6. HDLC Bus LAN

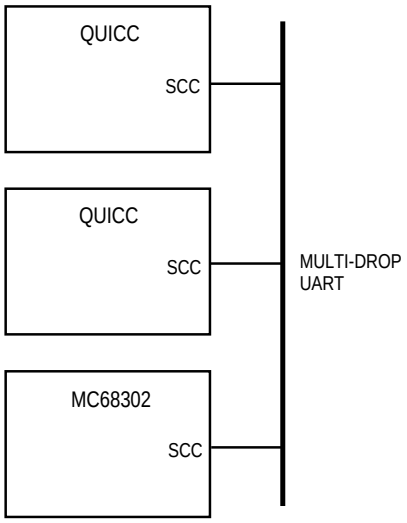
Figure 1-7 shows the original SDLC application, which can be implemented by both QUICCs and MC68302s.



NOTE: No collisions are allowed in this master-slave approach. Also available on the MC68302.

Figure 1-7. FSDLC Bus Implementation

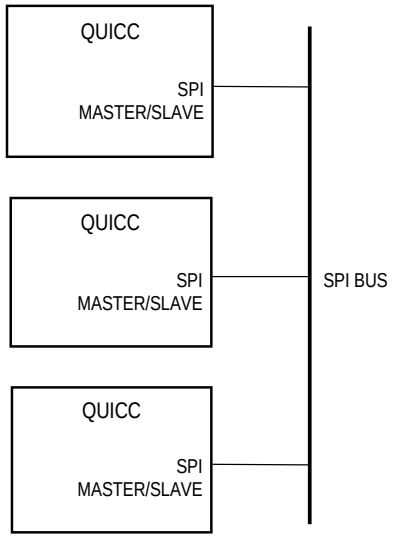
Figure 1-8 shows a UART LAN configuration that is supported by both the QUICC and the MC68302, as well as many other industry UARTs.



NOTES:
1. Simple LAN based on UART mode.
2. Ninth bit is an "address" bit.

Figure 1-8. UART LAN Implementation

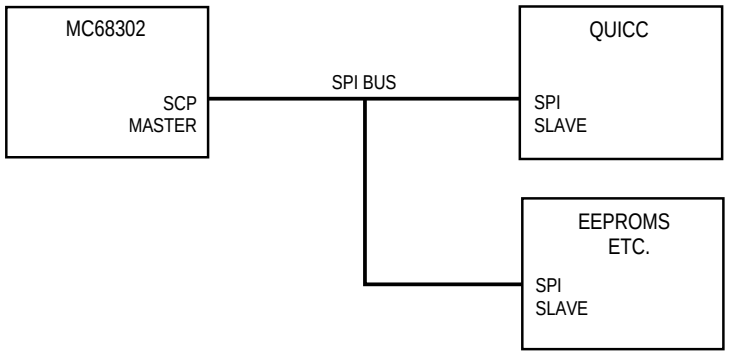
Figure 1-9 shows how the SPIs on the QUICC can be used to connect devices together into a local bus. The SPI exists on many other Motorola devices, such as the MC68HC11 microcontroller, and a number of peripherals such as A/D and D/A converters, LED drivers, LCD drivers, real-time clocks, serial EEPROM, PLL frequency synthesizers, and shift registers.



NOTE: SPI bus configuration—each QUICC can be the master in turn.

Figure 1-9. SPI Local Bus Implementation

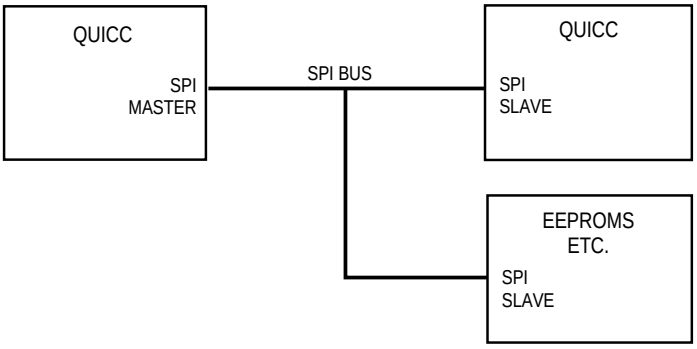
Figure 1-10 shows how the SCP on the MC68302 can be used to interface to the QUICC SPI.



NOTE: The MC68302 SCP can communicate with the QUICC SPI.

Figure 1-10. SPI Implementation Using SCP

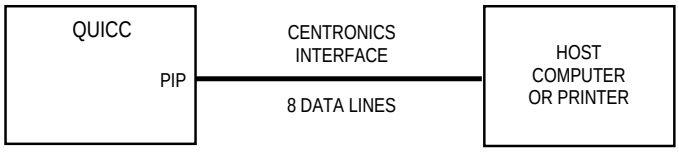
Figure 1-11 shows how the SPI on the QUICC can interface to another QUICC or SPI-based peripherals.



NOTE: Two QUICCs configured for a master-slave SPI connection.

Figure 1-11. SPI Master-Slave Implementation

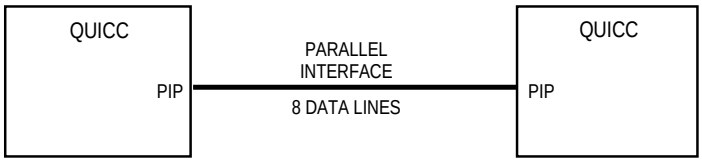
Figure 1-12 shows how the parallel interface port (PIP) can be used to implement the Centronics interface connection. The QUICC may be the peripheral or the host.



NOTE: The QUICC can communicate over a Centronics Interface.

Figure 1-12. Centronics Interface Implementation

Figure 1-13 shows how the PIP can also be used to implement a fast parallel connection between devices.



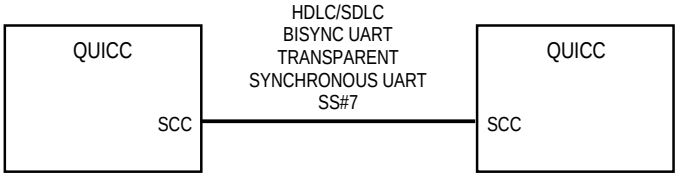
NOTE: Fast parallel connection between QUICCs.

Figure 1-13. Fast Parallel Connection Implementation

Figure 1-14 shows which SCC protocols may be used to connect SCCs on the QUICC and the MC68302.

Figure 1-14. SCC Protocol Implementation

Figure 1-15 shows which SCC protocols may be used to connect SCCs on multiple QUICCs or to other devices supporting such protocols.



NOTE: Point-to-point (WAN) configurations are available on the QUICC.

Figure 1-15. Multiple QUICC Point-to-Point Implementation

Figure 1-16 shows other point-to-point options that are possible with the QUICC and the MC68302.

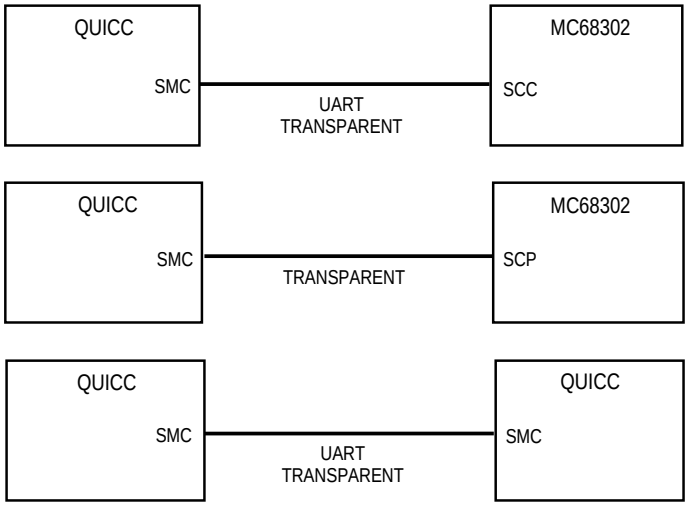
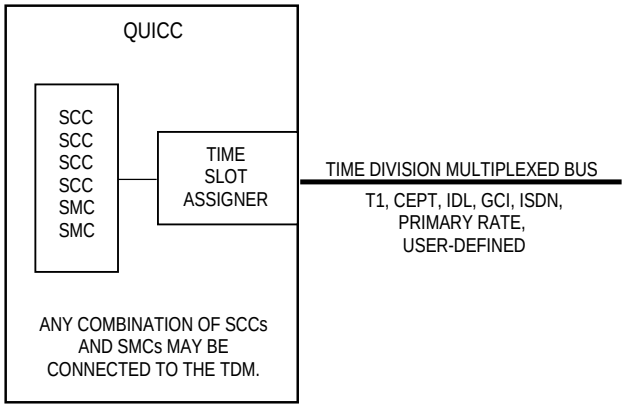


Figure 1-16. Other Point-to-Point Implementations

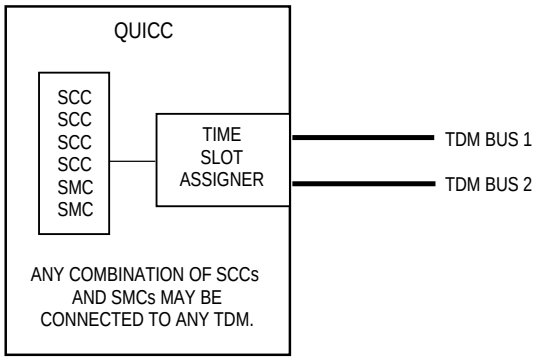
Figure 1-17 shows how up to six of the serial channels can connect to a TDM interface. The QUICC provides a built-in time-slot assigner for access to the TDM time slots. Other channels can work with their own set of pins, allowing possibilities like an Ethernet to T1 bridge, etc.



NOTE: Independent receive and transmit clocking, routing, and syncs are supported.

Figure 1-17. Serial Channel to TDM Bus Implementation

Figure 1-18 shows that the QUICC time-slot assigner can support two TDM buses. Each TDM bus can be of a different format—for example, one TDM can be a T1 line, and one can be a CEPT line. Also this technique could be used to bridge frames from basic rate ISDN to a T1/CEPT line, etc.

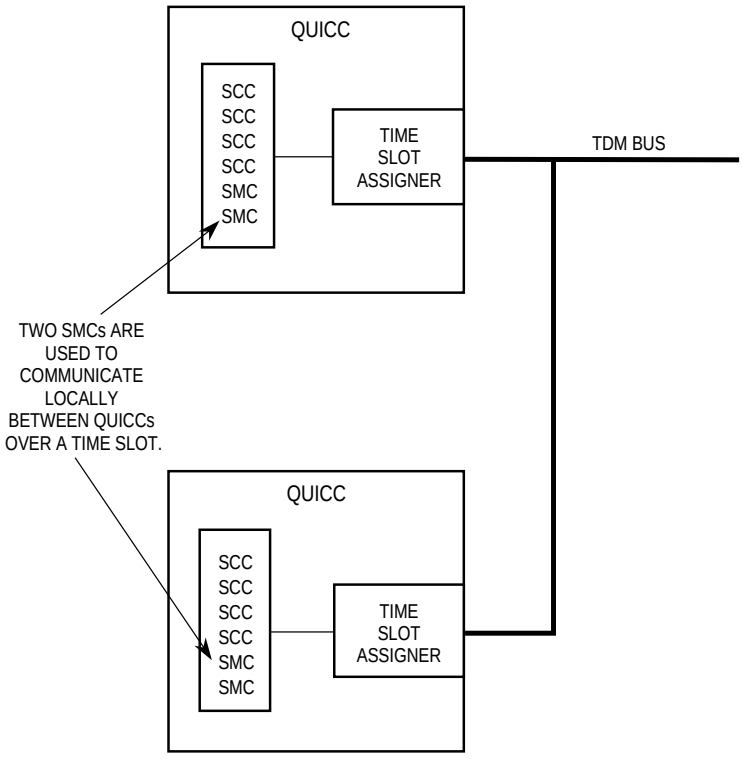


NOTE: Two TDM buses may be simultaneously supported with the time slot assigner.

Figure 1-18. Dual TDM Bus Implementation

1.6 QUICC SERIAL CONFIGURATION EXAMPLES

Figure 1-19 shows a situation where multiple QUICCs can communicate over a TDM line. This can be used, for instance, to implement an 8-channel line card. The SCCs implement the line interfaces, and the SMCs provide the local on-board communication between the QUICCs. The additional SMC on each QUICC can be used as a serial debug port. The SPI can be used to interface to peripherals, such as a serial EEPROM.



NOTE: The eight SCCs and two SMCs support 10 time slots on the TDM bus. The length and position of the time slots are made with time slot assigners.

Figure 1-19. Multiple QUICC TDM Bus Implementation

Figure 1-20 shows a general-purpose application that includes Ethernet, AppleTalk, an HDLC connection to a T1 line, an HDLC connection to frame relay, a UART debug monitor port, a totally transparent data stream port, and an SPI connection to a serial EEPROM.

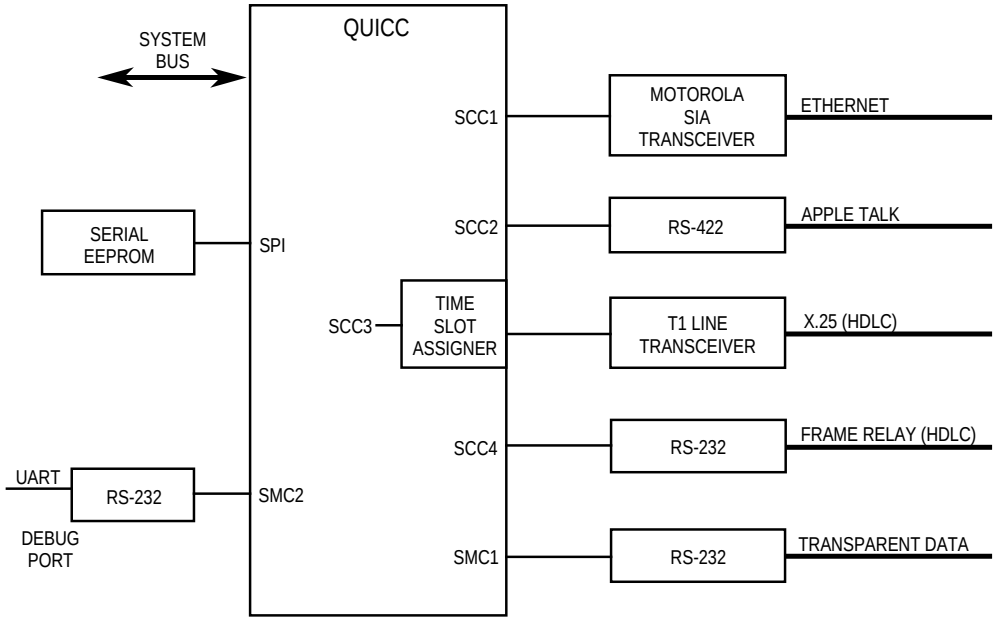


Figure 1-20. General-Purpose Application

1.7 QUICC SYSTEM BUS CONFIGURATIONS

Figure 1-21 shows a master-slave QUICC configuration. This system gives eight SCCs, four SMCs, two SPIs, four IDMAs, etc. Each QUICC uses its own DMA capability, but the CPU32+ is the only processor in the system. More QUICCs can be easily supported on the system bus, if desired.

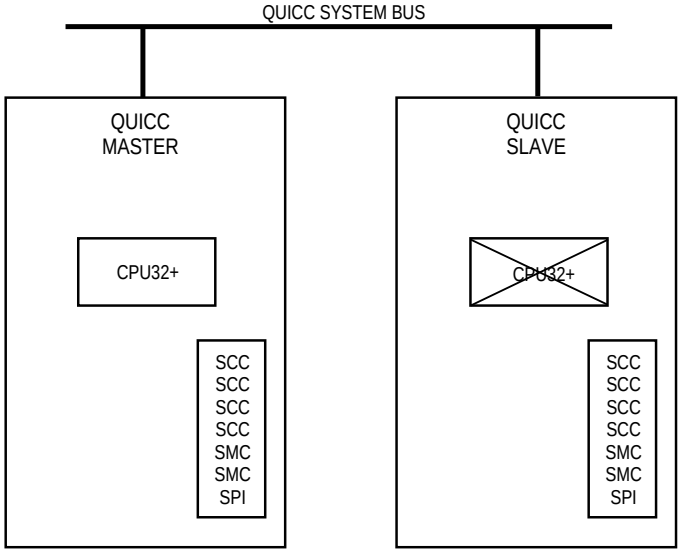


Figure 1-21. Master-Slave QUICC Implementation

The QUICC has special features in slave mode to support the M68040 family. When the QUICC is used in this way, it is said to be in MC68040 companion mode. Figure 1-22 shows how a QUICC in slave mode can interface to a MC68EC040. (The MC68EC040 is a low-cost version of the MC68040 with identical integer performance, but without the memory management unit (MMU) and the floating-point unit (FPU).) The DRAM controller on the QUICC will control the accesses of the MC68EC040 (including the burst modes). This configuration does require external address multiplexers, but the QUICC controls the multiplexers. The QUICC supports the MC68EC040 in other ways, such as interrupt handling and system protection features. When it is in slave mode, the QUICC can also be interfaced to any MC68030-type bus master instead of the MC68EC040.

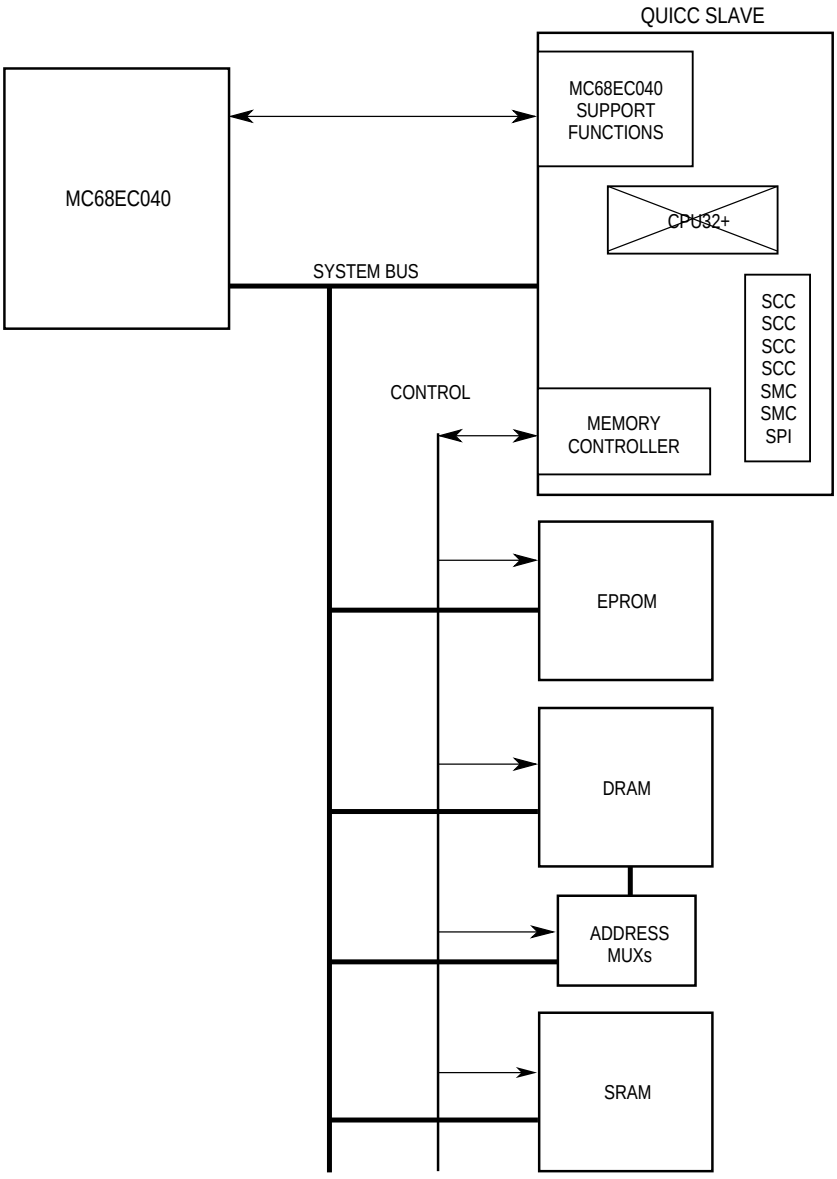


Figure 1-22. MC68040 Companion Mode

SECTION 2
SIGNAL DESCRIPTIONS

This section contains brief descriptions of the QUICC input and output signals in their functional groups as shown in Figure 2-1.

2.1 SYSTEM BUS SIGNAL INDEX

The QUICC system bus signals consist of two groups. The first group, listed in Table 2-1, consists of system bus signals that exist when the QUICC is in the normal mode (CPU32+ enabled). The second group consists of system bus signals that exist when the QUICC is in the slave mode (CPU32+ disabled). They are listed in Table 2-7 and may also be identified in Figure 2-1 as those with an italic font. In Table 2-1, the signal name, mnemonic, and a brief functional description are presented. For more detail on each signal, refer to the paragraphs that discuss each signal.

2.1.1 Address Bus

The address bus consists of the following two groups. Refer to Section 4 Bus Operation for information on the address bus and its relationship to bus operation.

2.1.1.1 ADDRESS BUS (A27–A0). This three-state bidirectional bus (along with A31–A28) provides the address for the current bus cycle, except in the CPU address space. Refer to Section 4 Bus Operation for more information on the CPU address space. A27 is the most significant address signal in this group.

2.1.1.2 ADDRESS BUS (A31–A28). These pins can be programmed as the most significant four address bits or as four byte write enables.

A31–A28—These pins can function as the most significant 4 address bits. A31 is the most significant address signal in this group.

$\overline{\text{WE3}}$–$\overline{\text{WE0}}$—On a write cycle, these active-low signals indicates which byte of the 32-bit data bus contains valid data.

$\overline{\text{WE0}}$—Corresponds to A31 and selects data bits 31–24. Also may be referred to as UUWE.

$\overline{\text{WE1}}$—Corresponds to A30 and selects data bits 23–16. Also may be referred to as UMWE.

$\overline{\text{WE2}}$—Corresponds to A29 and selects data bits 15–8. Also may be referred to as LMWE.

$\overline{\text{WE3}}$—Corresponds to A28 and selects data bits 7–0. Also may be referred to as LLWE.

NOTE

Write enable does not have the capability to follow dynamic bus sizing with external assertion of DSACK. Write enable will always follow the port size that is programed in GMR and the OR. For more information see 6.10 Memory Controller.

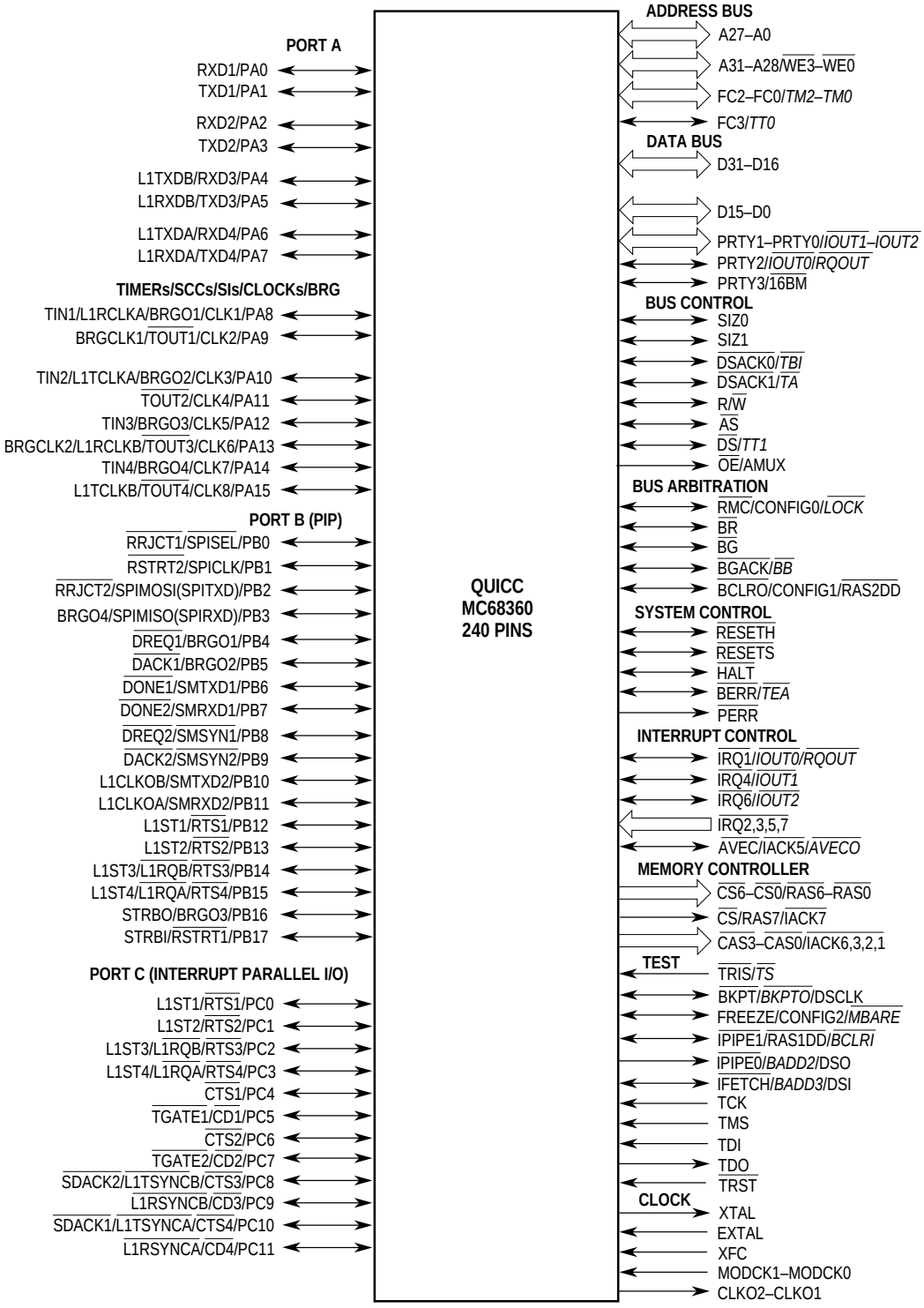


Figure 2-1. QUICC Functional Signal Groups

Table 2-1. System Bus Signal Index (Normal Operation)

Group	Signal Name	Mnemonic	Function
Address	Address Bus	A27–A0	Lower 27 bits of address bus. (I/O)
	Address Bus/Byte Write Enables	A31–A28/ WE0–WE3	Upper four bits of address bus (I/O), or byte write enable signals (O) for accesses to external memory or peripherals.
	Function Codes	FC3–FC0	Identifies the processor state and the address space of the current bus cycle. (I/O)
Data	Data Bus 31–16	D31–D16	Upper 16-bit data bus used to transfer byte or word data. Used in 16-bit bus mode (I/O).
	Data Bus 15–0	D15–D0	Lower 16-bit data bus used to transfer 3-byte or long-word data (I/O). Not used in 16-bit bus mode.
Parity	Parity 2–0	PRTY2–PRTY0	Parity signals for byte writes/reads from/to external memory module (I/O).
	Parity3/$\overline{16BM}$	PRTY3/$\overline{16BM}$	Parity signals for byte writes/reads from/to external memory module or defines 16-bit bus mode. (I/O)
	Parity Error	$\overline{PERR}$	Indicates a parity error during a read cycle. (O)
Memory Controller	Chip Select/Row Address Select 7/ Interrupt Acknowledge 7	$\overline{CS}$/RAS7/$\overline{IACK7}$	Enables peripherals or DRAMs at programmed addresses (O) or interrupt level 7 acknowledge line (O).
	Chip Select 6–0/ Row Address Select 6–0	$\overline{CS6}$–$\overline{CS0}$/ RAS6–RAS0	Enables peripherals or DRAMs at programmed addresses. (O)
	Column Address Select 3–0/Interrupt Acknowledge 1, 2, 3, 6	$\overline{CAS3}$-$\overline{CAS0}$/ IACK6,3,2,1	DRAM column address select or interrupt level acknowledge lines. (O)
Bus Arbitration	Bus Request	$\overline{BR}$	Indicates that an external device requires bus mastership. (I)
	Bus Grant	$\overline{BG}$	Indicates that the current bus cycle is complete and the QUICC has relinquished the bus. (O)
	Bus Grant Acknowledge	$\overline{BGACK}$	Indicates that an external device has assumed bus mastership. (I)
	Read-Modify-Write Cycle/Initial Configuration 0	$\overline{RMC}$/CONFIG0	Identifies the bus cycle as part of an indivisible read-modify-write operation (I/O) or initial QUICC configuration select (I).
	Bus Clear Out/ Initial Configuration 1/Row Address Select 2 Double-Drive	$\overline{BCLR}$/$\overline{CONF}$IG1/ RAS2DD	Indicates that an internal device requires the external bus (Open-Drain O) or initial QUICC configuration select (I) or row address select 2 double-drive output (O).
Bus Control	Data and Size Acknowledge	$\overline{DSACK1}$–$\overline{DSACK0}$	Provides asynchronous data transfer acknowledgement and dynamic bus sizing (open-drain I/O but driven high before three-stated).
	Address Strobe	$\overline{AS}$	Indicates that a valid address is on the address bus. (I/O)
	Data Strobe	$\overline{DS}$	During a read cycle, $\overline{DS}$ indicates that an external device should place valid data on the data bus. During a write cycle, $\overline{DS}$ indicates that valid data is on the data bus. (I/O)
	Size	SIZ1–SIZ0	Indicates the number of bytes remaining to be transferred for this cycle. (I/O)
	Read/Write	R/$\overline{W}$	Indicates the direction of data transfer on the bus. (I/O)
	Output Enable/ Address Multiplex	$\overline{OE}$/AMUX	Active during a read cycle indicates that an external device should place valid data on the data bus (O) or provides a strobe for external address multiplexing in DRAM accesses if internal multiplexing is not used (O).
Interrupt Control	Interrupt Request Level 7–1	$\overline{IRQ7}$–$\overline{IRQ1}$	Provides external interrupt requests to the CPU32+ at priority levels 7–1. (I)
	Autovector/Interrupt Acknowledge 5	$\overline{AVEC}$/$\overline{IACK5}$	Autovector request during an interrupt acknowledge cycle (open-drain I/O) or interrupt level 5 acknowledge line (O).

Table 2-1. System Bus Signal Index (Normal Operation)(Continued)

Group	Signal Name	Mnemonic	Function
System Control	Soft Reset	$\overline{\text{RESETS}}$	Sft system reset. (open-drain I/O)
	Hard Reset	$\overline{\text{RESETH}}$	Hard system reset. (open-drain I/O)
	Halt	$\overline{\text{HALT}}$	Suspends external bus activity. (open-drain I/O)
	Bus Error	$\overline{\text{BERR}}$	Indicates an erroneous bus operation is being attempted. (open-drain I/O)
Clock and Test	System Clock Out 1	CLKO1	Internal system clock output 1. (O)
	System Clock Out 2	CLKO2	Internal system clock output 2—normally 2x CLKO1. (O)
	Crystal Oscillator	EXTAL, XTAL	Connections for an external crystal to the internal oscillator circuit. EXTAL (I), XTAL (O).
	External Filter Capacitor	XFC	Connection pin for an external capacitor to filter the circuit of the PLL (I).
	Clock Mode Select 1–0	MODCK1–MODCK0	Selects the source of the internal system clock. (I) THESE PINS SHOULD NOT BE SET TO 00
	Instruction Fetch/ Development Serial Input	$\overline{\text{IFETCH}}$/DSI	Indicates when the CPU32+ is performing an instruction word prefetch (O) or input to the CPU32+ background debug mode (I).
	Instruction Pipe 0/ Development Serial Output	$\overline{\text{IPIPE0}}$/DSO	Used to track movement of words through the instruction pipeline (O) or output from the CPU32+ background debug mode (O).
	Instruction Pipe 1/ Row Address Select 1 Double-Drive	$\overline{\text{IPIPE1}}$/$\overline{\text{RAS1DD}}$	Used to track movement of words through the instruction pipeline (O), or a row address select 1 “double-drive” output (O).
Clock and Test (Cont'd)	Breakpoint/ Development Serial Clock	$\overline{\text{BKPT}}$/DSCLK	Signals a hardware breakpoint to the QUICC (open-drain I/O), or clock signal for CPU32+ background debug mode (I).
	Freeze/Initial Configuration 2	FREEZE/ CONFIG2	Indicates that the CPU32+ has acknowledged a breakpoint (O), or initial QUICC configuration select (I).
	Three-State	$\overline{\text{TRIS}}$	Used to three-state all pins if QUICC is configured as a master. Sampled during system reset. (I)
	Test Clock	TCK	Provides a clock for Scan test logic. (I)
	Test Mode Select	TMS	Controls test mode operations. (I)
	Test Data In	TDI	Serial test instructions and test data signal. (I)
	Test Data Out	TDO	Serial test instructions and test data signal. (O)
	Test Reset	$\overline{\text{TRST}}$	Provides an asynchronous reset to the test controller. (I)
Power	Clock Synthesizer Power	VCCSYN	Power supply to the PLL of the clock synthesizer.
	Clock Synthesizer Ground	GNDSYN	Ground supply to the PLL of the clock synthesizer.
	Clock Out Power	VCCCLK	Power supply to clock out pins.
	Clock Out Ground	GNDCLK	Ground supply to clock out pins.
	Special Ground 1	GNDS1	Special ground for fast AC timing on certain system bus signals.
	Special Ground 2	GNDS2	Special ground for fast AC timing on certain system bus signals.
	System Power Supply and Return	VCC, GND	Power supply and return to the QUICC.
—	No Connect	NC4–NC1	Four no-connect pins.

NOTE: I denotes input, 0 denotes output, and I/O is input/output.

2.1.2 Function Codes (FC3–FC0)

These three-state bidirectional signals identify the processor state and the address space of the current bus cycle as noted in Table 2-2. The function code pins provide the purpose of each bus cycle to external logic.

Other bus masters besides the QUICC may also output function codes during their bus cycles. On the QUICC, this capability is provided for each potential internal bus master (i.e., the IDMA, SDMA, and DRAM refresh units). Provision is also made for the decoding of function codes that are output from external bus masters (e.g., in the memory controller chip-select generation logic).

In computer design, function code information can be used to protect certain portions of the address map from unauthorized access or to extend the addressable range beyond the address limit. However, in controller applications, function codes are most often used as a debugging aid. Furthermore, in most controller applications, the QUICC stays continuously in the supervisor state.

Refer to Section 4 Bus Operation for more information.

Table 2-2. Address Space Encoding

Function Code Bits				Address Space
3	2	1	0	
0	0	0	0	Reserved (Motorola)
0	0	0	1	User Data Space
0	0	1	0	User Program Space
0	0	1	1	Reserved (User)
0	1	0	0	Reserved (Motorola)
0	1	0	1	Supervisor Data Space
0	1	1	0	Supervisor Program Space
0	1	1	1	Supervisor CPU Space
1	x	x	x	DMA Space

NOTE

FC3-0 may not be set to 0xF

2.1.3 Data Bus

The data bus consists of the following two groups. Refer to Section 4 Bus Operation for information on the data bus and its relationship to bus operation.

2.1.3.1 DATA BUS (D31–D16). These three-state bidirectional signals (along with D15–D0) provide the general-purpose data path between the QUICC and all other devices. Although the data path is a maximum of 32 bits wide, it can be dynamically sized to support 8-, 16-, or 32-bit transfers. D31 is the MSB of the data bus. Byte and word operations occur on D31–D16. Additionally, if the QUICC is configured into 16-bit bus mode, the D31–D16

pins are the only data pins used. Refer to Section 4 Bus Operation for information on the data bus and its relationship to bus operation.

2.1.3.2 DATA BUS (D15–D0). These pins can function as 16 additional data pins used in long-word and 3-byte transfers. They are three-stated and not used if the QUICC is configured into 16-bit bus mode.

2.1.4 Parity

These three-state bidirectional signals provide parity generation/checking for the data path between the QUICC or external masters and other devices. There are four parity lines—one for every eight data bits. The parity lines consists of two groups. Refer to Section 6 System Integration Module (SIM60) for more information on parity generation/checking.

2.1.4.1 PARITY (PRTY0). This pin is the parity value for data bits 31–24.

2.1.4.2 PARITY (PRTY1). This pin is the parity value for data bits 23–16.

2.1.4.3 PARITY (PRTY2). This pin is the parity value for data bits 15–8.

2.1.4.4 PARITY (PRTY3). This pin has two functions. During total system reset, it is the $\overline{16BM}$ pin to determine whether 16-bit data bus mode is to be enabled. After system reset, it functions as the parity line 3.

PRTY3—This pin is the parity value for data bits 0–7.

$\overline{16BM}$—This pin selects the 16-bit data bus mode. To choose a 32-bit data bus during total system reset, this pin can be left floating (it has an internal pullup resistor) or can be driven/pulled high. To choose a 16-bit data bus during total system reset, this pin should be driven/pulled low.

2.1.5 Memory Controller

The following signals are used to control an external memory device.

2.1.5.1 CHIP SELECT/ROW ADDRESS SELECT ($\overline{CS6}$–$\overline{CS0}$/$\overline{RAS6}$–$\overline{RAS0}$). The chip-select output signals enable peripherals or memory arrays at programmed addresses. CS0 is the global chip select for the boot ROM containing the user's reset vector and initialization program. Refer to Section 6 System Integration Module (SIM60) for more information on chip selects.

NOTE

In addition, $\overline{RAS1}$ can be simultaneously output on the $\overline{RAS1DD}$ pin to increase the $\overline{RAS1}$ line drive capability, and $\overline{RAS2}$ can be simultaneously output on the $\overline{RAS2DD}$ pin to increase the $\overline{RAS2}$ line drive capability.

2.1.5.2 CHIP SELECT/ROW ADDRESS SELECT/INTERRUPT ACKNOWLEDGE ($\overline{CS7}$/$\overline{RAS7}$/$\overline{IACK7}$). This pin can be programmed as a $\overline{CS7}$/$\overline{RAS7}$ pin or as the $\overline{IACK7}$ line. See Section 6 System Integration Module (SIM60) for more information on this selection.

$\overline{RAS7}$/$\overline{CS7}$—Row address select 7 or chip select 7 output signal.

$\overline{IACK7}$—The QUICC asserts this pin to indicate a level 7 external interrupt during an interrupt acknowledge cycle. Peripherals can use the $\overline{IACKx}$ strobes instead of monitoring the address bus and function codes to determine that an interrupt acknowledge cycle is in progress and to obtain the current interrupt level. $\overline{IACKx}$ lines need not be used when the vector is generated internally by the QUICC. See Section 4 Bus Operation for more information.

2.1.5.3 COLUMN ADDRESS SELECT/INTERRUPT ACKNOWLEDGE ($\overline{CAS3}$–$\overline{CAS0}$/ $\overline{IACK6}$, $\overline{3}$, $\overline{2}$, $\overline{1}$). These pins can be programmed as four column address selects for DRAMs or as interrupt acknowledge lines.

$\overline{CAS3}$–$\overline{CAS0}$—The DRAM column address select output signal enables the DRAM columns:

$\overline{CAS0}$ selects data bits 31–24.

$\overline{CAS1}$ selects data bits 23–16.

$\overline{CAS2}$ selects data bits 15–8.

$\overline{CAS3}$ selects data bits 7–0.

$\overline{IACK1}$, $\overline{IACK2}$, $\overline{IACK3}$, $\overline{IACK6}$—The QUICC asserts one of these pins to indicate the level of an external interrupt during an interrupt acknowledge cycle. Peripherals can use the $\overline{IACKx}$ strobes instead of monitoring the address bus and function codes to determine that an interrupt acknowledge cycle is in progress and to obtain the current interrupt level. $\overline{IACKx}$ lines need not be used when the vector is generated internally by the QUICC. See Section 4 Bus Operation for more information.

$\overline{IACK1}$ corresponds to $\overline{CAS0}$.

$\overline{IACK2}$ corresponds to $\overline{CAS1}$.

$\overline{IACK3}$ corresponds to $\overline{CAS2}$.

$\overline{IACK6}$ corresponds to $\overline{CAS3}$.

2.1.5.4 ADDRESS MULTIPLEX (AMUX). See 2.1.7.7 Output Enable/Address Multiplex (OE/AMUX) for more information.

2.1.6 Interrupt Request Level ($\overline{IRQ7}$–$\overline{IRQ1}$)

These pins are prioritized interrupt request lines. $\overline{IRQ7}$, the highest priority, is nonmaskable; $\overline{IRQ6}$–$\overline{IRQ1}$ are internally maskable interrupts. Refer to Section 5 CPU32+ for more information on the interrupt request lines.

2.1.7 Bus Control Signals

These signals control the bus transfer operations of the QUICC. Refer to Section 4 Bus Operation for more information on these signals.

2.1.7.1 DATA AND SIZE ACKNOWLEDGE ($\overline{\text{DSACK1}}$–$\overline{\text{DSACK0}}$). These two active-low bidirectional signals allow asynchronous data transfers and dynamic data bus sizing between the QUICC and external devices (see Table 2-3).

Table 2-3. $\overline{\text{DSACKx}}$ Encoding

DSACK1	DSACK0	Result
1 (Negated)	1 (Negated)	Insert wait states in current bus cycle.
1 (Negated)	0 (Asserted)	Complete cycle—data bus port size is 8 bits.
0 (Asserted)	1 (Negated)	Complete cycle—data bus port size is 16 bits.
0 (Asserted)	0 (Asserted)	Complete cycle—data bus port size is 32 bits.

2.1.7.2 AUTOVECTOR/INTERRUPT ACKNOWLEDGE ($\overline{\text{AVEC}}$/$\overline{\text{IACK5}}$). This pin can be programmed to be an autovector input or the interrupt acknowledge 5 line output.

$\overline{\text{AVEC}}$—This signal requests an automatic vector during an interrupt acknowledge cycle. Refer to Section 6 System Integration Module (SIM60) for more information on the autovector function. $\overline{\text{AVEC}}$ need not be used if the QUICC supplies the vector internally.

$\overline{\text{IACK5}}$—The QUICC asserts this pin to indicate the level of an external interrupt during an interrupt acknowledge cycle at level 5. Peripherals can use the $\overline{\text{IACKx}}$ strobes instead of monitoring the address bus and function codes to determine that an interrupt acknowledge cycle is in progress and to obtain the current interrupt level. $\overline{\text{IACKx}}$ lines need not be used when the vector is generated internally by the QUICC.

2.1.7.3 ADDRESS STROBE ($\overline{\text{AS}}$). This bidirectional signal is driven by the bus master to indicate a valid address on the address bus. The function code, size, and read/write signals are also valid when $\overline{\text{AS}}$ is asserted.

2.1.7.4 DATA STROBE ($\overline{\text{DS}}$). During a read cycle, this input/output signal is driven by the bus master to indicate that an external device should place valid data on the data bus. During a write cycle, the data strobe indicates that valid data is on the data bus.

2.1.7.5 TRANSFER SIZE (SIZ1, SIZ0). These bidirectional signals are driven by the bus master to indicate the number of operand bytes remaining to be transferred in the current bus cycle (see Table 2-4).

Table 2-4. SIZx Encoding

SIZ1	SIZ0	Transfer Size
0	1	Byte
1	0	Word
1	1	3 Bytes
0	0	Long Word

2.1.7.6 READ/WRITE (R/$\overline{\text{W}}$). This active-high bidirectional signal is driven by the bus master to indicate the direction of data transfer on the bus. A logic one indicates a read from a slave device; a logic zero indicates a write to a slave device.

2.1.7.7 OUTPUT ENABLE/ADDRESS MULTIPLEX ($\overline{OE}$/AMUX). This pin can be programmed as the output enable ($\overline{OE}$) output or as the address multiplex output.

$\overline{OE}$—During a read cycle, this output signal is driven by the bus master to indicate that an external device should place valid data on the data bus. $\overline{OE}$ may used to save an external inversion of the R/$\overline{W}$ signal.

AMUX—This output signal is driven by the DRAM controller to the external address multiplexer. AMUX need not be used if the DRAM addresses are multiplexed internally by the QUICC.

2.1.7.8 BYTE WRITE ENABLE ($\overline{WE3}$–$\overline{WE0}$). See 2.1.1.2 Address Bus (A31–A28) for the description.

2.1.8 Bus Arbitration Signals

The following signals are the four bus arbitration control signals used to determine the bus master. Refer to Section 4 Bus Operation for more information concerning these signals.

2.1.8.1 BUS REQUEST ($\overline{BR}$). This active-low input signal indicates that an external device needs to become the bus master. This input is typically wire-ORed.

2.1.8.2 BUS GRANT ($\overline{BG}$). Assertion of this active-low output signal indicates that the bus master has relinquished the bus.

2.1.8.3 BUS GRANT ACKNOWLEDGE ($\overline{BGACK}$). Assertion of this active-low input indicates that an external device has become the bus master.

2.1.8.4 READ-MODIFY-WRITE CYCLE/INITIAL CONFIGURATION ($\overline{RMC}$/CONFIG0). This pin can be programmed as the read-modify-write cycle output or as the initial configuration pin 0 input signal during system reset.

$\overline{RMC}$—This output signal identifies the bus cycle as part of an indivisible read-modify-write operation; it remains asserted during all bus cycles of the read-modify-write operation to indicate that bus ownership cannot be transferred.

NOTE

RMC is muxed with a CONFIG0 pin. RMC only functions when the CPU32+ is enabled, and is an output unless an external master ownes the bus, in which case it is an input.

CONFIG0—See 2.1.13 Initial Configuration Pins (CONFIG) for the description.

2.1.8.5 BUS CLEAR OUT/INITIAL CONFIGURATION/ROW ADDRESS SELECT DOUBLE-DRIVE ($\overline{BCLRO}$/CONFIG1/$\overline{RAS2DD}$). This pin can be programmed as the bus clear out output or as the initial configuration pin 1 input signal during system reset or as the $\overline{RAS2DD}$ output double-drive signal.

$\overline{\text{BCLRO}}$—This active-low open-drain output indicates that one of the QUICC internal bus masters is requesting the external bus master to release the bus.

CONFIG1—See 2.1.13 Initial Configuration Pins (CONFIG) for the description.

$\overline{\text{RAS2}}$—See 2.1.5.1 Chip Select/Row Address Select (CS6–CS0/RAS6–RAS0) for the description.

2.1.9 System Control Signals

The QUICC uses these signals to recover from an exception. Refer to Section 4 Bus Operation for more information on these signals.

2.1.9.1 SOFT RESET ($\overline{\text{RESETS}}$). This active-low, open-drain, bidirectional signal is used to initiate reset. An external reset signal (as well as a reset from the SIM60) resets the QUICC as well as all external devices. A reset signal from the CPU32+ (asserted as part of the RESET instruction) resets external devices only—the internal state of the CPU32+ is not affected; other on-chip modules are reset, but the configuration is not altered. When asserted by the QUICC, this signal is guaranteed to be asserted for a minimum of 512 clock cycles. For more information see 4.7 Reset Operation.

2.1.9.2 HARD RESET ($\overline{\text{RESETH}}$). This active-low, open-drain, bidirectional signal is used to initiate reset. An external hard reset signal (as well as an hard reset from the SIM60) resets the QUICC as well as all external devices and the internal state of the CPU32+; other on-chip modules are reset as well as the QUICC configuration. When asserted by the QUICC, this signal is guaranteed to be asserted for a minimum of 512 clock cycles. For more information see 4.7 Reset Operation.

During a hard reset, the address, data, and bus control pins are all three-stated. The $\overline{\text{BG}}$ pin output is the same as that on the $\overline{\text{BR}}$ input. The general-purpose I/O pins are all configured as inputs. The NC4–NC1 pins are undefined outputs. The XTAL, CLKO1, and CLKO2 pins are active outputs, except for CLKO1 which does not oscillate while the on-chip PLL is attaining a lock. The $\overline{\text{RESETS}}$ pin is an output.

2.1.9.3 HALT ($\overline{\text{HALT}}$). This active-low, open-drain, bidirectional signal is asserted to suspend external bus activity, to request a retry when used with $\overline{\text{BERR}}$, or to perform a single-step operation. As an output, $\overline{\text{HALT}}$ indicates a double bus fault by the CPU32+.

2.1.9.4 BUS ERROR ($\overline{\text{BERR}}$). This active-low, open-drain, bidirectional signal indicates that an invalid bus operation is being attempted or, when used with $\overline{\text{HALT}}$, that the bus master should retry the current cycle.

2.1.10 Clock Signals

These signals are used by the QUICC for controlling or generating the system clocks. Refer to Section 6 System Integration Module (SIM60) for more information on these clock signals.

2.1.10.1 SYSTEM CLOCK OUTPUTS (CLKO2–CLKO1). These output signals reflect the general system clock and are used as the bus timing reference by external devices. CLKO1

is the general system clock. CLKO2 is 2× CLKO1 if the on-chip clock synthesizer PLL is used, and is 1× CLKO1 otherwise.

2.1.10.2 CRYSTAL OSCILLATOR (EXTAL, XTAL). These two pins are the connections for an external crystal to the internal oscillator circuit. If an external oscillator is used, it should be connected to EXTAL, with XTAL left open.

2.1.10.3 EXTERNAL FILTER CAPACITOR (XFC). This pin is used to add an external capacitor to the filter circuit of the PLL. The capacitor should be connected between XFC and VCCSYN.

2.1.10.4 CLOCK MODE SELECT (MODCK1–MODCK0). The state of these active-high input signals during reset selects the type of external clock that is used by the PLL in the clock synthesizer to generate the system clocks. Table 2-5 lists the default values of the PLL.

Table 2-5. Default Operation Mode of the PLL

MODCK 1–0	PLL	Prescaled by 128	Multi. Factor (MF + 1)	EXTAL Freq. (examples)	CLKIN to the PLL	Initial Freq. (VCO/2)
00[1]	Disabled	Reserved	Reserved	Reserved	Reserved	Reserved
01	Enabled	No	1	>10 MHz	=EXTAL	=EXTAL
10	Enabled	Yes	401	4.192 MHz	32.75 kHz	13.14 MHz
11	Enabled	No	401	32.768 kHz	32.768 kHz	13.14 MHz

[1]This mode is reserved.

2.1.11 Instrumentation and Emulation Signals

These signals are used for test or software debugging. Refer to Section 5 CPU32+ for more information on these signals.

2.1.11.1 INSTRUCTION FETCH/DEVELOPMENT SERIAL INPUT ($\overline{\text{IFETCH}}$/$\overline{\text{DSI}}$). This active-low output signal indicates when the CPU32+ is performing an instruction word prefetch and when the instruction pipeline has been flushed. Additionally, this signal is the serial input to the CPU32+ in its background debug mode to issue background commands, etc.

2.1.11.2 INSTRUCTION PIPE/DEVELOPMENT SERIAL OUTPUT (IPIPE0/DSO). This active-low output signal is used to track movement of words through the instruction pipeline. Additionally, this signal is the serial output from the CPU32+ in its background debug mode to issue background status, etc.

2.1.11.3 INSTRUCTION PIPE/ROW ADDRESS SELECT DOUBLE-DRIVE (IPIPE1/ RAS1DD). This active-low output signal is used to track movement of words through the instruction pipeline. This signal also functions as a second output of the $\overline{\text{RAS1}}$ signal to increase fanout capability.

2.1.11.4 BREAKPOINT/DEVELOPMENT SERIAL CLOCK ($\overline{\text{BKPT}}$/DSCLK). This active-low input signal is used to signal a hardware breakpoint to the CPU32+. Additionally, this

signal is the serial clock used to transfer commands/status to and from the CPU32+ during background debug mode.

2.1.11.5 FREEZE/INITIAL CONFIGURATION (FREEZE/CONFIG2). This pin can be programmed as the freeze output or as the initial configuration pin 2 input signal during system reset.

FREEZE—Assertion of this active-high output signal indicates that the CPU32+ has acknowledged a breakpoint and has initiated background mode operation.

CONFIG2—See 2.1.13 Initial Configuration Pins (CONFIG) for the description.

2.1.12 Test Signals

The following signals are used with the on-board test logic . See Section 8 Scan Chain Test Access Port for more information on the use of these signals.

2.1.12.1 TRI-STATE SIGNAL ($\overline{\text{TRIS}}$). TThe TRIS pin is enabled as a tristate control pin only when the CPU32+ is enabled, and it is not sampled during reset. When asserted, TRIS immediately tristates the pins.

2.1.12.2 TEST RESET ($\overline{\text{TRST}}$). This input provides asynchronous reset to the test logic.

2.1.12.3 TEST CLOCK (TCK). This input provides a clock for on-board test logic.

2.1.12.4 TEST MODE SELECT (TMS). This input controls test mode operations for on-board test logic.

2.1.12.5 TEST DATA IN (TDI). This input is used for serial test instructions and test data for on-board test logic.

2.1.12.6 TEST DATA OUT (TDO). This output is used for serial test instructions and test data for on-board test logic.

2.1.13 Initial Configuration Pins (CONFIG)

The CONFIG2–CONFIG0 pins select the QUICC initial configuration during reset (see Table 2-6). They decide whether the CPU32+ core will be enabled or disabled, the global chip select port will be 8-, 16-, or 32-bits, and the MBAR address will be $003FF00 or $0033FF04. After reset, these pins may be programmed to their other function. The CONFIG2–CONFIG0 lines have internal pullup resistors so that if they are left floating, the default selection will be 111. See Section 6 System Integration Module (SIM60) for more information.

Table 2-6. Initial Configuration

Configuration Pins			
CONFIG2/ FREEZE	**CONFIG1/ $\overline{\text{BCLRO}}$**	**CONFIG0/ $\overline{\text{RMC}}$**	**Result**
0	0	0	Slave mode; global CS 8-bit size; MBAR at $003FF00.
0	0	1	Slave mode; global CS 32-bit size; MBAR at $003FF00; not MC68040 companion mode; $\overline{\text{BR}}$ output, $\overline{\text{BG}}$ input.
0	1	0	Slave mode; global CS 16-bit size; MBAR at $003FF00.
0	1	1	MC68040 companion mode; global CS 32-bit size; MBAR at $003FF00; $\overline{\text{BR}}$ input, $\overline{\text{BG}}$ output.
1	0	0	CPU enabled; global CS 32-bit size; MBAR at $003FF00.
1	0	1	CPU enabled; global CS 16-bit size; MBAR at $003FF00.
1	1	0	Slave mode; global CS disabled; MBAR at $003FF04.
1	1	1	CPU enabled; global CS 8-bit size; MBAR at $003FF00. (Default)

NOTE

All CONFIG pins do have an internal pull-up resistor during reset. If a configuration other than the default (CONFIG2-1 = 111) is desired, these pins should be driven by an active open collector device during the assertion of RESETH.

2.1.14 Power Signals

The following signals are used for power and ground to the QUICC.

2.1.14.1 VCCSYN AND GNDSYN. These pins provide power and ground to the clock synthesizer. They should be bypassed to each other with a 0.1-µF capacitor. See the system clock generation description in Section 6 System Integration Module (SIM60) for more details.

2.1.14.2 VCCCLK AND GNDCLK. These pins provide power and ground to the clock output pins (CLKO1 and CLKO2). They should be bypassed to each other with a 0.1-µF capacitor. See the system clock generation description in Section 6 System Integration Module (SIM60) for more detail.

2.1.14.3 GNDS1 AND GNDS2. These two pins are special ground pins that, if used properly, allow more aggressive timing to be provided on certain system bus pins. These pins include $\overline{\text{AS}}$, $\overline{\text{CASx}}$, and $\overline{\text{IPIPE}}$. Section 10 Electrical Characteristics already shows the aggressive timing; the user does not need to modify any values in the section. GNDS1 and GNDS2 should be connected to a quiet ground source or to a low-noise ground plane.

2.1.14.4 VCC AND GND. These pins are the rest of the power and ground connections for the QUICC.

2.1.14.5 NC4–NC1. These four pins should not be connected on the QUICC package. They are reserved for future enhancements.

2.2 SYSTEM BUS SIGNAL INDEX IN SLAVE MODE

The CONFIG2–CONFIG0 pins are used to cause the QUICC to enter the slave mode. The signal name, mnemonic, and a brief functional description are presented in Table 2-7. The rest of the QUICC pins maintain their functionality in slave mode. See Section 4 Bus Operation for details.

Additionally, the QUICC provides special support for the MC68EC040 bus (or other MC68040 family members) during slave mode. The MC68EC040 signals are marked in boldface in the table. For more information on MC68EC040 bus operation, see M68040UM/AD, *M68040 User's Manual*. The QUICC MC68EC040 support is described in Section 4 Bus Operation and Section 6 System Integration Module (SIM60).

Table 2-7. System Bus Signal Index (Slave Mode)

Master Mode Mnemonic	Slave Mode Signal Name	Slave Mode Mnemonic	Slave Mode Function
FC2–FC0	Function Codes/ **Transfer Modifier**	FC2–FC0/ **TM2–TM0**	Identifies the processor state and the address space of the current bus cycle (I/O), **or indicates the MC68EC040 supplement information about the access (I)**.
FC3	Function Code/ **Transfer Type**	FC3/**TT0**	Identifies the DMA address space of the current bus cycle (I/O), **or indicates the MC68EC040 general transfer type: normal, MOVE16, alternate logical function code, and acknowledge (I).**
$\overline{DS}$	Data Strobe/ **Transfer Type**	$\overline{DS}$/**TT1**	Data strobe (I/O), **or indicates the MC68EC040 general transfer type: normal, MOVE16, alternate logical function code, and acknowledge (I).**
$\overline{DSACK1}$	Data and Size Acknowledge/ **Transfer Acknowledge**	$\overline{DSACK1}$/**$\overline{TA}$**	Provides asynchronous data transfers and dynamic bus sizing; **for the MC68EC040, asserted to acknowledge bus transfer.** (Both are open-drain I/O but driven high before three-stated.)
$\overline{DSACK0}$	Data and Size Acknowledge/ **Transfer Burst Inhibit**	$\overline{DSACK0}$/ **TBI**	Provides asynchronous data transfers and dynamic bus sizing; **for the MC68EC040, indicates that a slave cannot handle a line burst access**. (Both are open-drain I/O but driven high before three-stated.)
$\overline{BERR}$	Bus Error/ **Transfer Error Acknowledge**	$\overline{BERR}$/ **TEA**	$\overline{BERR}$ indicates an erroneous bus operation is being attempted by the QUICC (open-drain I/O); **TEA indicates the same for the MC68EC040 (open-drain I/O)**
$\overline{TRIS}$	Transfer Start	TS	Indicates the beginning of an MC68040 bus transfer. (I)
$\overline{IPIPE0}$/$\overline{IFETCH}$	Burst Address	BADD3–BADD2	Address lines 2,3 generated by the QUICC on behalf of the MC68EC040, for MC68EC040 burst memory cycles. (O)
$\overline{BR}$	Bus Request	$\overline{BR}$ **BR**	Asserted by the QUICC to request bus mastership (O.D. O), **or bus request input from the MC68040. (I)**
$\overline{BG}$	Bus Grant	$\overline{BG}$ **BG**	Asserted by external logic to grant bus mastership to the QUICC (I), **or bus grant output to the MC68040. (O)**
$\overline{BGACK}$	Bus Grant Acknowledge **Bus Busy**	$\overline{BGACK}$ **BB**	Indicates that an external device or the QUICC has assumed bus mastership. (Open-drain I/O but driven high before three-stated).
$\overline{RMC}$/CONFIG0	040 Lock Cycle/ Configuration 0	**$\overline{LOCK}$**/ CONFIG0	**An MC68040 $\overline{LOCK}$ signal input to prevent the QUICC from obtaining the system bus during locked cycles (I)**, and the initial QUICC configuration select (I).
$\overline{BKPT}$	Breakpoint Out	$\overline{BKPTO}$	Signals a hardware breakpoint to the external CPU. (O)
FREEZE/ CONFIG2	Freeze/Initial Configuration Pin 2	$\overline{MBARE}$/ CONFIG2	Provides an MBAR access enable (I), or the initial QUICC configuration select. (I)
$\overline{IRQ1}$,$\overline{4}$,$\overline{6}$	Interrupt Request/ Interrupt Outputs	$\overline{IRQ6}$,$\overline{4}$,$\overline{1}$/ $\overline{IOUT2}$–$\overline{IOUT0}$/ $\overline{IRQOUT}$	Provides an interrupt request to the QUICC interrupt controller (I), or interrupt output signals (O) (either $\overline{RQOUT}$ as a single request or $\overline{IOUT2}$–$\overline{IOUT0}$ encoded).

Table 2-7. System Bus Signal Index (Slave Mode) (Continued)

Master Mode Mnemonic	Slave Mode Signal Name	Slave Mode Mnemonic	Slave Mode Function
PRTY0	Parity 0/Interrupt Output 2	PRTY0/$\overline{\text{IOUT2}}$	Parity signals for D31–D24 writes/reads from/to external memory bank (I/O), or interrupt output 2 signal (O).
PRTY1	Parity 1/Interrupt Output 1	PRTY1/$\overline{\text{IOUT1}}$	Parity signals for D23–D16 writes/reads from/to external memory bank (I/O) or interrupt output 1 signal. (O)
PRTY2	Parity 2/ Interrupt Output 0/ Request Output	PRTY2/$\overline{\text{IOUT0}}$/ $\overline{\text{RQOUT}}$	Parity signals for D15–D8 writes/reads from/to external memory bank (I/O), or interrupt output 0 signal (O), or $\overline{\text{RQOUT}}$ as a single interrupt request output (O).
$\overline{\text{AVEC}}$/$\overline{\text{IACK5}}$	Autovector Output	$\overline{\text{AVECO}}$	Signal output to the external processor to generate an internal vector number during an interrupt acknowledge cycle. (three-stated O)
$\overline{\text{IPIPE1}}$/ $\overline{\text{RAS1DD}}$	Bus Clear Input/ Row Address Select 1 Double-Drive	$\overline{\text{BCLRI}}$/ $\overline{\text{RAS1DD}}$	Signals that an external device requests the QUICC to release the external bus (I), or row address select 1 double-drive (O).

2.3 ON-CHIP PERIPHERALS SIGNAL INDEX

The input and output system signals for the QUICC peripherals are listed in Table 2-8. The signal name, mnemonic, and a brief functional description are presented. For more detail on each signal, refer to the specific module section. The peripherals pins are divided into three ports: A, B, and C.

Port A has 16 pins, port B has 18 pins, and port C has 12 pins. All the following signals are multiplexed with either port A, B, or C. All pins may be inputs or outputs; in addition, some pins may be configured to be open-drain. See 7.14 Parallel I/O Ports for further details.

Table 2-8. Peripherals Signal Index

Group	Signal Name	Mnemonic	Function
SCC	Receive Data	RXD4–RXD1	Serial receive data input to the SCCs. (I)
	Transmit Data	TXD4–TXD1	Serial transmit data output from the SCCs. (O)
	Request to Send	$\overline{\text{RTS4}}$–$\overline{\text{RTS1}}$	Request to send outputs indicate that the SCC is ready to transmit data. (O)
	Clear to Send	$\overline{\text{CTS4}}$–$\overline{\text{CTS1}}$	Clear to send inputs indicate to the SCC that data transmission may begin. (I)
	Carrier Detect	$\overline{\text{CD4}}$–$\overline{\text{CD1}}$	Carrier detect inputs indicate that the SCC should begin reception of data. (I)
	Receive Start	$\overline{\text{RSTRT1}}$	This output from SCC1 identifies the start of a receive frame. Can be used by an Ethernet CAM to perform address matching. (O)
	Receive Reject	RRJCT1	This input to SCC1 allows a CAM to reject the current Ethernet frame after it determines the frame address did not match. (I)
	Clocks	CLK8–CLK1	Input clocks to the SCCs, SMCs, SI, and the baud rate generators. (I)
IDMA	DMA Request	$\overline{\text{DREQ2}}$–$\overline{\text{DREQ1}}$	A request (input) to an IDMA channel to start an IDMA transfer. (I)
	DMA Acknowledge	$\overline{\text{DACK2}}$–$\overline{\text{DACK1}}$	An acknowledgement (output) by the IDMA that an IDMA transfer is in progress. (O)
	DMA Done	$\overline{\text{DONE2}}$–$\overline{\text{DONE1}}$	A bidirectional signal that indicates the last IDMA transfer in a block of data. (I/O)
TIMER	Timer Gate	$\overline{\text{TGATE2}}$–$\overline{\text{TGATE1}}$	An input to a timer that enables/disables the counting function. (I)
	Timer Input	TIN4–TIN1	Time reference input to the timer that allows it to function as a counter. (I)

Table 2-8. Peripherals Signal Index (Continued)

Group	Signal Name	Mnemonic	Function
	Timer Output	$\overline{\text{TOUT4}}$–$\overline{\text{TOUT1}}$	Output waveform (pulse or toggle) from the timer as a result of a reference value being reached. (O)
SPI	SPI Master-In Slave-Out	SPIMISO	Serial data input to the SPI master (I); serial data output from an SPI slave (O).
	SPI Master-Out Slave-In	SPIMOSI	Serial data output from the SPI master (O).; serial data input to an SPI slave (I).
	SPI Clock	SPICLK	Output clock from the SPI master (O); input clock to the SPI slave (I).
	SPI Select	$\overline{\text{SPISEL}}$	SPI slave select input. (I)
SMC	SMC Receive Data	SMRXD2–SMRXD1	Serial data input to the SMCs. (I)
	SMC Transmit Data	SMTXD2–SMTXD1	Serial data output from the SMCs. (O)
	SMC Sync	$\overline{\text{SMSYN2}}$–$\overline{\text{SMSYN1}}$	SMC synchronization signal. (I)
SI	SI Receive Data	L1RXDA, L1RXDB	Serial input to the time division multiplexed (TDM) channel A or channel B.
	SI Transmit Data	L1TXDA, L1TXDB	Serial output from the TDM channel A or channel B.
	SI Receive Clock	L1RCLKA, L1RCLKB	Input receive clock to TDM channel A or channel B.
	SI Transmit Clock	L1TCLKA, L1TCLKB	Input transmit clock to TDM channel A or channel B.
	SI Transmit Sync Signals	L1TSYNCA, L1TSYNCB	Input transmit data sync signal to the TDM channel A or channel B.
	SI Receive Sync Signals	L1RSYNCA, L1RSYNCB	Input receive data sync signal to TDM channel A or channel B.
	IDL Interface Request	L1RQA, L1RQB	IDL interface request to transmit on the D channel. Output from the SI.
	SI Output Clock	L1CLKOA, L1CLKOB	Output serial data rate clock. Can output a data rate clock when the input clock is 2x the data rate.
	SI Data Strobes	L1ST4– L1ST1	Serial data strobe outputs can be used to gate clocks to external devices that do not have a built-in time slot assigner (TSA).
BRG	Baud Rate Generator Out 4–1	BRGO4–BRGO1	Baud rate generator output clock allows baud rate generator to be used externally.
	BRG Input Clock	CLK2, CLK6	Baud rate generator input clock from which BRG will derive the baud rates.
PIP	Port B 15–0	PB15–PB0	PIP Data I/O Pins
	Strobe Out	STRBO	This input causes the PIP output data to be placed on the PIP data pins.
	Strobe In	STRBI	This input causes data on the PIP data pins to be latched by the PIP as input data.
SDMA	SDMA Acknowledge 2–1	$\overline{\text{SDACK2}}$–$\overline{\text{SDACK1}}$	SDMA output signals used in RISC receiver to mark fields in the Ethernet receive frame.

SECTION 3
QUICC MEMORY MAP

The following tables present a programmer's model (register map) of all registers in the QUICC. For more information about a particular register, refer to the description for the module or sub-module indicated in the right column. The address column indicates the offset of the register from the address stored in the module base address register (MBAR). This register in the SIM block controls the location of all internal memory/registers as well as their supervisor/user access space (see Section 6 System Integration Module (SIM60)). Bold letters mark registers that are restricted to supervisor access. Other registers are programmable to exist in either supervisor or user space. Registers that are reset only by hard reset are marked with an H in the reset value column. All of the registers are memory-mapped.

All internal memory and registers occupy a single 8-Kbyte memory block that is relocatable along 8-Kbyte boundaries. The location is fixed by writing the desired base address of the 8-Kbyte memory block to the MBAR using the MOVES instruction. The MBAR is the only exception since it resides at a fixed location in $03FF00.

The 8-Kbyte block is divided into two 4-Kbyte sections. The RAM occupies the first section; the internal registers occupy the second section. The location of the QUICC registers is shown in Figure 3-1.

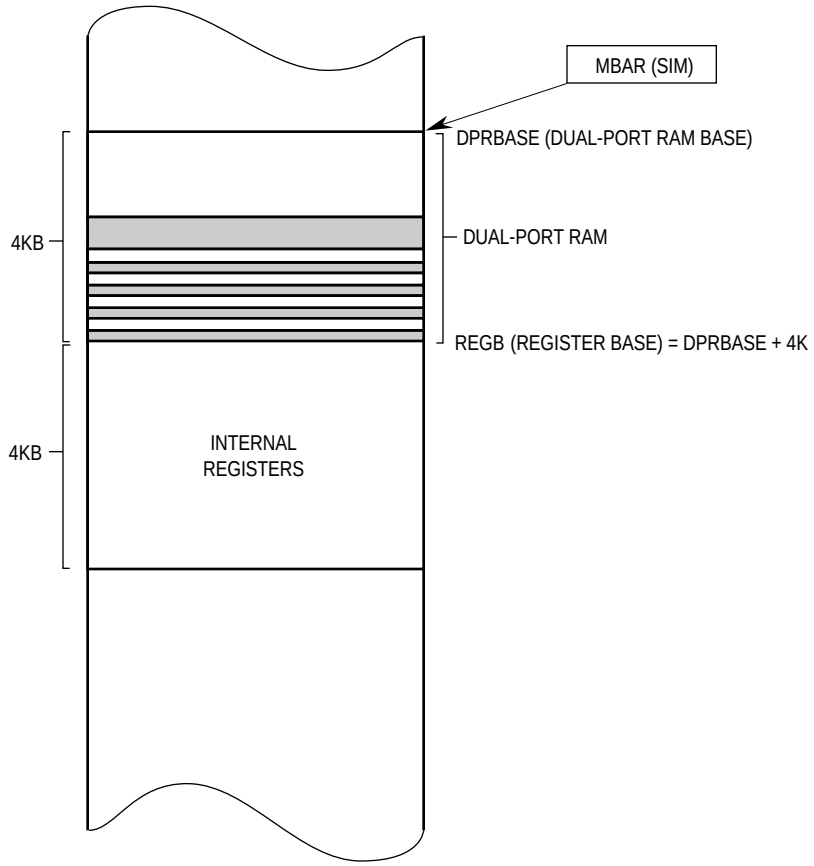


Figure 3-1. QUICC Memory Map

3.1 DUAL-PORT RAM MEMORY MAP

The internal 2816-byte (2560-byte on REV A and B mask) dual-port RAM is partitioned to 1792 bytes (1536 bytes on REV A and B mask) of system RAM, 256-byte microcode scratch area, and 768 bytes of parameter RAM (see Table 3-1). Its base address, called dual-port RAM base (DPRBASE), is the address pointed to by the MBAR.

NOTE

Rev A mask is C63T, Rev B mask are C69T, and F35G

The system RAM may be used for microcode program area, data area, and buffer descriptors (BDs). It may be partitioned in several ways, allowing programmable partition sizes to fit the system requirements. This is described in Section 7 Communication Processor Module (CPM).

The parameter RAM contains the protocol-specific parameters. For detailed information about the use of the buffer descriptors and protocol parameters in a specific protocol, see Section 7 Communication Processor Module (CPM).

Table 3-1. Dual-Port RAM Map

Address	Size	Block	Description
DPRBASE + 0 DPRBASE + 3FF	1024 Bytes	Dual-Port RAM	User Data / BDs / Microcode Program
DPRBASE + 400 DPRBASE + 5FF	512 Bytes	Dual-Port RAM	User Data / BDs
DPRBASE + 600 DPRBASE + 6FF	256 Bytes	Dual-Port RAM	User Data / BDs / Microcode Scratch
DPRBASE + 700 DPRBASE + BFF	256 Bytes	Dual-Port RAM	User Data / BDs
DPRBASE + C00 DPRBASE + CBF	192 Bytes	Dual-Port RAM	Parameter RAM Page 1
DPRBASE + CC0 DPRBASE + CFF		Reserved	Reserved
DPRBASE + D00 DPRBASE + DBF	192 Bytes	Dual-Port RAM	Parameter RAM Page 2
DPRBASE + DC0 DPRBASE + DFF		Reserved	Reserved
DPRBASE + E00 DPRBASE + EBF	192 Bytes	Dual-Port RAM	Parameter RAM Page 3
DPRBASE + EC0 DPRBASE + EFF		Reserved	Reserved
DPRBASE + F00 DPRBASE + FBF	192 Bytes	Dual-Port RAM	Parameter RAM Page 4
DPRBASE + FC0 DPRBASE + FFF		Reserved	Reserved

3.2 CPM SUB-MODULE BASE ADDRESSES

Within the four parameter RAM pages are the base addresses for the CPM sub-modules such as the SCCs, SMCs, etc. The base addresses for the sub-modules are shown in Table 3-2. See the particular sub-module description within Section 7 Communication Processor Module (CPM) for further information.

Table 3-2. CPM Sub-Module Base Addresses

Parameter RAM Page	Sub-Module	Base Address
1	SCC1 Base	DPRBASE + $C00
1	Misc Base	DPRBASE + $CB0
2	SCC2 Base	DPRBASE + $D00
2	SPI Base	DPRBASE + $D80
2	Timer Base	DPRBASE + $DB0
3	SCC3 Base	DPRBASE + $E00
3	IDMA1 Base	DPRBASE + $E70

Table 3-2. CPM Sub-Module Base Addresses

3	SMC1 Base	DPRBASE + $E80
4	SCC4 Base	DPRBASE + $F00
4	IDMA2 Base	DPRBASE + $F70
4	SMC2 Base	DPRBASE + $F80

3.3 INTERNAL REGISTERS MEMORY MAP

In addition to the internal dual-port RAM, there are a number of internal registers to support the functions of the various CPU32+ core peripherals. The internal registers (see Table 3-3 and Table 3-4) are memory-mapped registers offset from the register base (REGBASE) pointer. REGBASE (abbreviated REGB) = DPRBASE + 4K. All registers are located on the internal IMB.

NOTES

All registers that are underlined in the following tables are special registers called event registers. In these registers, bits are set by the QUICC and cleared by the user. To clear a bit, the user must write a one to that bit. For example, to clear bit 2 in SCCE1, the MOVE.B #$04,SCCE1 instruction may be used. Do NOT use read-modify-write instructions (such as BSET, BCLR, AND, OR, etc.) with these registers, or ALL bits in that register will inadvertently be cleared. See the individual register descriptions for more information.

All undefined and reserved bits within registers and parameter RAM values written by the user should be written with zero to allow for future enhancements to the device.

Bold letters mark registers that are restricted to supervisor access.

3.3.1 SIM Registers Memory Map

Table 3-3 lists the SIM registers memory map.

Table 3-3. QUICC SIM Registers Memory Map

Address	Name	Width	Description	Reset Value		Block
REGB + 0000	MCR	32	Module Configuration Register	0000 7cff	H	SIM
REGB + 0004		32	Reserved			
REGB + 0008	AVR	8	Autovector Register	00	H	
REGB + 0009	RSR	8	Reset Status Register		H/S	
REGB + 000a		16	Reserved			
REGB + 000c	CLKOCR	8	CLKO Control Register	f(MODCK1)	H	
REGB + 000d			Reserved			
REGB + 0010	PLLCR	16	PLL Control Register	f(MODCK1–0)	H	
REGB + 0012		16	Reserved			
REGB + 0014	CDVCR	16	Clock Divider Control Register	0000	H	
REGB + 0016	PEPAR	16	Port E Pin Assignment Register	0000	H	
REGB + 0018 to REGB + 0021			Reserved			
REGB + 0022	SYPCR	8	System Protection Control	f(MODCK1–0)	H	
REGB + 0023	SWIV	8	Software Interrupt Vector	0F	H	
REGB + 0024		16	Reserved			
REGB + 0026	PICR	16	Periodic Interrupt Control Register	000F	H	
REGB + 0028		16	Reserved			
REGB + 002a	PITR	16	Periodic Interrupt Timing Register	0000/0300	H	
REGB + 002c		24	Reserved			
REGB + 002f	SWSR	8	Software Service Register	00	H	
REGB + 0030	BKAR	32	Breakpoint Address Register	XXXX	—	
REGB + 0034	BKCR	32	Breakpoint Control Register	0000 0000	H	
REGB + 0038 to REGB + 003f			Reserved			
REGB + 0040	GMR	32	Global Memory Register	0000 1200	H	MEMC
REGB + 0044	MSTAT	16	Memory Controller Status Register	0000	H	
REGB + 0046 to REGB + 004f			Reserved			
REGB + 0050	BR0	32	Base Register 0	0000 0051	H	
REGB + 0054	OR0	32	Option Register 0	F000 0000	H	
REGB + 0058 to REGB + 005f			Reserved			
REGB + 0060	BR1	32	Base Register 1	0000 0050	H	
REGB + 0064	OR1	32	Option Register 1	F000 000x	H	
REGB + 0068 to REGB +006f			Reserved			
REGB + 0070	BR2	32	Base Register 2	0000 0050	H	
REGB + 0074	OR2	32	Option Register 2	F000 000x	H	

Table 3-3. QUICC SIM Registers Memory Map

REGB + 0078 to REGB + 007f			Reserved			
REGB + 0080	BR3	32	Base Register 3	0000 0050	H	
REGB + 0084	OR3	32	Option Register 3	F000 000x	H	
REGB + 0088 to REGB + 008f			Reserved			
REGB + 0090	BR4	32	Base Address Register 4	0000 0050	H	
REGB + 0094	OR4	32	Option Register 4	F000 000x	H	
REGB + 0098 to REGB + 009f			Reserved			
REGB + 00a0	BR5	32	Base Address Register 5	0000 0050	H	
REGB + 00a4	OR5	32	Option Register 5	F000 000x	H	
REGB + 00a8 to REGB + 00af			Reserved			
REGB + 00b0	BR6	32	Base Address Register 6	0000 0050	H	
REGB + 00b4	OR6	32	Option Register 6	F000 000x	H	
REGB + 00b8 to REGB + 00bf			Reserved			
REGB + 00c0	BR7	32	Base Address Register 7	0000 0050	H	
REGB + 00c4	OR7	32	Option Register 7	F000 000x	H	
REGB + 00c8 to REGB + 00ef			Reserved			
REGB + 00f0 to REGB + 00ff			Reserved			

3.3.2 CPM Registers Memory Map

Table 3-4 lists the CPM registers memory map.

Table 3-4. QUICC CPM Registers Memory Map

Address	Name	Width	Description	Reset Value		Block
REGB + 400 to REGB + 4ff			Reserved			
REGB + 500	ICCR	16	Channel Configuration Register	0000	H	IDMA1
REGB + 502		16	Reserved			
REGB + 504	CMR1	16	IDMA1 Mode Register	0000		
REGB + 506		16	Reserved			
REGB + 508	SAPR1	32	IDMA1 Source Address Pointer	0000 0000		
REGB + 50C	DAPR1	32	IDMA1 Destination Address Pointer	0000 0000		
REGB + 510	BCR1	32	IDMA1 Byte Count Register	0000 0000		
REGB + 514	FCR1	8	IDMA1 Function Code Register	00		

Table 3-4. QUICC CPM Registers Memory Map

REGB + 515		8	Reserved			
REGB + 516	CMAR1	8	Channel Mask Register	00		
REGB + 517		8	Reserved			
REGB + 518	CSR1	8	IDMA1 Channel Status Register	00		
REGB + 519		24	Reserved			
REGB + 51C	SDSR	8	SDMA Status Register	00		SDMA
REGB + 51D		8	Reserved			
REGB + 51E	SDCR	16	SDMA Configuration Register	0000	H	
REGB + 520	SDAR	32	SDMA Address Register	XXXX XXXX		
REGB + 524		16	Reserved			IDMA2
REGB + 526	CMR2	16	IDMA2 Mode Register	0000		
REGB + 528	SAPR2	32	IDMA2 Source Address Pointer	0000 0000		
REGB + 52C	DAPR2	32	IDMA2 Destination Address Pointer	0000 0000		
REGB + 530	BCR2	32	IDMA2 Byte Count Register	0000 0000		
REGB + 534	FCR2	8	IDMA2 Function Code Register	00		
REGB + 535		8	Reserved			
REGB + 536	CMAR2	8	Channel Mask Register	00		
REGB + 537		8	Reserved			
REGB + 538	CSR2	8	IDMA2 Channel Status Register	00		
REGB + 539 to REGB + 53F			Reserved			
REGB + 540	CICR	24	CP Interrupt Configuration Register	xx00 0000	H	CPIC
REGB + 544	CIPR	32	CP Interrupt Pending Register	0000 0000		
REGB + 548	CIMR	32	CP Interrupt Mask Register	0000 0000		
REGB + 54C	CISR	32	CP In-Service Register	0000 0000		
REGB + 550	PADIR	16	Port A Data Direction Register	0000	H	Parallel I/O
REGB + 552	PAPAR	16	Port A Pin Assignment Register	0000	H	
REGB + 554	PAODR	16	Port A Open Drain Register	0000	H	
REGB + 556	PADAT	16	Port A Data Register	XXXX		
REGB + 558 to REGB + 55f			Reserved			
REGB + 560	PCDIR	16	Port C Data Direction Register	0000	H	
REGB + 562	PCPAR	16	Port C Pin Assignment Register	0000	H	
REGB + 564	PCSO	16	Port C Special Options	0000	H	
REGB + 566	PCDAT	16	Port C Data Register	XXXX		
REGB + 568	PCINT	16	Port C Interrupt Control Register	0000	H	
REGB + 56a to REGB + 57f			Reserved			
REGB + 580	TGCR	16	Timer Global Configuration Register	0000	H	TIMER

Table 3-4. QUICC CPM Registers Memory Map

REGB + 582 to REGB + 58f			Reserved			
REGB + 590	TMR1	16	Timer1 Mode Register	0000		
REGB + 592	TMR2	16	Timer2 Mode Register	0000		
REGB + 594	TRR1	16	Timer1 Reference Register	FFFF		
REGB + 596	TRR2	16	Timer2 Reference Register	FFFF		
REGB + 598	TCR1	16	Timer1 Capture Register	0000		
REGB + 59A	TCR2	16	Timer2 Capture Register	0000		
REGB + 59C	TCN1	16	Timer1 Counter	0000		
REGB + 59E	TCN2	16	Timer2 Counter	0000		
REGB + 5A0	TMR3	16	Timer3 Mode Register	0000		
REGB + 5A2	TMR4	16	Timer4 Mode Register	0000		
REGB + 5A4	TRR3	16	Timer3 Reference Register	FFFF		
REGB + 5A6	TRR4	16	Timer4 Reference Register	FFFF		
REGB + 5A8	TCR3	16	Timer3 Capture Register	0000		
REGB + 5AA	TCR4	16	Timer4 Capture Register	0000		
REGB + 5AC	TCN3	16	Timer3 Counter	0000		
REGB + 5AE	TCN4	16	Timer4 Counter	0000		
REGB + 5B0	TER1	16	Timer1 Event Register	0000		
REGB + 5B2	TER2	16	Timer2 Event Register	0000		
REGB + 5B4	TER3	16	Timer3 Event Register	0000		
REGB + 5B6	TER4	16	Timer4 Event Register	0000		
REGB + 5b8 to REGB + 5bf			Reserved			
REGB + 5CO	CR	16	Command Register	0000		CP
REGB + 5C4	RCCR	16	RISC Configuration Register	0000	H	
REGB + 5c6 to REGB + 5d5			Reserved			
REGB + 5D6	RTER	16	RISC Timers Event Register	0000		
REGB + 5DA	RTMR	16	RISC Timers Mask Register	0000		
REGB + 5dc to REGB + 5ef			Reserved			
REGB + 5F0	BRGC1	24	BRG1 Configuration Register	xx00 0000	H	BRG
REGB + 5F4	BRGC2	24	BRG2 Configuration Register	xx00 0000	H	
REGB + 5F8	BRGC3	24	BRG3 Configuration Register	xx00 0000	H	
REGB + 5FC	BRGC4	24	BRG4 Configuration Register	xx00 0000	H	
REGB + 600	GSMR_L1	32	SCC1 General Mode Register	0000 0000		SCC1
REGB + 604	GSMR_H1	32	SCC1 General Mode Register	0000 0000		
REGB + 608	PSMR1	16	SCC1 Protocol-Specific Mode Register	0000		
REGB + 60c	TODR1	16	SCC1 Transmit on Demand	0000		

Table 3-4. QUICC CPM Registers Memory Map

REGB + 60e	DSR1	16	SCC1 Data Sync. Register	7E7E		
REGB + 610	SCCE1	16	SCC1 Event Register	0000		
REGB + 614	SCCM1	16	SCC1 Mask Register	0000		
REGB + 617	SCCS1	8	SCC1 Status Register	00		
REGB + 618 to REGB + 61f			Reserved			
REGB + 620	GSMR_L2	32	SCC2 General Mode Register	0000 0000		SCC2
REGB + 624	GSMR_H2	32	SCC2 General Mode Register	0000 0000		
REGB + 628	PSMR2	16	SCC2 Protocol-Specific Mode Register	0000		
REGB + 62c	TODR2	16	SCC2 Transmit on Demand	0000		
REGB + 62e	DSR2	16	SCC2 Data Sync. Register	7E7E		
REGB + 630	SCCE2	16	SCC2 Event Register	0000		
REGB + 634	SCCM2	16	SCC2 Mask Register	0000		
REGB + 637	SCCS2	8	SCC2 Status Register	00		
REGB + 638 to REGB + 63f			Reserved			
REGB + 640	GSMR_L3	32	SCC3 General Mode Register	0000 0000		SCC3
REGB + 644	GSMR_H3	32	SCC3 General Mode Register	0000 0000		
REGB + 648	PSMR3	16	SCC3 Protocol-Specific Mode Register	0000		
REGB + 64c	TODR3	16	SCC3 Transmit on Demand	0000		
REGB + 64e	DSR3	16	SCC3 Data Sync. Register	7E7E		
REGB + 650	SCCE3	16	SCC3 Event Register	0000		
REGB + 654	SCCM3	16	SCC3 Mask Register	0000		
REGB + 657	SCCS3	8	SCC3 Status Register	00		
REGB + 658 to REGB + 65f			Reserved			
REGB + 660	GSMR_L4	32	SCC4 General Mode Register	0000 0000		SCC4
REGB + 664	GSMR_H4	32	SCC4 General Mode Register	0000 0000		
REGB + 668	PSMR4	16	SCC4 Protocol-Specific Mode Register	0000		
REGB + 66c	TODR4	16	SCC4 Transmit on Demand	0000		
REGB + 66e	DSR4	16	SCC4 Data Sync. Register	7E7E		
REGB + 670	SCCE4	16	SCC4 Event Register	0000		
REGB + 674	SCCM4	16	SCC4 Mask Register	0000		
REGB + 677	SCCS4	8	SCC4 Status Register	00		
REGB + 678 to REGB + 681			Reserved			
REGB + 682	SMCMR1	16	SMC1 Mode Register	0000		SMC1
REGB + 686	SMCE1	8	SMC1 Event Register	00		
REGB + 68a	SMCM1	8	SMC1 Mask Register	00		
REGB + 68C			Reserved			

Table 3-4. QUICC CPM Registers Memory Map

REGB + 692	SMCMR2	16	SMC2 Mode Register	0000		SMC2
REGB + 696	SMCE2	8	SMC2 or PIP Event Register	00		
REGB + 69a	SMCM2	8	SMC2 Mask Register	00		
REGB + 69C			Reserved			
REGB + 6A0	SPMODE	16	SPI Mode Register	0000	H	SPI
REGB + 6A6	SPIE	8	SPI Event Register	00		
REGB + 6AA	SPIM	8	SPI Mask Register	00		
REGB + 6AD	SPCOM	8	SPI Command Register	00		
REGB + 6B2	PIPC	16	PIP Configuration Register	0000	H	PIP
REGB + 6B6	PTPR	16	PIP Timing Parameters Register	0000		
REGB + 6B8	PBDIR	18	Port B Data Direction Register	xxx0 0000	H	
REGB + 6BC	PBPAR	18	Port B Pin Assignment Register	xxx0 0000	H	
REGB + 6C2	PBODR	16	Port B Open Drain Register	0000	H	
REGB + 6C4	PBDAT	18	Port B Data Register	xxxX XXXX		
REGB + 6c8 to REGB + 6df			Reserved			
REGB + 6E0	SIMODE	32	SI Mode Register	0000 0000	H	SI
REGB + 6E4	SIGMR	8	SI Global Mode Register	00	H	
REGB + 6E6	SISTR	8	SI Status Register	00	H	
REGB + 6E7	SICMR	8	SI Command Register	00		
REGB + 6E8		32	Reserved			
REGB+ 6EC	SICR	32	SI Clock Route	0000 0000	H	
REGB + 6F2	SIRP	32	SI RAM Pointers	0000 0000		
REGB + 6F6 to REGB + 6FF	RES		Reserved			
REGB + 700 to REGB + 7ff	SIRAM	256 Bytes	SI Routing RAM	XXXX		

Notes:
1.Reset value field.
2.H=Effected only upon $\overline{\text{RESETH}}$ assertion
3.S=Effected only upon $\overline{\text{RESETS}}$ assertion
4.Blank field = Effected by both $\overline{\text{RESETH}}$ or $\overline{\text{RESETS}}$ assertion.

SECTION 4
BUS OPERATION

This section provides a functional description of the system bus, the signals that control it, and the bus cycles provided for data transfer operations. It also describes the error and halt conditions, bus arbitration, and reset operation. Operation of the external bus is the same whether the QUICC or an external device is the bus master; the names and descriptions of bus cycles are from the viewpoint of the bus master. For exact timing specifications, refer to Section 10 Electrical Characteristics.

NOTE

The bus operation of the QUICC is very similar to the bus operation of the MC68030 and the MC68340. Much of the text and figures of the bus operation of those devices is common to this section.

The QUICC also supports the MC68EC040 (or other M68040 family members) as an external bus master. The MC68EC040 can access QUICC registers and use QUICC peripherals. The QUICC has a glueless MC68EC040 interface and special logic for acting as the MC68EC040 memory controller, interrupt controller, and the provider of system protection logic. The MC68EC040 bus operation is described in the *M68040 User Manual*. When the QUICC is the bus master of an M68040 system, its bus operation remains the same when it is the only bus master in the system. See 4.6.7 Internal Accesses for a description and timing diagram of the MC68EC040 internal read/write cycles (i.e., MC68EC040 reading/writing the QUICC) and interrupt acknowledge cycles. See 6.11 General-Purpose Chip-Select Overview (SRAM Banks) and 6.12 DRAM Controller Overview (DRAM Banks) for more information on the timing diagrams of MC68EC040 DRAM and SRAM accesses.

The QUICC architecture supports byte, word, and long-word operands allowing access to 8-, 16-, and 32-bit data ports through the use of asynchronous cycles controlled by the size outputs (SIZ1, SIZ0) and data size acknowledge inputs ($\overline{\text{DSACK1}}$, $\overline{\text{DSACK0}}$).

The QUICC allows byte, word, and long-word operands to be located in memory on any byte boundary. For a misaligned transfer, more than one bus cycle may be required to complete the transfer, regardless of port size. For a port less than 32 bits wide, multiple bus cycles may be required for an operand transfer due to either misalignment or a port width smaller than the operand size. Instruction words and their associated extension words must be aligned on word boundaries. The user should be aware that misalignment of word or long-word operands can cause the CPU32+ to perform multiple bus cycles for operand transfers; therefore, processor performance is optimized if word and long-word memory operands are

aligned on word or long-word boundaries, respectively. The QUICC IDMAs, when used, reduce the misalignment overhead to a minimum.

4.1 BUS TRANSFER SIGNALS

The bus transfers information between the QUICC and external memory or a peripheral device. External devices can accept or provide 8, 16, or 32 bits in parallel and must follow the handshake protocol described in this section. The maximum number of bits accepted or provided during a bus transfer is defined as the port width. The QUICC contains an address bus that specifies the address for the transfer and a data bus that transfers the data. Control signals indicate the beginning and type of the cycle as well as the address space and size of the transfer. The selected device then controls the length of the cycle with the signal(s) used to terminate the cycle. Strobe signals, one for the address bus and another for the data bus, indicate the validity of the address and provide timing information for the data.

Both asynchronous and synchronous operation is possible for any port width. In asynchronous operation, the bus and control input signals are internally synchronized to the QUICC clock, introducing a delay. This delay is the time required for the QUICC to sample an input signal, synchronize the input to the internal clocks, and determine whether it is high or low. In synchronous mode, the bus and control input signals must be timed to setup and hold times. Since no synchronization is needed, bus cycles can be completed in three clock cycles in this mode. Additionally, using the fast-termination option of the chip-select signals, two-clock operation is possible.

Furthermore, for all inputs, the QUICC latches the level of the input during a sample window around the falling edge of the clock signal. This window is illustrated in Figure 4-1, where t_{su} and t_h are the input setup and hold times, respectively. To ensure that an input signal is recognized on a specific falling edge of the clock, that input must be stable during the sample window. If an input makes a transition during the window time period, the level recognized by the QUICC is not predictable; however, the QUICC always resolves the latched level to either a logic high or low before using it. In addition to meeting input setup and hold times for deterministic operation, all input signals must obey the protocols described in this section.

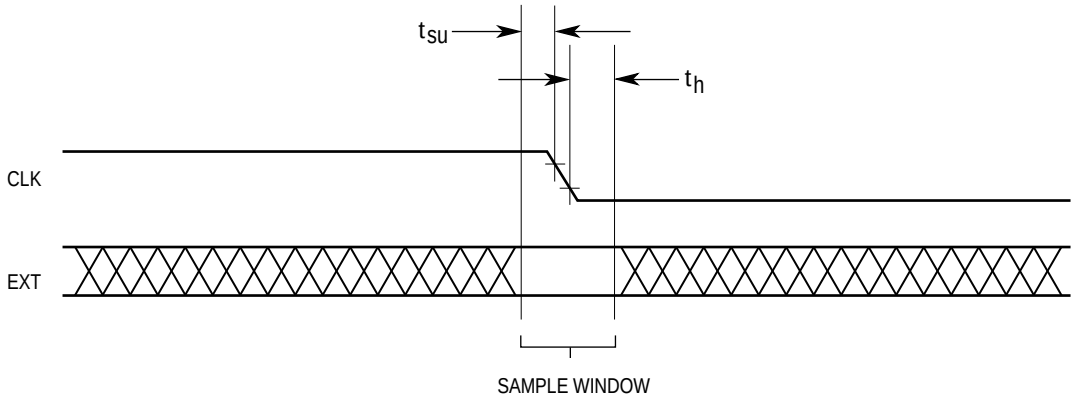


Figure 4-1. Input Sample Window

4.1.1 Bus Control Signals

The QUICC initiates a bus cycle by driving the address, size, function code, and read/write outputs. At the beginning of a bus cycle, SIZ1 and SIZ0 are driven with the FC signals. SIZ1 and SIZ0 indicate the number of bytes remaining to be transferred during an operand cycle (consisting of one or more bus cycles). Table 4-3 lists the encoding of SIZ1 and SIZ0. These signals are valid while $\overline{AS}$ is asserted.

The R/$\overline{W}$ signal determines the direction of the transfer during a bus cycle. Driven at the beginning of a bus cycle, R/$\overline{W}$ is valid while $\overline{AS}$ is asserted. R/$\overline{W}$ only transitions when a write cycle is preceded by a read cycle or vice versa. The signal may remain low for consecutive write cycles.

The $\overline{RMC}$ signal is asserted at the beginning of the first bus cycle of a read-modify-write operation and remains asserted until completion of the final bus cycle of the operation.

4.1.2 Function Codes (FC3–FC0)

The FCx signals are outputs that indicate one of 16 address spaces to which the address applies. Fifteen of these spaces are designated as either a normal or DMA cycle, user or supervisor, and program or data spaces. One other address space is designated as CPU space to allow the CPU32+ to acquire specific control information not normally associated with read or write bus cycles. The FCx signals are valid while $\overline{AS}$ is asserted.

Function codes (see Table 4-1) can be considered as extensions of the 32-bit address that can provide up to eight different 4-Gbyte address spaces. Function codes are automatically generated by the CPU32+ to select address spaces for data and program at both user and supervisor privilege levels, and a CPU address space for processor functions. User programs access only their own program and data areas to increase protection of system integrity and can be restricted from accessing other information. The S-bit in the CPU32+ status register is set for supervisor accesses and cleared for user accesses to provide differentiation. Refer to 4.4 CPU Space Cycles for more information.

Table 4-1. Address Space Encoding

Function Code Bits				Address Spaces
3	2	1	0	
0	0	0	0	Reserved (Motorola)
0	0	0	1	User Data Space
0	0	1	0	User Program Space
0	0	1	1	Reserved (User)
0	1	0	0	Reserved (Motorola)
0	1	0	1	Supervisor Data Space
0	1	1	0	Supervisor Program Space
0	1	1	1	Supervisor CPU Space
1	x	x	x	DMA space

4.1.3 Address Bus (A31–A0)

The address bus signals are outputs that define the address of the byte (or the most significant byte) to be transferred during a bus cycle. The QUICC places the address on the bus at the beginning of a bus cycle. The address is valid while $\overline{AS}$ is asserted.

4.1.4 Address Strobe ($\overline{AS}$)

$\overline{AS}$ is an output timing signal that indicates the validity of an address on the address bus and of many control signals. $\overline{AS}$ is asserted approximately one-half clock cycle after the beginning of a bus cycle.

4.1.5 Data Bus (D31-D0)

The data bus is a bidirectional, nonmultiplexed, parallel bus that contains the data being transferred to or from the QUICC. A read or write operation may transfer 8, 16, 24, or 32 bits of data (one, two, three, or four bytes) in one bus cycle. During a read cycle, the data is latched by the QUICC on the last falling edge of the clock for that bus cycle. For a write cycle, all 32 bits of the data bus are driven, regardless of the port width or operand size. The QUICC places the data on the data bus approximately one-half clock cycle after $\overline{AS}$ is asserted in a write cycle.

4.1.6 Data Strobe ($\overline{DS}$)

$\overline{DS}$ is an output timing signal that applies to the data bus. For a read cycle, the QUICC asserts $\overline{DS}$ and $\overline{AS}$ simultaneously to signal the external device to place data on the bus. For a write cycle, $\overline{DS}$ signals to the external device that the data to be written is valid. The QUICC asserts $\overline{DS}$ approximately one clock cycle after the assertion of $\overline{AS}$ during a write cycle.

4.1.7 Output Enable ($\overline{OE}$)

$\overline{OE}$ is an output timing signal that applies to the data bus. On a read cycle, the QUICC asserts $\overline{OE}$ to signal the external device to place data on the bus. $\overline{OE}$ is asserted during read cycles with timing similar to $\overline{AS}$.

$\overline{OE}$ is not shown in the diagrams in this section. Use $\overline{AS}$ timing instead during read cycles.

4.1.8 Byte Write Enable ($\overline{WE0}$, $\overline{WE1}$, $\overline{WE2}$, $\overline{WE3}$)

The upper upper write enable ($\overline{WE0}$) indicates that the upper eight bits of the data bus (D31–D24) contain valid data during a write cycle. The upper middle write enable ($\overline{WE1}$) indicates that the upper middle eight bits of the data bus (D23–D16) contain valid data during a write cycle. The lower middle write enable ($\overline{WE2}$) indicates that the lower middle eight bits of the data bus (D15–D8) contain valid data during a write cycle. The lower write enable ($\overline{WE3}$) indicates that the lower eight bits of the data bus contain valid data during a write cycle.

The equations of the byte write enables for 32-bit port (16BM = 1) are as follows:

```
WE0 = R/W + AS + A0 + A1
WE1 = R/W + AS + not {(A1 * SIZ0) + (A0 * A1) + (A1 * SIZ1)}
WE2 = R/W + AS + not {(A0 * A1) + (A1 * SIZ0 * SIZ1) + (A1 * SIZ0 * SIZ1) +
       (A0 * A1 * SIZ0)}
WE3 = R/W + AS + not {(A0 * SIZ0 * SIZ1) + (SIZ0 * SIZ1) + (A0 * A1) + (A1 *
       SIZ1)}
```

These signals have the same timing as AS. The equations are valid only for a 32-bit port.

The equations of the byte write enables for 16-bit port (B16M = 0) are as follows:

```
WE0 = R/W + AS + A0
WE1 = R/W + AS + (A0 * SIZ0 * SIZ1)
```

These signals have the same timing as AS. The equations are valid only for a 16-bit port.

WEx signals are not shown in the diagrams in this section. Use AS timing instead during write cycles. The particular WEx signals that are active in a given bus cycle depend on which bytes are being written.

NOTE

Note that the WE signals are not affected by dynamic bus sizing. External assertion of DSACKx will have no effect on which WEx signal gets asserted.

When 16-bit mode is selected and Bit 7 of PEPAR is set, WE2 and WE3 are used as address lines A29 and A28 respectively.

4.1.9 Bus Cycle Termination Signals

The following signals can terminate a bus cycle.

4.1.9.1 DATA TRANSFER AND SIZE ACKNOWLEDGE (DSACK1 AND DSACK0). During bus cycles, external devices assert DSACK1 and/or DSACK0 as part of the bus protocol. During a read cycle, this signals the QUICC to terminate the bus cycle and to latch the data. During a write cycle, this indicates that the external device has successfully stored the data and that the cycle may terminate. These signals also indicate to the QUICC the size of the port for the bus cycle just completed (see Table 4-3). Refer to 4.3.1 Read Cycle for timing relationships of DSACK1 and DSACK0.

Additionally, the system integration module (SIM60) can be programmed to internally generate DSACK1 and DSACK0 for external accesses, eliminating logic required to generate these signals. The SIM60 can alternatively be programmed to generate a fast termination, providing a two-cycle external access. Refer to 4.2.6 Fast Termination Cycles for additional information on these cycles.

4.1.9.2 BUS ERROR (BERR). This signal is also a bus cycle termination indicator and can be used in the absence of DSACKx to indicate a bus error condition. BERR can also be asserted in conjunction with DSACKx to indicate a bus error condition, provided it meets the

appropriate timing described in this section and in Section 10 Electrical Characteristics. Additionally, BERR and HALT can be asserted together to indicate a retry termination. Refer to 4.5 Bus Exception Control Cycles for additional information on the use of these signals.

See the memory controller description in Section 6 System Integration Module (SIM60) for precautions about asserting BERR externally too early during DRAM and SRAM cycles controlled by the memory controller.

The internal bus monitor can be used to generate the BERR signal for internal and external transfers in all the following descriptions.

4.1.9.3 AUTOVECTOR (AVEC). This signal can be used to terminate interrupt acknowledge cycles, indicating that the QUICC should internally generate a vector (autovector) number to locate an interrupt handler routine. AVEC can be generated either externally or internally by the SIM60 (refer to Section 6 System Integration Module (SIM60) for additional information). AVEC is ignored during all other bus cycles.

4.2 DATA TRANSFER MECHANISM

The QUICC supports byte, word, and long-word operands, allowing access to 8-,16-, and 32-bit data ports through the use of asynchronous cycles controlled by DSACK1 and DSACK0. The QUICC also supports byte, word, and long-word operands, allowing access to 8-, 16, and 32-bit data ports through the use of synchronous cycles controlled by the fast-termination capability of the SIM60.

4.2.1 Dynamic Bus Sizing

The QUICC dynamically interprets the port size of the addressed device during each bus cycle, allowing operand transfers to or from 8-, 16-, and 32-bit ports. During an operand transfer cycle, the slave device signals its port size (byte, word, or long word) and indicates completion of the bus cycle to the QUICC through the use of the DSACKx inputs. Refer to Table 4-2 for DSACKx encoding.

Table 4-2. DSACKx Encoding

DSACK1	DSACK0	Result
1	1	Insert Wait States in Current Bus Cycle
1	0	Complete Cycle—Data Bus Port Size is 8 Bits
0	1	Complete Cycle—Data Bus Port Size is 16 Bits
0	0	Complete Cycle—Data Bus Port Size is 32 Bits

For example, if the QUICC is executing an instruction that reads a long-word operand from a long-word aligned address, it attempts to read 32 bits during the first bus cycle. (Refer to 4.2.2 Misaligned Operands for the case of a word or byte address.) If the port responds that it is 32 bits wide, the QUICC latches all 32 bits of data and continues with the next operation. If the port responds that it is 16 bits wide, the QUICC latches the 16 bits of valid data and runs another bus cycle to obtain the other 16 bits. The operation for an 8-bit port is similar, but requires four read cycles. The addressed device uses the DSACKx signals to indicate

the port width. For instance, a 32-bit device always returns $\overline{\text{DSACKx}}$ for a 32-bit port (regardless of whether the bus cycle is a byte, word, or long-word operation).

Dynamic bus sizing requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 32-bit port must reside on data bus bits 0–31, a 16-bit port must reside on data bus bits 16–32, and an 8-bit port must reside on data bus bits 24–31. This requirement minimizes the number of bus cycles needed to transfer data to 8- and 16-bit ports and ensures that the QUICC correctly transfers valid data. The QUICC always attempts to transfer the maximum amount of data on all bus cycles; for a long-word operation, it always assumes that the port is 32 bit wide when beginning the bus cycle.

The bytes of operands are designated as shown in Figure 4-2. The most significant byte of a long-word operand is OP0, and OP3 is the least significant byte. The two bytes of a word-length operand are OP2 (most significant) and OP3. The single byte of a byte-length operand is OP3. These designations are used in the figures and descriptions that follow.

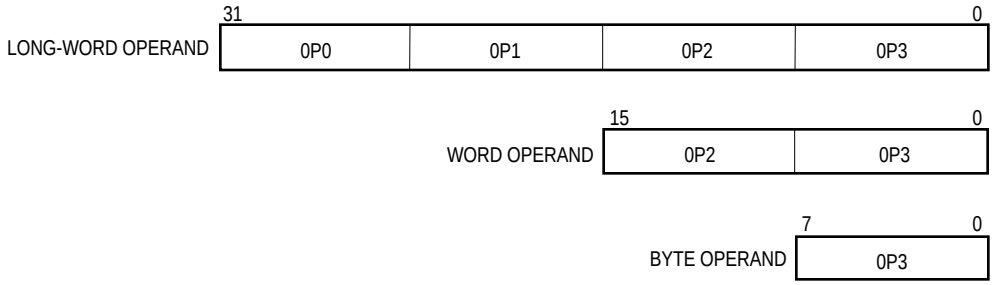


Figure 4-2. Internal Operand Representation

Figure 4-3 shows the required organization of data ports on the QUICC bus for 8, 16, and 32-bit devices. The four bytes shown are connected through the internal data bus and data multiplexer to the external data bus. This path is the means through which the QUICC supports dynamic bus sizing and operand misalignment. Refer to 4.2.2 Misaligned Operands for the definition of misaligned operand. The data multiplexer establishes the necessary connections for different combinations of address and data sizes.

The multiplexer takes the four bytes of the 32-bit bus and routes them to their required positions. For example, OP0 can be routed to D24–D31, as would be the normal case, or it can be routed to any other byte position to support a misaligned transfer. The same is true for any of the operand bytes. The positioning of bytes is determined by the size and address outputs.

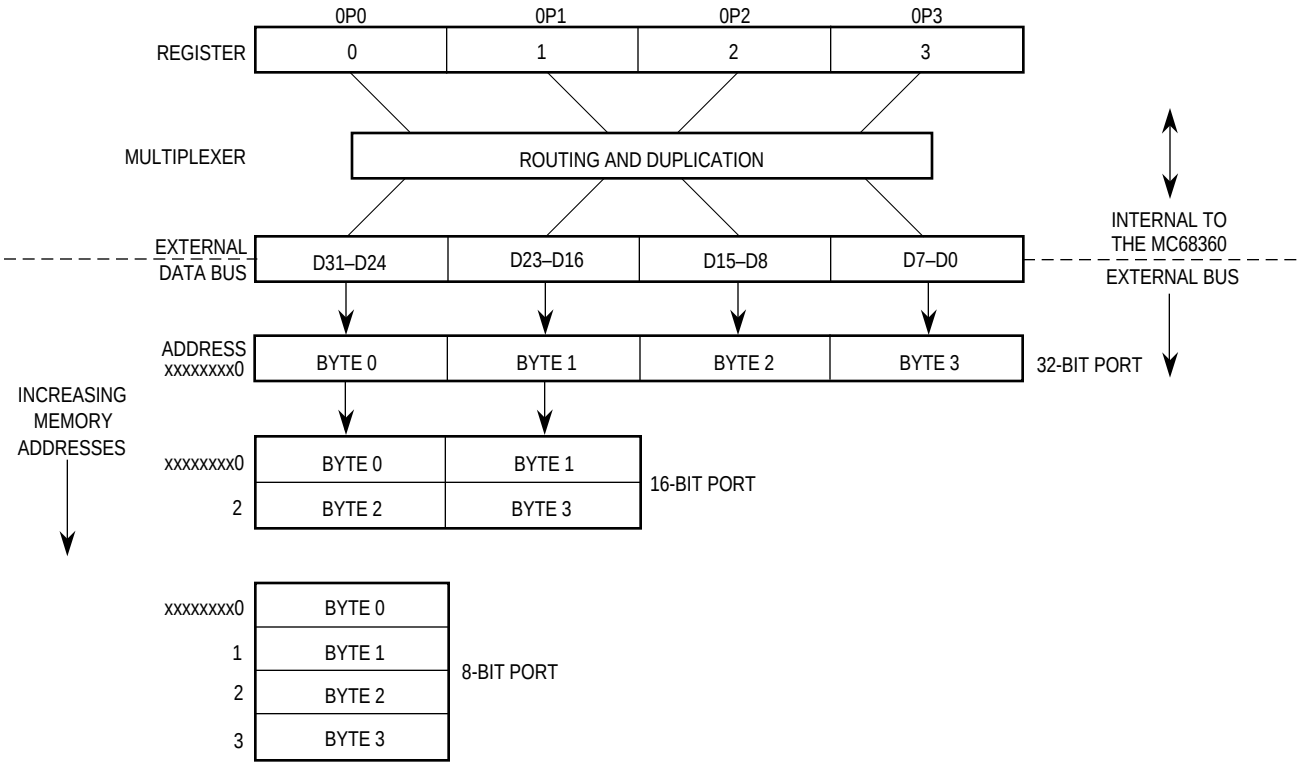


Figure 4-3. QUICC Interface to Various Port Sizes

The SIZ0 and SIZ1 outputs indicate the remaining number of bytes to be transferred during the current bus cycle (see Table 4-3).

Table 4-3. SIZx Encoding

SIZ1	SIZ0	Size
0	1	Byte
1	0	Word
1	1	3 Bytes
0	0	Long Word

The number of bytes transferred during a write or read bus cycle is equal to or less than the size indicated by the SIZx outputs, depending on port width and operand alignment. For example, during the first bus cycle of a long-word transfer to a word port, the SIZx outputs indicate that four bytes are to be transferred, although only two bytes are moved on that bus cycle.

A0 and A1 also affect operation of the data multiplexer. During an operand transfer, A2-A31 indicate the long-word base address of that portion of the operand to be accessed; A0 and A1 indicate the byte offset from the base. Table 4-4 lists the encoding of A0 and A1 and the corresponding byte offset from the long-word base.

Table 4-4. Address Offset Encoding

A1	A0	Offset
0	0	+0 Byte
0	1	+1 Byte
1	0	+2 Bytes
1	1	+3 Bytes

Table 4-5 lists the bytes required on the data bus for read cycles. The entries shown as OPx are portions of the requested operand that are read during that bus cycle and are defined by SIZ0, SIZ1, A0, and A1 for the bus cycle. Bytes labeled x are “don’t cares” and are not required during that read cycle.

Table 4-5. Data Bus Requirements for Read Cycles

Transfer Size	Size		Address		Long-Word Port External Data Bytes Required				Word Port External Data Bytes Required		Byte Port External Data Bytes Required
	SIZ1	SIZ0	A1	A0	D31:D24	D23:D16	D15:D8	D7:D0	D31:D24	D23:D16	D31:D24
Byte	0	1	0	0	OP3	x	x	x	OP3	x	OP3
	0	1	0	1	x	OP3	x	x	x	OP3	OP3
	0	1	1	0	x	x	OP3	x	OP3	x	OP3
	0	1	1	1	x	x	x	OP3	x	OP3	OP3
Word	1	0	0	0	OP2	OP3	x	x	OP2	OP3	OP2
	1	0	0	1	x	OP2	OP3	x	x	OP2	OP2
	1	0	1	0	x	x	OP2	OP3	OP2	OP3	OP2
	1	0	1	1	x	x	x	OP2	x	OP2	OP2
3 Bytes	1	1	0	0	OP1	OP2	OP3	x	OP1	OP2	OP1
	1	1	0	1	x	OP1	OP2	OP3	x	OP1	OP1
	1	1	1	0	x	x	OP1	OP2	OP1	OP2	OP1
	1	1	1	1	x	x	x	OP1	x	OP1	OP1
Long Word	0	0	0	0	OP0	OP1	OP2	OP3	OP0	OP1	OP0
	0	0	0	1	x	OP0	OP1	OP2	x	OP0	OP0
	0	0	1	0	x	x	OP0	OP1	OP0	OP1	OP0
	0	0	1	1	x	x	x	OP0	x	OP0	OP0

Table 4-6 lists the combinations of SIZ0, SIZ1, A0, and A1 and the corresponding pattern of the data transfer for write cycles from the internal multiplexer of the QUICC to the external data bus. Bytes labeled x are “don't care.”

Figure 4-4 shows the transfer of a long-word operand to a word port. In the first bus cycle, the QUICC places the four operand bytes on the external bus. Since the address is long-word aligned in this example, the multiplexer follows the pattern in the entry of Table 4-6 corresponding to SIZ0, SIZ1, A0, A1 = 0000. The port latches the data on bits D16–D31 of the data bus, asserts $\overline{\text{DSACK1}}$ ($\overline{\text{DSACK0}}$ remains negated), and the QUICC terminates the bus cycle. It then starts a new bus cycle with SIZ0, SIZ1, A0, A1 = 1010 to transfer the remaining

16 bits. SIZ0 and SIZ1 indicate that a word remains to be transferred; A0 and A1 indicate that the word corresponds to an offset of two from the base address. The multiplexer follows the pattern corresponding to this configuration of the size and address signals and places the two least significant bytes of the long word on the word portion of the bus (D16–D31). The bus cycle transfers the remaining bytes to the word-size port. Figure 4-5 shows the timing of the bus transfer signals for this operation.

Table 4-6. QUICC Internal to External Data Bus Multiplexer—Write Cycle

Transfer Size	Size		Address		External Data Bus Connection			
	SIZ1	SIZ0	A1	A0	D31:D24	D23:D16	D15:D8	D7:D0
Byte	0	1	0	0	OP3	x	x	x
	0	1	0	1	OP3	OP3	x	x
	0	1	1	0	OP3	x	OP3	x
	0	1	1	1	OP3	OP3	x	OP3
Word	1	0	0	0	OP2	OP3	x	x
	1	0	0	1	OP2	OP2	OP3	x
	1	0	1	0	OP2	OP3	OP2	OP3
	1	0	1	1	OP2	OP2	x	OP2
3 Bytes	1	1	0	0	OP1	OP2	OP3	x
	1	1	0	1	OP1	OP1	OP2	OP3
	1	1	1	0	OP1	OP2	OP1	OP2
	1	1	1	1	OP1	x	OP2	OP1
Long Word	0	0	0	0	OP0	OP1	OP2	OP3
	0	0	0	1	OP0	OP0	OP1	OP2
	0	0	1	0	OP0	OP1	OP0	OP1
	0	0	1	1	OP0	OP0	x	OP0

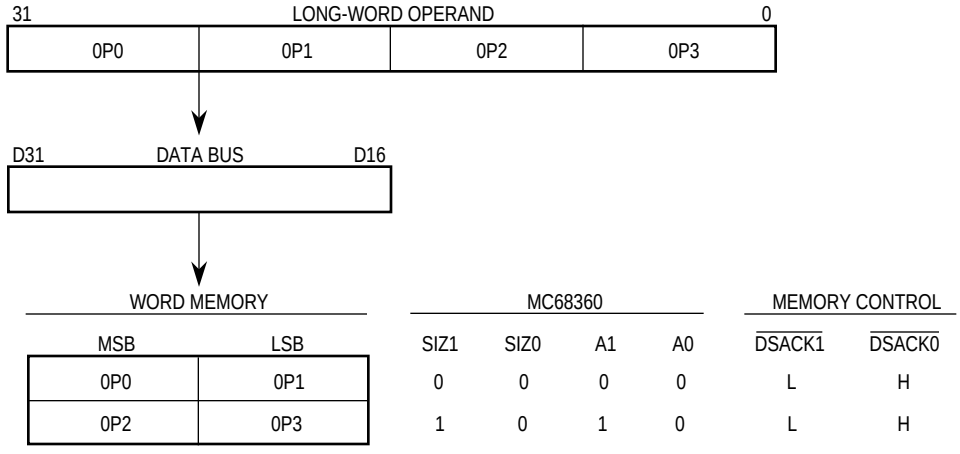


Figure 4-4. Example of Long-Word Transfer to Word Port

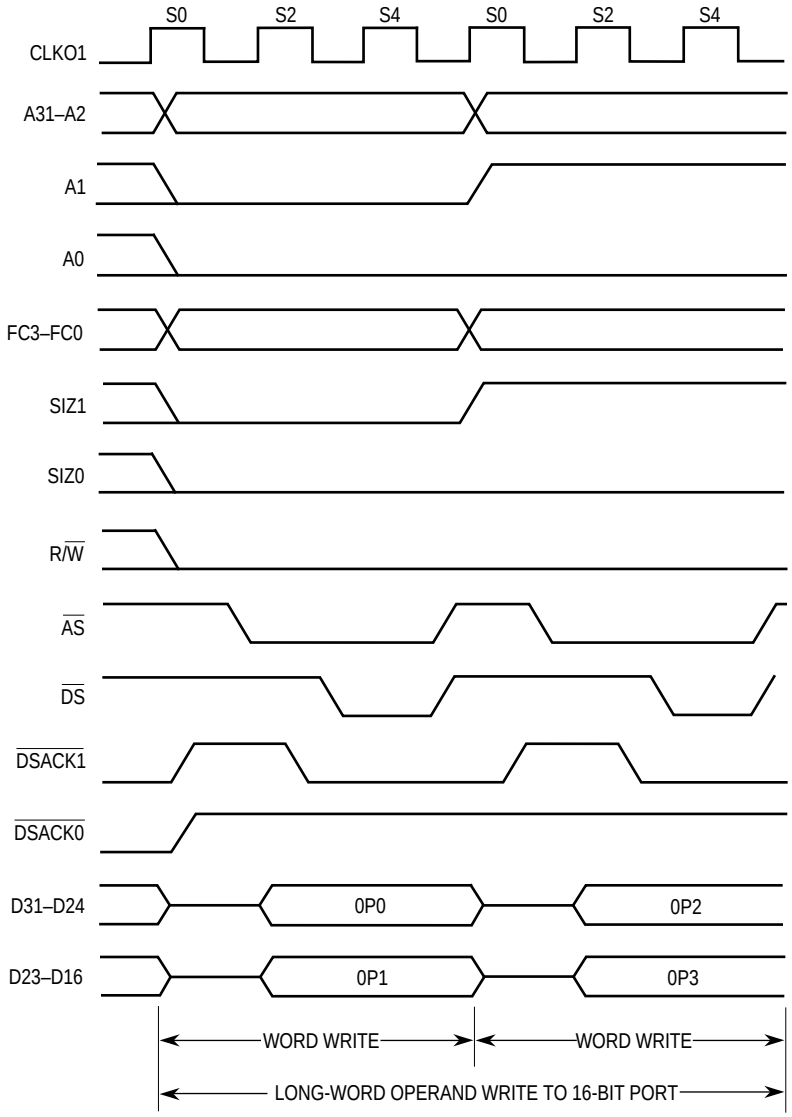


Figure 4-5. Long-Word Operand Write Timing (16-Bit Data Port)

Figure 4-6 shows a word transfer to an 8-bit bus port. Like the preceding example, this example requires two bus cycles. Each bus cycle transfers a single byte. The size signals for the first cycle specify two bytes; for the second cycle, they specify one byte. Figure 4-7 shows the associated bus transfer signal timing.

4.2.2 Misaligned Operands

Since operands may reside at any byte boundaries, they may be misaligned. A byte operand is properly aligned at any address; a word operand is misaligned at an odd address; a long word is misaligned at an address that is not evenly divisible by four. The MC68302, MC68000/MC68008, MC68010, and MC68340 implementations allow long-word transfers on odd-word boundaries but force exceptions if word or long-word operand transfers are attempted at odd-byte addresses. Although the QUICC does not enforce any alignment restrictions for data operands (including PC relative data addresses), some performance degradation occurs when additional bus cycles are required for long-word or word operands

that are misaligned. For maximum performance, data items should be aligned on their natural boundaries. All instruction words and extension words must reside on word boundaries. Attempting to prefetch an instruction word at an odd address causes an address error exception.

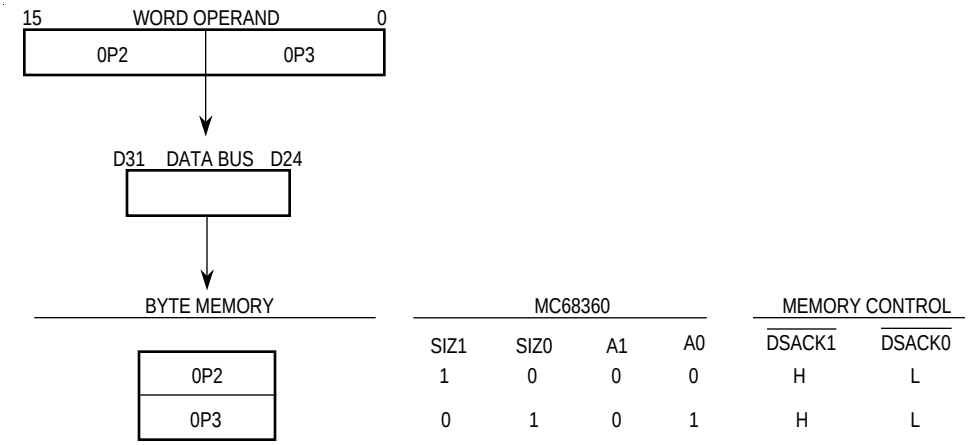


Figure 4-6. Example of Word Transfer to Byte Port

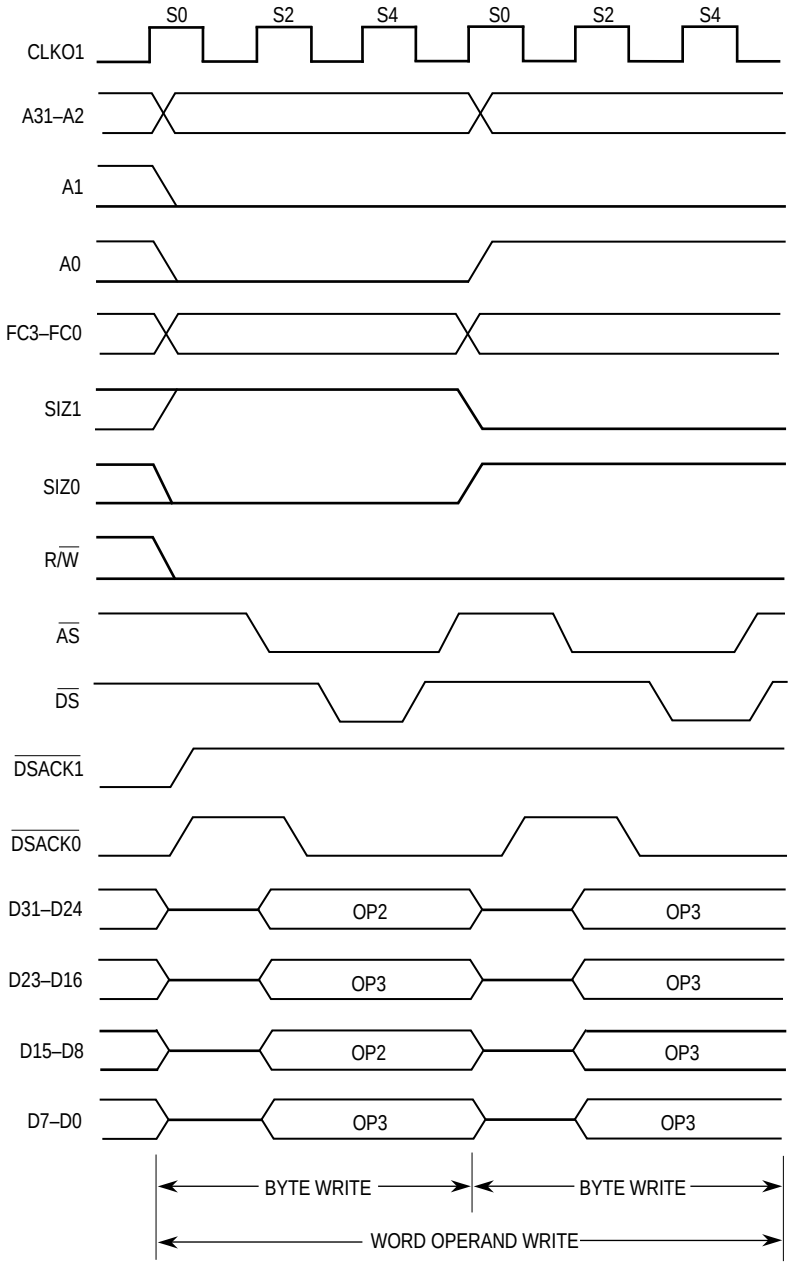
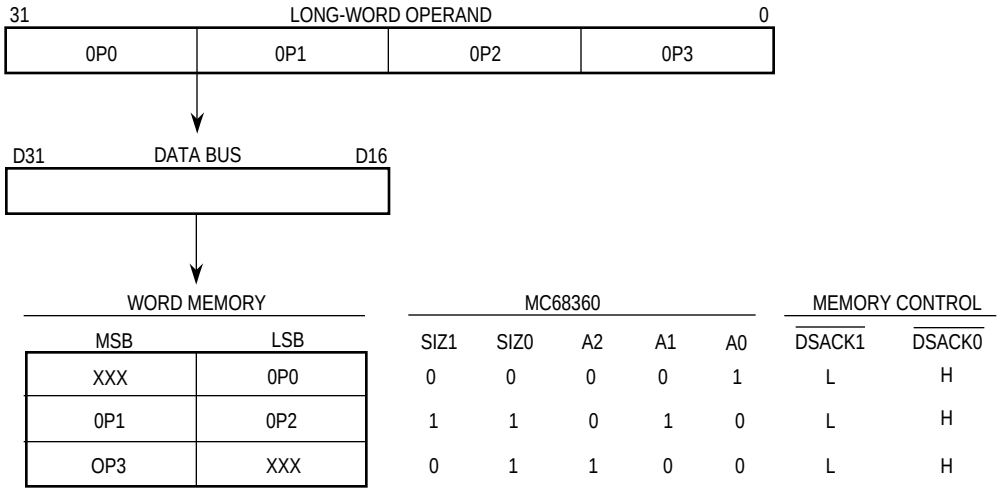


Figure 4-7. Word Operand Write Timing (8-Bit Data Port)

Figure 4-8 shows the transfer of a long-word operand to an odd address in word-organized memory, which requires three bus cycles. For the first cycle, the SIZx signals specify a long-word transfer, and the address offset (A2–A0) is 001. Since the port width is 16 bits, only the first byte of the long word is transferred. The slave device latches the byte and acknowledges the data transfer, indicating that the port is 16 bits wide. When the processor starts the second cycle, the SIZx signals specify that three bytes remain to be transferred with an address offset (A2–A0) of 010. The next two bytes are transferred during this cycle. The processor then initiates the third cycle, with the SIZx signals indicating one byte remaining to be transferred. The address offset (A2–A0) is now 100; the port latches the final byte, and the operation is complete. Figure 4-9 shows the associated bus transfer signal timing.



WORD MEMORY		MC68360					MEMORY CONTROL	
MSB	LSB	SIZ1	SIZ0	A2	A1	A0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$
XXX	0P0	0	0	0	0	1	L	H
0P1	0P2	1	1	0	1	0	L	H
OP3	XXX	0	1	1	0	0	L	H

Figure 4-8. Misaligned Long-Word Transfer to Word Port Example

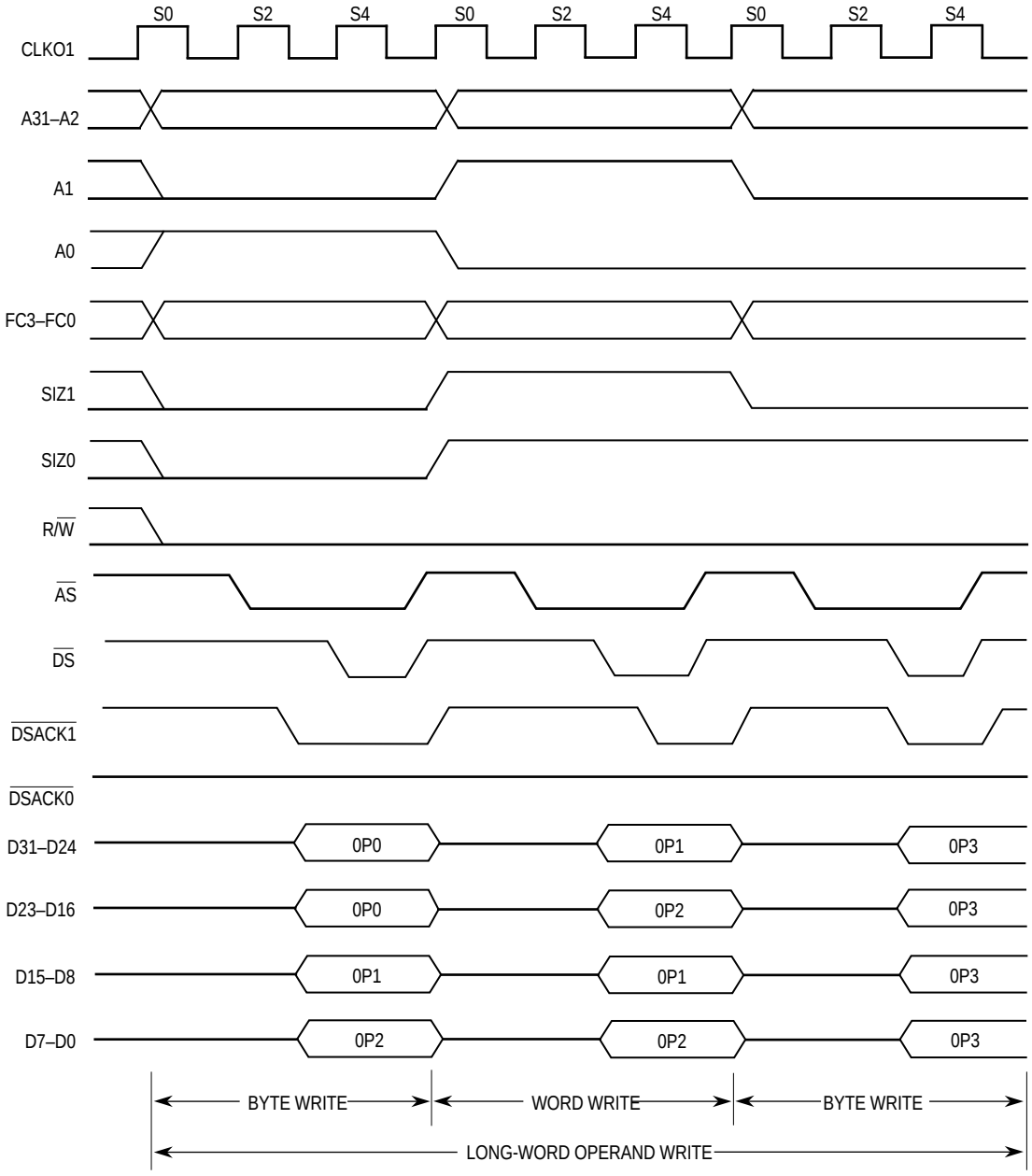


Figure 4-9. Misaligned Long-Word Transfer to Word Port Timing

Figure 4-10 and Figure 4-11 show a word transfer to an odd address in word-organized memory. This example is similar to the one shown in Figure 4-8 and Figure 4-9 except that the operand is word sized and the transfer requires only two bus cycles.

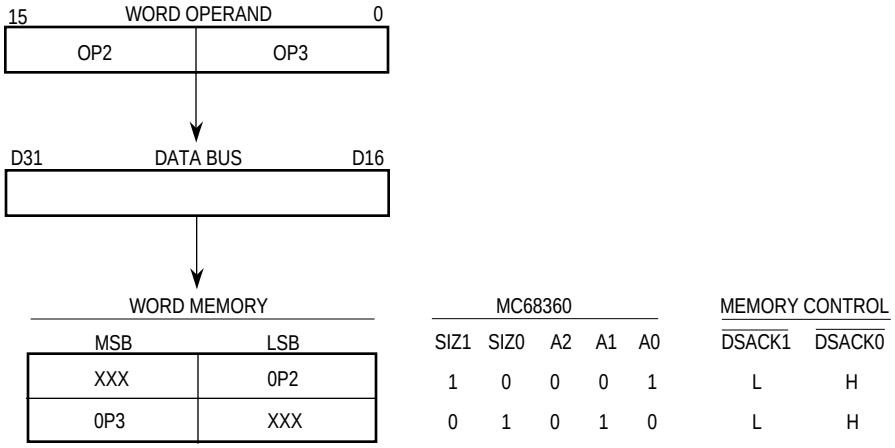


Figure 4-10. Misaligned Word Transfer to Word Port Example

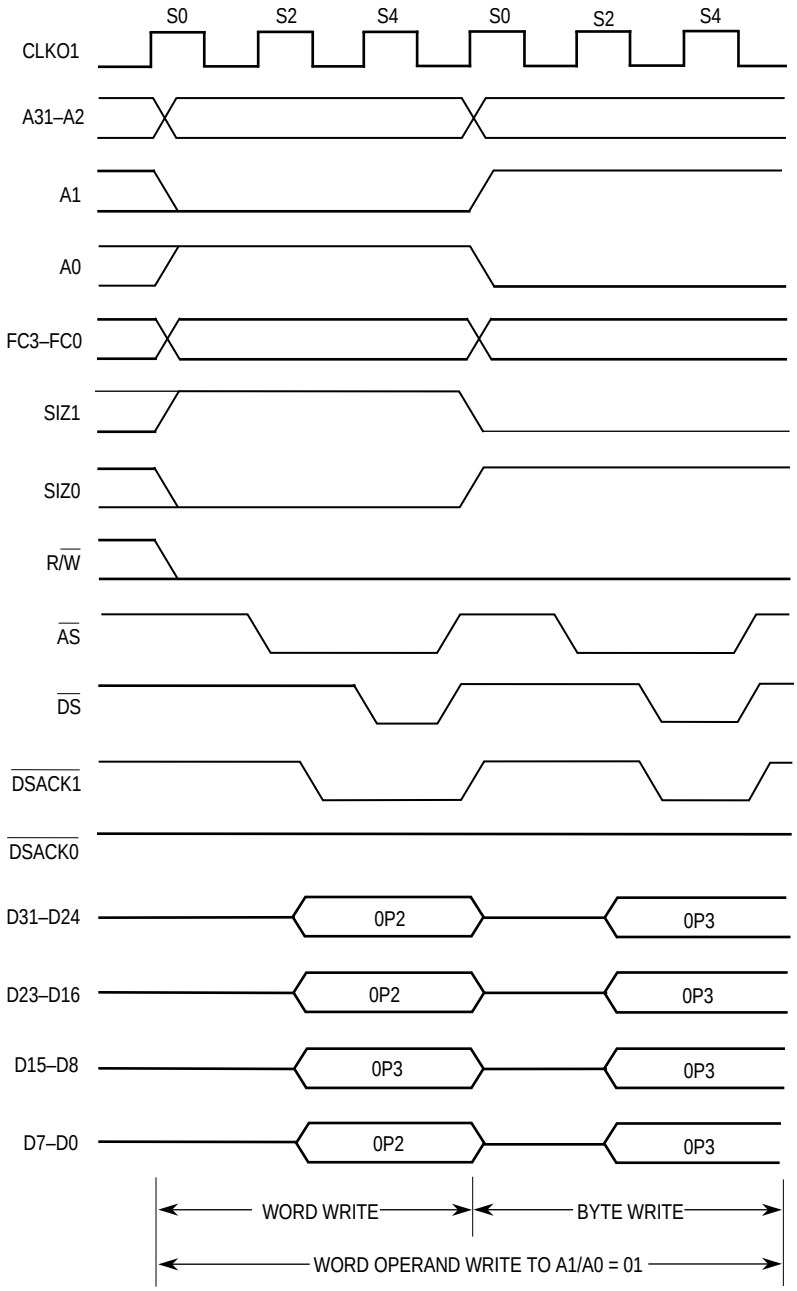


Figure 4-11. Misaligned Word Transfer to Word Port Timing

Figure 4-12 and Figure 4-13 show an example of a long-word transfer to an odd address in long-word-organized memory. In this example, a long-word access is attempted beginning at the least significant byte of a long-word-organized memory. Only one byte can be transferred in the first bus cycle. The second bus cycle then consists of a three-byte access to a long-word boundary. Since the memory is long-word organized, no further bus cycles are necessary.

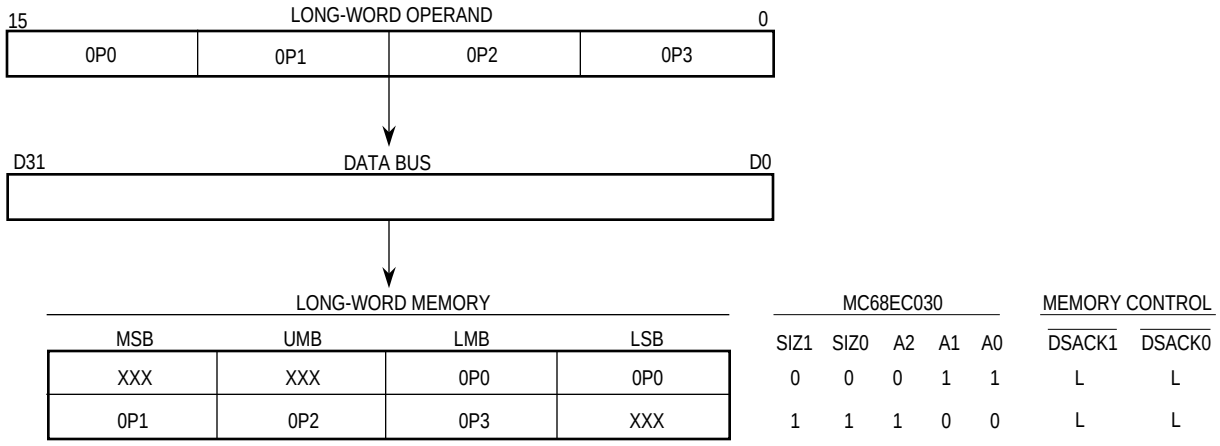


Figure 4-12. Misaligned Long-Word Transfer to Long-Word Port Example

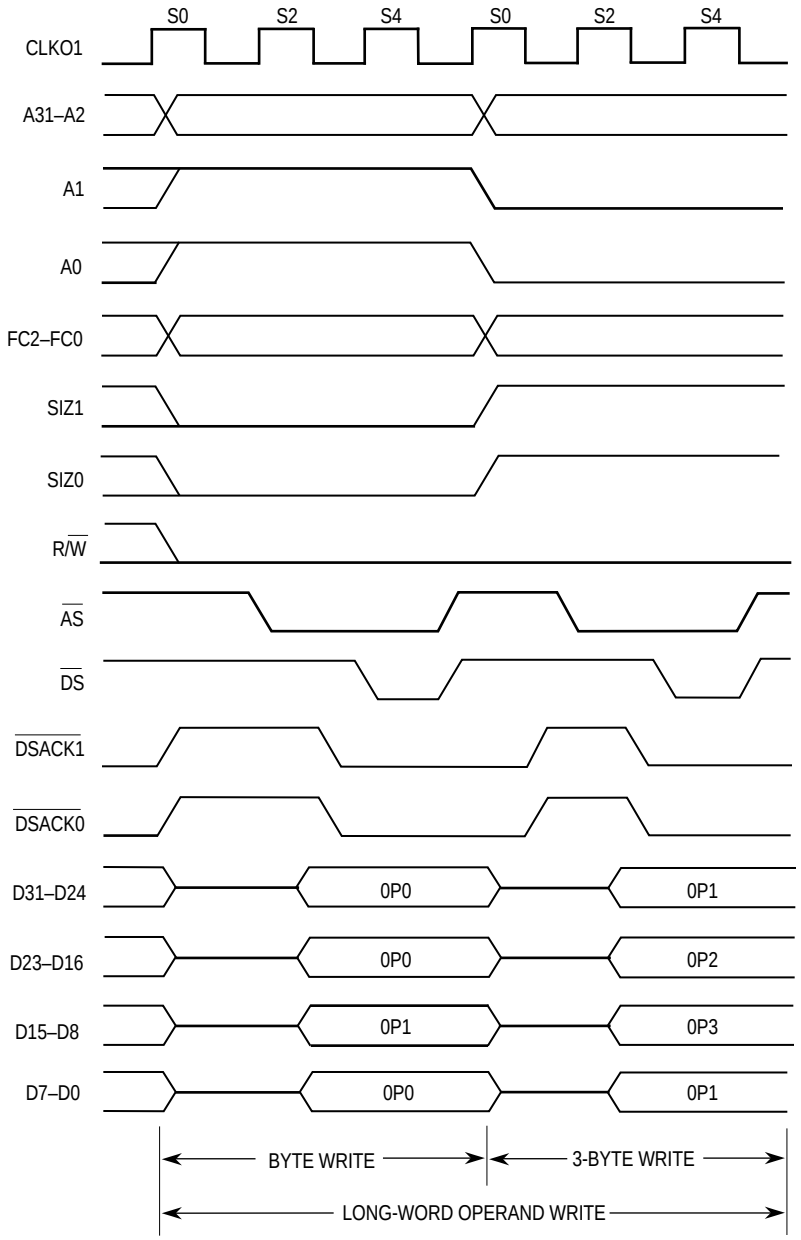


Figure 4-13. Misaligned Long-Word Transfer to Long-Word Port Timing

4.2.3 Effects of Dynamic Bus Sizing and Operand Misalignment

The combination of operand size, operand alignment, and port size determines the number of bus cycles required to perform a particular memory access. Table 4-7 lists the number of bus cycles required for different operand sizes to different port sizes with all possible alignment conditions for write cycles and read cycles.

Table 4-7. Memory Alignment and Port Size Influence on Write Bus Cycles

A1–A0	Number of Bus Cycles			
	00	01	10	11
Instruction[1]	1:2:4	N/A	N/A	N/A
Byte Operand	1:1:1	1:1:1	1:1:1	1:1:1
Word Operand	1:1:2	1:2:2	1:1:2	2:2:2
Long-Word Operand	1:2:4	2:3:4	2:2:4	2:3:4

Notes:
1. Data Port Size—32 Bits:16 Bits:8 Bits
2. Instruction reads can either be two words from an even-word boundary, or one word from an odd-word boundary.

This table verifies that bus cycle throughput is significantly affected by port size and alignment. The QUICC system designer and programmer should be aware of and account for these effects, particularly in time-critical applications.

If the required instruction begins at an even-word boundary, the processor prefetches a long word (up to two instructions) by reading a long word from a long-word address (A1–A0 = 00), regardless of port size. When the required instruction begins at an odd-word boundary, the processor reads 16-bits only, from the odd-word boundary. Refer to Section 5 CPU32+ for a complete description of the pipeline operation.

4.2.4 Bus Operation

The QUICC bus is asynchronous, allowing external devices connected to the bus to operate at clock frequencies different from the clock for the QUICC. Bus operation uses the handshake lines ($\overline{AS}$, $\overline{DS}$, $\overline{DSACK1}$, $\overline{DSACK0}$, $\overline{BERR}$, and $\overline{HALT}$) to control data transfers. $\overline{AS}$ signals a valid address on the address bus, and $\overline{DS}$ is used as a condition for valid data on a write cycle. Decoding the SIZx outputs and lower address lines (A1–A0) provides strobes that select the active portion of the data bus. The slave device (memory or peripheral) responds by placing the requested data on the correct portion of the data bus for a read cycle or by latching the data on a write cycle; the slave asserts the $\overline{DSACK1}$/$\overline{DSACK0}$ combination that corresponds to the port size to terminate the cycle.

Alternatively, the SIM60 can be programmed to assert the $\overline{DSACK1}$/$\overline{DSACK0}$ combination internally and respond for the slave. If no slave responds or the access is invalid, external control logic may assert $\overline{BERR}$ or $\overline{BERR}$ with $\overline{HALT}$ to abort or retry the bus cycle, respectively. $\overline{DSACKx}$ can be asserted before the data from a slave device is valid on a read cycle. The length of time that $\overline{DSACKx}$ may precede data must not exceed a specified value in any asynchronous system to ensure that valid data is latched into the QUICC. (See Section 10 Electrical Characteristics for timing parameters.)

Note that no maximum time is specified from the assertion of $\overline{AS}$ to the assertion of $\overline{DSACKx}$. Although the QUICC can transfer data in a minimum of three clock cycles when the cycle is terminated with $\overline{DSACKx}$, the QUICC inserts wait cycles in clock-period increments until $\overline{DSACKx}$ is recognized. $\overline{BERR}$ and/or $\overline{HALT}$ can be asserted after $\overline{DSACKx}$ is asserted. $\overline{BERR}$ and/or $\overline{HALT}$ must be asserted within the time specified after $\overline{DSACKx}$ is

asserted in any asynchronous system. If this maximum delay time is violated, the QUICC may exhibit erratic behavior.

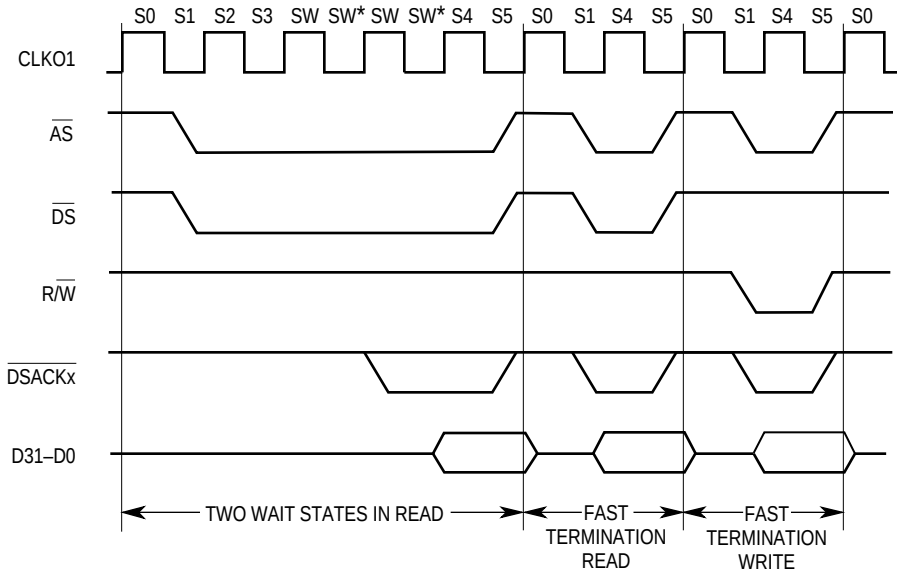
4.2.5 Synchronous Operation with DSACKx

Although cycles terminated with DSACKx are classified as asynchronous, cycles terminated with DSACKx can also operate synchronously in that signals are interpreted relative to clock edges. The devices that use these cycles must synchronize the response to the QUICC clock (CLKO1) to be synchronous. Since the devices terminate bus cycles with DSACKx, the dynamic bus sizing capabilities of the QUICC are available. The minimum cycle time for these cycles is also three clocks. To support systems that use the system clock to generate DSACKx and other asynchronous inputs, the asynchronous input setup time and the asynchronous input hold time are given. If the setup and hold times are met for the assertion or negation of a signal, such as DSACKx, the QUICC is guaranteed to recognize that signal level on that specific falling edge of the system clock. If the assertion of DSACKx is recognized on a particular falling edge of the clock, valid data is latched into the QUICC (for a read cycle) on the next falling clock edge if the data meets the data setup time. In this case, the parameter for asynchronous operation can be ignored. The timing parameters are described in Section 10 Electrical Characteristics.

If a system asserts DSACKx for the required window around the falling edge of S2 and obeys the proper bus protocol by maintaining DSACKx (and/or BERR/HALT) until and throughout the clock edge that negates AS (with the appropriate asynchronous input hold time), no wait states are inserted. The bus cycle runs at its maximum speed for bus cycles terminated with DSACKx (three clocks per cycle). When BERR (or BERR and HALT) is asserted after DSACKx, BERR (and HALT) must meet the appropriate setup time prior to the falling clock edge one clock cycle after DSACKx is recognized. This setup time is critical, and the QUICC may exhibit erratic behavior if it is violated. When operating synchronously, the data-in setup and hold times for synchronous cycles may be used instead of the timing requirements for data relative to DS.

4.2.6 Fast Termination Cycles

With an external device that has a fast access time, the memory controller circuits can provide a two-clock external bus transfer. Since the memory controller circuits are driven from the system clock, the bus cycle termination is inherently synchronized with the system clock. Refer to Section 6 System Integration Module (SIM60) for more information on chip selects and the DRAM controller. To use the fast termination (cycle length is two clocks) option, an external device should be fast enough to have data ready, within the specified setup time, by the falling edge of S4. Figure 4-14 shows the DSACKx timing for a read with two wait states, followed by a fast termination read and write.



* DSACKx only internally asserted for fast termination cycles.

Figure 4-14. Fast Termination Timing

NOTES

When using the fast termination option (cycle length is two clocks), DS is asserted only in a read cycle, not in a write cycle.

DSACKx is only internally asserted for fast termination cycles.

4.3 DATA TRANSFER CYCLES

The transfer of data between the QUICC and other devices involves the following signals:

- Address Bus A31–A0
- Data Bus D31–D0
- Control Signals

The address and data buses are both parallel, nonmultiplexed buses. The bus master moves data on the bus by issuing control signals, and the bus uses a handshake protocol to ensure correct movement of the data. In all bus cycles, the bus master is responsible for deskewing all signals it issues at both the start and end of the cycle. In addition, the bus master is responsible for deskewing the acknowledge and data signals from the slave devices. The following paragraphs define read, write, and read-modify-write cycle operations. Each bus cycle is defined as a succession of states that apply to the bus operation. These states are different from the QUICC states described for the CPU32+. The clock cycles used in the descriptions and timing diagrams of data transfer cycles are independent of the clock frequency. Bus operations are described in terms of external bus states.

4.3.1 Read Cycle

During a read cycle, the QUICC receives data from a memory or peripheral device. If the instruction specifies a long-word operation, the QUICC attempts to read four bytes at once. For a word operation, the QUICC attempts to read two bytes at once. For a byte operation, the QUICC reads one byte. The section of the data bus from which each byte is read depends on the operand size, address signals (A1, A0), and the port size. Refer to 4.2.1 Dynamic Bus Sizing and 4.2.2 Misaligned Operands for more information.

Figure 4-15 shows a long-word read cycle flowchart and Figure 4-16 illustrates a byte read cycle flowchart. Figure 4-17 and Figure 4-18 show functional read cycles timing diagrams specified in terms of clock periods.

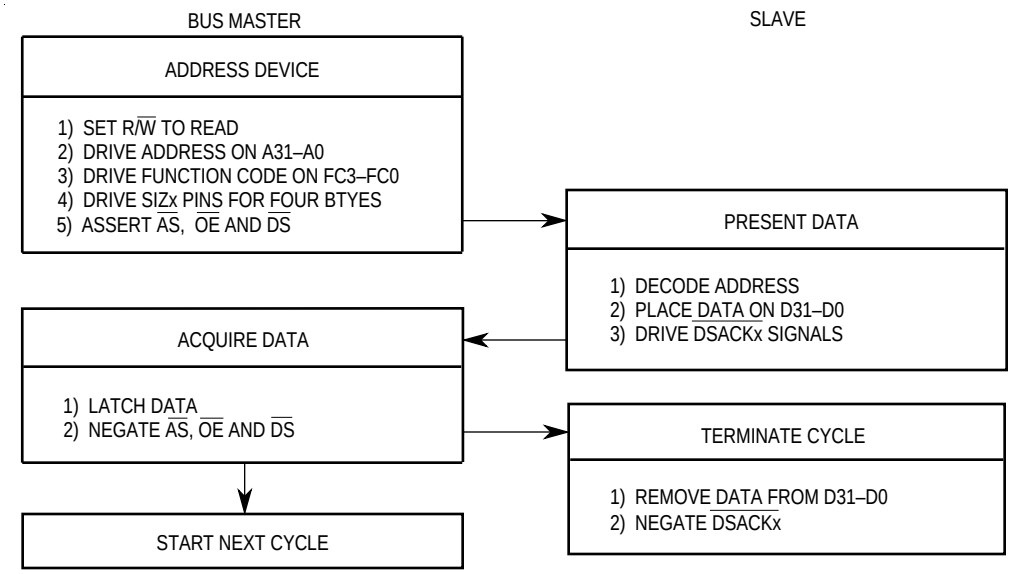


Figure 4-15. Long-Word Read Cycle Flowchart

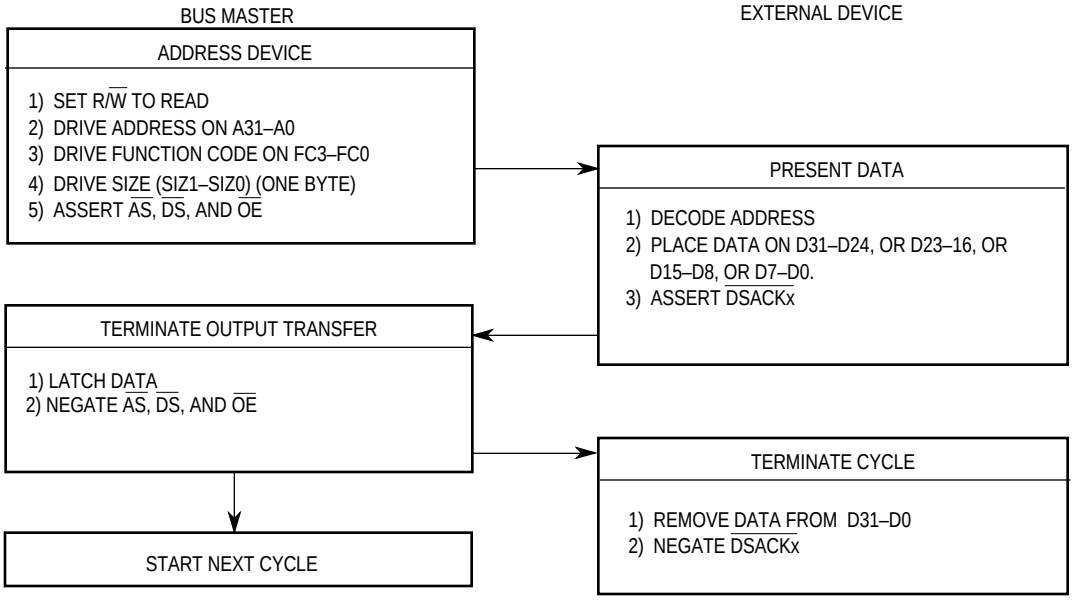


Figure 4-16. Byte Read Cycle Flowchart

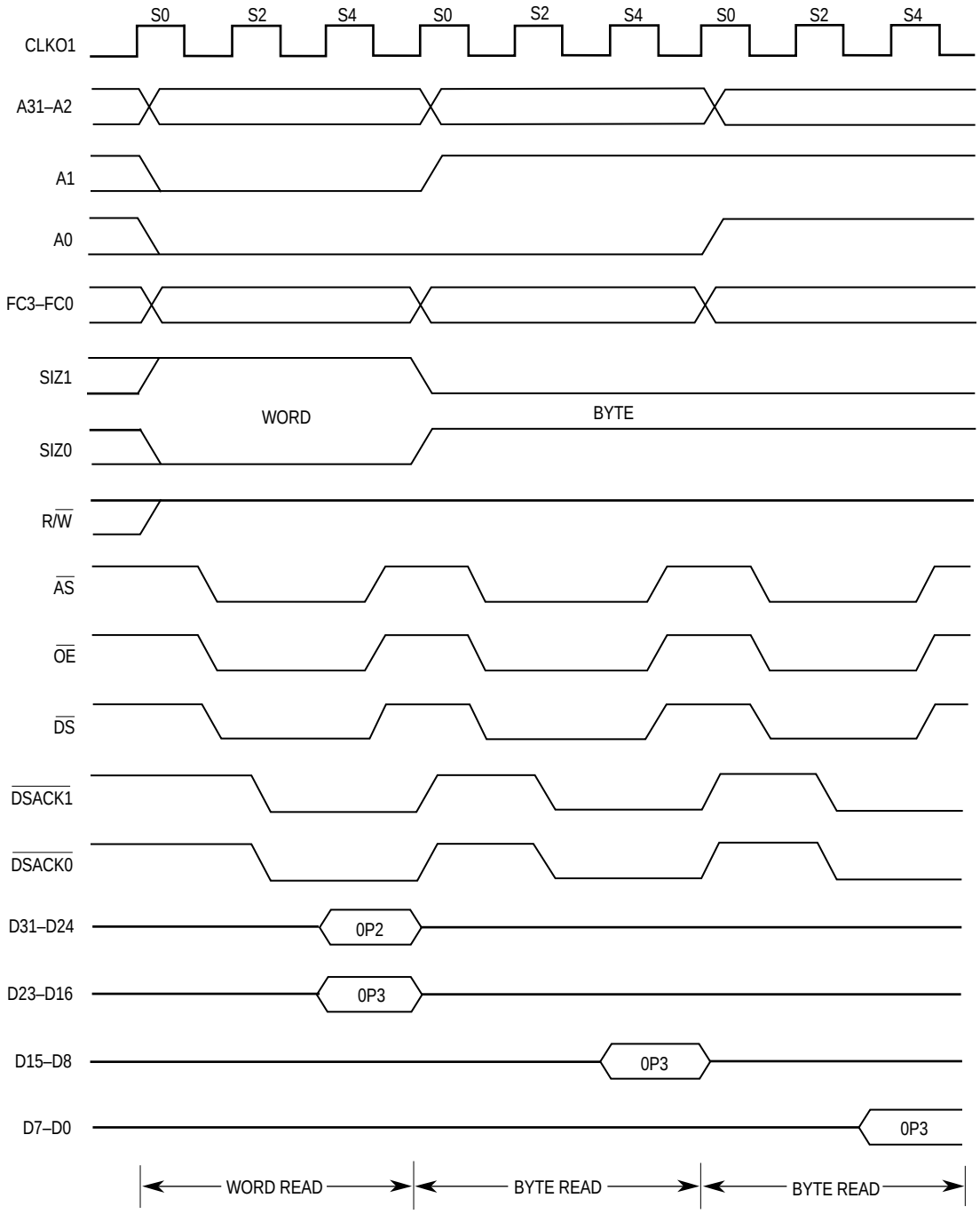


Figure 4-17. Byte and Word Read Cycles—32-Bit Port Timing

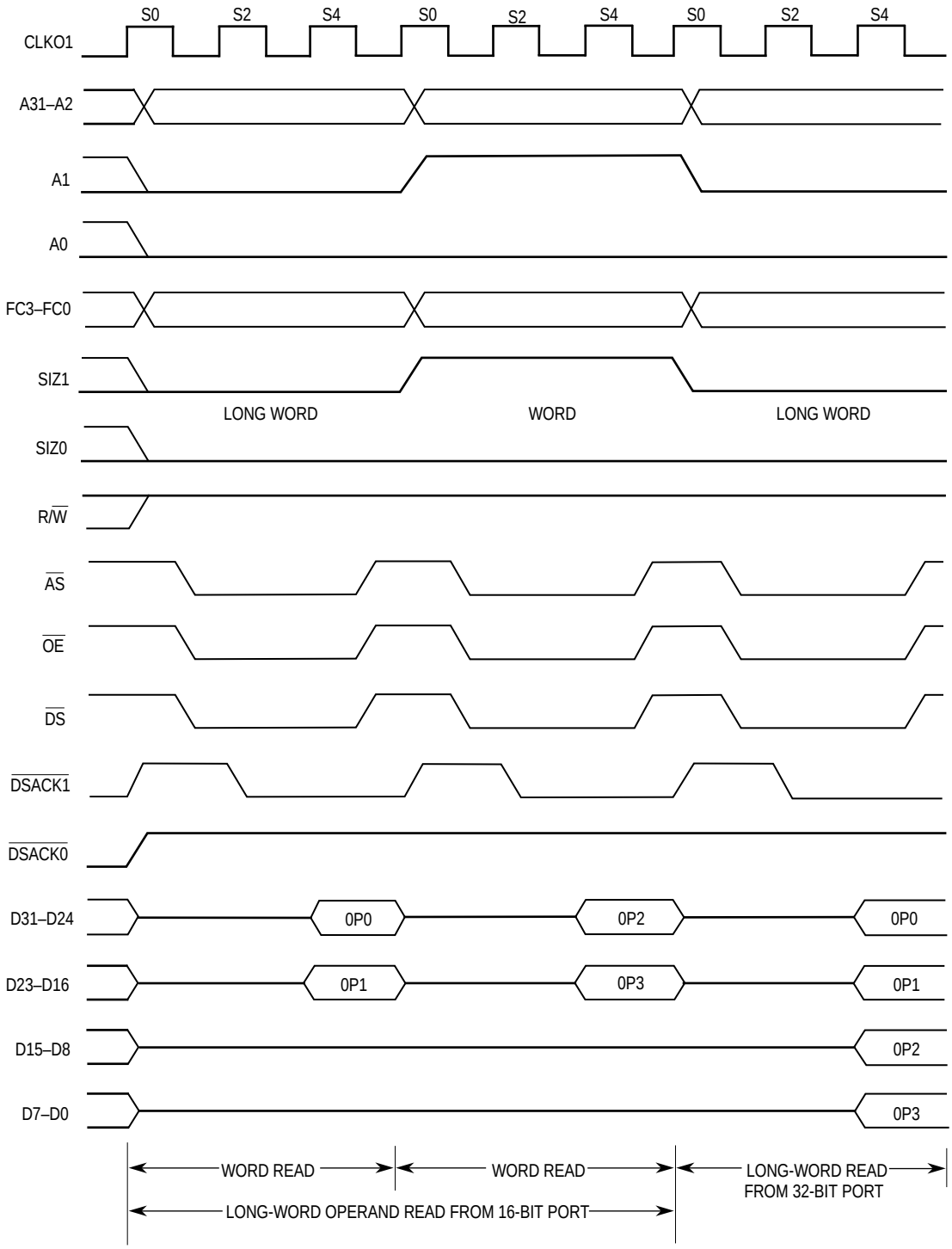


Figure 4-18. Long-Word Read—16-Bit and 32-Bit Port Timing

State 0—The read cycle starts in state 0 (S0). During S0, the QUICC places a valid address on A31–A0 and valid function codes on FC3–FC0. The function codes select the address space for the cycle. The QUICC drives R/$\overline{W}$ high for a read cycle. SIZ1 and SIZ0 become valid, indicating the number of bytes requested for transfer.

State 1—One-half clock later, in state 1 (S1), the QUICC asserts $\overline{AS}$ indicating a valid address on the address bus. The QUICC also asserts $\overline{DS}$ and $\overline{OE}$ during S1. The selected device uses R/$\overline{W}$, SIZ1 or SIZ0, A0, A1, $\overline{DS}$, and $\overline{OE}$ to place its information on the data bus. Any or all of the bytes (D31–D24, D23–D16, D15–D8, and D7–D0) are selected by SIZ1, SIZ0, A1, and A0. Concurrently, the selected device asserts $\overline{DSACKx}$.

State 2—As long as at least one of the $\overline{DSACKx}$ signals is recognized on the falling edge of S2 (meeting the asynchronous input setup time requirement), data is latched on the falling edge of S4, and the cycle terminates.

State 3—If $\overline{DSACKx}$ is not recognized by the start of state 3 (S3), the QUICC inserts wait states instead of proceeding to states 4 and 5. To ensure that wait states are inserted, both $\overline{DSACK1}$ and $\overline{DSACK0}$ must remain negated throughout the asynchronous input setup and hold times around the end of S2. If wait states are added, the QUICC continues to sample $\overline{DSACKx}$ on the falling edges of the clock until one is recognized.

State 4—At the falling edge of state 4 (S4), the QUICC latches the incoming data and samples $\overline{DSACKx}$ to get the port size.

State 5—The QUICC negates $\overline{AS}$, $\overline{DS}$, and $\overline{OE}$ during state 5 (S5). It holds the address valid during S5 to provide address hold time for memory systems. R/$\overline{W}$, SIZ1, SIZ0, and FC3–FC0 also remain valid throughout S5. The external device keeps its data and $\overline{DSACKx}$ signals asserted until it detects the negation of $\overline{AS}$, $\overline{DS}$, or $\overline{OE}$ (whichever it detects first). The device must remove its data and negate $\overline{DSACKx}$ within approximately one clock period after sensing the negation of $\overline{AS}$, $\overline{DS}$, or $\overline{OE}$. $\overline{DSACKx}$ signals that remain asserted beyond this limit may be prematurely detected for the next bus cycle.

4.3.2 Write Cycle

During a write cycle, the QUICC transfers data to memory or a peripheral device. Figure 4-19 is a flowchart of a write cycle operation for a long-word transfer. Figure 4-20 shows the functional write cycle timing diagram specified in clock periods for two write cycles (between two read cycles with no idle time) for a 32-bit port.

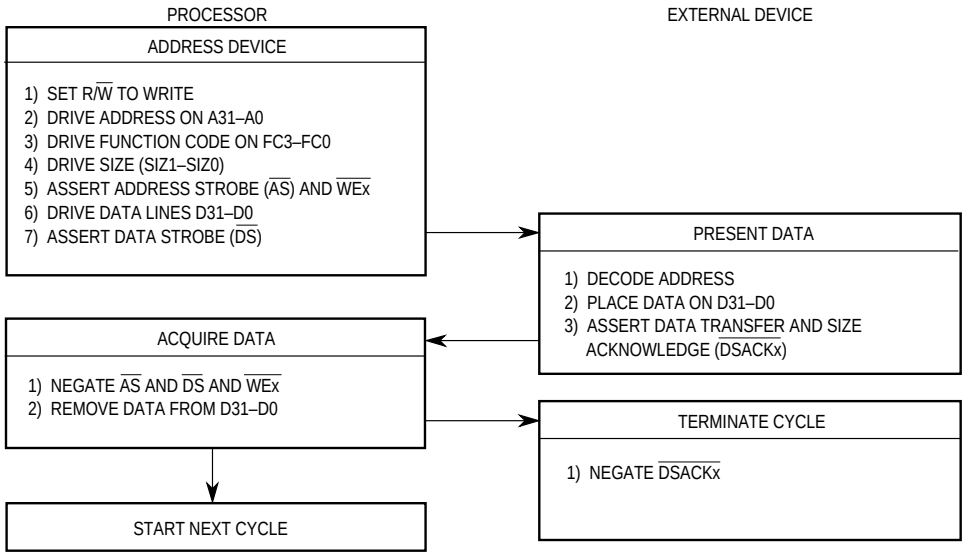
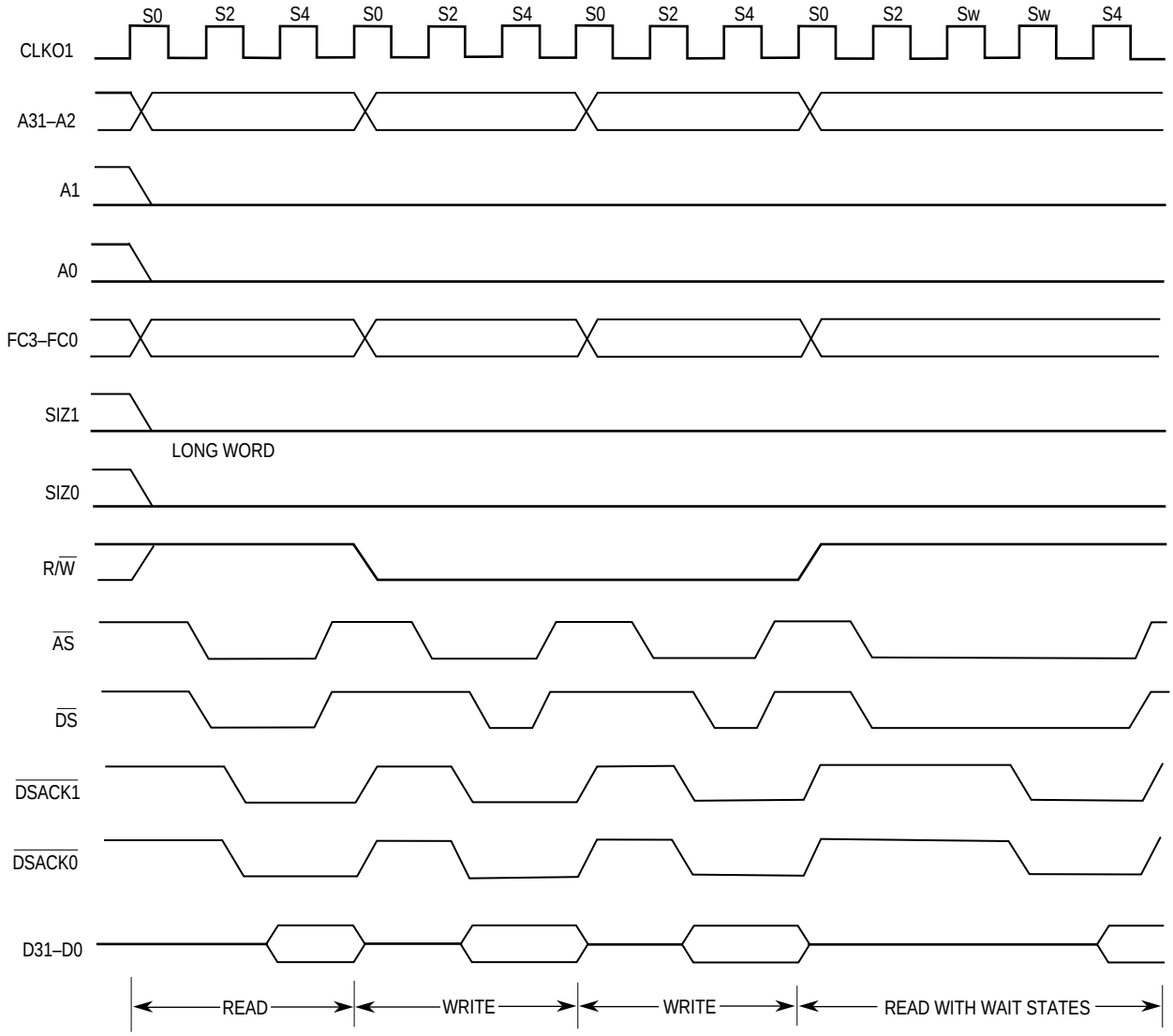


Figure 4-19. Write Cycle Flowchart



NOTE: $\overline{WE3}$–$\overline{WE0}$ is not shown.

Figure 4-20. Read-Write-Read Cycles—32-Bit Port

State 0—The write cycle starts in S0. During S0, the QUICC places a valid address on A31–A0 and valid function codes on FC3–FC0. The function codes select the address space for the cycle. The QUICC drives R/$\overline{W}$ low for a write cycle. SIZ1 and SIZ0 become valid, indicating the number of bytes to be transferred.

State 1—One-half clock later during S1, the QUICC asserts $\overline{AS}$, indicating a valid address on the address bus. During this state, any or all of the byte write enables ($\overline{WE0}$, $\overline{WE1}$, $\overline{WE2}$, and $\overline{WE3}$) are asserted simultaneously with $\overline{AS}$.

State 2—During S2, the QUICC places the data to be written onto D31–D0 and samples $\overline{DSACKx}$ at the end of S2.

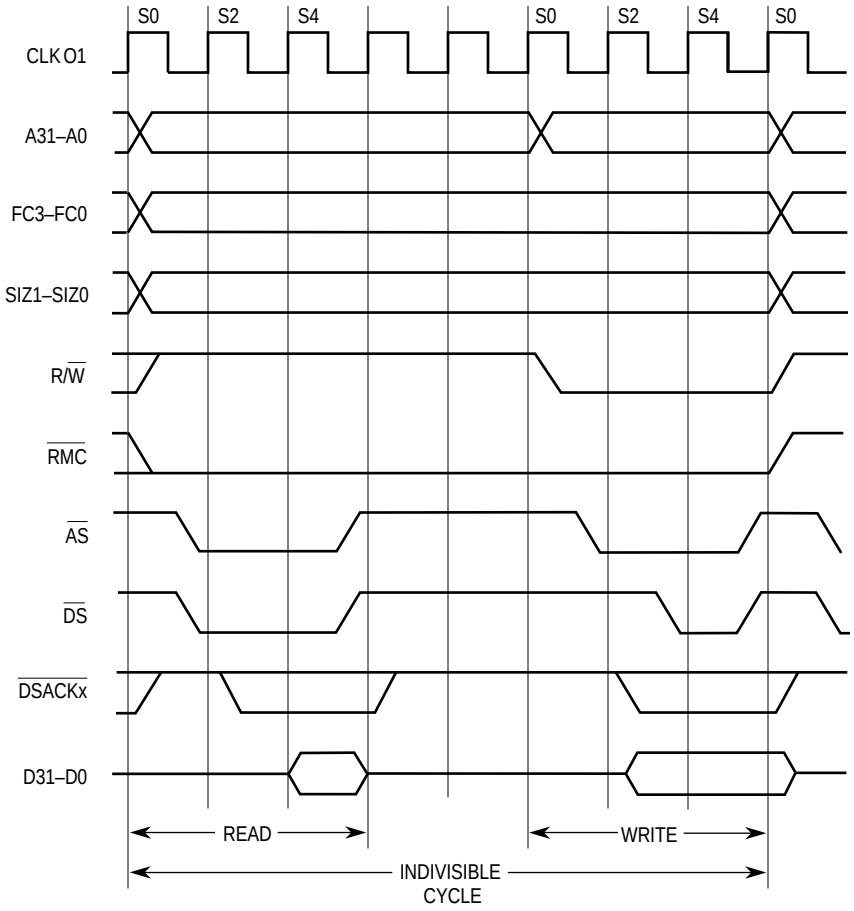
State 3—The QUICC asserts DS during S3, indicating that data is stable on the data bus. As long as at least one of the DSACKx signals is recognized by the end of S2 (meeting the asynchronous input setup time requirement), the cycle terminates one clock later. If DSACKx is not recognized by the start of S3, the QUICC inserts wait states instead of proceeding to S4 and S5. To ensure that wait states are inserted, both DSACK1 and DSACK0 must remain negated throughout the asynchronous input setup and hold times around the end of S2. If wait states are added, the QUICC continues to sample DSACKx on the falling edges of the clock until one is recognized. The selected device uses the four write enables lines or R/W, SIZ1, SIZ0, A1, and A0 to latch data from the appropriate byte(s) of the data bus (D31–D24, D23–D16, D15–D8, and D7–D0). WE3–WE0 or SIZ1, SIZ0, A1, and A0 select the bytes of the data bus. If it has not already done so, the device asserts DSACKx to signal that it has successfully stored the data.

State 4—The QUICC issues no new control signals during S4.

State 5—The QUICC negates WE3–WE0, AS, and DS during S5. It holds the address and data valid during S5 to provide address hold time for memory systems. R/W, SIZ1, SIZ0, and FC3–FC0 also remain valid throughout S5. The external device must keep DSACKx asserted until it detects the negation of AS or DS (whichever it detects first). The device must negate DSACKx within approximately one clock period after sensing the negation of AS or DS. DSACKx signals that remain asserted beyond this limit may be prematurely detected for the next bus cycle.

4.3.3 Read-Modify-Write Cycle

The read-modify-write cycle performs a read, conditionally modifies the data in the arithmetic logic unit, and may write the data out to memory. In the QUICC, this operation is indivisible, providing semaphore capabilities for multiprocessor systems. During the entire read-modify-write sequence, the QUICC asserts RMC to indicate that an indivisible operation is occurring. The QUICC does not issue a bus grant (BG) signal in response to a bus request (BR) signal during this operation. Figure 4-21 is an example of a functional timing diagram of a read-modify-write instruction specified in terms of clock periods.



NOTE: OE and WE3–WE0 are not shown.

Figure 4-21. Read-Modify-Write Cycle Timing

State 0—The QUICC asserts RMC in S0 to identify a read-modify-write cycle. The QUICC places a valid address on A31–A0 and valid function codes on FC3–FC0. The function codes select the address space for the operation. SIZ1 and SIZ0 become valid in S0 to indicate the operand size. The QUICC drives R/W high for the read cycle.

State 1—One-half clock later in S1, the QUICC asserts AS, indicating a valid address on the address bus. The QUICC also asserts OE and DS during S1.

State 2—The selected device uses OE, R/W, SIZ1, SIZ0, A0, and DS to place information on the data bus. Any of the bytes (D31–D24, D23–D16, D15–D8, and D7–D0) are selected by SIZ1, SIZ0, A1, and A0. Concurrently, the selected device may assert DSACKx.

State 3—As long as at least one of the DSACKx signals is recognized by the end of S2 (meeting the asynchronous input setup time requirement), data is latched on the next falling edge of the clock, and the cycle terminates. If DSACKx is not recognized by the start of S3, the QUICC inserts wait states instead of proceeding to S4 and S5. To ensure that wait states are inserted, both DSACK1 and DSACK0 must remain negated throughout the asynchro-

nous input setup and hold times around the end of S2. If wait states are added, the QUICC continues to sample $\overline{\text{DSACKx}}$ on the falling edges of the clock until one is recognized.

State 4—At the end of S4, the QUICC latches the incoming data.

State 5—The QUICC negates $\overline{\text{OE}}$, $\overline{\text{AS}}$, and $\overline{\text{DS}}$ during S5. If more than one read cycle is required to read in the operand(s), S0–S5 are repeated for each read cycle. When finished reading, the QUICC holds the address, R/$\overline{\text{W}}$, and FC3–FC0 valid in preparation for the write portion of the cycle. The external device keeps its data and $\overline{\text{DSACKx}}$ signals asserted until it detects the negation of $\overline{\text{AS}}$ or $\overline{\text{DS}}$ (whichever it detects first). The device must remove the data and negate $\overline{\text{DSACKx}}$ within approximately one clock period after sensing the negation of $\overline{\text{AS}}$ or $\overline{\text{DS}}$. $\overline{\text{DSACKx}}$ signals that remain asserted beyond this limit may be prematurely detected for the next portion of the operation.

Idle States—The QUICC does not assert any new control signals during the idle states, but it may internally begin the modify portion of the cycle at this time. S0–S5 are omitted if no write cycle is required. If a write cycle is required, R/$\overline{\text{W}}$ remains in the read mode until S0 to prevent bus conflicts with the preceding read portion of the cycle; the data bus is not driven until S2.

State 0—The QUICC drives R/$\overline{\text{W}}$ low for a write cycle. Depending on the write operation to be performed, the address lines may change during S0.

State 1—In S1, the QUICC asserts $\overline{\text{AS}}$, indicating a valid address on the address bus. During this state, $\overline{\text{WE0}}$, $\overline{\text{WE1}}$, $\overline{\text{WE2}}$, and/or $\overline{\text{WE3}}$ assert simultaneously with $\overline{\text{AS}}$.

State 2—During S2, the QUICC places the data to be written onto D31–D0.

State 3—The QUICC asserts $\overline{\text{DS}}$ during S3, indicating stable data on the data bus. As long as at least one of the $\overline{\text{DSACKx}}$ signals is recognized by the end of S2 (meeting the asynchronous input setup time requirement), the cycle terminates one clock later. If $\overline{\text{DSACKx}}$ is not recognized by the start of S3, the QUICC inserts wait states instead of proceeding to S4 and S5. To ensure that wait states are inserted, both $\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$ must remain negated throughout the asynchronous input setup and hold times around the end of S2. If wait states are added, the QUICC continues to sample $\overline{\text{DSACKx}}$ on the falling edges of the clock until one is recognized. The selected device uses $\overline{\text{WE3}}$–$\overline{\text{WE0}}$ or R/$\overline{\text{W}}$, $\overline{\text{DS}}$, SIZ1, SIZ0, A1, and A0 to latch data from the appropriate section(s) of the data bus (D31–D24, D23–D16, D15–D8, and D7–D0). $\overline{\text{WE3}}$–$\overline{\text{WE0}}$ or SIZ1, SIZ0, A1, and A0 select the data bus sections. If it has not already done so, the device asserts $\overline{\text{DSACKx}}$ when it has successfully stored the data.

State 4—The QUICC issues no new control signals during S4.

State 5—The QUICC negates $\overline{\text{WE3}}$–$\overline{\text{WE0}}$, $\overline{\text{AS}}$, and $\overline{\text{DS}}$ during S5. It holds the address and data valid during S5 to provide address hold time for memory systems. R/$\overline{\text{W}}$ and FC3–FC0 also remain valid throughout S5. If more than one write cycle is required, S0–S5 are repeated for each write cycle. The external device keeps $\overline{\text{DSACKx}}$ asserted until it detects the negation of $\overline{\text{AS}}$ or $\overline{\text{DS}}$ (whichever it detects first). The device must remove its data and

negate DSACKx within approximately one clock period after sensing the negation of AS or DS.

4.4 CPU SPACE CYCLES

FC2–FC0 select user and supervisor program and data areas. The area selected by function code FC3–FC0 = $7 is classified as the CPU space. The breakpoint acknowledge, LPSTOP broadcast, module base address register access, and interrupt acknowledge cycles described in the following paragraphs use CPU space. The CPU space type, which is encoded on A19–A16 during a CPU space operation, indicates the function that the QUICC is performing. On the QUICC, four of the encodings are implemented as shown in Figure 4-22. All unused values are reserved by Motorola for additional CPU space types.

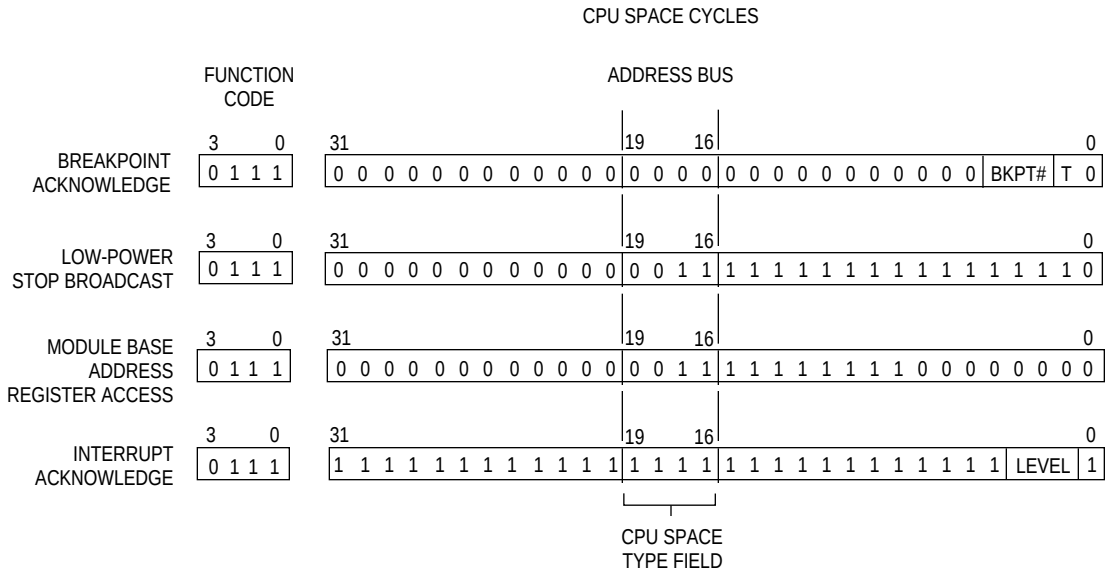


Figure 4-22. CPU Space Address Encoding

4.4.1 Breakpoint Acknowledge Cycle

The breakpoint acknowledge cycle allows external hardware to insert an instruction directly into the instruction pipeline as the program executes. The breakpoint acknowledge cycle is generated by the execution of the BKPT instruction, the internal breakpoint logic, or the assertion of the BKPT pin. The T-bit state (shown in Figure 4-22) differentiates a software breakpoint cycle (T = 0) from a hardware breakpoint cycle (T = 1).

When a software BKPT is executed, the QUICC performs a word read from CPU space, type 0, at an address corresponding to the breakpoint number (bits [2–0] of the BKPT opcode) on A4–A2, and the T-bit (A1) is cleared. If this bus cycle is terminated with BERR (i.e., no instruction word is available), the QUICC then performs illegal instruction exception processing. If the bus cycle is terminated by DSACKx, the QUICC uses the data on the bus to replace the BKPT instruction in the internal instruction pipeline and then begins execution of that instruction.

When the CPU32+ acknowledges hardware breakpoint ($\overline{\text{BKPT}}$ pin assertion or internal breakpoint logic) with background mode disabled, the CPU32+ performs a word read from CPU space, type 0, at an address corresponding to all ones on A4–A2 (BKPT#7), and the T-bit (A1) is set. If this bus cycle is terminated by $\overline{\text{BERR}}$, the QUICC performs hardware breakpoint exception processing. If this bus cycle is terminated by $\overline{\text{DSACK}}$x, the QUICC ignores data on the data bus and continues execution of the next instruction.

NOTE

The $\overline{\text{BKPT}}$ pin is sampled on the same clock phase as data and is latched with data as it enters the CPU32+ pipeline. If $\overline{\text{BKPT}}$ is asserted for only one bus cycle and a pipeline flush occurs before $\overline{\text{BKPT}}$ is detected by the CPU32+, $\overline{\text{BKPT}}$ is ignored. To ensure detection of $\overline{\text{BKPT}}$ by the CPU32+, $\overline{\text{BKPT}}$ can be asserted until a breakpoint acknowledge cycle is recognized.

When the QUICC is configured for a 32-bit bus, the CPU32+ can fetch two instructions simultaneously. Since there is only one $\overline{\text{BKPT}}$ pin, the external user cannot break individually on those instructions, but rather must break on both, causing the BKPT exception to be taken after the first instruction and before the second instruction. The internal breakpoint logic, however, can individually assert a breakpoint for either instruction. (See the BKAR and BKCR discussion in Section 6 System Integration Module (SIM60) for details).

The breakpoint operation flowchart is shown in Figure 4-23. Figure 4-24 and Figure 4-25 show the timing diagrams for the breakpoint acknowledge cycle with instruction opcodes supplied on the cycle and with an exception signaled, respectively.

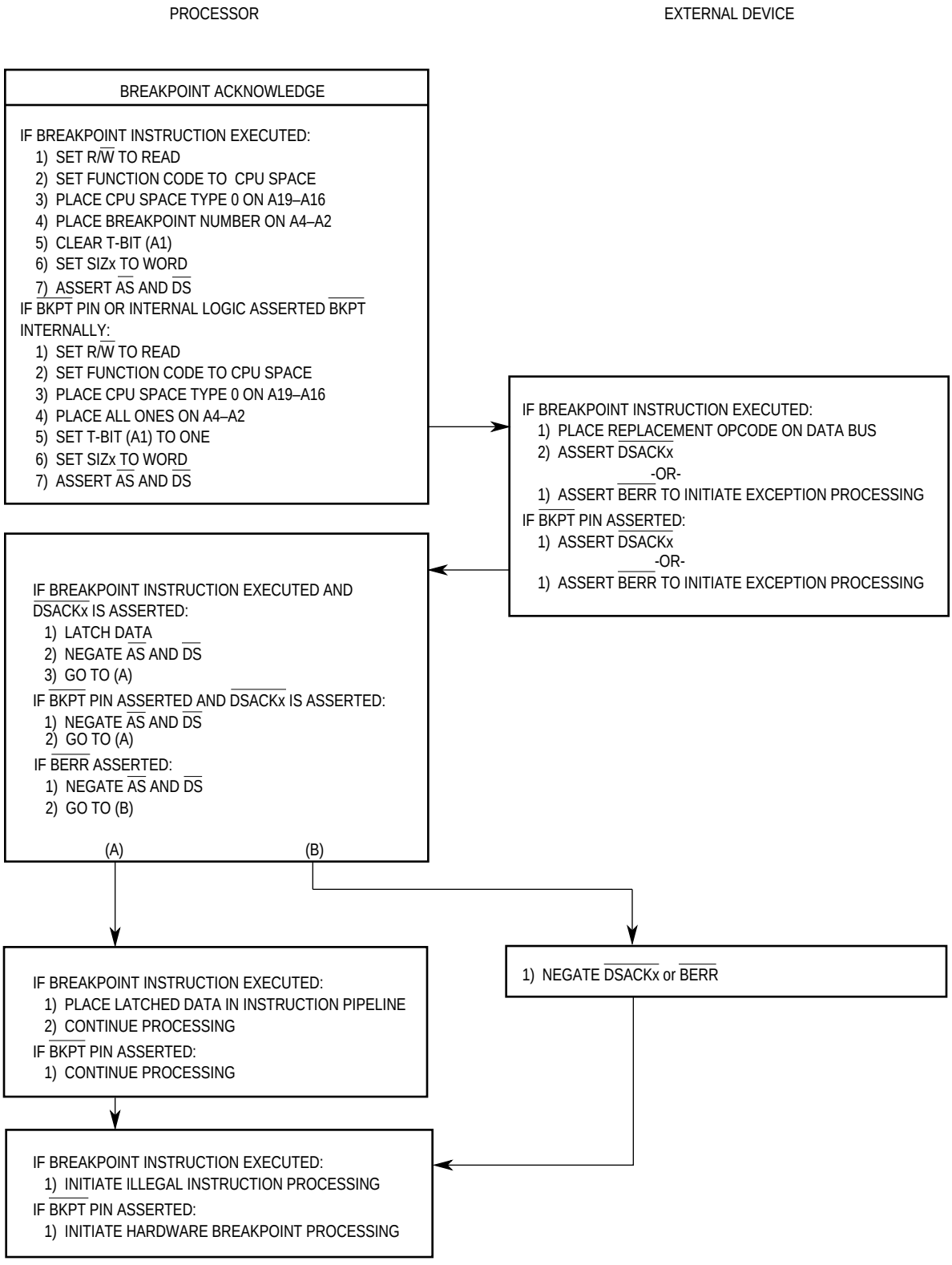


Figure 4-23. Breakpoint Operation Flowchart

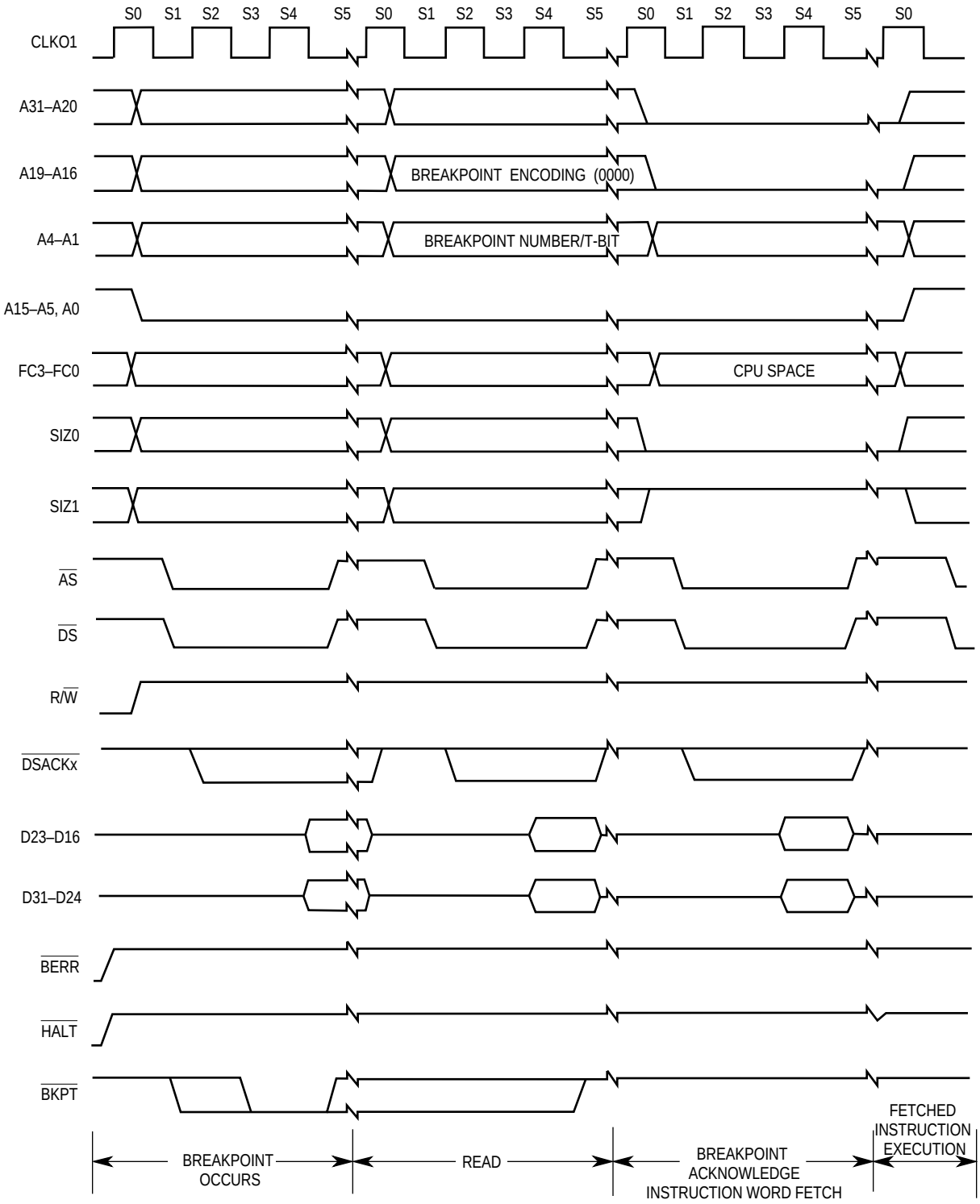


Figure 4-24. Breakpoint Acknowledge Cycle Timing (Opcode Returned)

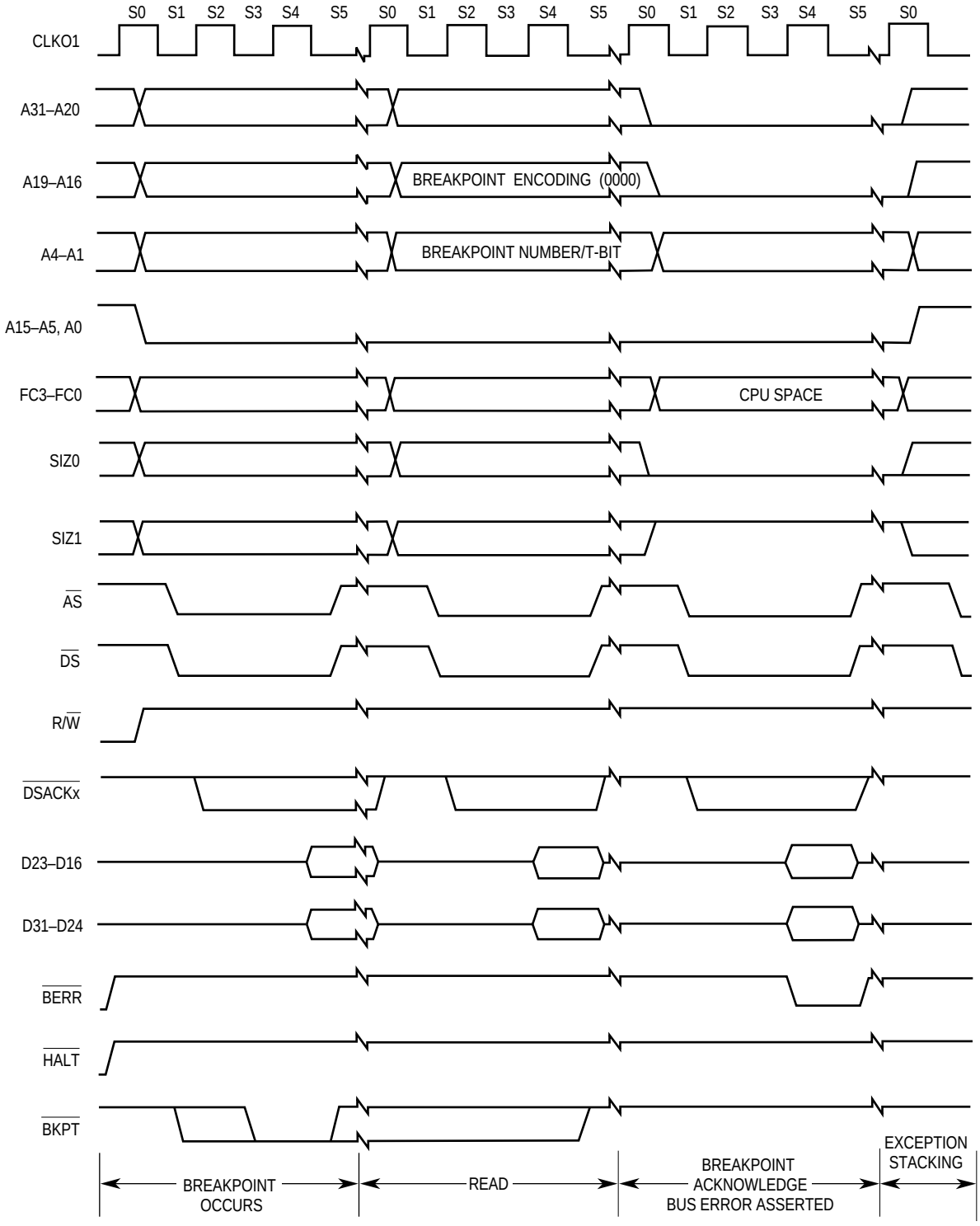


Figure 4-25. Breakpoint Acknowledge Cycle Timing (Exception Signaled)

4.4.2 LPSTOP Broadcast Cycle

The LPSTOP broadcast cycle is generated by the CPU32+ executing the LPSTOP instruction. The external bus interface must get a copy of the interrupt mask level from the CPU32+,

so the CPU32+ performs a CPU space type 3 write with the interrupt mask level (I2–I0) encoded on bits 2–0 of the data bus, as shown in the following figure. The CPU space type 3 cycle waits for the bus to be available, and is shown externally to indicate to external devices that the QUICC is going into LPSTOP mode. If an external device requires additional time to prepare for entry into LPSTOP mode, entry can be delayed by asserting $\overline{\text{HALT}}$. The SIM60 provides internal $\overline{\text{DSACKx}}$ response to this cycle. For more information on how the SIM60 responds to LPSTOP mode, see Section 6 System Integration Module (SIM60) for details.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—													I2	I1	I0

4.4.3 Module Base Address Register (MBAR) Access

All internal module registers, including the SIM60, occupy a single 8-kbyte block that is locatable along 8-kbyte boundaries. The location is fixed by writing the desired base address of the SIM60 block to the MBAR using the MOVES instruction. The MBAR is only accessible in CPU space at address $0003FF00. The SFC or DFC register must indicate CPU space (FC2–FC0 = $7), using the MOVEC instruction, before accessing MBAR. Refer to Section 6 System Integration Module (SIM60) for additional information on the MBAR.

4.4.4 Interrupt Acknowledge Bus Cycles

The CPU32+ makes an interrupt pending in three cases. The first case occurs when a peripheral device signals the CPU32+ (with the $\overline{\text{IRQ7}}$–$\overline{\text{IRQ1}}$ signals) that the device requires service and the internally synchronized value on these signals indicates a higher priority than the interrupt mask in the status register. The second case occurs when a transition has occurred in the case of a level 7 interrupt. A recognized level 7 interrupt must be removed for one clock cycle before a second level 7 can be recognized. The third case occurs if, upon returning from servicing a level 7 interrupt, the request level stays at 7 and the processor mask level changes from 7 to a lower level, a second level 7 is recognized. The CPU32+ takes an interrupt exception for a pending interrupt within one instruction boundary (after processing any other pending exception with a higher priority). The following paragraphs describe the various kinds of interrupt acknowledge bus cycles that can be executed as part of interrupt exception processing.

4.4.4.1 INTERRUPT ACKNOWLEDGE CYCLE—TERMINATED NORMALLY. When the CPU32+ processes an interrupt exception, it performs an interrupt acknowledge cycle to obtain the number of the vector that contains the starting location of the interrupt service routine. Some interrupting devices have programmable vector registers that contain the interrupt vectors for the routines they use. The following paragraphs describe the interrupt acknowledge cycle for these devices. Other interrupting conditions or devices cannot supply a vector number and use the autovector cycle described in 4.4.4.2 Autovector Interrupt Acknowledge Cycle.

The interrupt acknowledge cycle is a read cycle. It differs from the read cycle described in 4.3.1 Read Cycle in that it accesses the CPU address space. Specifically, the differences are as follows:

1. FC3–FC0 are set to $7 (FC3/FC2/FC1/FC0 = 0111) for CPU address space.
2. A3, A2, and A1 are set to the interrupt request level, and the $\overline{\text{IACKx}}$ strobe corresponding to the current interrupt level is asserted. (Either the function codes and address signals or the $\overline{\text{IACKx}}$ strobes can be monitored to determine that an interrupt acknowledge cycle is in progress and the current interrupt level.)
3. The CPU32+ space type field (A19–A16) is set to $F (interrupt acknowledge).
4. Other address signals (A31–A20, A15–A4, and A0) are set to one.

The responding device places the vector number on the data bus during the interrupt acknowledge cycle. Beyond this, the cycle is terminated normally with $\overline{\text{DSACKx}}$.

Figure 4-26 is a flowchart of the interrupt acknowledge cycle; Figure 4-27 shows the timing for an interrupt acknowledge cycle terminated with $\overline{\text{DSACKx}}$.

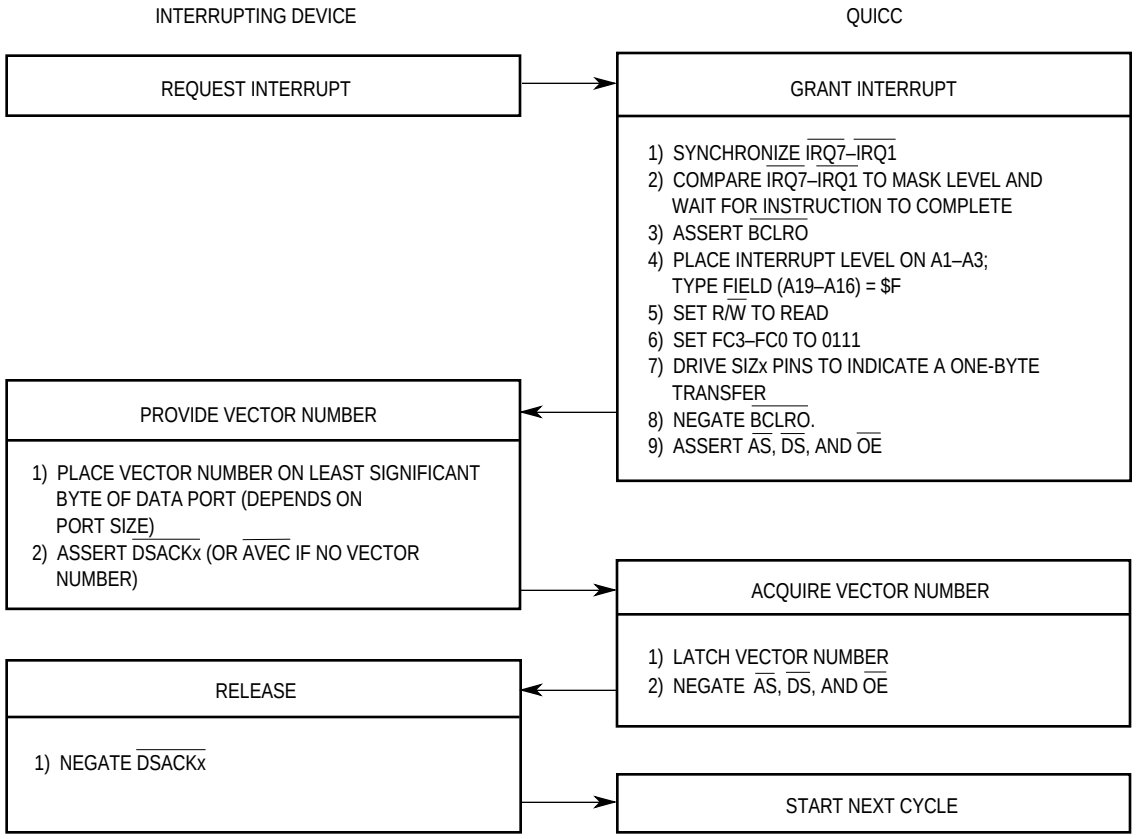
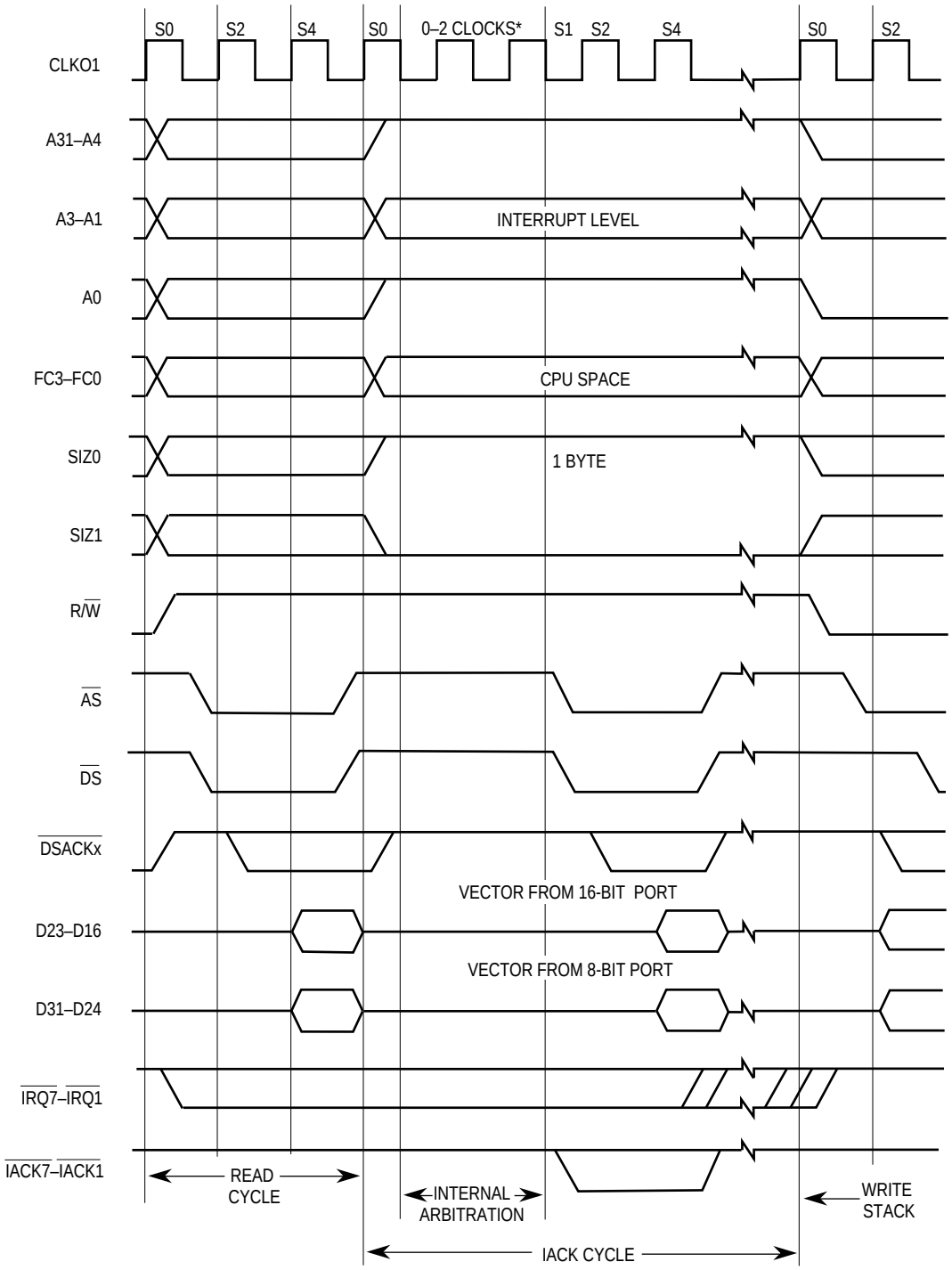


Figure 4-26. Interrupt Acknowledge Cycle Flowchart



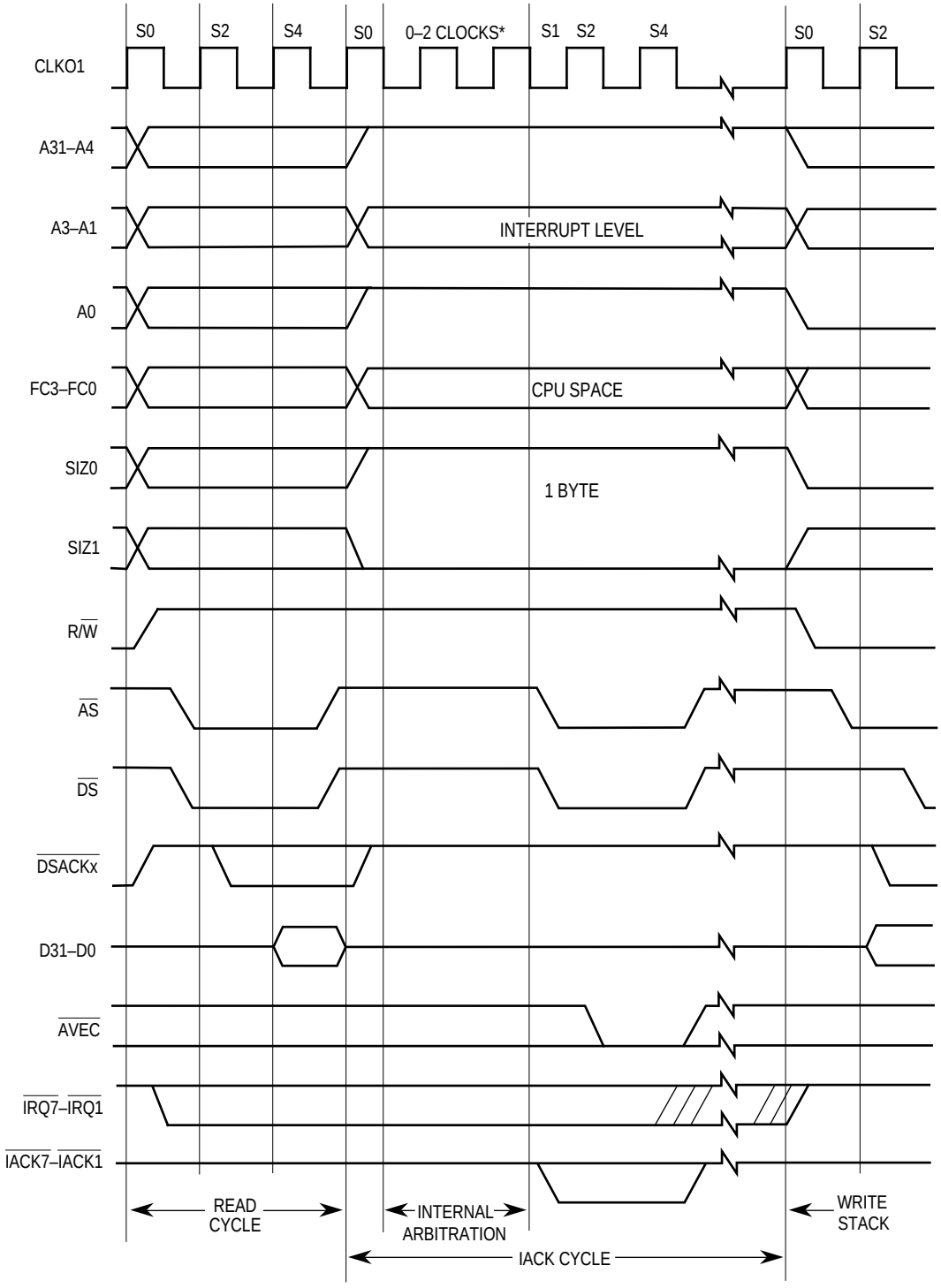
* Internal arbitration may take between 0–2 clock cycles.

Figure 4-27. Interrupt Acknowledge Cycle Timing

4.4.4.2 AUTOVECTOR INTERRUPT ACKNOWLEDGE CYCLE. When the interrupting device cannot supply a vector number, it requests an automatically generated vector (autovector). Instead of placing a vector number on the data bus and asserting $\overline{\text{DSACKx}}$, the device asserts $\overline{\text{AVEC}}$ to terminate the cycle. The $\overline{\text{DSACKx}}$ signals may not be asserted

during an interrupt acknowledge cycle terminated by $\overline{\text{AVEC}}$. The vector number supplied in an autovector operation is derived from the interrupt level of the current interrupt. When the $\overline{\text{AVEC}}$ signal is asserted instead of $\overline{\text{DSACK}}$x during an interrupt acknowledge cycle, the QUICC ignores the state of the data bus and internally generates the vector number (the sum of the interrupt level plus 24 ($18)).

$\overline{\text{AVEC}}$ is multiplexed with $\overline{\text{IACK5}}$. The $\overline{\text{AVEC}}$ bit in the port E pin assignment register (PEPAR) controls whether the $\overline{\text{AVEC}}$/$\overline{\text{IACK5}}$ pin is used as an autovector input or as $\overline{\text{IACK5}}$ (see Section 6 System Integration Module (SIM60) for additional information). $\overline{\text{AVEC}}$ is only sampled during an interrupt acknowledge cycle; during all other cycles, $\overline{\text{AVEC}}$ is ignored. Additionally, $\overline{\text{AVEC}}$ can be internally generated for external devices by programming the autovector register (note that in this case $\overline{\text{AVEC}}$ pin will not be asserted externally). Seven distinct autovectors can be used, corresponding to the seven levels of interrupt available with signals $\overline{\text{IRQ7}}$–$\overline{\text{IRQ1}}$. Figure 4-28 shows the timing for an autovector operation.



* Internal Arbitration may take between 0–2 clock cycles.

Figure 4-28. Autovector Operation Timing

4.4.4.3 SPURIOUS INTERRUPT CYCLE. Requested interrupts, whether internal or external, are arbitrated internally. When no internal module (including the SIM60, which responds for external requests) responds during an interrupt acknowledge cycle by arbitrating for the

interrupt acknowledge cycle internally, the spurious interrupt monitor generates an internal bus error signal to terminate the vector acquisition. The QUICC automatically generates the spurious interrupt vector number, 24, instead of the interrupt vector number in this case. When an external device does not respond to an interrupt acknowledge cycle with AVEC or DSACKx, a bus monitor must assert BERR, which results in the CPU32+ taking the spurious interrupt vector. If HALT is also asserted, the QUICC retries the interrupt acknowledge cycle instead of using the spurious interrupt vector.

4.5 BUS EXCEPTION CONTROL CYCLES

The bus architecture requires assertion of DSACKx from an external device to signal that a bus cycle is complete. Neither DSACKx nor AVEC is asserted in the following cases:

1. DSACKx in fast-termination cycles.
2. AVEC when programmed to respond internally.
3. The external device does not respond.
4. Various other application-dependent errors occur.

The QUICC provides BERR when no device responds by asserting DSACKx/AVEC within an appropriate period of time after the QUICC asserts AS. This mechanism allows the cycle to terminate and the QUICC to enter exception processing for the error condition. HALT is also used for bus exception control. This signal can be asserted by an external device for debugging purposes to cause single bus cycle operation or, in combination with BERR, a retry of a bus cycle in error. To properly control termination of a bus cycle for a retry or a bus error condition, DSACKx, BERR, and HALT can be asserted and negated with the rising edge of the QUICC clock. This assures that when two signals are asserted simultaneously, the required setup and hold time for both is met for the same falling edge of the QUICC clock. This or an equivalent precaution should be designed into the external circuitry to provide these signals. Alternatively, the internal bus monitor could be used. The acceptable bus cycle terminations for asynchronous cycles are summarized in relation to DSACKx assertion as follows (case numbers refer to Table 4-8):

1. Normal Termination: DSACKx is asserted; BERR and HALT remain negated (case 1).
2. Halt Termination: HALT is asserted at the same time or before DSACKx, and BERR remains negated (case 2).
3. Bus Error Termination: BERR is asserted in lieu of, at the same time, or before DSACKx (case 3) or after DSACKx (case 4), and HALT remains negated; BERR is negated at the same time or after DSACKx.
4. Retry Termination: HALT and BERR are asserted in lieu of, at the same time, or before DSACKx (case 5) or after DSACKx (case 6); BERR is negated at the same time or after DSACKx, and HALT may be negated at the same time or after BERR.

Table 4-8 shows various combinations of control signal sequences and the resulting bus cycle terminations. To ensure predictable operation, BERR and HALT should be negated according to the specifications in Section 10 Electrical Characteristics. DSACKx, BERR, and HALT may be negated after AS. If DSACKx or BERR remain asserted into S2 of the next bus cycle, that cycle may be terminated prematurely.

EXAMPLE A: A system uses a bus monitor timer to terminate accesses to an unpopulated address space. The timer asserts $\overline{\text{BERR}}$ after timeout (case 3).

EXAMPLE B: A system uses error detection and correction on RAM contents. The designer may:

1. Delay $\overline{\text{DSACKx}}$ until data is verified and assert $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ simultaneously to indicate to the QUICC to automatically retry the error cycle (case 5), or, if data is valid, assert $\overline{\text{DSACKx}}$ (case 1).
2. Delay $\overline{\text{DSACKx}}$ until data is verified and assert $\overline{\text{BERR}}$ with or without $\overline{\text{DSACKx}}$ if data is in error (case 3). This initiates exception processing for software handling of the condition.
3. Return $\overline{\text{DSACKx}}$ prior to data verification; if data is invalid, $\overline{\text{BERR}}$ is asserted on the next clock cycle (case 4). This initiates exception processing for software handling of the condition.
4. Return $\overline{\text{DSACKx}}$ prior to data verification; if data is invalid, assert $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ on the next clock cycle (case 6). The memory controller can then correct the RAM prior to or during the automatic retry.

Table 4-8. $\overline{\text{DSACKx}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ Assertion Results

Case Num	Control Signal	Asserted on Rising Edge of State		Result
		N	N + 2	
1	$\overline{\text{DSACKx}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA NA	S NA X	Normal cycle terminate and continue.
2	$\overline{\text{DSACKx}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA A/S	S NA S	Normal cycle terminate and halt; continue when $\overline{\text{HALT}}$ negated.
3	$\overline{\text{DSACKx}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	NA/A A NA	X S NA	Terminate and take bus error exception, possibly deferred.
4	$\overline{\text{DSACKx}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA NA	X A NA	Terminate and take bus error exception, possibly deferred.
5	$\overline{\text{DSACKx}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	NA/A A A/S	X S S	Terminate and retry when $\overline{\text{HALT}}$ negated.
6	$\overline{\text{DSACKx}}$ $\overline{\text{BERR}}$ $\overline{\text{HALT}}$	A NA NA	X A A	Terminate and retry when $\overline{\text{HALT}}$ negated.

NOTES:
N —The number of current even bus state (e.g., S2, S4, etc.)
A —Signal is asserted in this bus state
NA —Signal is not asserted in this state
X —Don't care
S —Signal was asserted in previous state and remains asserted in this state

4.5.1 Bus Errors

$\overline{\text{BERR}}$ can be used to abort the bus cycle and the instruction being executed. $\overline{\text{BERR}}$ takes precedence over $\overline{\text{DSACKx}}$ provided it meets the timing constraints described in Section 10 Electrical Characteristics. If $\overline{\text{BERR}}$ does not meet these constraints, it may cause unpredict-

able operation of the QUICC. If BERR remains asserted into the next bus cycle, it may cause incorrect operation of that cycle. When BERR is issued to terminate a bus cycle, the QUICC may enter exception processing immediately following the bus cycle, or it may defer processing the exception.

The instruction prefetch mechanism requests instruction words from the bus controller before it is ready to execute them. If a bus error occurs on an instruction fetch, the QUICC does not take the exception until it attempts to use that instruction word. Should an intervening instruction cause a branch or should a task switch occur, the bus error exception does not occur. The bus error condition is recognized during a bus cycle in any of the following cases:

1. DSACKx and HALT are negated, and BERR is asserted.
2. HALT and BERR are negated, and DSACKx is asserted. BERR is then asserted within one clock cycle (HALT remains negated).

When the QUICC recognizes a bus error condition, it terminates the current bus cycle in the normal way. Figure 4-29 shows the timing of a bus error for the case in which DSACKx is not asserted. Figure 4-30 shows the timing for a bus error that is asserted after DSACKx. Exceptions are taken in both cases. (Refer to Section 5 CPU32+ for details of bus error exception processing.)

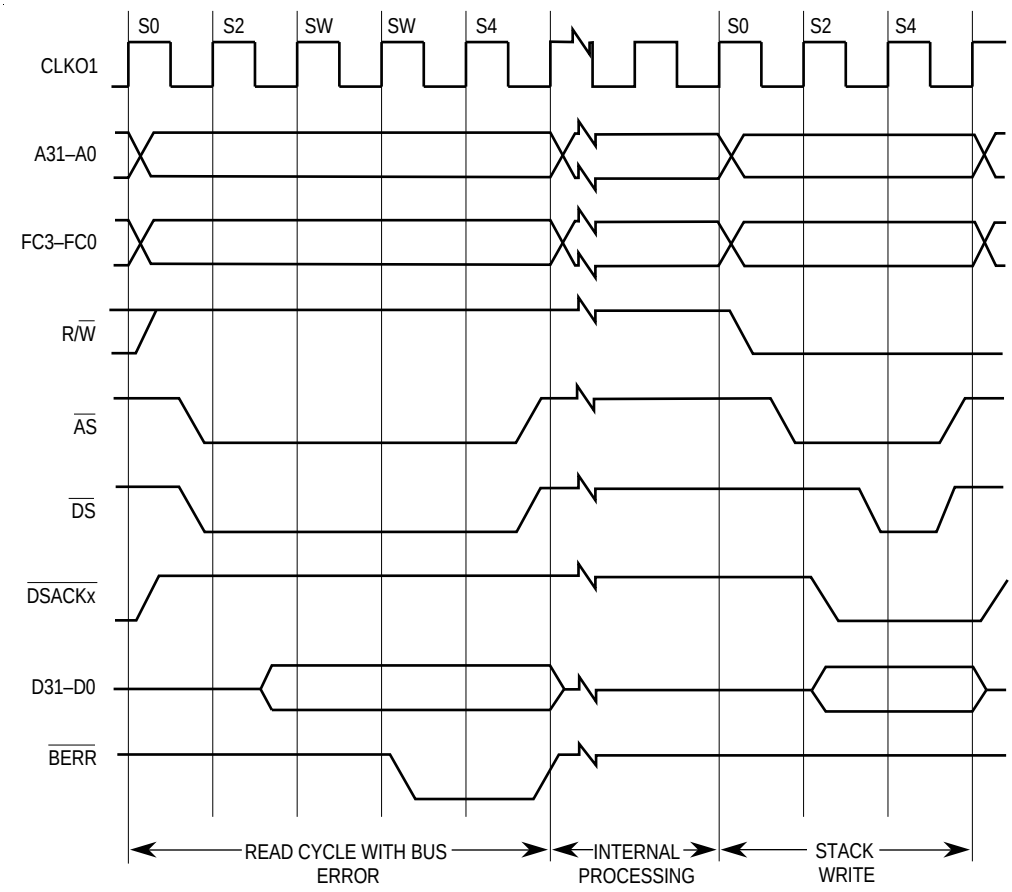


Figure 4-29. Bus Error without DSACKx

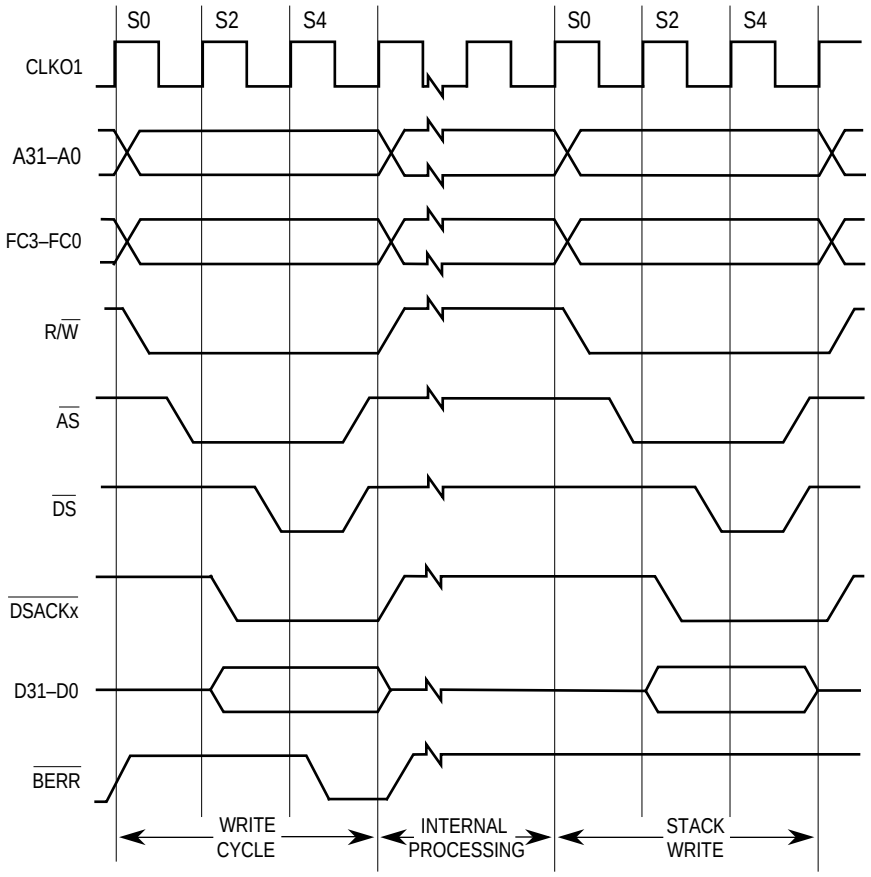


Figure 4-30. Late Bus Error with DSACKx

In the second case, in which BERR is asserted after DSACKx is asserted, BERR must be asserted within the time specified for purely asynchronous operation, or it must be asserted and remain stable during the sample window around the next falling edge of the clock after DSACKx is recognized. If BERR is not stable at this time, the QUICC may exhibit erratic behavior. BERR has priority over DSACKx. In this case, data may be present on the bus but may not be valid. This sequence can be used by systems that have memory error detection and correction logic and by external cache memories.

4.5.2 Retry Operation

When both BERR and HALT are asserted by an external device during a bus cycle, the QUICC enters the retry sequence shown in Figure 4-31. A delayed retry, which is similar to the delayed bus error signal described previously, can also occur (see Figure 4-32). The QUICC terminates the bus cycle, places the control signals in their inactive state, and does not begin another bus cycle until the BERR and HALT signals are negated by external logic. After a synchronization delay, the QUICC retries the previous cycle using the same access information (address, function code, size, etc.). BERR should be negated before S2 of the retried cycle to ensure correct operation of the retried cycle.

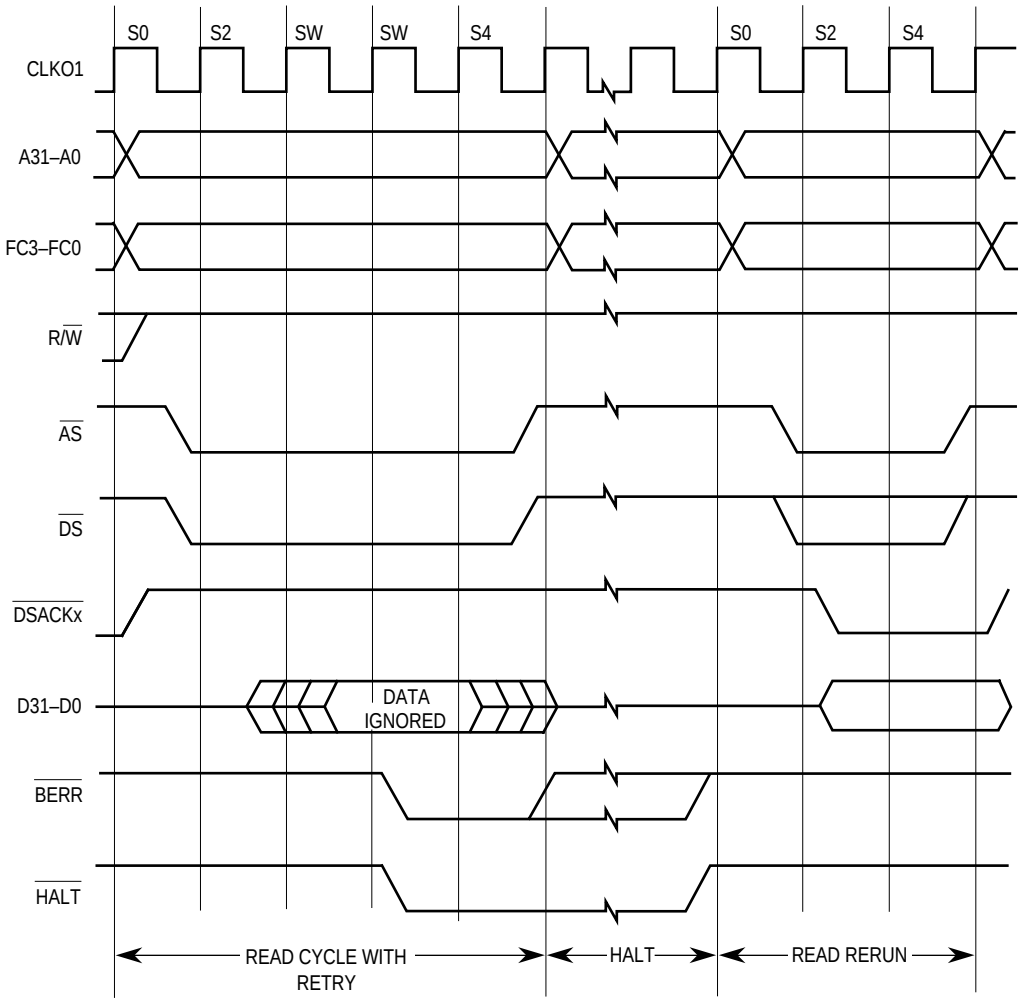


Figure 4-31. Retry Sequence

The QUICC retries any read or write cycle of a read-modify-write operation separately; RMC remains asserted during the entire retry sequence.

Asserting BR at the same time as BERR and HALT provides a relinquish and retry operation. The QUICC does not relinquish the bus during a read-modify-write cycle, but may relinquish the bus between any other bus cycles. (i.e. relinquish-and-retry has priority over bus coherency, except in the case of read-modify-write cycles). Any device that requires the QUICC to give up the bus and retry a bus cycle during a read-modify-write cycle must assert BERR and BR only (HALT must not be included). The bus error handler software should examine the read-modify-write bit in the special status word (refer to Section 5 CPU32+) and take the appropriate action to resolve this type of fault when it occurs.

NOTE

When the relinquish and retry is asserted during an internal master's word access to an 8-bit port, and the external master that takes the bus performs an external-to-internal bus cycle, the en-

tire word access will be retried. This is true even if the relinquish and retry was asserted on the second access and the first 8-bit access was completed normally.

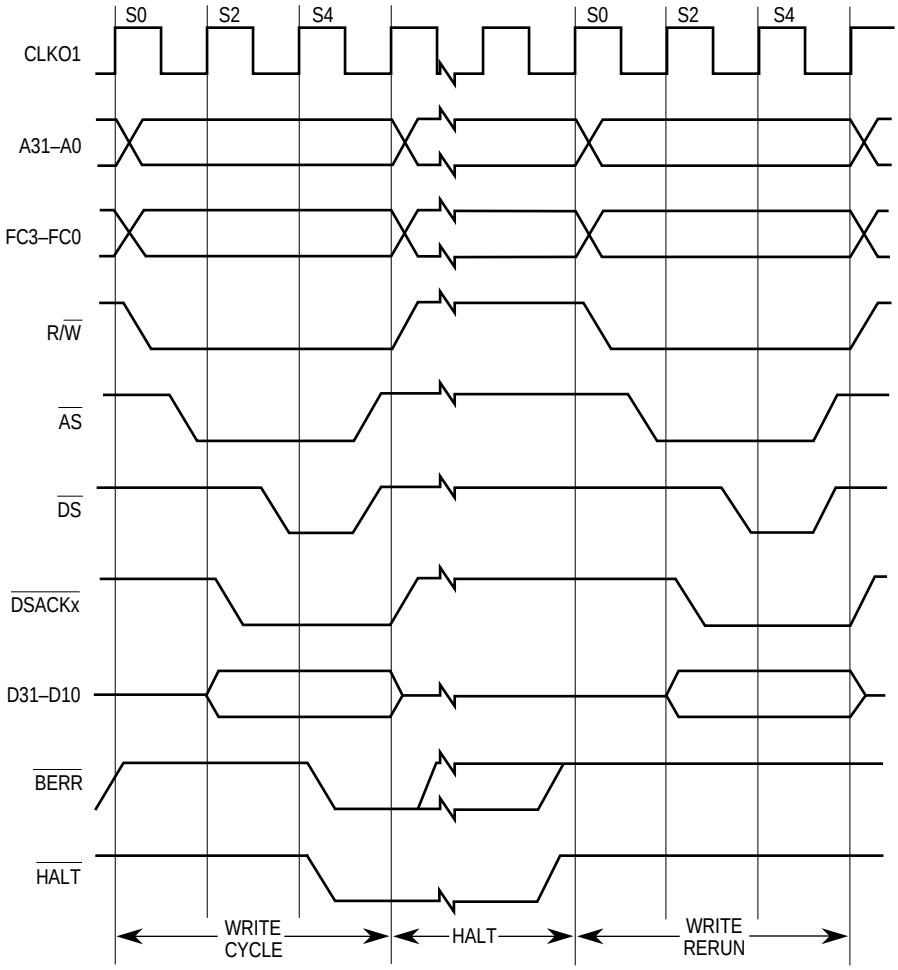


Figure 4-32. Late Retry Sequence

4.5.3 Halt Operation

When HALT is asserted and BERR is not asserted, the QUICC halts external bus activity at the next bus cycle boundary (see Figure 4-33). HALT by itself does not terminate a bus cycle. HALT affects external bus cycles only; thus, a program that does not require use of the external bus may continue executing until it requires use of the external bus.

Negating and reasserting HALT in accordance with the correct timing requirements provides a single step (bus cycle to bus cycle) operation. The single-cycle mode allows the user to proceed through (and debug) external QUICC operations, one bus cycle at a time. Since the occurrence of a bus error while HALT is asserted causes a retry operation, the user must anticipate retry cycles while debugging in the single-cycle mode. The single-step operation and the software trace capability allow the system debugger to trace single bus cycles, single instructions, or changes in program flow.

When the QUICC completes a bus cycle with $\overline{\text{HALT}}$ asserted, D31–D0 is placed in the high-impedance state, and bus control signals are driven inactive (not high-impedance state); the address, function code, size, and read/write signals remain in the same state. The halt operation has no effect on bus arbitration (refer to 4.6 Bus Arbitration). When bus arbitration occurs while the QUICC is halted, the address and control signals are also placed in the high-impedance state. Once bus mastership is returned to the QUICC, if $\overline{\text{HALT}}$ is still asserted, the address, function code, size, and read/write signals are again driven to their previous states. The QUICC does not service interrupt requests while it is halted.

NOTES

In Figure 4-33, note that $\overline{\text{BR}}$ is not asserted until after the halt operation is complete. If $\overline{\text{BR}}$ is asserted at the same time as $\overline{\text{HALT}}$, the user should note that the $\overline{\text{BG}}$ signal may not be asserted immediately (as in other M68000 family devices) but rather after the full operand transfer is complete. This difference in behavior is due to the coherency rules imposed by the QUICC and other IMB-based M68300 family members. Refer to 4.6 Bus Arbitration for more details. To override the coherency rules, a relinquish and retry cycle may be used.

In the MCR of the SIM60, if the show cycles enable bits SHEN1-SHEN0 = 1x to enable show cycles mode, and $\overline{\text{HALT}}$ is asserted externally, the following behavior is possible. It is possible that the QUICC may not show the last bus cycle externally, if that bus cycle happens to be an internal-to-internal bus cycle. This is due to a pipelining characteristic of the QUICC coupled with the $\overline{\text{HALT}}$ signal being asserted late into an internal-to-external bus cycle. Note that show cycles mode is not the normal configuration for the QUICC.

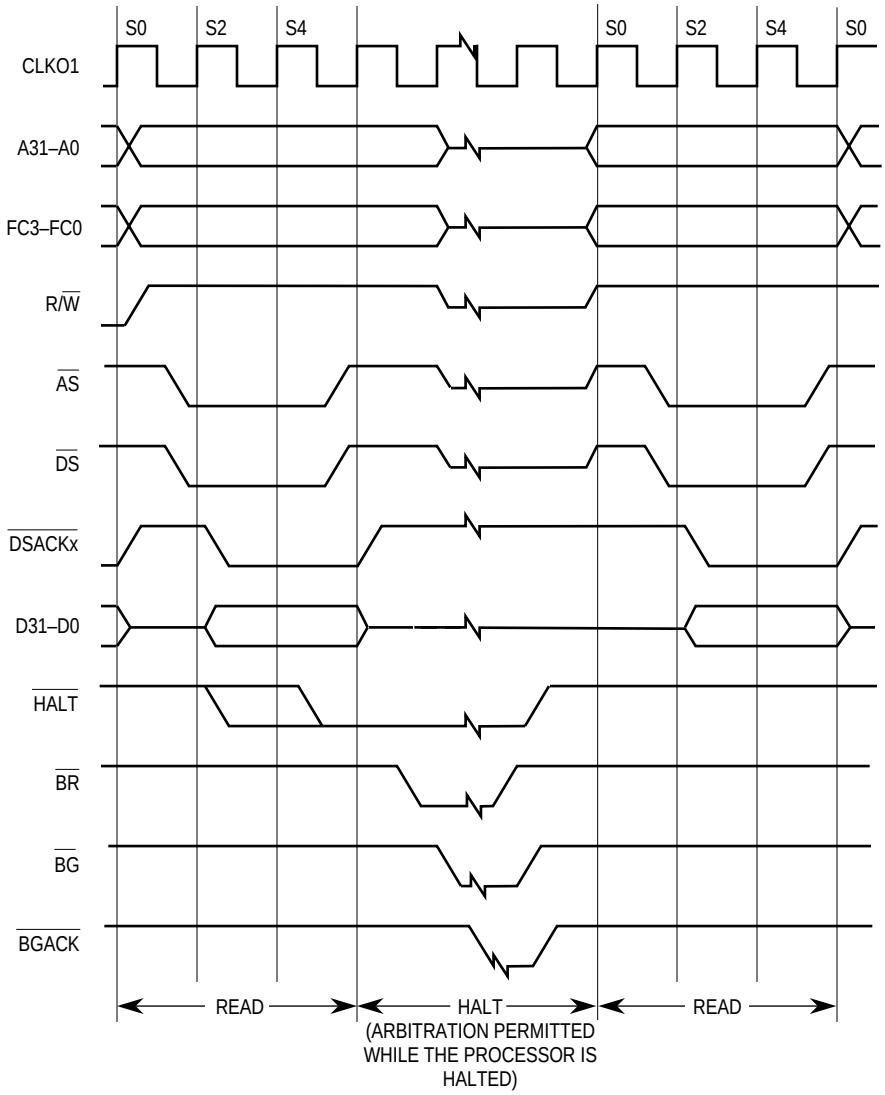


Figure 4-33. $\overline{\text{HALT}}$ Timing

4.5.4 Double Bus Fault

A double bus fault results when a bus error or an address error occurs during the exception processing sequence for any of the following:

1. A previous bus error
2. A previous address error
3. A reset

For example, the QUICC attempts to stack several words containing information about the state of the machine while processing a bus error exception. If a bus error exception occurs during the stacking operation, the second error is considered a double bus fault. When a double bus fault occurs, the QUICC halts and drives the $\overline{\text{HALT}}$ line low. Only a reset operation can restart a halted QUICC. However, bus arbitration can still occur (refer to 4.6 Bus Arbitration). A second bus error or address error that occurs after exception processing has

completed (during execution of the exception handler routine or later) does not cause a double bus fault. A bus cycle that is retried does not constitute a bus error or contribute to a double bus fault. The QUICC continues to retry the same bus cycle as long as the external hardware requests it.

Reset can also be generated internally by the halt monitor (see Section 5 CPU32+).

4.6 BUS ARBITRATION

The bus design of the QUICC provides for a single bus master at any one time, either the QUICC or an external device. One or more of the external devices on the bus can have the capability of becoming bus master for the external bus and the QUICC internal bus. Bus arbitration is the protocol by which an external device becomes bus master; the bus controller in the QUICC manages the bus arbitration signals so that the QUICC has the lowest priority.

NOTE

The QUICC may assert the $\overline{\text{BCLRO}}$ signal for one or more of its internal bus masters, IDMA, SDMA, or DRAM refresh cycle, or when an interrupt request is pending on a level that is greater than a programmable level. The user can use $\overline{\text{BCLRO}}$ to negate the $\overline{\text{BR}}$ line asserted by an external master to reduce the interrupt latency for programmable interrupt levels and to increase the QUICC internal master arbitration priority over external masters.

External devices that need to obtain the bus must assert the bus arbitration signals in the sequences described in the following paragraphs. Systems that include several devices that can become bus master require external circuitry to assign priorities to the devices, so that when two or more external devices attempt to become bus master at the same time, the one having the highest priority becomes bus master first. The sequence of the protocol is as follows:

1. An external device asserts $\overline{\text{BR}}$.
2. The QUICC asserts $\overline{\text{BG}}$ to indicate that the bus is available.
3. The external device asserts $\overline{\text{BGACK}}$ to indicate that it has assumed bus mastership.

$\overline{\text{BR}}$ may be issued any time during a bus cycle or between cycles. $\overline{\text{BG}}$ is asserted in response to $\overline{\text{BR}}$. To guarantee operand coherency, $\overline{\text{BG}}$ is only asserted at the end of an operand transfer. (For example if any internal master such as the CPU, SDMA or IDMA on the QUICC is writing a 32-bit operand to an 8-bit port size, $\overline{\text{BG}}$ is not asserted until the fourth byte is written.) Additionally, $\overline{\text{BG}}$ is not asserted until the end of a read-modify-write operation (when $\overline{\text{RMC}}$ is negated) in response to a $\overline{\text{BR}}$ signal. When the requesting device receives $\overline{\text{BG}}$ and more than one external device can be bus master, the requesting device should begin whatever arbitration is required. When it assumes bus mastership, the external device asserts $\overline{\text{BGACK}}$ and maintains $\overline{\text{BGACK}}$ during the entire bus cycle (or cycles) for which it is bus master. The following conditions must be met for an external device to assume mastership of the bus through the normal bus arbitration procedure: it must have

received $\overline{BG}$ through the arbitration process, and $\overline{BGACK}$ must be inactive, indicating that no other bus master has claimed ownership of the bus.

Figure 4-34 is a flowchart showing the detail involved in bus arbitration for a single device. This technique allows processing of bus requests during data transfer cycles.

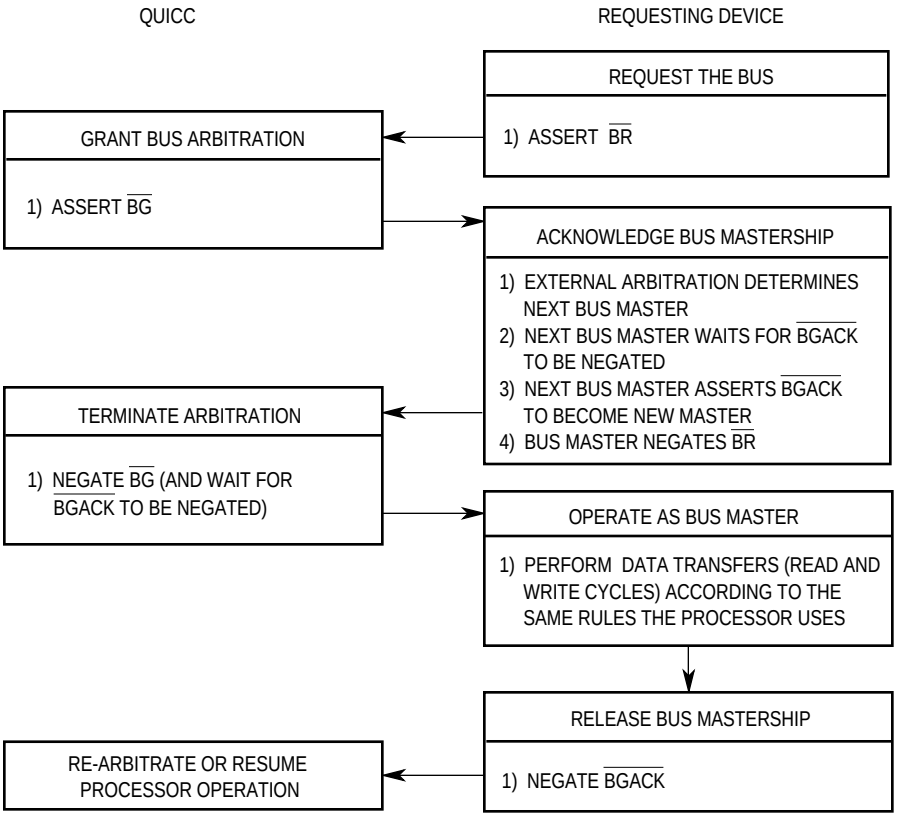
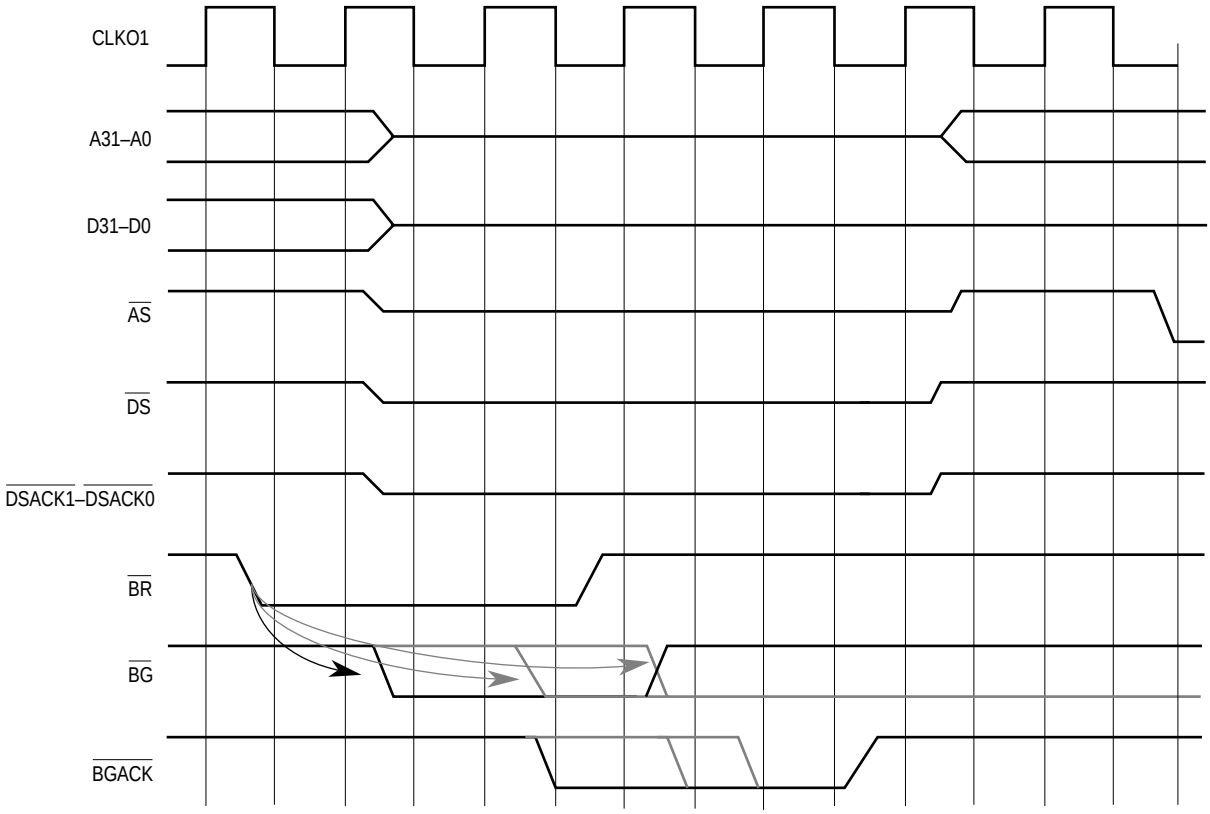


Figure 4-34. Bus Arbitration Flowchart for Single Request

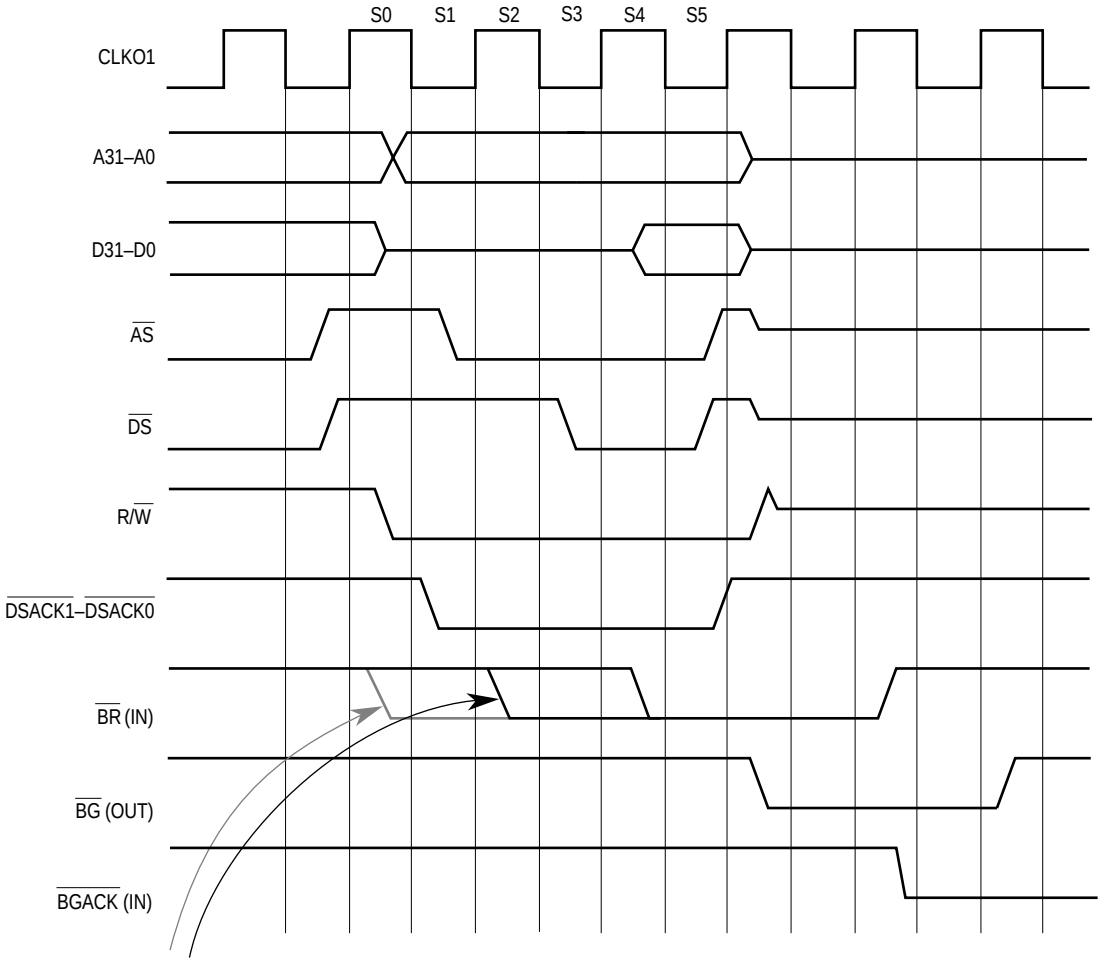
The QUICC has a synchronous arbitration timing mode to reduce the $\overline{BR}$ to $\overline{BG}$ delay to one clock in the idle bus case (see Figure 4-35). Figure 4-36 illustrates the active bus case.

$\overline{BR}$ is negated at the time that $\overline{BGACK}$ is asserted. This type of operation applies to a system consisting of the QUICC and one device capable of bus mastership. In a system having a number of devices capable of bus mastership, $\overline{BR}$ from each device can be wire-ORed to the QUICC. In such a system, more than one bus request could be asserted simultaneously. $\overline{BG}$ is negated a few clock cycles after the transition of $\overline{BGACK}$. However, if bus requests are still pending after the negation of $\overline{BG}$, the QUICC asserts another $\overline{BG}$ within a few clock cycles after it was negated. This additional assertion of $\overline{BG}$ allows external arbitration circuitry to select the next bus master before the current bus master has finished using the bus. The following paragraphs provide additional information about the three steps in the arbitration process. Bus arbitration requests are recognized during normal processing, $\overline{HALT}$ assertion, and when the CPU32+ has halted due to a double bus fault.



NOTE:
- BR has synchronous timing.
- BR has asynchronous timing.

Figure 4-35. Bus Arbitration Timing Diagram—Idle Bus Case



NOTE:
BR has synchronous timing.
BR has synchronous timing.

Figure 4-36. Bus Arbitration Timing Diagram—Active Bus Case

4.6.1 Bus Request

External devices capable of becoming bus masters request the bus by asserting $\overline{BR}$. This signal can be wire-ORed to indicate to the QUICC that some external device requires control of the bus. The QUICC is effectively at a lower bus priority level than the external device and relinquishes the bus after it has completed the current bus cycle (if one has started). If no $\overline{BGACK}$ is received while the $\overline{BR}$ is active, the QUICC remains bus master once $\overline{BR}$ is negated. This prevents unnecessary interference with ordinary processing if the arbitration circuitry inadvertently responds to noise or if an external device determines that it no longer requires use of the bus before it has been granted mastership.

4.6.2 Bus Grant

The QUICC supports operand coherency; thus, if an operand transfer requires multiple bus cycles, the QUICC does not release the bus until the entire transfer is complete. The assertion of $\overline{BG}$ is therefore subject to the following constraints:

- The minimum time for $\overline{BG}$ assertion after $\overline{BR}$ is asserted depends on internal synchronization.
- When working in synchronous mode (ASTM bit in the MCR is set), the minimum time can be one clock.
- During an external operand transfer, the QUICC does not assert $\overline{BG}$ until after the last cycle of the transfer (determined by SIZx and $\overline{DSACKx}$).
- During an external operand transfer, the QUICC does not assert $\overline{BG}$ as long as $\overline{RMC}$ is asserted.
- If the show cycle bits SHEN1–SHEN0 = 1x and if one of the QUICC internal masters is making internal accesses, the QUICC does not assert $\overline{BG}$ until the transfer is terminated.
- If SHEN1–SHEN0 = 00 and if one of the QUICC internal masters is making internal accesses, the external bus is granted away, and the QUICC continues to execute internal bus cycles. In this case, the arbitration overhead (external bus idle time) is minimal.
- If SHEN1–SHEN0 = 01, the QUICC does not assert $\overline{BG}$ to an external master.

Externally, the $\overline{BG}$ signal can be routed through a daisy-chained network or a priority-encoded network. The QUICC is not affected by the method of arbitration as long as the protocol is obeyed.

4.6.3 Bus Grant Acknowledge

An external device cannot request and be granted the external bus while another device is the active bus master. A device that asserts $\overline{BGACK}$ remains the bus master until it negates $\overline{BGACK}$. $\overline{BGACK}$ should not be negated until all required bus cycles are completed. Bus mastership is terminated at the negation of $\overline{BGACK}$. When no other device requests the bus after $\overline{BGACK}$ is negated, the QUICC will regain bus mastership.

The minimum time for the first bus cycle after $\overline{BGACK}$ negation depends on internal synchronization and internal bus arbitration. This timing is therefore subject to the following constraints:

- When working in synchronous mode (ASTM bit in the MCR is set) and SHEN0–SHEN1 = 00 and one of the QUICC internal masters requests an external accesses, the minimum time can be one clock.
- When working in asynchronous mode (ASTM bit in the MCR is cleared) and SHEN0–1 = 00 and one of the QUICC internal masters requests an external accesses, the minimum time depends on internal synchronization plus one clock.
- If SHEN1–SHEN0 = 1×, another clock is added for internal bus arbitration.

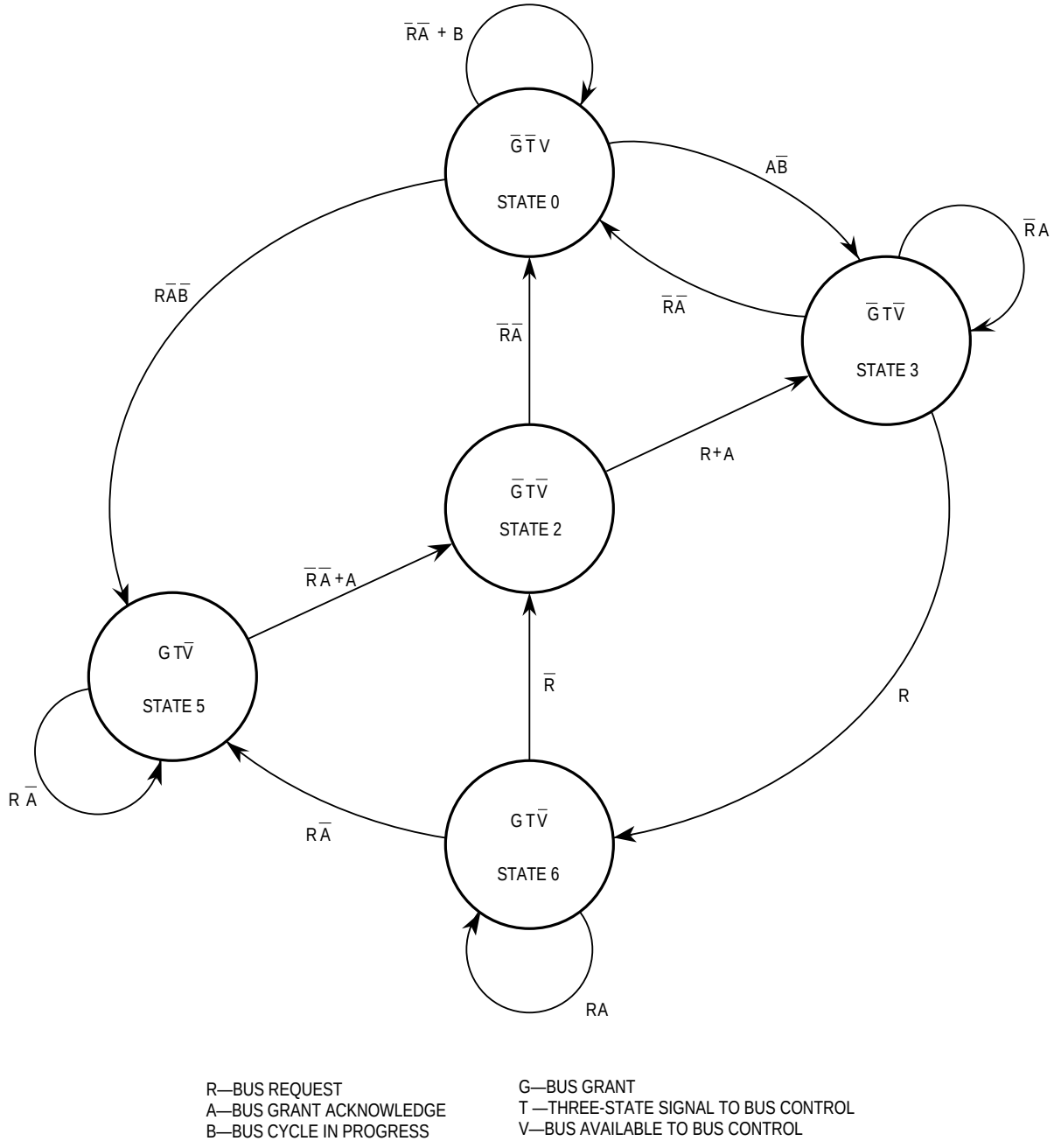
Once an external device receives the bus and asserts $\overline{\text{BGACK}}$, it should negate $\overline{\text{BR}}$. If $\overline{\text{BR}}$ remains asserted after $\overline{\text{BGACK}}$ is asserted, the QUICC assumes that another device is requesting the bus and prepares to issue another $\overline{\text{BG}}$.

4.6.4 Bus Arbitration Control

The bus arbitration control unit in the QUICC is implemented with a finite state machine. As discussed previously, all asynchronous inputs to the QUICC are internally synchronized in a maximum of two cycles of the clock. As shown in Figure 4-37, input signals labeled R and A are internally synchronized versions of $\overline{\text{BR}}$ and $\overline{\text{BGACK}}$, respectively. The $\overline{\text{BG}}$ output is labeled G, and the internal high-impedance control signal is labeled T. If T is true, the address, data, and control buses are placed in the high-impedance state after the next rising edge following the negation of $\overline{\text{AS}}$ and $\overline{\text{RMC}}$. All signals are shown in positive logic (active high), regardless of their true active voltage level. The state machine shown in Figure 4-37 does not have a state 1 or state 4.

State changes occur on the next rising edge of the clock after the internal signal is valid. The $\overline{\text{BG}}$ signal transitions on the rising edge of the clock after a state is reached during which G changes. The bus control signals (controlled by T) are driven by the QUICC immediately following a state change, when bus mastership is returned to the QUICC. State 0, in which G and T are both negated, is the state of the bus arbiter while the QUICC is bus master. R and A keep the arbiter in state 0 as long as they are both negated.

The QUICC does not allow arbitration of the external bus during the $\overline{\text{RMC}}$ sequence. For the duration of this sequence, the QUICC ignores the $\overline{\text{BR}}$ input. If mastership of the bus is required during an $\overline{\text{RMC}}$ operation, $\overline{\text{BERR}}$ must be used to abort the $\overline{\text{RMC}}$ sequence.



R—BUS REQUEST
A—BUS GRANT ACKNOWLEDGE
B—BUS CYCLE IN PROGRESS

G—BUS GRANT
T —THREE-STATE SIGNAL TO BUS CONTROL
V—BUS AVAILABLE TO BUS CONTROL

Figure 4-37. Bus Arbitration State Diagram

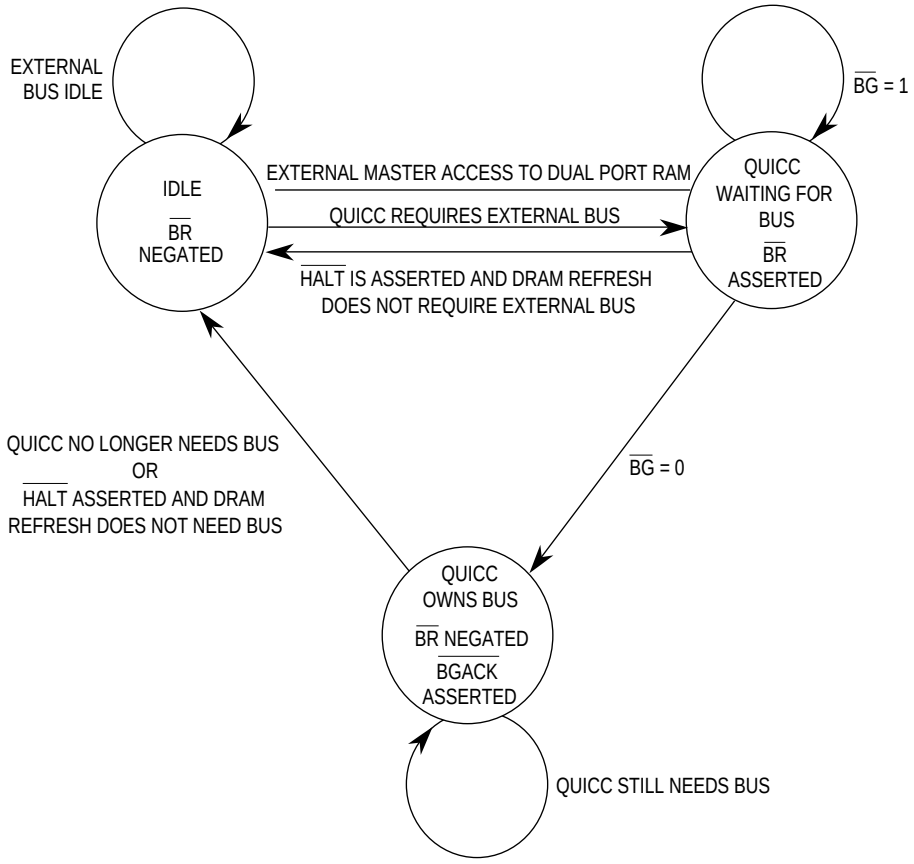
4.6.5 Slave (Disable CPU32+) Mode Bus Arbitration

When configured in the slave mode, the QUICC follows the bus arbitration mechanism described in 4.6 Bus Arbitration. When acting as one or more of the QUICC internal masters (refresh cycles, IDMA, and SDMA), the QUICC will output the $\overline{BR}$ signal. Systems that include several devices that can become bus master require external circuitry to assign priorities to the devices, so that when two or more external devices attempt to become bus

master at the same time, the one having the highest priority becomes bus master first. The sequence of the protocol in normal slave mode is as follows:

1. The QUICC asserts $\overline{BR}$.
2. The QUICC waits for the assertion of $\overline{BG}$ and the negation of $\overline{BGACK}$ to indicate that the bus is available.
3. The QUICC asserts $\overline{BGACK}$ to indicate that it has assumed the bus.

The state machine for the normal slave mode arbitration is shown in Figure 4-38.



NOTE: $\overline{BGACK}$ is only asserted by QUICC during the state "QUICC Owns Bus", otherwise $\overline{BGACK}$ is three-stated by the QUICC.

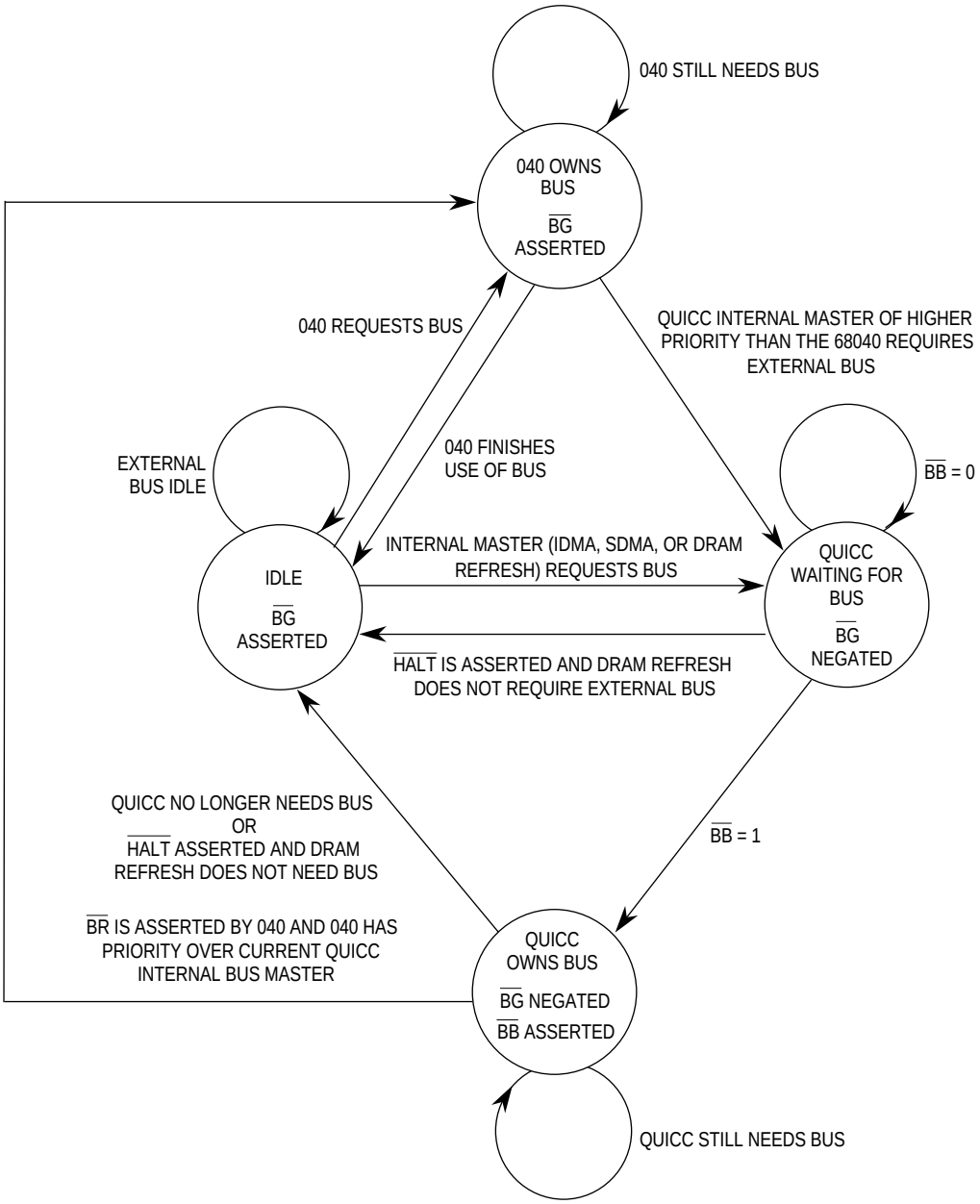
Figure 4-38. Slave Mode Bus Arbitration State Machine

In 68040 companion mode, the QUICC changes its bus arbitration sequence to match that needed by the 68040. It is as follows:

1. The QUICC asserts $\overline{BG}$ continuously whenever the QUICC does not need the bus.
2. When the QUICC needs the bus, and the 68040 is not requesting the bus, it will deassert $\overline{BG}$ from the 68040 and assert $\overline{BB}$ to indicate that it has assumed the bus. If the 68040 then requests the bus using the $\overline{BR}$ pin, while the QUICC is asserting $\overline{BB}$, the BR040ID bits in the MCR will be used to determine if the 68040 has a high enough bus request priority to cause the QUICC to give up the bus (i.e. deassert $\overline{BB}$ and assert $\overline{BG}$.)

3. If the 68040 requests the bus at the same time that a QUICC internal master is requesting the bus, the BR040ID bits are used to determine who will acquire the bus first.
4. When the QUICC no longer needs the bus, it deasserts $\overline{BB}$ and asserts $\overline{BG}$.

The state machine for the MC68040 companion mode arbitration is shown in Figure 4-39.



NOTES:
1. If the 68040 and the QUICC Internal Master requests the bus at the same time, the highest priority requester wins.
2. The transition from "040 Owns Bus" to "QUICC Waiting for Bus" may be delayed, until the write portion of an 040 locked cycle if an 040 locked cycle is in progress when the higher priority QUICC internal master requests the bus.
3. $\overline{BB}$ is only asserted by QUICC during the state "QUICC Owns Bus", otherwise $\overline{BB}$ is three-stated by the QUICC.

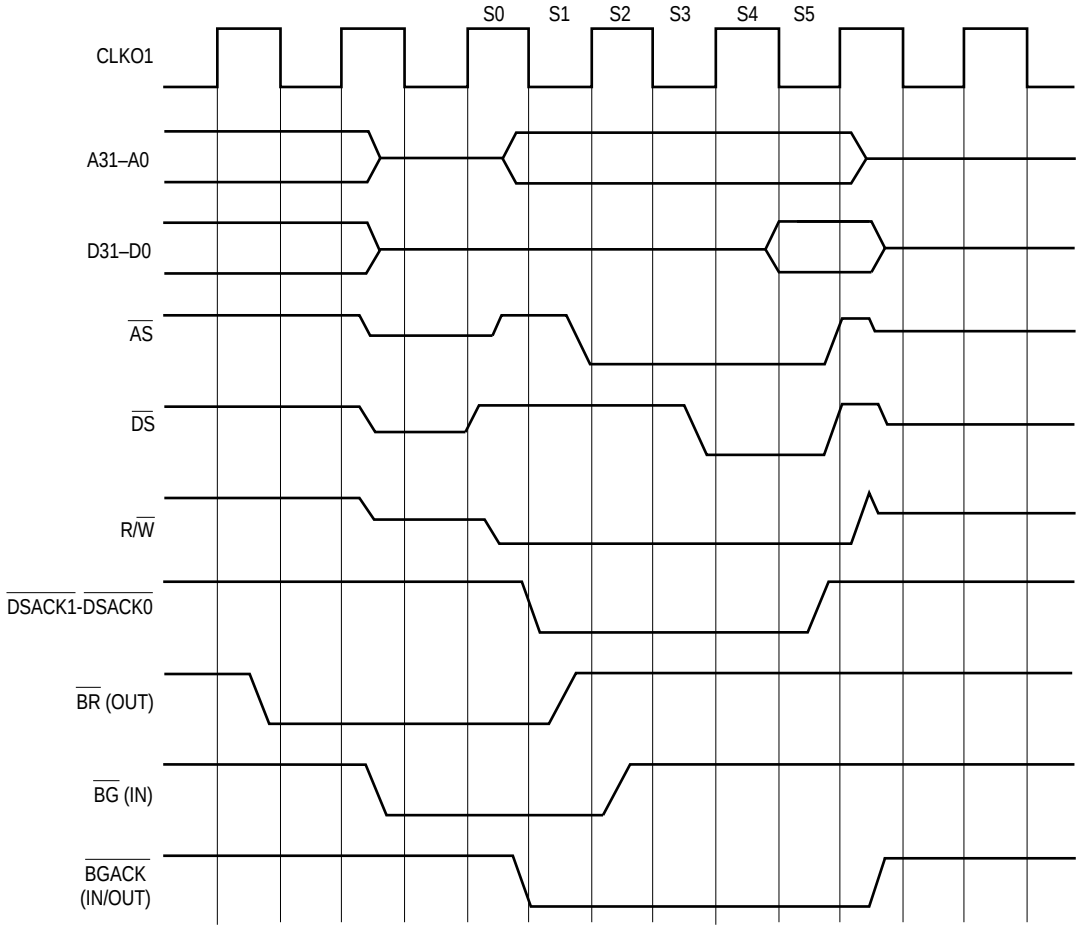
Figure 4-39. MC68040 Companion Mode Bus Arbitration State Machine

The QUICC has another mechanism to assign priorities to the bus masters. A new pin called bus clear in (BCLRI) is defined. BCLRI indicates to the QUICC that a request is being made for the QUICC to release the system bus. The QUICC will then clear all internal bus masters with an arbitration ID smaller than the programmed value of the bus clear in ID (BCLRIID) in the MCR.

Slave (disable CPU32+) mode bus arbitration has fewer arbitration modes than exist in a normal mode, since in slave mode, the SHEN1-SHEN0 bits are forced to be "00":

- In synchronous mode (ASTM bit in the MCR is set), $\overline{BG}$ and $\overline{BGACK}$ have synchronous timing, and the minimal delay between the assertion of $\overline{BG}$ (negation of $\overline{BGACK}$) and the assertion of $\overline{BGACK}$ is one clock.
- In asynchronous mode, the minimum time for $\overline{BGACK}$ assertion after $\overline{BG}$ is asserted ($\overline{BGACK}$ is negated) depends on internal synchronization.
- The QUICC will not request the external bus (assert $\overline{BR}$) when one of its internal masters is making an internal access. The QUICC will request the external bus only when one of its internal masters is beginning an external access. In this case, the arbitration overhead (external bus idle time is minimal).

See Figure 4-40 for the slave mode bus arbitration timing diagram.



NOTES:
1. Synchronous arbitration with SHEN1–SHEN0 = 00.
2. Minimum bus idle time.

Figure 4-40. Slave Mode Bus Arbitration Timing Diagram

4.6.6 Slave (Disable CPU32+) Mode Bus Exceptions

The reset and bus error master mode support also applies to the slave mode. There is a difference, however, in supporting halt and retry as explained in the following paragraphs.

4.6.6.1 $\overline{\text{HALT}}$. The QUICC transfer operation may be suspended at any time by asserting $\overline{\text{HALT}}$ to the QUICC. In response, any bus cycle in progress is completed (after $\overline{\text{DSACKx}}$ is asserted), and bus ownership is released. No further bus cycles will be started while $\overline{\text{HALT}}$ remains asserted. When the QUICC is in the middle of an operand transfer when halted and when a new transfer request is pending, the QUICC will arbitrate for the bus and continue normal operation.

NOTE

When the QUICC is doing a word access to an 8-bit port and $\overline{\text{HALT}}$ is asserted during the first access to an 8-bit port, the QUICC will access this byte again after bus ownership is granted to the QUICC.

NOTE

In slave mode $\overline{\text{HALT}}$ has more priority than bus coherency, whereas in normal mode (CPU32+ is enabled) $\overline{\text{HALT}}$ has less priority than bus coherency.

4.6.6.2 RETRY. When $\overline{\text{HALT}}$ and $\overline{\text{BERR}}$ are asserted during a bus cycle, the QUICC terminates the bus cycle, releases the bus, and suspends any further operation until these signals are negated. The QUICC will then arbitrate for the bus, re-execute the previous bus cycle, and continue normal operation. Thus, in slave mode, a retry is actually a relinquish and retry.

NOTE

When the relinquish and retry is asserted during a word access to an 8-bit port, and the external master that takes the bus performs an external-to-internal bus cycle, the entire word access will be retried. This is true even if the relinquish and retry was asserted on the second access and the first 8-bit access was completed normally.

4.6.7 Internal Accesses

The QUICC supports an external-master access to its internal registers with a glueless interface. The QUICC internal register port size is always 32 bits. External QUICC/MC68EC030 accesses have the same bus operation as the QUICC (see 4.3 Data Transfer Cycles**)**. The QUICC supports the interrupt acknowledge cycles presented in 4.4.4 Interrupt Acknowledge Bus Cycles. The QUICC also supports the MC68EC040 read and write accesses and interrupt acknowledge cycles (see Figure 4-41–Figure 4-44).

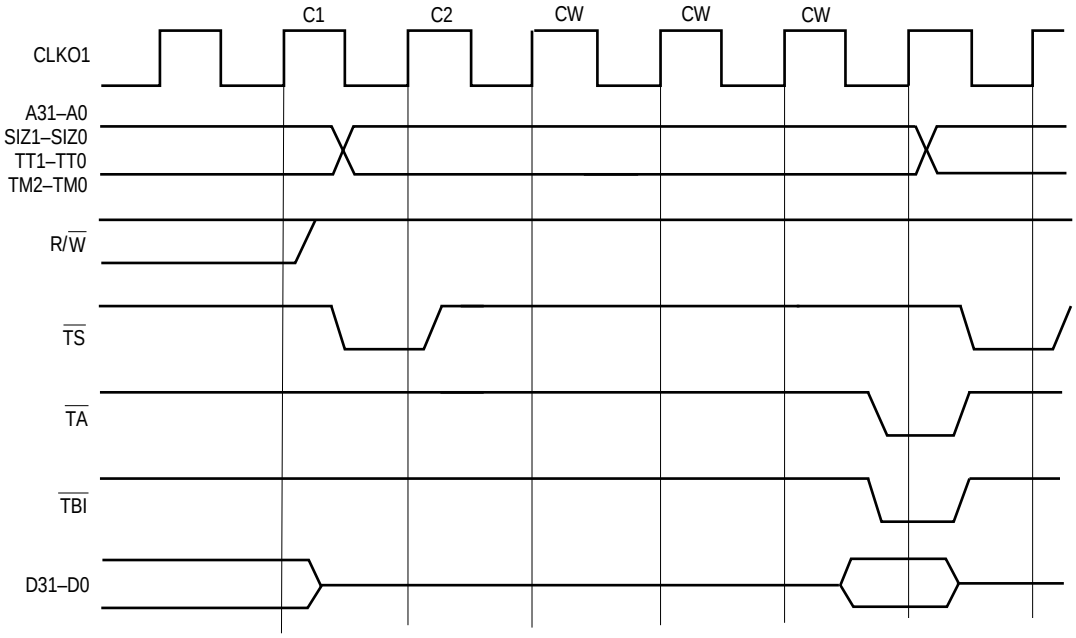


Figure 4-41. MC68EC040 Internal Registers Read Cycle

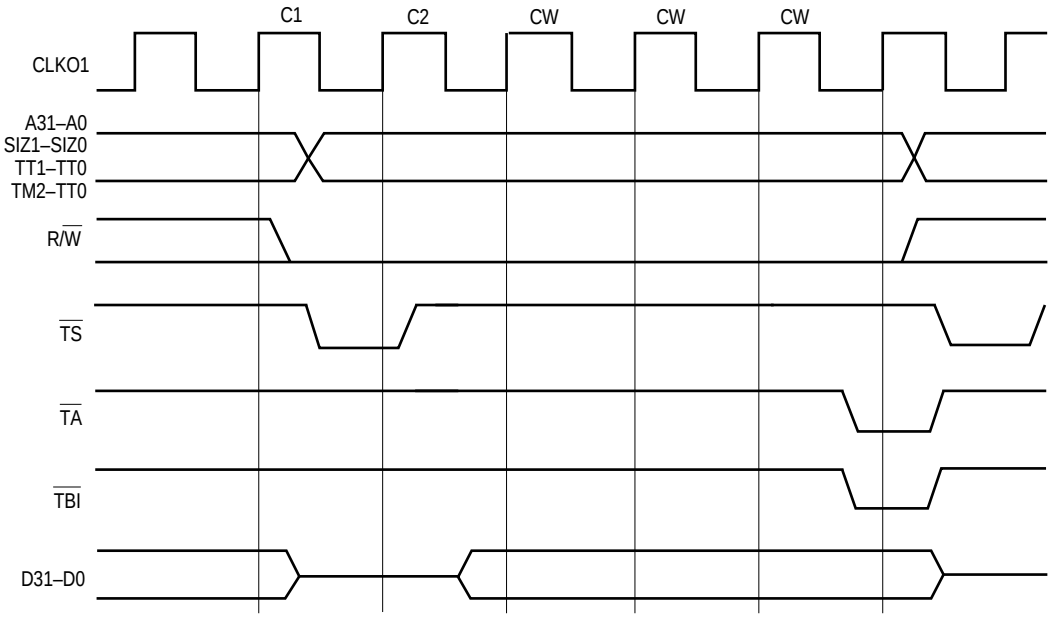


Figure 4-42. MC68EC040 Internal Registers Write Cycle

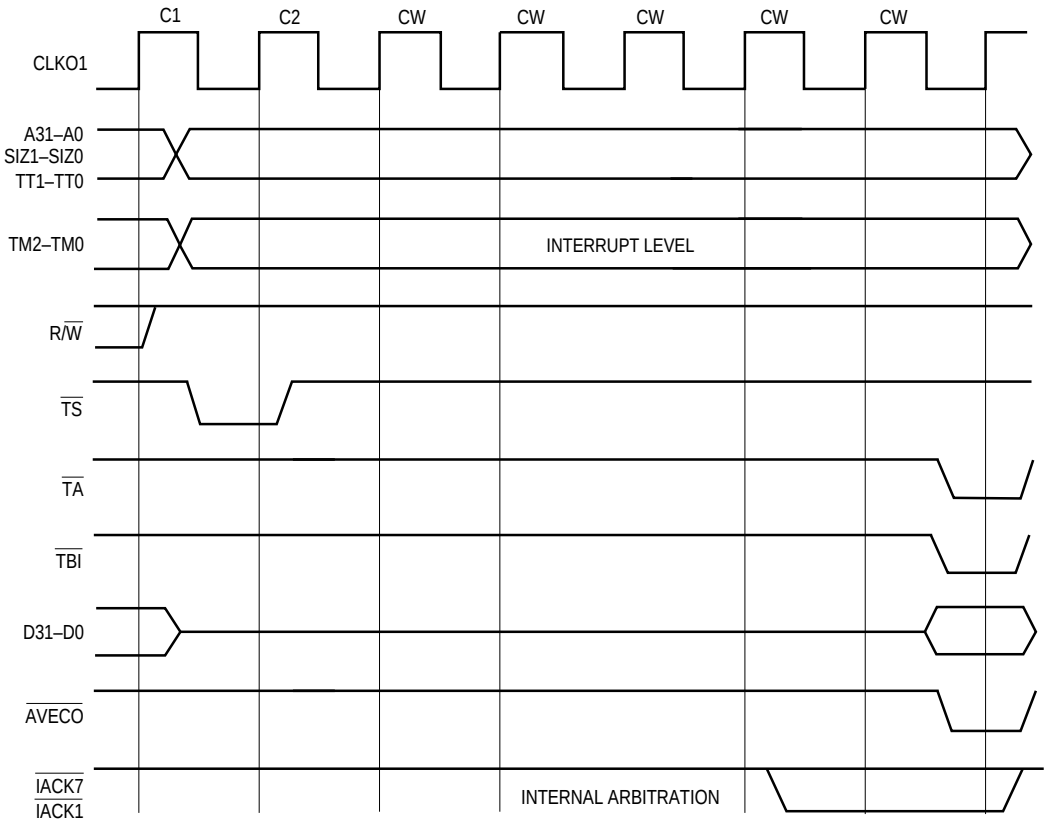


Figure 4-43. MC68EC040 Autovector Operation Timing

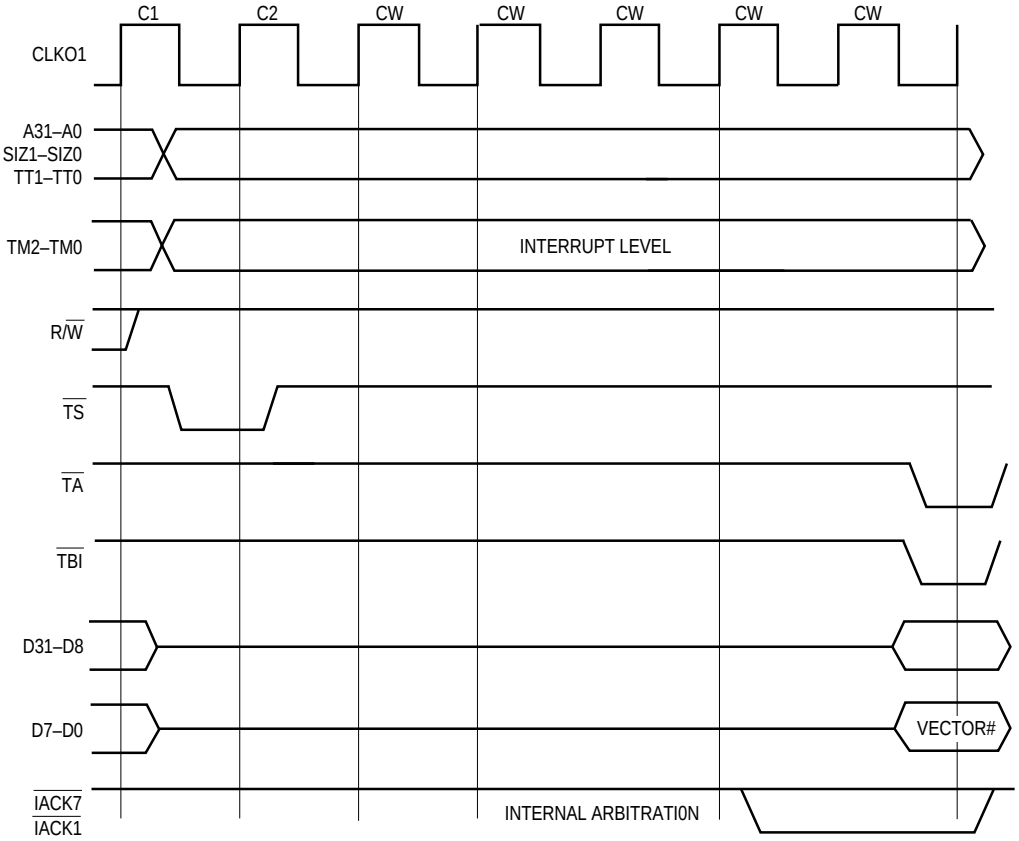


Figure 4-44. MC68EC040 Interrupt Acknowledge Cycle

4.6.8 Show Cycles

The QUICC can perform data transfers with its internal modules without using the external bus, but when debugging, it is desirable to have address and data information appear on the external bus. These external bus cycles, called show cycles, are distinguished by the fact that $\overline{AS}$ is not asserted externally. $\overline{DS}$ is used to signal address strobe timing in show cycles.

After reset, show cycles are disabled and must be enabled by writing to the SHEN bits in the module configuration register. When show cycles are disabled, the address bus, function codes, size, and read/write signals continue to reflect internal bus activity. However, $\overline{AS}$ and $\overline{DS}$ are not asserted externally, and the external data bus remains in a high impedance state. When show cycles are enabled, $\overline{DS}$ indicates address strobe timing and the external data bus contains data. The following paragraphs are a state-by-state description of show cycles, and Figure 4-45 illustrates a show cycle timing diagram. Refer to Section 10 Electrical Characteristics for specific timing information.

State 0 – During state 0, the address and function codes become valid, R/$\overline{W}$ is driven to indicate a show read or write cycle, and the size pins indicate the number of bytes to transfer. During a read, the addressed peripheral is driving the data bus, and the user must take care to avoid bus conflicts.

State 41 – One-half clock cycle later, $\overline{DS}$ (rather than $\overline{AS}$) is asserted to indicate that address information is valid.

State 42– No action occurs in state 42. The bus controller remains in state 42 (wait states will be inserted) until the internal read cycle is complete.

State 43– When $\overline{DS}$ is negated, show data is valid on the next falling edge of the system clock. The external data bus drivers are enabled so that data becomes valid on the external bus as soon as it is available on the internal bus.

State 0 – The address, function codes, read/write, and size pins change to begin the next cycle. Data from the preceding cycle is valid through state 0.

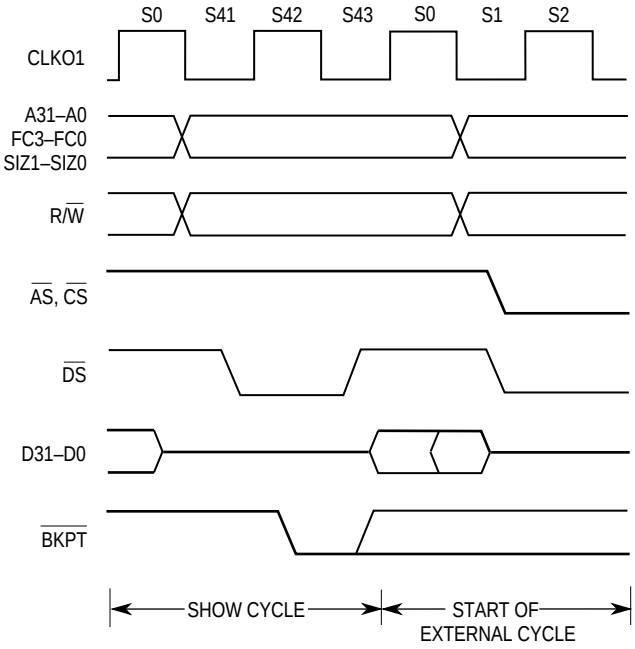


Figure 4-45. Show Cycle Timing Diagram

4.7 RESET OPERATION

The QUICC has reset control logic to determine the cause of reset, synchronize it if necessary, and assert the appropriate reset lines. The reset control logic can independently drive five different internal lines:

1. EXTSYSRST (external system reset) drives the external hard and soft reset pins ($\overline{\text{RESETH}}$ and $\overline{\text{RESETS}}$).
2. EXTRST (external reset) drives the external soft reset pin ($\overline{\text{RESETS}}$).
3. CLKRST (clock reset) resets the clock module.
4. INTSYSRST (internal system reset) resets the memory controller, system protection logic, serial interface, interrupt controller, and parallel I/O modules.
5. INTRST (internal reset) goes to all other internal circuits.

Table 4-9 summarizes the result of each reset source. Synchronous reset sources are not asserted until the end of the current bus cycle, regardless of whether $\overline{\text{RMC}}$ is asserted. The internal bus monitor is automatically enabled for synchronous resets; therefore, if the current bus cycle does not terminate normally, the bus monitor terminates it. Only single-byte or word transfers are guaranteed valid for synchronous resets. Asynchronous reset sources indicate a catastrophic failure, and the reset controller logic immediately resets the system. Resetting the QUICC causes any bus cycle in progress to terminate as if $\overline{\text{DSACKx}}$ or $\overline{\text{BERR}}$ had been asserted. In addition, the QUICC appropriately initializes registers for a reset exception.

Table 4-9. Reset Source Summary

Type	Source	Timing	Reset Lines Asserted by Controller			
External Hard Reset ($\overline{\text{RESETH}}$)	External	Asynchronous	INTRST	INTSYSRST	CLKRST	EXTSYSRST
External Soft Reset ($\overline{\text{RESETS}}$)	External	Synchronous	INTRST	—	—	EXTRST
Power-Up	EBI	Asynchronous	INTRST	INTSYSRST	CLKRST	EXTSYSRST
Software Watchdog	Sys Prot	Asynchronous	INTRST	INTSYSRST	—	EXTSYSRST
Double Bus Fault	Sys Prot	Asynchronous	INTRST	INTSYSRST	CLKRST	EXTSYSRST
Loss of Clock[1]	Clock	Asynchronous	INTRST	INTSYSRST	CLKRST	EXTSYSRST
Reset Instruction	CPU32+	Asynchronous	INTRST[2]	—	—	EXTRST

NOTES:
1. The reset behavior is this case is dependent on the PLL programming (see 6.9.3.9 CLKO Control Register (CLKOCR)).
2. Doesn't cause a CPU32 reset exception nor does it affect any of its internal registers.

If an external device drives $\overline{\text{RESETS}}$ or $\overline{\text{RESETH}}$ low, they should be asserted for at least 32 clock periods to ensure that the QUICC resets. When the reset control logic detects that an external device drives $\overline{\text{RESETS}}$ low, it starts driving both internal and external $\overline{\text{RESETS}}$ low for 512 cycles to guarantee this length of reset to the entire system. When the reset control logic detects that an external device drives $\overline{\text{RESETH}}$ low, it starts driving both internal and external $\overline{\text{RESETS}}$ and $\overline{\text{RESETH}}$ low for 512 cycles to guarantee this length of reset to the entire system. The external and the internal resets are released after the external device stops driving the external reset signal low or after the 512 cycles, whatever is later. Figure 4-46 shows the reset timing.

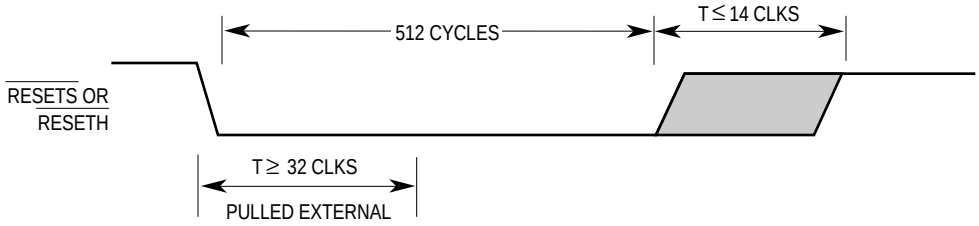


Figure 4-46. Timing for External Devices Driving $\overline{\text{RESET}}$

NOTE

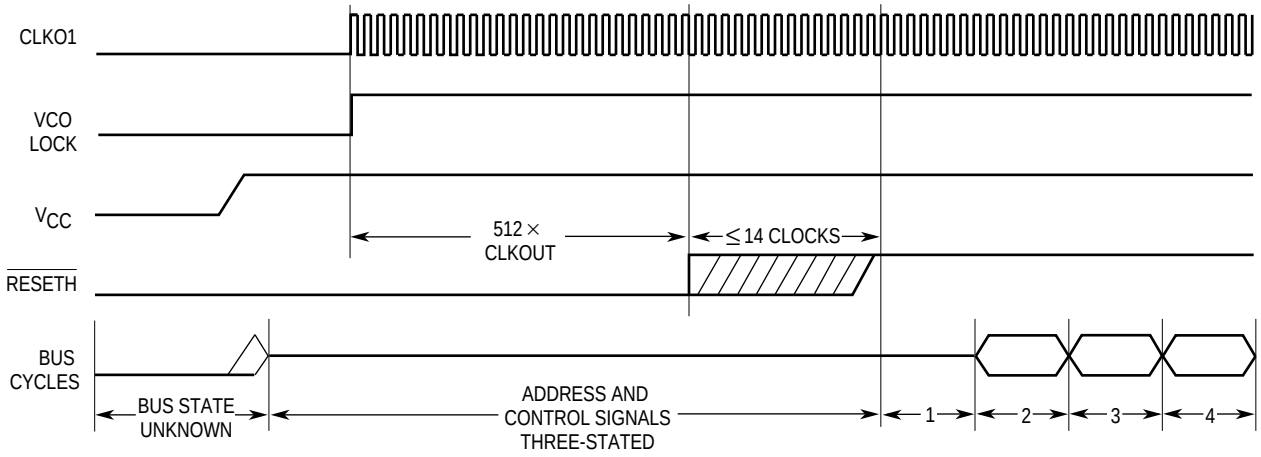
RESETS signal will always be negated after 512 cycles after assertion.

If reset is asserted from any other source, the reset control logic asserts a reset for a minimum of 512 cycles and until the source of reset is negated.

After any internal reset occurs, a 14-cycle rise time is allowed before testing for the presence of an external reset. If no external reset is detected, the CPU32+ begins its vector fetch.

Figure 4-47 is a timing diagram of the power-up reset operation, showing the relationships between $\overline{\text{RESETH}}$, $\overline{\text{RESETS}}$, VCC, and bus signals. During the reset period, the entire bus three-states (except for non-three-statable signals, which are driven to their inactive state).

Once $\overline{\text{RESETH}}$ and $\overline{\text{RESETS}}$ negate, all control signals are driven to their inactive state, the data bus is in read mode, and the address bus is driven. After this, the first bus cycle of the reset exception processing begins.



NOTES:
1. Internal start-up time.
2. SSP read here.
3. PC read here.
4. First instruction fetched here.
5. This figure is true when MODCK is 11 or 10.
   When MODCK is 01 CLKO1 will be driven high at power up.

Figure 4-47. Initial Reset Operation Timing

NOTE

The PLL samples the MODCLK pins while in the first 512 clocks of RESET. The process starts with RESET being asserted, then MODCLK pins are sampled and the PLL is initialized according to the MODCLK pins. For the next 500-2000 EXTAL cycles the PLL is synchronizing. 512 clocks after the PLL synchronizes, the QUICC no longer drives RESET and does not sample the MOD-CLK pins.

User should make suer the ramp up time of Vcc will never be faster than 4mSec to ensure proper power on reset sequence.

When a RESET instruction is executed, the QUICC drives the $\overline{\text{RESETS}}$ signal for 512 clock cycles. In this case, the QUICC resets the external devices of the system, and many of the internal registers of the QUICC (see Section 3 QUICC Memory Map for a list of registers affected by each type of reset).

The bus arbitration circuitry is only reset during a power-on reset. It may be used during all other resets.

In QUICC slave mode (disable CPU32+) the reset operates the same as in the normal (mas-ter) mode except that the RESET instruction does not exist.

NOTE

$\overline{\text{RESETS}}$ does not restore the Boot CS0 since the intent of $\overline{\text{RESETS}}$ is to not reset the memory controller. Note that the CPU will still fetch the SP and PC from $0 and $4, therefore a system implementing $\overline{\text{RESETS}}$ must have a device or register mapped to 0 and 4 at all times.

In the case where the CP32+ excutes a RESET command, the QUICC drives $\overline{\text{RESETS}}$ pin. In that case $\overline{\text{RESETS}}$ will be driven from CLOCK low (not CLOCK high as in all other cases). This requires a special AC timing parameter which is spec 58A in 10.9 Bus Operation AC Timing Specifications.

SECTION 5
CPU32+

The CPU32+, the second instruction processing module of the M68300 family, is based on the industry-standard MC68000 core processor. Like the original CPU32, it has many features of the MC68010 and MC68020 as well as unique features suited for high-performance processor applications. The CPU32+ provides a significant performance increase over the MC68000 CPU, yet maintains source-code and binary-code compatibility with the M68000 family.

The CPU32+ differs from the original CPU32 in two ways: it allows an option of a 32-bit data interface and allows byte-misaligned accesses to data operands.

5.1 OVERVIEW

The CPU32+ is designed to interface to the intermodule bus (IMB), allowing interaction with other IMB submodules. In this manner, integrated processors can be developed that contain useful peripherals on chip. This integration provides high-speed accesses among the IMB submodules, increasing system performance.

The CPU32+ core is a CPU32 core with its bus interface unit modified to connect directly to the 32-bit IMB and take advantage of the larger bus width. Although the original CPU32 core already had a 32-bit internal data path and 32-bit arithmetic hardware, its external interface (i.e., to the internal IMB) was 16 bits. The CPU32+ core, however, can operate on 32-bit external operands with one bus cycle. This capability allows the CPU32+ core to fetch a long-word instruction or two word-length instructions in one bus cycle, allowing the internal instruction queue to be filled more quickly. The CPU32+ core can also read and write 32-bits of data in one bus cycle. The CPU32+ has an additional word in its instruction pipeline when fetching from a 32-bit port. When fetching from a 16-bit port, this additional word is disabled. The performance of the CPU32+ on a 16-bit bus is the same as the CPU32 performance.

The CPU32+ also supports byte-misaligned operands. Since operands can reside at any byte boundary, they may occasionally become misaligned. A byte operand is properly aligned at any address; a word operand is misaligned at a odd address; a long-word operand is misaligned at an address that is not evenly divisible by four. Devices such as the MC68302, MC68000/8, MC68010, and CPU32-based MC68300 allow long-word operand transfers at odd-word addresses, but force exceptions if word or long-word operand transfers are attempted at odd-byte addresses. Although the CPU32+ does not enforce any alignment restrictions for data operands (including PC relative data addresses), some performance degradation occurs when additional bus cycles are required for long-word or word operands that are misaligned. For maximum performance, data items should be

aligned on their natural boundaries. All instruction words and extension words must reside on word boundaries. Attempting to prefetch an instruction word at an odd address causes an address error exception.

The CPU32+ has four bits (SZ1, SZ0 and SZC1, SCZ0) in the software status word (SSW) that are new or have changed definitions.

The CPU32+ offers low power consumption. The CPU32+ is implemented in high-speed complementary metal-oxide semiconductor (HCMOS) technology, providing low power use during normal operation. During periods of inactivity, the low-power stop (LPSTOP) instruction can be executed, shutting down the CPU32+ and other IMB modules, greatly reducing power consumption.

Ease of programming is an important consideration when using an integrated processor. The CPU32+ instruction format reflects a predominant register-memory interaction philosophy. All data resources are available to operations that require them. The programming model includes eight multifunction data registers and seven general-purpose addressing registers. The data registers support 8-bit (byte), 16-bit (word), and 32-bit (long-word) operand lengths for all operations. Address manipulation is supported by word and long-word operations. Although the program counter (PC) and stack pointers (SP) are special-purpose registers, they are also available for most data addressing activities. Ease of program checking and diagnosis is enhanced by trace and trap capabilities at the instruction level.

As processor applications become more complex and programs become larger, high-level languages (HLLs) become the system designer's choice in programming languages. HLLs aid in the rapid development of complex algorithms with less error and are readily portable. The CPU32+ instruction set efficiently support HLLs.

5.1.1 Features

Features of the CPU32+ are as follows:

- Fully Upward Object-Code Compatible with M68000 Family
- Loop Mode of Instruction Execution
- Fast Multiply, Divide, and Shift Instructions
- Fast Bus Interface with Dynamic Bus Port Sizing
- Improved Exception Handling
- Additional Addressing Modes

  —Scaled Index
  —Address Register Indirect with Base Displacement and Index
  —Expanded PC Relative Modes
  —32-Bit Branch Displacements
- Instruction Set Additions

  —High-Precision Multiply and Divide
  —Trap on Condition Codes
  —Upper and Lower Bounds Checking

- Enhanced Breakpoint Instruction
- Trace on Change of Flow
- Table Lookup and Interpolate (TBL) Instruction
- LPSTOP Instruction
- Hardware $\overline{\text{BKPT}}$ Signal, Background Mode
- Fully Static Implementation

A block diagram of the CPU32+ is shown in Figure 5-1. The major blocks depicted operate in a highly independent fashion that maximizes concurrences of operation while managing the essential synchronization of instruction execution and bus operation. The bus controller loads instructions from the data bus into the decode unit. The sequencer and control unit provide overall chip control by managing the internal buses, registers, and functions of the execution unit.

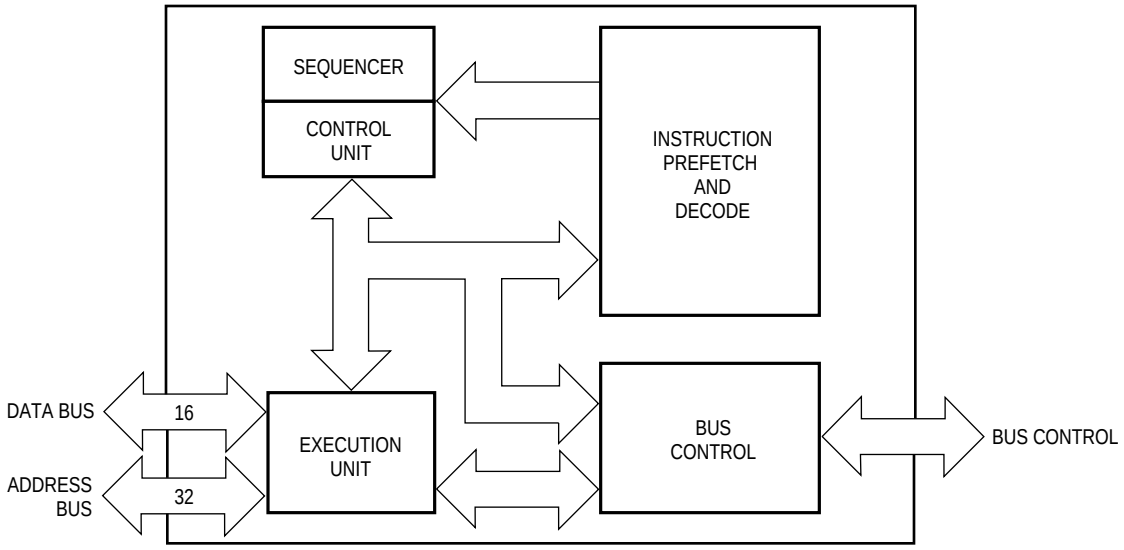


Figure 5-1. CPU32+ Block Diagram

5.1.2 Loop Mode Instruction Execution

The CPU32+ has several features that provide efficient execution of program loops. One of these features is the DBcc looping primitive instruction. To increase the performance of the CPU32+, a loop mode has been added to the processor. The loop mode is used by any single-word instruction that does not change the program flow. Loop mode is implemented in conjunction with the DBcc instruction. Figure 5-2 shows the required form of an instruction loop for the processor to enter loop mode.

The loop mode is entered when the DBcc instruction is executed and the loop displacement is –4. Once in loop mode, the processor performs only the data cycles associated with the instruction and suppresses all instruction fetches. The termination condition and count are checked after each execution of the data operations of the looped instruction. The CPU32+ automatically exits the loop mode during interrupts or other exceptions.

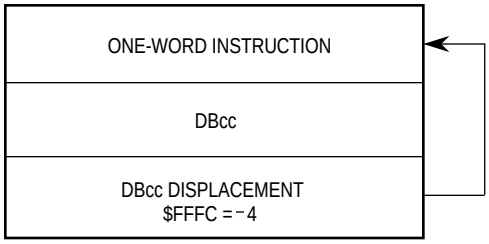


Figure 5-2. Loop Mode Instruction Sequence

5.1.3 Vector Base Register

The vector base register (VBR) contains the base address of the 1024-byte exception vector table, which consists of 256 exception vectors. Exception vectors contain the memory addresses of routines that begin execution at the completion of exception processing. These routines perform a series of operations appropriate for the corresponding exceptions. Because the exception vectors contain memory addresses, each vector consists of one long word, except the reset vector. The reset vector consists of two long words: the address used to initialize the supervisor stack pointer (SSP) and the address used to initialize the PC.

The address of an interrupt exception vector is derived from an 8-bit vector number and the VBR. The vector numbers for some exceptions are obtained from an external device; other numbers are supplied automatically by the processor. The processor multiplies the vector number by 4 to calculate the vector offset, which is added to the VBR. The sum is the memory address of the vector. All exception vectors are located in supervisor data space, except the reset vector, which is located in supervisor program space. Only the initial reset vector is fixed in the processor's memory map; once initialization is complete, there are no fixed assignments. Since the VBR provides the base address of the vector table, the vector table can be located anywhere in memory; it can even be dynamically relocated for each task that is executed by an operating system. Refer to 5.5 Exception Processing for additional details.

5.1.4 Exception Handling

The processing of an exception occurs in four steps, with variations for different exception causes. During the first step, a temporary internal copy of the status register (SR) is made, and the SR is set for exception processing. During the second step, the exception vector is determined. During the third step, the current processor context is saved. During the fourth step, a new context is obtained, and the processor then proceeds with instruction processing.

Exception processing saves the most volatile portion of the current context by pushing it on the supervisor stack. This context is organized in a format called the exception stack frame. This information always includes the SR and PC context of the processor when the exception occurred. To support generic handlers, the processor places the vector offset in the exception stack frame. The processor also marks the frame with a frame format. The format

field allows the return-from-exception (RTE) instruction to identify what information is on the stack so that it may be properly restored.

5.1.5 Addressing Modes

Addressing in the CPU32+ is register oriented. Most instructions allow the results of the specified operation to be placed either in a register or directly in memory; this flexibility eliminates the need for extra instructions to store register contents in memory.

The seven basic addressing modes are as follows:

- Register Direct
- Register Indirect
- Register Indirect with Index
- Program Counter Indirect with Displacement
- Program Counter Indirect with Index
- Absolute
- Immediate

Included in the register indirect addressing modes are the capabilities to postincrement, predecrement, and offset. The PC relative mode also has index and offset capabilities. In addition to these addressing modes, many instructions implicitly specify the use of the SR, SP and/or PC. Addressing is explained fully in the M68000PM/AD, *M68000 Family Programmer's Reference Manual***.**

5.2 ARCHITECTURE SUMMARY

The CPU32+ is upward source- and object-code compatible with the MC68000 and MC68010. It is downward source- and object-code compatible with the MC68020. Within the M68000 family, architectural differences are limited to the supervisory operating state. User programs can be executed unchanged on upward-compatible devices.

The major CPU32+ features are as follows:

- 32-Bit Internal Data Path and Arithmetic Hardware
- 32-Bit Address Bus Supported by 32-Bit Calculations
- Rich Instruction Set
- Eight 32-Bit General-Purpose Data Registers
- Seven 32-Bit General-Purpose Address Registers
- Separate User and Supervisor Stack Pointers (USP and SSP)
- Separate User and Supervisor Address Spaces
- Separate Program and Data Address Spaces
- Many Data Types
- Flexible Addressing Modes
- Full Interrupt Processing
- Expansion Capability

5.2.1 Programming Model

The CPU32+ programming model consists of two groups of registers that correspond to the user and supervisor privilege levels. User programs can only use the registers of the user model. The supervisor programming model, which supplements the user programming model, is used by CPU32+ system programmers who wish to protect sensitive operating system functions. The supervisor model is identical to that of MC68010 and later processors.

The CPU32+ has eight 32-bit data registers, seven 32-bit address registers, a 32-bit PC, separate 32-bit SSP and USP, a 16-bit SR, two alternate function code registers, and a 32-bit VBR (see Figure 5-3 and Figure 5-4).

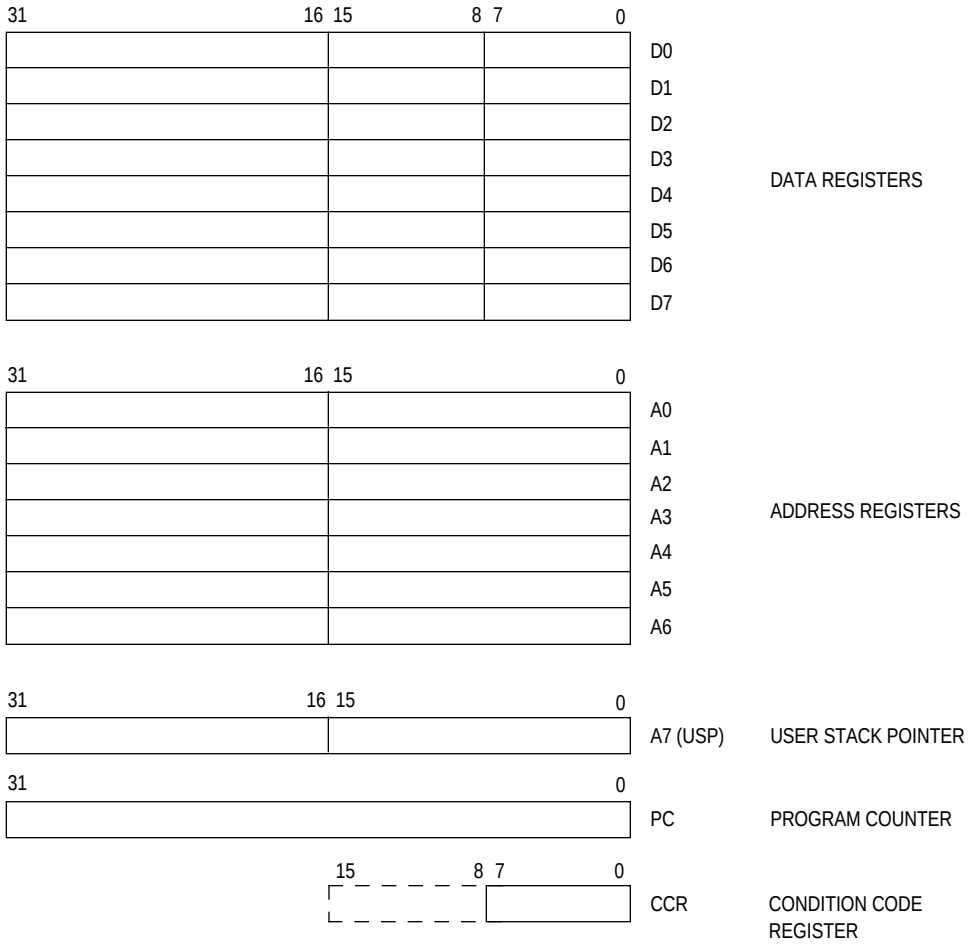


Figure 5-3. User Programming Model

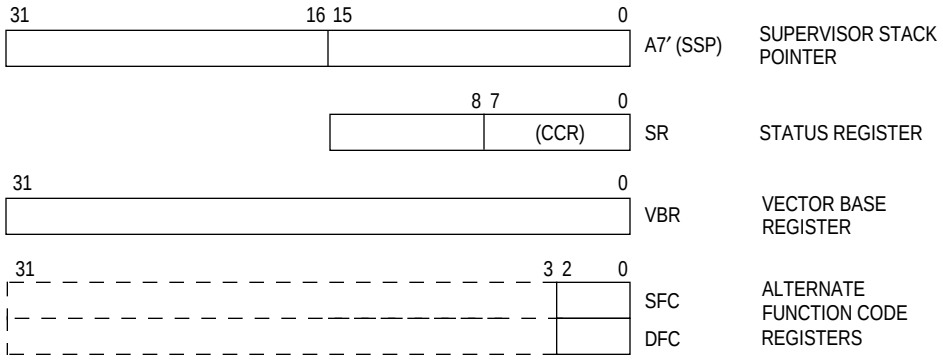


Figure 5-4. Supervisor Programming Model Supplement

5.2.2 Registers

Registers D7–D0 are used as data registers for bit, byte (8-bit), word (16-bit), long-word (32-bit), and quad-word (64-bit) operations. Registers A6 to A0 and the USP and SSP are address registers that may be used as software SPs or base address registers. Register A7 (shown as A7 and A7' in Figure 5-3 and Figure 5-4) is a register designation that applies to the USP in the user privilege level and to the SSP in the supervisor privilege level. In addition, address registers may be used for word and long-word operations. All 16 general-purpose registers (D7–D0, A7–A0) may be used as index registers.

The Program Counter (PC) contains the address of the next instruction to be executed by the CPU32+. During instruction execution and exception processing, the processor automatically increments the contents of the PC or places a new value in the PC, as appropriate.

The Status Register (SR) (see Figure 5-5) contains condition codes, an interrupt priority mask (three bits), and three control bits. Condition codes reflect the results of a previous operation. The codes are contained in the low byte (CCR) of the SR. The interrupt priority mask determines the level of priority an interrupt must have to be acknowledged. The control bits determine trace mode and privilege level. At user privilege level, only the CCR is available. At supervisor privilege level, software can access the full SR.

The Vector Base Register (VBR) contains the base address of the exception vector table in memory. The displacement of an exception vector is added to the value in this register to access the vector table.

Alternate source and destination function code registers (SFC and DFC) contain 3-bit function codes. The CPU32+ generates a function code each time it accesses an address. Specific codes are assigned to each type of access. The codes can be used to select eight dedicated 4-Gbyte address spaces. The MOVEC instruction can use registers SFC and DFC to specify the function code of a memory address.

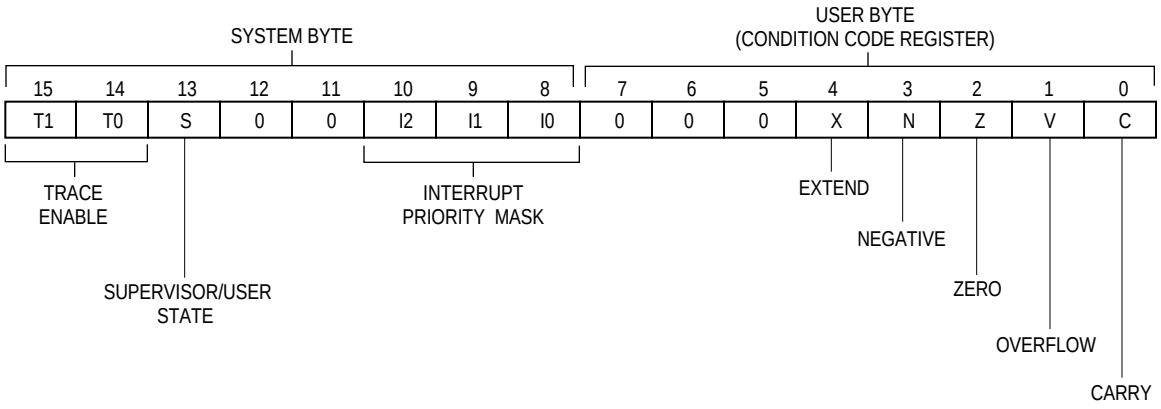


Figure 5-5. Status Register

5.3 INSTRUCTION SET

The following paragaphs describe the CPU32+ instruction set. A description of the instruction format, the operands used by the instructions, and a summary of the instructions by category are included. Complete programming information is provided in the M68000PM/AD, *M68000 Family Programmer's Reference Manual.*

The CPU32+ instructions include machine functions for all the following operations:

- Data Movement
- Arithmetic Operations
- Logical Operations
- Shifts and Rotates
- Bit Manipulation
- Conditionals and Branches
- System Control

The large instruction set encompasses a complete range of capabilities and, combined with the enhanced addressing modes, provides a flexible base for program development.

The instruction set of the CPU32+ is very similar to that of the MC68020 (see Table 5-1). The following M68020 instructions are not implemented on the CPU32+:

BFxx	—	Bit Field Instructions (BFCHG, BFCLR, BFEXTS, BFEXTU, BFFFO, BFINS, BFSET, BFTST)
CALLM, RTM	—	Call Module, Return Module
CAS, CAS2	—	Compare and Set (Read-Modify-Write Instructions)
cpxxx	—	Coprocessor Instructions (cpBcc, cpDBcc, cpGEN, cpRESTORE, cpSAVE, cpScc, cpTRAPcc)
PACK, UNPK	—	Pack, Unpack BCD Instructions

The CPU32+ traps on unimplemented instructions or illegal effective addressing modes, allowing user-supplied code to emulate unimplemented capabilities or to define special-purpose functions. However, Motorola reserves the right to use all currently unimplemented instruction operation codes for future M68000 core enhancements.

Table 5-1. Instruction Set

Mnemonic	Description		Mnemonic	Description
ABCD	Add Decimal with Extend		MOVEA	Move Address
ADD	Add		MOVE CCR	Move Condition Code Register
ADDA	Add Address		MOVE SR	Move to/from Status Register
ADDI	Add Immediate		MOVE USP	Move User Stack Pointer
ADDQ	Add Quick		MOVEC	Move Control Register
AND	Logical AND		MOVEM	Move Multiple Registers
ANDI	Logical AND Immediate		MOVEP	Move Peripheral Data
ASL	Arithmetic Shift Left		MOVEQ	Move Quick
ASR	Arithmetic Shift Right		MOVES	Move Alternate Address Space
Bcc	Branch Conditionally (16 Tests)		MULS	Signed Multiply
BCHG	Bit Test and Change		MULU	Unsigned Multiply
BCLR	Bit Test and Clear		NBCD	Negate Decimal with Extend
BGND	Enter Background Mode		NEG	Negate
BKPT	Breakpoint		NEGX	Negate with Extend
BRA	Branch Always		NOP	No Operation
BSET	Bit Test and Set		NOT	Ones Complement
BSR	Branch to Subroutine		OR	Logical Inclusive OR
BTST	Bit Test		ORI	Logical Inclusive OR Immediate
CHK	Check Register against Bounds		PEA	Push Effective Address
CHK2	Check Register against Upper and		RESET	Reset External Devices
	Lower Bounds		ROL, ROR	Rotate Left and Right
CLR	Clear Operand		ROXL, ROXR	Rotate with Extend Left and Right
CMP	Compare		RTD	Return and Deallocate
CMPA	Compare Address		RTE	Return from Exception
CMPI	Compare Immediate		RTR	Return and Restore
CMPM	Compare Memory		RTS	Return from Subroutine
CMP2	Compare Register against Upper		SBCD	Subtract Decimal with Extend
	and Lower Bounds		Scc	Set Conditionally
DBcc	Test Condition, Decrement and		STOP	Stop
	Branch (16 Tests)		SUB	Subtract
DIVS, DIVSL	Signed Divide		SUBA	Subtract Address
DIVU, DIVUL	Unsigned Divide		SUBI	Subtract Immediate
EOR	Logical Exclusive OR		SUBQ	Subtract Quick
EORI	Logical Exclusive OR Immediate		SUBX	Subtract with Extend
EXG	Exchange Registers		SWAP	Swap Data Register Halves
EXT, EXTB	Sign Extend		TAS	Test and Set Operand
ILLEGAL	Take Illegal Instruction Trap		TBLS, TBLSN	Table Lookup and Interpolate,
JMP	Jump			Signed
JSR	Jump to Subroutine		TBLU, TBLUN	Table Lookup and Interpolate,
LEA	Load Effective Address			Unsigned
LINK	Link and Allocate		TRAPcc	Trap Conditionally (16 Tests)
LPSTOP	Low-Power Stop		TRAPV	Trap on Overflow
LSL, LSR	Logical Shift Left and Right		TST	Test
MOVE	Move		UNLK	Unlink

5.3.1 M68000 Family Compatibility

It is the philosophy of the M68000 Family that all user-mode programs should execute unchanged on a more advanced processor and that supervisor-mode programs and exception handlers should require only minimal alteration.

The CPU32+ can be thought of as an intermediate member of the M68000 family. Object code from an MC68000 or MC68010 may be executed on the CPU32+, and many of the instruction and addressing mode extensions of the MC68020 are also supported.

5.3.1.1 NEW INSTRUCTIONS. Two instructions have been added to the M68000 instruction set: LPSTOP and TBL.

5.3.1.2 LOW-POWER STOP (LPSTOP). In applications where power consumption is a consideration, the CPU32+ can force the device into a low-power standby mode when immediate processing is not required. The low-power mode is entered by executing the LPSTOP instruction. The processor remains in this mode until a user-specified or higher level interrupt or a reset occurs.

5.3.1.3 TABLE LOOKUP AND INTERPOLATE (TBL). To maximize throughput for real-time applications, reference data is often precalculated and stored in memory for quick access. The storage of sufficient data points can require an inordinate amount of memory. The TBL instruction uses linear interpolation to recover intermediate values from a sample of data points, thus conserving memory.

When the TBL instruction is executed, the CPU32+ looks up two table entries bounding the desired result and performs a linear interpolation between them. Byte, word, and long-word operand sizes are supported. The result can be rounded according to a round-to-nearest algorithm or returned unrounded along with the fractional portion of the calculated result (byte and word results only). This extra precision can be used to reduce cumulative error in complex calculations. See 5.3.4 Using the TBL Instructions for examples.

5.3.1.4 UNIMPLEMENTED INSTRUCTIONS. The ability to trap on unimplemented instructions allows user-supplied code to emulate unimplemented capabilities or to define special-purpose functions. However, Motorola reserves the right to use all currently unimplemented instruction operation codes for future M68000 enhancements. See 5.5.2.8 Illegal or Unimplemented Instructions for more details.

5.3.2 Instruction Format and Notation

All instructions consist of at least one word. Some instructions can have as many as seven words, as shown in Figure 5-6. The first word of the instruction, called the operation word, specifies instruction length and the operation to be performed. The remaining words, called extension words, further specify the instruction and operands. These words may be immediate operands, extensions to the effective address mode specified in the operation word, branch displacements, bit number, special register specifications, trap operands, or argument counts.

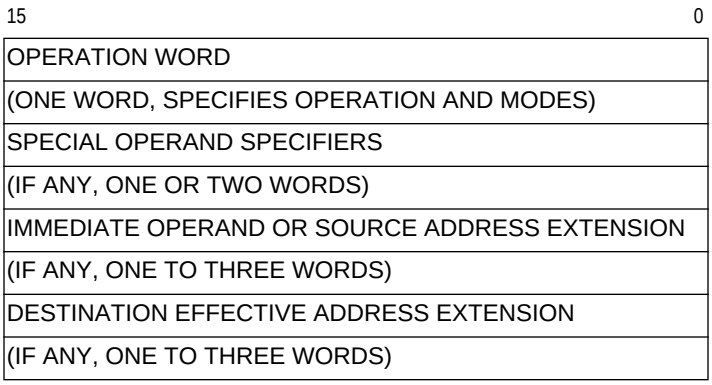


Figure 5-6. Instruction Word General Format

Besides the operation code, which specifies the function to be performed, an instruction defines the location of every operand for the function. Instructions specify an operand location in one of three ways:

- Register Specification — A register field of the instruction contains the number of the register.
- Effective Address — An effective address field of the instruction contains address mode information.
- Implicit Reference — The definition of an instruction implies the use of specific registers.

The register field within an instruction specifies the register to be used. Other fields within the instruction specify whether the register is an address or data register and how it is to be used. The M68000PM/AD, *M68000 Family Programmer's Reference Manual*, contains detailed register information.

Except where noted, the following notation is used in this section:

Data	Immediate data from an instruction
Destination	Destination contents
Source	Source contents
Vector	Location of exception vector
An	Any address register (A7–A0)
Ax, Ay	Address registers used in computation
Dn	Any data register (D7–D0)
Rc	Control register (VBR, SFC, DFC)
Rn	Any address or data register
Dh, Dl	Data registers, high- and low-order 32 bits of product
Dr, Dq	Data registers, division remainder, division quotient
Dx, Dy	Data registers, used in computation
Dym, Dyn	Data registers, table interpolation values

Xn	Index register
[An]	Address extension
cc	Condition code
d#	Displacement
	Example: d_{16} is a 16-bit displacement
⟨ea⟩	Effective address
#⟨data⟩	Immediate data; a literal integer
label	Assembly program label
list	List of registers
	Example: D3–D0
[...]	Bits of an operand
	Examples: [7] is bit 7; [31:24] are bits 31–24
(...)	Contents of a referenced location
	Example: (Rn) refers to the contents of Rn
CCR	Condition code register (lower byte of SR)
	X—extend bit
	N—negative bit
	Z—zero bit
	V—overflow bit
	C—carry bit
PC	Program counter
SP	Active stack pointer
SR	Status register
SSP	Supervisor stack pointer
USP	User stack pointer
FC	Function code
DFC	Destination function code register
SFC	Source function code register
+	Arithmetic addition or postincrement
–	Arithmetic subtraction or predecrement
/	Arithmetic division or conjunction symbol
×	Arithmetic multiplication
=	Equal to

≠	Not equal to
>	Greater than
≥	Greater than or equal to
<	Less than
≤	Less than or equal to
Λ	Logical AND
V	Logical OR
⊕	Logical exclusive OR
~	Invert; operand is logically complemented
BCD	Binary-coded decimal, indicated by subscript Example: $Source_{10}$ is a BCD source operand.
LSW	Least significant word
MSW	Most significant word
{R/W}	Read/write indicator

In a description of an operation, a destination operand is placed to the right of source operands and is indicated by an arrow (⇒).

5.3.3 Instruction Summary

The instructions form a set of tools to perform the following operations:

Data Movement	Bit Manipulation
Integer Arithmetic	Binary-Coded Decimal Arithmetic
Logic	Program Control
Shift and Rotate	System Control

The complete range of instruction capabilities combined with the addressing modes described previously provide flexibility for program development. All CPU32+ instructions are summarized in Table 5-2.

Table 5-2. Instruction Set Summary

Opcode	Operation	Syntax
ABCD	$Source_{10}$ + $Destination_{10}$ + X ⇒ Destination	ABCD Dy,Dx ABCD –(Ay),–(Ax)
ADD	Source + Destination ⇒ Destination	ADD ⟨ea⟩,Dn ADD Dn,⟨ea⟩
ADDA	Source + Destination ⇒ Destination	ADDA ⟨ea⟩,An
ADDI	Immediate Data + Destination ⇒ Destination	ADDI #⟨data⟩,⟨ea⟩
ADDQ	Immediate Data + Destination ⇒ Destination	ADDQ #⟨data⟩,⟨ea⟩
ADDX	Source + Destination + X ⇒ Destination	ADDX Dy,Dx ADDX –(Ay),–(Ax)
AND	Source Λ Destination ⇒ Destination	AND ⟨ea⟩,Dn AND Dn,⟨ea⟩
ANDI	Immediate Data Λ Destination ⇒ Destination	ANDI #⟨data⟩,⟨ea⟩
ANDI to CCR	Source Λ CCR ⇒ CCR	ANDI #⟨data⟩,CCR
ANDI to SR	If supervisor state the Source Λ SR ⇒ SR else TRAP	ANDI #⟨data⟩,SR
ASL,ASR	Destination Shifted by ⟨count⟩ ⇒ Destination	ASd Dx,Dy ASd #⟨data⟩,Dy ASd ⟨ea⟩
Bcc	If (condition true) then PC + d ⇒ PC	Bcc ⟨label⟩
BCHG	~(⟨number⟩ of Destination) ⇒ Z; ~(⟨number⟩ of Destination) ⇒ ⟨bit number⟩ of Destination	BCHG Dn,⟨ea⟩ BCHG #data⟩,⟨ea⟩
BCLR	~(⟨number⟩ of Destination) ⇒ Z; 0 ⇒ ⟨bit number⟩ of Destination	BCLR Dn,⟨ea⟩ BCLR #⟨data⟩,⟨ea⟩
BGND	If (background mode enabled) then enter background mode else Format/Vector offset ⇒ –(SSP) PC ⇒ –(SSP) SR ⇒ –(SSP) (Vector) ⇒ PC	BGND
BKPT	Run breakpoint acknowledge cycle; TRAP as illegal instruction	BKPT #⟨data⟩
BRA	PC + d ⇒ PC	BRA ⟨label⟩
BSET	~(⟨number⟩ of Destination) ⇒ Z; 1 ⇒ ⟨bit number⟩ of Destination	BSET Dn,⟨eaÒ BSET #⟨data⟩,⟨ea⟩
BSR	SP – 4 ⇒ SP; PC ⇒ (SP); PC + d ⇒ PC	BSR ⟨label⟩
BTST	– (⟨number⟩ of Destination) ⇒ Z;	BTST Dn,⟨ea⟩ BTST #⟨data⟩,⟨ea⟩
CHK	If Dn < 0 or Dn > Source then TRAP	CHK ⟨ea⟩,Dn
CHK2	If Rn < lower bound or If Rn > upper bound then TRAP	CHK2 ⟨ea⟩,Rn
CLR	0 ⇒ Destination	CLR ⟨ea⟩
CMP	Destination Source ⇒ cc	CMP ⟨ea⟩,Dn
CMPA	Destination — Source	CMPA ⟨ea⟩,An
CMPI	Destination — Immediate Data	CMPI #⟨data⟩,⟨ea⟩
CMPM	Destination — Source ⇒ cc	CMPM (Ay)+,(Ax)+

Table 5-2. Instruction Set Summary (Continued)

Opcode	Operation	Syntax
CMP2	Compare Rn < lower-bound or Rn > upper-bound and Set Condition Codes	CMP2 ⟨ea⟩,Rn
DBcc	If condition false then (Dn – 1 ⇒ Dn; If Dn ≠ –1 then PC + d ⇒ PC)	DBcc Dn,⟨label⟩
DIVS DIVSL	Destination/Source ⇒ Destination	DIVS.W ⟨ea⟩,Dn 32/16 ⇒ 16r:16q DIVS.L ⟨ea⟩,Dq 32/32 ⇒ 32q DIVS.L ⟨ea⟩,Dr:Dq 64/32 ⇒ 32r:32q DIVSL.L ⟨ea⟩,Dr:Dq 32/32 ⇒ 32r:32q
DIVU DIVUL	Destination/Source ⇒ Destination	DIVU.W ⟨ea⟩,Dn 32/16 ⇒ 16r:16q DIVU.L ⟨ea⟩,Dq 32/32 ⇒ 32q DIVU.L ⟨ea⟩,Dr:Dq 64/32 ⇒ 32r:32q DIVUL.L ⟨ea⟩,Dr:Dq 32/32 ⇒ 32r:32q
EOR	Source ⊕ Destination ⇒ Destination	EOR Dn,⟨ea⟩
EORI	Immediate Data ⊕ Destination ⇒ Destination	EORI #⟨data⟩,⟨ea⟩
EORI to CCR	Source ⊕ CCR ⇒ CCR	EORI #⟨data⟩,CCR
EORI to SR	If supervisor state the Source ⊕ SR ⇒ SR else TRAP	EORI #⟨data⟩,SR
EXG	Rx ⇔ Ry	EXG Dx,Dy EXG Ax,Ay EXG Dx,Ay EXG Ay,Dx
EXT EXTB	Destination Sign-Extended ⇒ Destination	EXT.W Dn extend byte to word EXT.L Dn extend word to long word EXTB.L Dn extend byte to long word
LLEGAL	SSP – 2 ⇒ SSP; Vector Offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SSp – 2 ⇒ SSP; SR ⇒ (SSP); Illegal Instruction Vector Address ⇒ PC	ILLEGAL
JMP	Destination Address ⇒ PC	JMP ⟨ea⟩
JSR	SP – 4 ⇒ SP; PC ⇒ (SP) Destination Address ⇒ PC	JSR ⟨ea⟩
LEA	⟨ea⟩ ⇒ An	LEA ⟨ea⟩,An
LINK	SP – 4 ⇒ SP; An ⇒ (SP) SP ⇒ An, SP + d ⇒ SP	LINK An,#⟨displacement⟩
LPSTOP	If supervisor state Immediate Data ⇒ SR Interrupt Mask ⇒ External Bus Interface (EBI) STOP else TRAP	LPSTOP #⟨data⟩
LSL,LSR	Destination Shifted by ⟨count⟩ ⇒ Destination	LSd[1] Dx,Dy LSd[1] #⟨data⟩,Dy LSd[1] ⟨ea⟩
MOVE	Source ⇒ Destination	MOVE ⟨ea⟩,⟨ea⟩
MOVEA	Source ⇒ Destination	MOVEA ⟨ea⟩,An
MOVE from CCR	CCR ⇒ Destination	MOVE CCR,⟨ea⟩
MOVE to CCR	Source ⇒ CCR	MOVE ⟨ea⟩,CCR
MOVE from SR	If supervisor state then SR ⇒ Destination else TRAP	MOVE SR,⟨ea⟩
MOVE to SR	If supervisor state then Source ⇒ SR else TRAP	MOVE ⟨ea⟩,SR

Table 5-2. Instruction Set Summary (Continued)

Opcode	Operation	Syntax
MOVE USP	If supervisor state then USP ⇒ An or An ⇒ USP else TRAP	MOVE USP,An MOVE An,USP
MOVEC	If supervisor state then Rc ⇒ Rn or Rn ⇒ Rc else TRAP	MOVEC Rc,Rn MOVEC Rn,Rc
MOVEM	Registers ⇒ Destination Source ⇒ Registers	MOVEM register list,⟨ea⟩ MOVEM ⟨ea⟩,register list
MOVEP	Source ⇒ Destination	MOVEP Dx,(d,Ay) MOVEP (d,Ay),Dx
MOVEQ	Immediate Data ⇒ Destination	MOVEQ #⟨data⟩,Dn
MOVES	If supervisor state then Rn ⇒ Destination [DFC] or Source [SFC] ⇒ Rn else TRAP	MOVES Rn,⟨ea⟩ MOVES ⟨ea⟩,Rn
MULS	Source × Destination ⇒ Destination	MULS.W ⟨ea⟩,Dn 16 × 16 ⇒ 32 MULS.L ⟨ea⟩,Dl 32 × 32 ⇒ 32 MULS.L ⟨ea⟩,Dh:Dl 32 × 32 ⇒ 64
MULU	Source × Destination ⇒ Destination	MULU.W ⟨ea⟩,Dn 16 × 16 ⇒ 32 MULU.L ⟨ea⟩,Dl 32 × 32 ⇒ 32 MULU.L ⟨ea⟩,Dh:Dl 32 × 32 ⇒ 64
NBCD	0 – (Destination$_{10}$) – X ⇒ Destination	NBCD ⟨ea⟩
NEG	0 – (Destination) ⇒ Destination	NEG ⟨ea⟩
NEGX	0 – (Destination) – X ⇒ Destination	NEGX ⟨ea⟩
NOP	None	NOP
NOT	~Destination ⇒ Destination	NOT ⟨ea⟩
OR	Source V Destination ⇒ Destination	OR ⟨ea⟩,Dn OR Dn,⟨ea⟩
ORI	Immediate Data V Destination ⇒ Destination	ORI #⟨data⟩,⟨ea⟩
ORI to CCR	Source V CCR ⇒ CCR	ORI #⟨data⟩,CCR
ORI to SR	If supervisor state then Source V SR ⇒ SR else TRAP	ORI #⟨data⟩,SR
PEA	Sp – 4 ⇒ SP; ⟨ea⟩ ⇒ (SP)	PEA ⟨ea⟩
RESET	If supervisor state then Assert $\overline{\text{RESET}}$ else TRAP	RESET
ROL,ROR	Destination Rotated by ⟨count ⟩⇒ Destination	ROd^1 Rx,Dy ROd^1 #⟨data⟩,Dy ROd^1 ⟨ea⟩
ROXL,ROXR	Destination Rotated with X by ⟨count⟩ ⇒ Destination	ROXd^1 Rx,Dy ROXd^1 #⟨data⟩,Dy ROXd^1 ⟨ea⟩
RTD	(SP) ⇒ PC; SP + 4 + d ⇒ SP	RTD #⟨displacement⟩
RTE	If supervisor state the (SP) ⇒ SR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP; restore state and deallocate stack according to (SP) else TRAP	RTE
RTR	(SP) ⇒ CCR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP	RTR

Table 5-2. Instruction Set Summary (Concluded)

Opcode	Operation	Syntax
RTS	(SP) ⇒ PC; SP + 4 ⇒ SP	RTS
SBCD	$Destination_{10}$ – $Source_{10}$ – X ⇒ Destination	SBCD Dx,Dy SBCD –(Ax),–(Ay)
Scc	If Condition True then 1s ⇒ Destination else 0s ⇒ Destination	Scc ⟨ea⟩
STOP	If supervisor state then Immediate Data ⇒ SR; STOP else TRAP	STOP #⟨data⟩
SUB	Destination – Source ⇒ Destination	SUB ⟨ea⟩,Dn SUB Dn,⟨ea⟩
SUBA	Destination – Source ⇒ Destination	SUBA ⟨ea⟩,An
SUBI	Destination – Immediate Data ⇒ Destination	SUBI #⟨data⟩,⟨ea⟩
SUBQ	Destination – Immediate Data ⇒ Destination	SUBQ #⟨data⟩,⟨ea⟩
SUBX	Destination – Source – X ⇒ Destination	SUBX Dx,Dy SUBX –(Ax),–(Ay)
SWAP	Register [31:16] ⇔ Register [15:0]	SWAP Dn
TAS	Destination Tested ⇒ Condition Codes; 1 ⇒ bit 7 of Destination	TAS ⟨ea⟩
TBLS	ENTRY(n) + {(ENTRY(n + 1) – ENTRY(n)) × Dx[7:0]} / 256 ⇒ Dx	TBLS.⟨size⟩ ⟨ea⟩, Dx TBLS.⟨size⟩ Dym:Dyn, Dx
TBLSN	ENTRY(n) × 256 + {(ENTRY(n + 1) – ENTRY(n)) × Dx [7:0]} ⇒ Dx	TBLSN.⟨size⟩ ⟨ea⟩,Dx TBLSN.⟨size⟩ Dym:Dyn, Dx
TBLU	ENTRY(n) + {(ENTRY(n + 1) – ENTRY(n)) × Dx[7:0]} / 256 ⇒ Dx	TBLU.⟨size⟩ ⟨ea⟩,Dx TBLU.⟨size⟩ Dym:Dyn, Dx
TBLUN	ENTRY(n) × 256 + {(ENTRY(n + 1) – ENTRY(n)) × Dx[7:0]} ⇒ Dx	TBLUN.⟨size⟩ ⟨ea⟩,Dx TBLUN.⟨size⟩ Dym:Dyn,Dx
TRAP	SSP – 2 ⇒ SSP; Format/Offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SSP – 2 ⇒ SSP; SR ⇒ (SSP); Vector Address ⇒ PC	TRAP #⟨vector⟩
TRAPcc	If cc then TRAP	TRAPcc TRAPcc.W #⟨data⟩ TRAPcc.L #⟨data⟩
TRAPV	If V then TRAP	TRAPV
TST	Destination Tested ⇒ Condition Codes	TST ⟨ea⟩
UNLK	An ⇒ SP; (SP) ⇒ An; SP + 4 ⇒ SP	UNLK An

NOTE 1: d is direction, L or R.

5.3.3.1 CONDITION CODE REGISTER. The CCR portion of the SR contains five bits that indicate the result of a processor operation. Table 5-2 lists the effect of each instruction on these bits. The carry bit and the multiprecision extend bit are separate in the M68000 Family to simplify programming techniques that use them. Refer to Table 5-3 as an example.

Table 5-3. Condition Code Computations

Operations	X	N	Z	V	C	Special Definition
ABCD	*	U	?	U	?	C = Decimal Carry Z = Z Λ Rm Λ ... Λ R0
ADD, ADDI, ADDQ	*	*	*	?	?	V = Sm Λ Dm Λ R̄m V S̄m Λ D̄m Λ Rm C = Sm Λ Dm V Rm Λ Dm V Sm Λ Rm
ADDX	*	*	?	?	?	V = Sm Λ Dm Λ R̄m V S̄m Λ D̄m Λ Rm C = Sm Λ Dm V Rm Λ Dm V Sm Λ Rm Z = Z Λ Rm Λ ... Λ R0
AND, ANDI, EOR, EORI, MOVEQ, MOVE, OR, ORI, CLR, EXT, NOT, TAS, TST	—	*	*	0	0	
CHK	—	*	U	U	U	
CHK2, CMP2	—	U	?	U	?	Z = (R = LB) V (R = UB) C = (LB < UB) Λ (IR < LB) V (R > UB) V (UB < LB) Λ (R > UB) Λ (R < LB)
SUB, SUBI, SUBQ	*	*	*	?	?	V = S̄m Λ Dm Λ R̄m V Sm Λ D̄m Λ Rm C = Sm Λ Dm V Rm Λ Dm V Sm Λ Rm
SUBX	*	*	?	?	?	V = S̄m Λ Dm Λ R̄m V Sm Λ D̄m Λ Rm C = Sm Λ Dm V Rm Λ Dm V Sm Λ Rm Z = Z Λ Rm Λ ... Λ R0
CMP, CMPI, CMPM	—	*	*	?	?	V = S̄m Λ Dm Λ R̄m V Sm Λ D̄m Λ Rm C = Sm Λ Dm V Rm Λ Dm V Sm Λ Rm
DIVS, DIVU	—	*	*	?	0	V = Division Overflow
MULS, MULU	—	*	*	?	0	V = Multiplication Overflow
SBCD, NBCD	*	U	?	U	?	C = Decimal Borrow Z = Z Λ Rm Λ ... Λ R0
NEG	*	*	*	?	?	V = Dm Λ Rm C = Dm V Rm
NEGX	*	*	?	?	?	V = Dm Λ Rm C = Dm V Rm Z = Z Λ Rm Λ ... Λ R0
ASL	*	*	*	?	?	V = Dm Λ (Dm – 1 V ... V D̄m – r) V D̄m Λ (Dm – 1 V ... + Dm – r) C = D̄m – r + 1
ASL (r = 0)	—	*	*	0	0	
LSL, ROXL	*	*	*	0	?	C = Dm – r + 1
LSR (r = 0)	—	*	*	0	0	
ROXL (r = 0)	—	*	*	0	?	C = X
ROL	—	*	*	0	?	C = Dm – r + 1
ROL (r = 0)	—	*	*	0	0	
ASR, LSR, ROXR	*	*	*	0	?	C = Dr – 1
ASR, LSR (r = 0)	—	*	*	0	0	
ROXR (r = 0)	—	*	*	0	?	C = X
ROR	—	*	*		0	?
ROR (r = 0)	—	*	*		0	0

Table 5-3. Condition Code Computations (Continued)

Note: The following notations apply to this table only.

—	=	Not affected	Sm	=	Source operand MSB
U	=	Undefined	Dm	=	Destination operand MSB
?	=	See special definition	Rm	=	Result operand MSB
*	=	General case	R	=	Register tested
X	=	C	n	=	Bit Number
N	=	Rm	r	=	Shift count
Z	=	$\overline{Rm} \Lambda ... \Lambda \overline{R0}$	LB	=	Lower bound
Λ	=	Boolean AND	UB	=	Upper bound
V	=	Boolean OR	$\overline{Rm}$	=	NOT Rm

5.3.3.2 DATA MOVEMENT INSTRUCTIONS. The MOVE instruction is the basic means of transferring and storing address and data. MOVE instructions transfer byte, word, and long-word operands from memory to memory, memory to register, register to memory, and register to register. Address movement instructions (MOVE or MOVEA) transfer word and long-word operands and ensure that only valid address manipulations are executed.

In addition to the general MOVE instructions, there are several special data movement instructions—move multiple registers (MOVEM), move peripheral data (MOVEP), move quick (MOVEQ), exchange registers (EXG), load effective address (LEA), push effective address (PEA), link stack (LINK), and unlink stack (UNLK). Table 5-4 is a summary of the data movement operations.

Table 5-4. Data Movement Operations

Instruction	Operand Syntax	Operand Size	Operation
EXG	Rn, Rn	32	Rn ⇒ Rn
LEA	⟨ea⟩, An	32	⟨ea⟩ ⇒ An
LINK	An, #⟨d⟩	16, 32	SP – 4 ⇒ SP, An ⇒ (SP); SP ⇒ An, SP + d ⇒ SP
MOVE	⟨ea⟩, ⟨ea⟩	8, 16, 32	Source ⇒ Destination
MOVEA	⟨ea⟩, An	16, 32 ⇒ 32	Source ⇒ Destination
MOVEM	list, ⟨ea⟩ ⟨ea⟩, list	16, 32 16, 32 ⇒ 32	Listed registers ⇒ Destination Source ⇒ Listed registers
MOVEP	Dn, (d_{16}, An) (d_{16}, An), Dn	16, 32	Dn [31:24] ⇒ (An + d); Dn [23:16] ⇒ (An + d + 2); Dn [15:8] ⇒ (An + d + 4); Dn [7:0] ⇒ (An + d + 6) (An + d) ⇒ Dn [31:24]; (An + d + 2) ⇒ Dn [23:16]; (An + d + 4) ⇒ Dn [15:8]; (An + d + 6) ⇒ Dn [7:0]
MOVEQ	#⟨data⟩, Dn	8 ⇒ 32	Immediate Data ⇒ Destination
PEA	⟨ea⟩	32	SP – 4 ⇒ SP; ⟨ea⟩ ⇒ SP
UNLK	An	32	An ⇒ SP; (SP) ⇒ An, SP + 4 ⇒ SP

5.3.3.3 INTEGER ARITHMETIC OPERATIONS. The arithmetic operations include the four basic operations of add (ADD), subtract (SUB), multiply (MUL), and divide (DIV) as well as arithmetic compare (CMP, CMPM, CMP2), clear (CLR), and negate (NEG). The instruction set includes ADD, CMP, and SUB instructions for both address and data operations with all

operand sizes valid for data operations. Address operands consist of 16 or 32 bits. The clear and negate instructions apply to all sizes of data operands.

Signed and unsigned MUL and DIV instructions include:

- Word multiply to produce a long-word product
- Long-word multiply to produce a long-word or quad-word product
- Division of a long-word dividend by a word divisor (word quotient and word remainder)
- Division of a long-word or quad-word dividend by a long-word divisor (long-word quotient and long-word remainder)

A set of extended instructions provides multiprecision and mixed-size arithmetic. These instructions are add extended (ADDX), subtract extended (SUBX), sign extend (EXT), and negate binary with extend (NEGX). Refer to Table 5-5 for a summary of the integer arithmetic operations.

Table 5-5. Integer Arithmetic Operations

Instruction	Operand Syntax	Operand Size	Operation
ADD	Dn, ⟨ea⟩ ⟨ea⟩, Dn	8, 16, 32 8, 16, 32	Source + Destination ⇒ Destination
ADDA	⟨ea⟩, An	16, 32	Source + Destination ⇒ Destination
ADDI	#⟨data⟩, ⟨ea⟩	8, 16, 32	Immediate Data + Destination ⇒ Destination
ADDQ	#⟨data⟩, ⟨ea⟩	8, 16, 32	Immediate Data + Destination ⇒ Destination
ADDX	Dn, Dn – (An), – (An)	8, 16, 32 8, 16, 32	Source + Destination + X ⇒ Destination
CLR	⟨ea⟩	8, 16, 32	0 ⇒ Destination
CMP	⟨ea⟩, Dn	8, 16, 32	(Destination – Source), CCR shows results
CMPA	⟨ea⟩, An	16, 32	(Destination – Source), CCR shows results
CMPI	#⟨data⟩, ⟨ea⟩	8, 16, 32	(Destination – Immediate Data), CCR shows results
CMPM	(An) +, (An) +	8, 16, 32	(Destination – Source), CCR shows results
CMP2	⟨ea⟩, Rn	8, 16, 32	Lower bound ≤ Rn ≤ Upper Bound, CCR shows results
DIVS/DIVU DIVSL/DIVUL	⟨ea⟩, Dn ea⟩, Dr:Dq ⟨ea⟩, Dq ⟨ea⟩, Dr:Dq	32/16 ⇒ 16:16 64/32 ⇒ 32:32 32/32 ⇒ 32 32/32 ⇒ 32:32	Destination/Source ⇒ Destination (signed or unsigned)
EXT	Dn Dn	8 ⇒ 16 16 ⇒ 32	Sign Extended Destination ⇒ Destination
EXTB	Dn	8 ⇒ 32	Sign Extended Destination ⇒ Destination
MULS/MULU	⟨ea⟩, Dn ⟨ea⟩, Dl ⟨ea⟩, Dh:Dl	16 × 16 ⇒ 32 32 × 32 ⇒ 32 32 × 32 ⇒ 64	Source × Destination ⇒ Destination (signed or unsigned)
NEG	⟨ea⟩	8, 16, 32	0 – Destination ⇒ Destination
NEGX	⟨ea⟩	8, 16, 32	0 – Destination – X ⇒ Destination
SUB	⟨ea⟩, Dn Dn, ⟨ea⟩	8, 16, 32	Destination – Source ⇒ Destination
SUBA	⟨ea⟩, An	16, 32	Destination – Source ⇒ Destination
SUBI	#⟨data⟩, ⟨ea⟩	8, 16, 32	Destination – Immediate Data ⇒ Destination
SUBQ	#⟨data⟩, ⟨ea⟩	8, 16, 32	Destination – Immediate Data ⇒ Destination
SUBX	Dn, Dn – (An), – (An)	8, 16, 32 8, 16, 32	Destination – Source – X ⇒ Destination
TBLS/TBLU	⟨ea⟩, Dn Dym:Dyn, Dn	8, 16, 32	Dyn – Dym ⇒ Temp (Temp × Dn [7:0]) ⇒ Temp (Dym × 256) + Temp ⇒ Dn
TBLSN/TBLUN	⟨ea⟩, Dn Dym:Dyn, Dn	8, 16, 32	Dyn – Dym ⇒ Temp (Temp × Dn [7:0]) / 256 ⇒ Temp Dym + Temp ⇒ Dn

5.3.3.4 LOGIC INSTRUCTIONS. The logical operation instructions (AND, OR, EOR, and NOT) perform logical operations with all sizes of integer data operands. A similar set of immediate instructions (ANDI, ORI, and EORI) provide these logical operations with all sizes of immediate data. The test (TST) instruction arithmetically compares the operand with zero, placing the result in the CCR. Table 5-6 summarizes the logical operations.

Table 5-6. Logic Operations

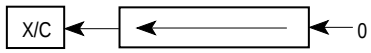
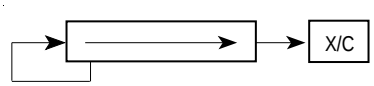
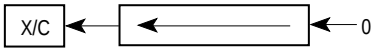
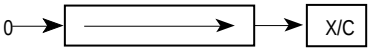
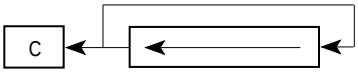
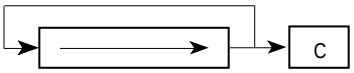
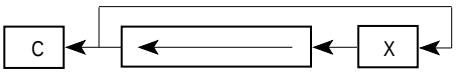
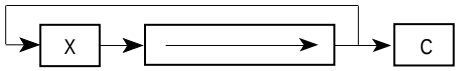
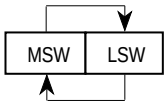
Instruction	Operand Syntax	Operand Size	Operation
AND	⟨ea⟩, Dn Dn, ⟨ea⟩	8, 16, 32 8, 16, 32	Source Λ Destination ⇒ Destination
ANDI	#⟨data⟩, ⟨ea⟩	8, 16, 32	Immediate Data Λ Destination ⇒ Destination
EOR	Dn, ⟨ea⟩	8, 16, 32	Source ⊕ Destination ⇒ Destination
EORI	#⟨data⟩, ⟨ea⟩	8, 16, 32	Immediate Data ⊕ Destination ⇒ Destination
NOT	⟨ea⟩	8, 16, 32	$\overline{\text{Destination}}$ ⇒ Destination
OR	⟨ea⟩, Dn Dn, ⟨ea⟩	8, 16, 32 8, 16, 32	Source V Destination ⇒ Destination
ORI	#⟨data⟩, ⟨ea⟩	8, 16, 32	Immediate Data V Destination ⇒ Destination
TST	⟨ea⟩	8, 16, 32	Source – 0, to set condition codes

5.3.3.5 SHIFT AND ROTATE INSTRUCTIONS. The arithmetic shift instructions, ASR and ASL, and logical shift instructions, LSR and LSL, provide shift operations in both directions. The ROR, ROL, ROXR, and ROXL instructions perform rotate (circular shift) operations, with and without the extend bit. All shift and rotate operations can be performed on either registers or memory.

Register shift and rotate operations shift all operand sizes. The shift count may be specified in the instruction operation word (to shift from 1 to 8 places) or in a register (modulo 64 shift count).

Memory shift and rotate operations shift word-length operands one bit position only. The SWAP instruction exchanges the 16-bit halves of a register. Performance of shift/rotate instructions is enhanced so that use of the ROR and ROL instructions with a shift count of eight allows fast byte swapping. Table 5-7 is a summary of the shift and rotate operations.

Table 5-7. Shift and Rotate Operations

Instruction	Operand Syntax	Operand Size	Operation
ASL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ASR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
LSL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
LSR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROXL	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
ROXR	Dn, Dn #⟨data⟩, Dn ⟨ea⟩	8, 16, 32 8, 16, 32 16	
SWAP	Dn	16	

5.3.3.6 BIT MANIPULATION INSTRUCTIONS. Bit manipulation operations are accomplished using the following instructions: bit test (BTST), bit test and set (BSET), bit test and clear (BCLR), and bit test and change (BCHG). All bit manipulation operations can be performed on either registers or memory. The bit number is specified as immediate data or in a data register. Register operands are 32 bits, and memory operands are 8 bits. Table 5-8 is a summary of bit manipulation instructions.

Table 5-8. Bit Manipulation Operations

Instruction	Operand Syntax	Operand Size	Operation
BCHG	Dn, ⟨ea⟩ #⟨data⟩, ⟨ea⟩	8, 32 8, 32	~(⟨bit number⟩ of destination) ⇒ Z ⇒ bit of destination
BCLR	Dn, ⟨ea⟩ #⟨data⟩, ⟨ea⟩	8, 32 8, 32	~(⟨bit number⟩ of destination) ⇒ Z; 0 ⇒ bit of destination
BSET	Dn, ⟨ea⟩ #⟨data⟩, ⟨ea⟩	8, 32 8, 32	~(⟨bit number⟩ of destination) ⇒ Z; 1 ⇒ bit of destination
BTST	Dn, ⟨ea⟩ #⟨data⟩, ⟨ea⟩	8, 32 8, 32	~(⟨bit number⟩ of destination) ⇒ Z

5.3.3.7 BINARY-CODED DECIMAL (BCD) INSTRUCTIONS. Five instructions support operations on BCD numbers. The arithmetic operations on packed BCD numbers are add decimal with extend (ABCD), subtract decimal with extend (SBCD), and negate decimal with extend (NBCD). Table 5-9 is a summary of the BCD operations.

Table 5-9. Binary-Coded Decimal Operations

Instruction	Operand Syntax	Operand Size	Operation
ABCD	Dn, Dn – (An), – (An)	8 8	$Source_{10} + Destination_{10} + X \Rightarrow Destination$
NBCD	⟨ea⟩	8 8	$0 - Destination_{10} - X \Rightarrow Destination$
SBCD	Dn, Dn – (An), – (An)	8 8	$Destination_{10} - Source_{10} - X \Rightarrow Destination$

5.3.3.8 PROGRAM CONTROL INSTRUCTIONS. A set of subroutine call and return instructions and conditional and unconditional branch instructions perform program control operations. Table 5-10 summarizes these instructions.

Table 5-10. Program Control Operations

Instruction	Operand Syntax	Operand Size	Operation
Conditional			
Bcc	⟨label⟩	8, 16, 32	If condition true, then PC + d ⇒ PC
DBcc	Dn, ⟨label⟩	16	If condition false, then Dn – 1 ⇒ PC; if Dn ≠ (– 1), then PC + d ⇒ PC
Scc	⟨ea⟩	8	If condition true, then destination bits are set to 1; else destination bits are cleared to 0
Unconditional			
BRA	⟨label⟩	8, 16, 32	PC + d ⇒ PC
BSR	⟨label⟩	8, 16, 32	SP – 4 ⇒ SP; PC ⇒ (SP); PC + d ⇒ PC
JMP	⟨ea⟩	none	Destination ⇒ PC
JSR	⟨ea⟩	none	SP – 4 ⇒ SP; PC ⇒ (SP); destination ⇒ PC
NOP	none	none	PC + 2 ⇒ PC
Returns			
RTD	#⟨d⟩	16	(SP) ⇒ PC; SP + 4 + d ⇒ SP
RTR	none	none	(SP) ⇒ CCR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP
RTS	none	none	(SP) ⇒ PC; SP + 4 ⇒ SP

To specify conditions for change in program control, condition codes must be substituted for the letters "cc" in conditional program control opcodes. Condition test mnemonics are given below. Refer to 5.3.3.10 Condition Tests for detailed information on condition codes.

—CC — Carry clear
—CS — Carry set
—EQ — Equal
—F — False*
—GE — Greater or equal
—GT — Greater than
—HI — High
—LE — Less or equal
LS — Low or same
LT — Less than
MI — Minus
NE — Not equal
PL — Plus
T — True
VC — Overflow clear
VS — Overflow set
—*Not applicable to the Bcc instruction

5.3.3.9 SYSTEM CONTROL INSTRUCTIONS. Privileged instructions, trapping instructions, and instructions that use or modify the CCR provide system control operations. All of these instructions cause the processor to flush the instruction pipeline. Table 5-11 summarizes the instructions. The preceding list of condition tests also applies to the TRAPcc instruction. Refer to 5.3.3.10 Condition Tests for detailed information on condition codes.

Table 5-11. System Control Operations

Instruction	Operand Syntax	Operand Size	Operation
Privileged			
ANDI	#⟨data⟩, SR	16	Immediate Data Λ SR ⇒ SR
EORI	#⟨data⟩, SR	16	Immediate Data ⊕ SR ⇒ SR
MOVE	⟨ea⟩, SR SR, ⟨ea⟩	16 16	Source ⇒ SR SR ⇒ Destination
MOVEA	USP, An An, USP	32 32	USP ⇒ An An ⇒ USP
MOVEC	Rc, Rn Rn, Rc	32 32	Rc ⇒ Rn Rn ⇒ Rc
MOVES	Rn, ⟨ea⟩ ⟨ea⟩, Rn	8, 16, 32	Rn ⇒ Destination using DFC Source using SFC ⇒ Rn
ORI	#⟨data⟩, SR	16	Immediate Data V SR ⇒ SR
RESET	none	none	Assert $\overline{\text{RESET}}$ line
RTE	none	none	(SP) ⇒ SR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP; restore stack according to format
STOP	#⟨data⟩	16	Immediate Data ⇒ SR; STOP
LPSTOP	#⟨data⟩	none	Immediate Data ⇒ SR; interrupt mask ⇒ EBI; STOP
Trap Generating			
BKPT	#⟨data⟩	none	If breakpoint cycle acknowledged, then execute returned operation word, else trap as illegal instruction.
BGND	none	none	If background mode enabled, then enter background mode, else format/vector offset ⇒ – (SSP); PC ⇒ – (SSP); SR ⇒ – (SSP); (vector) ⇒ PC
CHK	⟨ea⟩, Dn	16, 32	If Dn < 0 or Dn > (ea), then CHK exception
CHK2	⟨ea⟩, Rn	8, 16, 32	If Rn < lower bound or Rn > upper bound, then CHK exception
ILLEGAL	none	none	SSP – 2 ⇒ SSP; vector offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SSP – 2 ⇒ SSP; SR ⇒ (SSP); Illegal instruction vector address ⇒ PC
TRAP	#⟨data⟩	none	SSP – 2 ⇒ SSP; format/vector offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SR ⇒ (SSP); vector address ⇒ PC
TRAPcc	none #⟨data⟩	none 16, 32	If cc true, then TRAP exception
TRAPV	none	none	If V set, then overflow TRAP exception
Condition Code Register			
ANDI	#⟨data⟩, CCR	8	Immediate Data Λ CCR ⇒ CCR
EORI	#⟨data⟩, CCR	8	Immediate Data ⊕ CCR ⇒ CCR
MOVE	⟨ea⟩, CCR CCR, ⟨ea⟩	16 16	Source ⇒ CCR CCR ⇒ Destination
ORI	#⟨data⟩, CCR	8	Immediate Data V CCR ⇒ CCR

5.3.3.10 CONDITION TESTS. Conditional program control instructions and the TRAPcc instruction execute on the basis of condition tests. A condition test is the evaluation of a logical expression related to the state of the CCR bits. If the result is 1, the condition is true. If

the result is 0, the condition is false. For example, the T condition is always true, and the EQ condition is true only if the Z-bit condition code is true. Table 5-12 lists each condition test.

Table 5-12. Condition Tests

Mnemonic	Condition	Encoding	Test
T	True	0000	1
F*	False	0001	0
HI	High	0010	C • Z
LS	Low or Same	0011	C + Z
CC	Carry Clear	0100	C
CS	Carry Set	0101	C
NE	Not Equal	0110	Z
EQ	Equal	0111	Z
VC	Overflow Clear	1000	V
VS	Overflow Set	1001	V
PL	Plus	1010	N
MI	Minus	1011	N
GE	Greater or Equal	1100	N • V + N • V
LT	Less Than	1101	N • V + N • V
GT	Greater Than	1110	N • V • Z + N • V • Z
LE	Less or Equal	1111	Z + N • V + N • V

* Not available for the Bcc instruction.
•=Boolean AND
+=Boolean OR
N=Boolean NOT

5.3.4 Using the TBL Instructions

There are four TBL instructions. TBLS returns a signed, rounded byte, word, or long-word result. TBLSN returns a signed, unrounded byte, word, or long-word result. TBLU returns an unsigned, rounded byte, word, or long-word result. TBLUN returns an unsigned, unrounded byte, word, or long-word result. All four instructions support two types of interpolation data: an n-element table stored in memory and a two-element range stored in a pair of data registers. The latter form provides a means of performing surface (3D) interpolation between two previously calculated linear interpolations.

The following examples show how to compress tables and use fewer interpolation levels between table entries. Example 1 (see Figure 5-7) demonstrates TBL for a 257-entry table, allowing up to 256 interpolation levels between entries. Example 2 (see Figure 5-8) reduces table length for the same data to four entries. Example 3 (see Figure 5-9) demonstrates use of an 8-bit independent variable with an instruction.

Two additional examples show how TBLSN can reduce cumulative error when multiple table lookup and interpolation operations are used in a calculation. Example 4 demonstrates addition of the results of three table interpolations. Example 5 illustrates use of TBLSN in surface interpolation.

5.3.4.1 TABLE EXAMPLE 1: STANDARD USAGE. The table consists of 257 word entries. As shown in Figure 5-7, the function is linear within the range 32768 ≤ X ≤ 49152. Table entries within this range are as given in Table 5-13 .

Table 5-13. Standard Usage Entries

Entry Number	X-Value	Y-Value
128*	32768	1311
162	41472	1659
163	41728	1669
164	41984	1679
165	42240	1690
192*	49152	1966

*These values are the end points of the range.
All entries between these points fall on the line.

Figure 5-7. Table Example 1

The table instruction is executed with the following bit pattern in Dx:

31 … 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0

Table Entry Offset ⇒ Dx [8:15] = $A3 = 163

Interpolation Fraction ⇒ Dx [0:7] = $80 = 128

Using this information, the table instruction calculates dependent variable Y:

Y = 1669 + (128 (1679 – 1669)) / 256 = 1674

5.3.4.2 TABLE EXAMPLE 2: COMPRESSED TABLE. In Example 2 (see Figure 5-8), the data from Example 1 has been compressed by limiting the maximum value of the independent variable. Instead of the range $0 \leq X = 65535$, X is limited to $0 \leq X \leq 1023$. The table has been compressed to only five entries, but up to 256 levels of interpolation are allowed between entries.

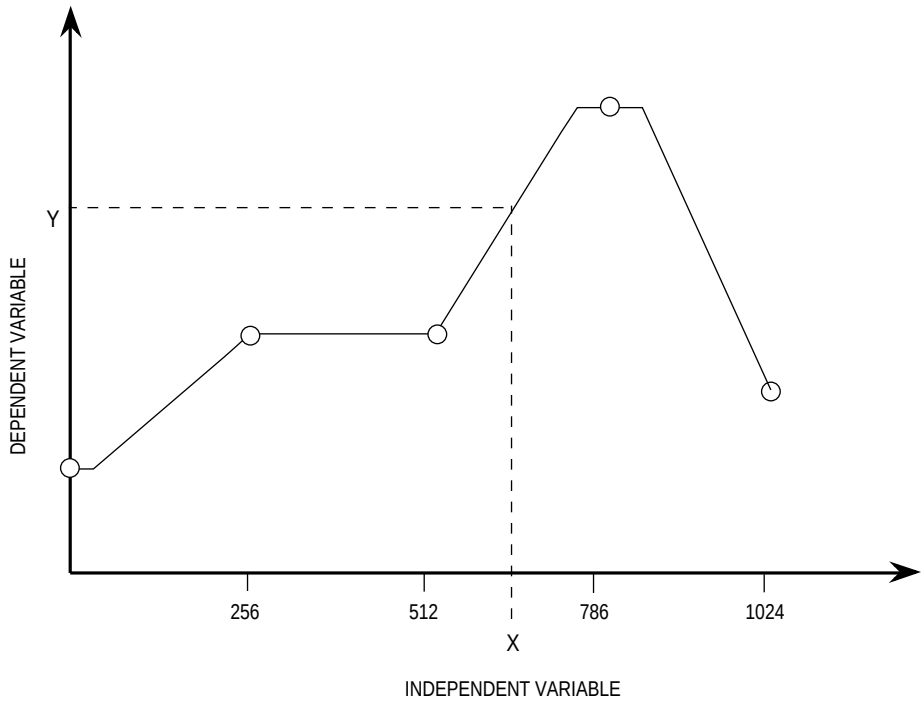


Figure 5-8. Table Example 2

NOTE

Extreme table compression with many levels of interpolation is possible only with highly linear functions. The table entries within the range of interest are listed in Table 5-14.

Table 5-14. Compressed Table Entries

Entry Number	X-Value	Y-Value
2	512	1311
3	786	1966

Since the table is reduced from 257 to 5 entries, independent variable X must be scaled appropriately. In this case the scaling factor is 64, and the scaling is done by a single instruction:

LSR.W #6,Dx

Thus, Dx now contains the following bit pattern:

31 … 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED	0	0	0	0	0	0	1	0	1	0	0	0	1	1	1	0

Table Entry Offset ⇒ Dx [8:15] = $02 = 2

Interpolation Fraction ⇒ Dx [0:7] = $8E = 142

Using this information, the table instruction calculates dependent variable Y:

Y = 1331 + (142 (1966 – 1311)) / 256 = 1674

The function chosen for Examples 1 and 2 is linear between data points. If another function had been used, interpolated values might not have been identical.

5.3.4.3 TABLE EXAMPLE 3: 8-BIT INDEPENDENT VARIABLE. This example shows how to use a table instruction within an interpolation subroutine. Independent variable X is calculated as an 8-bit value, allowing 16 levels of interpolation on a 17-entry table. X is passed to the subroutine, which returns an 8-bit result. The subroutine uses the data listed in Table 5-15, based on the function shown in Figure 5-9.

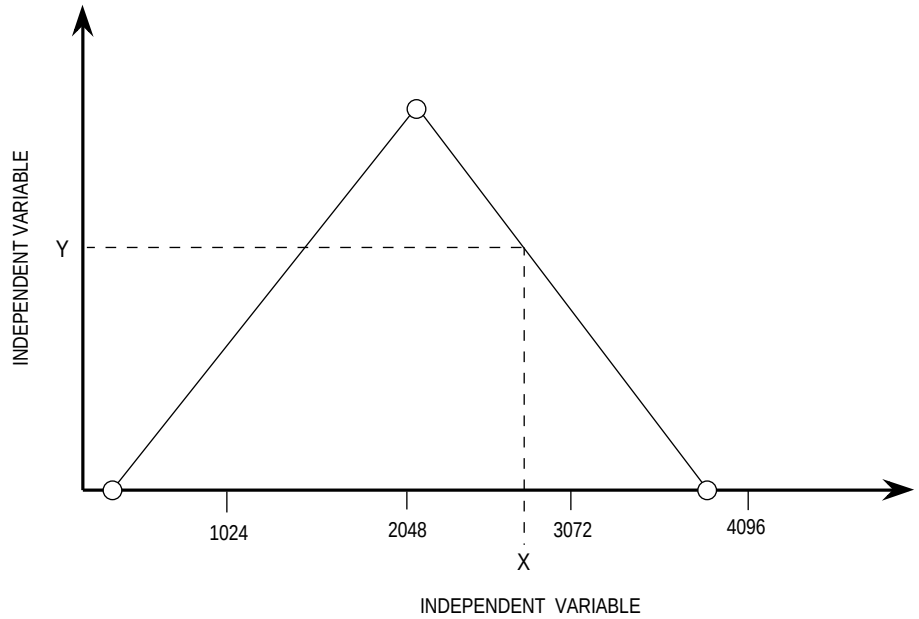


Figure 5-9. Table Example 3

Table 5-15. T8-Bit Independent Variable Entries

X (Subroutine)	X (Instruction)	Y
0	0	0
1	256	16
2	512	32
3	768	48
4	1024	64
5	1280	80
6	1536	96
7	1792	112
8	2048	128
9	2304	112
10	2560	96
11	2816	80
12	3072	64
13	3328	48
14	3584	32
15	3840	16
16	4096	0

The first column is the value passed to the subroutine, the second column is the value expected by the table instruction, and the third column is the result returned by the subroutine.

The following value has been calculated for independent variable X:

31 … 16	15															0
NOT USED	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	1

Since X is an 8-bit value, the upper four bits are used as a table offset, and the lower four bits are used as an interpolation fraction. The following results are obtained from the subroutine:

Table Entry Offset ⇒ Dx [4:7] = $B = 11

Interpolation Fraction ⇒ Dx [0:3] = $D = 13

Thus, Y is calculated as follows:

$Y = 80 + (13\ (64 - 80)) / 16 = 67$

If the 8-bit value for X were used directly by the table instruction, interpolation would be incorrectly performed between entries 0 and 1. Data must be shifted to the left four places before use:

LSL.W #4, Dx

The new range for X is $0 \leq X \leq 4096$; however, since a left shift fills the least significant digits of the word with zeros, the interpolation fraction can only have one of 16 values.

After the shift operation, Dx contains the following value:

31 … 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED	0	0	0	0	1	0	1	1	1	1	0	1	0	0	0	0

Execution of the table instruction using the new value in Dx yields:

Table Entry Offset $\Rightarrow$ Dx [8:15] = $0B = 11

Interpolation Fraction $\Rightarrow$ Dx [0:7] = $D0 = 208

Thus, Y is calculated as follows:

$Y = 80 + (208\ (64 - 80)) / 256 = 67$

5.3.4.4 TABLE EXAMPLE 4: MAINTAINING PRECISION. In this example, three TBL operations are performed and the results are summed. The calculation is done once with the result of each TBL rounded before addition and once with only the final result rounded. Assume that the result of the three interpolations are as follows (a "." indicates the binary radix point).

TBL # 1	0010 0000 . 0111 0000
TBL# 2	0011 1111 . 0111 0000
TBL # 3	0000 0001 . 0111 0000

First, the results of each TBL are rounded with the TBLS round-to-nearest-even algorithm. The following values would be returned by TBLS:

TBL # 1	0010 0000 .
TBL # 2	0011 1111 .
TBL # 3	0000 0001 .

Summing, the following result is obtained:

0010 0000 .
0011 1111 .
0000 0001 .
0110 0000 .

Now, using the same TBL results, the sum is first calculated and then rounded according to the same algorithm:

0010 0000 . 0111 0000
0011 1111 . 0111 0000
0000 0001 . 0111 0000
0110 0001 . 0101 0000

Rounding yields:

0110 0001 .

The second result is preferred. The following code sequence illustrates how addition of a series of table interpolations can be performed without loss of precision in the intermediate results:

```
L0:
TBLSN.B     ⟨ea⟩, Dx
TBLSN.B     ⟨ea⟩, Dx
TBLSN.B     ⟨ea⟩, Dl
ADD.L       Dx, Dm    Long addition avoids problems with carry
ADD.L       Dm, Dl
ASR.L       #8, Dl    Move radix point
BCC.B       L1        Fraction MSB in carry
ADDQ.B      #1, Dl
L1: . . .
```

5.3.4.5 TABLE EXAMPLE 5: SURFACE INTERPOLATIONS. The various forms of table can be used to perform surface (3D) TBLs. However, since the calculation must be split into a series of 2D TBLs, it is possible to lose precision in the intermediate results. The following code sequence, incorporating both TBLS and TBLSN, eliminates this possibility.

```
L0:
MOVE.W      Dx, Dl        Copy entry number and fraction number
TBLSN.B     ⟨ea⟩, Dx
TBLSN.B     ⟨ea⟩, Dl
TBLS.W      Dx:Dl, Dm     Surface interpolation, with round
ASR.L       #8, Dm        Read just the result
BCC.B       L1            No round necessary
ADDQ.B      #1, Dl        Half round up
L1: . . .
```

Before execution of this code sequence, Dx must contain fraction and entry numbers for the two TBL, and Dm must contain the fraction for surface interpolation. The ⟨ea⟩ fields in the TBLSN instructions point to consecutive columns in a 3D table. The TBLS size parameter must be word if the TBLSN size parameter is byte, and must be long word if TBLSN is word. Increased size is necessary because a larger number of significant digits is needed to accommodate the scaled fractional results of the 2D TBL.

5.3.5 Nested Subroutine Calls

The LINK instruction pushes an address onto the stack, saves the stack address at which the address is stored, and reserves an area of the stack for use. Using this instruction in a series of subroutine calls will generate a linked list of stack frames.

The UNLK instruction removes a stack frame from the end of the list by loading an address into the SP and pulling the value at that address from the stack. When the instruction operand is the address of the link address at the bottom of a stack frame, the effect is to remove the stack frame from both the stack and the linked list.

5.3.6 Pipeline Synchronization with the NOP Instruction

Although the no operation (NOP) instruction performs no visible operation, it does force synchronization of the instruction pipeline, since all previous instructions must complete execution before the NOP begins.

5.4 PROCESSING STATES

This section describes the processing states of the CPU32+. It includes a functional description of the bits in the supervisor portion of the SR and an overview of actions taken by the processor in response to exception conditions.

5.4.1 State Transitions

The processor is always in one of four processing states: normal, background, exception, or halted.

When the processor fetches instructions and operands or executes instructions, it is in the normal processing state. The stopped condition, which the processor enters when a STOP or LPSTOP instruction is executed, is a variation of the normal state in which no further bus cycles are generated.

Background state is an alternate operational mode used for system debugging. Refer to 5.6 Development Support for more information.

Exception processing refers specifically to the transition from normal processing of a program to normal processing of system routines, interrupt routines, and other exception handlers. Exception processing includes the stack operations, the exception vector fetch, and the filling of the instruction pipeline caused by an exception. Exception processing ends when execution of an exception handler routine begins. Refer to 5.5 Exception Processing for comprehensive information.

A catastrophic system failure occurs if the processor detects a bus error or generates an address error while in the exception processing state. This type of failure halts the processor. For example, if a bus error occurs during exception processing caused by another bus error, the CPU32+ assumes that the system is not operational and halts.

The halted condition should not be confused with the stopped condition. After the processor executes a STOP or LPSTOP instruction, execution of instructions can resume when a trace, interrupt, or reset exception occurs.

5.4.2 Privilege Levels

To protect system resources, the processor can operate with either of two levels of access—user or supervisor. Supervisor level is more privileged than user level. All instructions are

available at the supervisor level, but execution of some instructions is not permitted at the user level. There are separate SPs for each level. The S-bit in the SR indicates privilege level and determines which SP is used for stack operations. The processor identifies each bus access (supervisor or user mode) via function codes to enforce supervisor and user access levels.

In a typical system, most programs execute at the user level. User programs can access only their own code and data areas and are restricted from accessing other information. The operating system executes at the supervisor privilege level, has access to all resources, performs the overhead tasks for the user level programs, and coordinates their activities.

5.4.2.1 SUPERVISOR PRIVILEGE LEVEL. If the S-bit in the SR is set, supervisor privilege level applies, and all instructions are executable. The bus cycles generated for instructions executed in supervisor level are normally classified as supervisor references, and the values of the function codes on FC2–FC0 refer to supervisor address spaces.

All exception processing is performed at the supervisor level. All bus cycles generated during exception processing are supervisor references, and all stack accesses use the SSP.

Instructions that have important system effects can only be executed at supervisor level. For instance, user programs are not permitted to execute STOP, LPSTOP, or RESET instructions. To prevent a user program from gaining privileged access, except in a controlled manner, instructions that can alter the S-bit in the SR are privileged. The TRAP #n instruction provides controlled user access to operating system services.

5.4.2.2 USER PRIVILEGE LEVEL. If the S-bit in the SR is cleared, the processor executes instructions at the user privilege level. The bus cycles for an instruction executed at the user privilege level are classified as user references, and the values of the function codes on FC2–FC0 specify user address spaces. While the processor is at the user level, implicit references to the system SP and explicit references to address register seven (A7) refer to the USP.

5.4.2.3 CHANGING PRIVILEGE LEVEL. To change from user privilege level to supervisor privilege level, a condition that causes exception processing must occur. When exception processing begins, the current values in the SR, including the S-bit, are saved on the supervisor stack, and then the S-bit is set to enable supervisor access. Execution continues at supervisor privilege level until exception processing is complete.

To return to user access level, a system routine must execute one of the following instructions: MOVE to SR, ANDI to SR, EORI to SR, ORI to SR, or RTE. These instructions execute only at supervisor privilege level and can modify the S-bit of the SR. After these instructions execute, the instruction pipeline is flushed, then refilled from the appropriate address space.

The RTE instruction causes a return to a program that was executing when an exception occurred. When RTE is executed, the exception stack frame saved on the supervisor stack can be restored in either of two ways.

If the frame was generated by an interrupt, breakpoint, trap, or instruction exception, the SR and PC are restored to the values saved on the supervisor stack, and execution resumes at the restored PC address, with access level determined by the S-bit of the restored SR.

If the frame was generated by a bus error or an address error exception, the entire processor state is restored from the stack.

5.5 EXCEPTION PROCESSING

An exception is a special condition that pre-empts normal processing. Exception processing is the transition from normal mode program execution to execution of a routine that deals with an exception. The following paragraphs discuss system resources related to exception handling, exception processing sequence, and specific features of individual exception processing routines.

5.5.1 Exception Vectors

An exception vector is the address of a routine that handles an exception. The VBR contains the base address of a 1024-byte exception vector table, which consists of 256 exception vectors. Sixty-four vectors are defined by the processor, and 192 vectors are reserved for user definition as interrupt vectors. Except for the reset vector, which is two long words, each vector in the table is one long word. Refer to Table 5-16 for information on vector assignment.

All exception vectors, except the reset vector, are located in supervisor data space. The reset vector is located in supervisor program space. Only the initial reset vector is fixed in the processor memory map. When initialization is complete, there are no fixed assignments. Since the VBR stores the vector table base address, the table can be located anywhere in memory. It can also be dynamically relocated for each task executed by an operating system.

Each vector is assigned an 8-bit number. Vector numbers for some exceptions are obtained from an external device; others are supplied by the processor. The processor multiplies the vector number by 4 to calculate vector offset, then adds the offset to the contents of the VBR. The sum is the memory address of the vector.

5.5.1.1 TYPES OF EXCEPTIONS. An exception can be caused by internal or external events.

An internal exception can be generated by an instruction or by an error. The TRAP, TRAPcc, TRAPV, BKPT, CHK, CHK2, RTE, and DIV instructions can cause exceptions during normal execution. Illegal instructions, instruction fetches from odd addresses, word or long-word operand accesses from odd addresses, and privilege violations also cause internal exceptions.

Sources of external exception include interrupts, breakpoints, bus errors, and reset requests. Interrupts are peripheral device requests for processor action. Breakpoints are used to support development equipment. Bus error and reset are used for access control and processor restart.

Table 5-16. Exception Vector Assignments

Vector Number	Vector Offset			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset: Initial Stack Pointer
1	4	004	SP	Reset: Initial Program Counter
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Division
6	24	018	SD	CHK, CHK2 Instructions
7	28	01C	SD	TRAPcc, TRAPV Instructions
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12	48	030	SD	Hardware Breakpoint
13	52	034	SD	(Reserved for Coprocessor Protocol Violation)
14	56	038	SD	Format Error
15	60	03C	SD	Uninitialized Interrupt
16–23	64 92	040 05C	SD	(Unassigned, Reserved) —
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32–47	128 188	080 0BC	SD	Trap Instruction Vectors (0–15) —
48–58	192 232	0C0 0E8	SD	(Reserved for Coprocessor) —
59–63	236 252	0EC 0FC	SD	(Unassigned, Reserved) —
64–255	256 1020	100 3FC	SD	User-Defined Vectors (192)

CAUTION

Because there is no protection on the 64 processor-defined vectors, external devices can access vectors reserved for internal purposes. This practice is strongly discouraged.

5.5.1.2 EXCEPTION PROCESSING SEQUENCE. For all exceptions other than a reset exception, exception processing occurs in the following sequence. Refer to 5.5.2.1 Reset for details of reset processing.

As exception processing begins, the processor makes an internal copy of the SR. After the copy is made, the processor state bits in the SR are changed—the S-bit is set, establishing supervisor access level, and bits T1 and T0 are cleared, disabling tracing. For reset and interrupt exceptions, the interrupt priority mask is also updated.

Next, the exception number is obtained. For interrupts, the number is fetched from CPU space $F (the bus cycle is an interrupt acknowledge). For all other exceptions, internal logic provides a vector number.

Next, current processor status is saved. An exception stack frame is created and placed on the supervisor stack. All stack frames contain copies of the SR and the PC for use by RTE. The type of exception and the context in which the exception occurs determine what other information is stored in the stack frame.

Finally, the processor prepares to resume normal execution of instructions. The exception vector offset is determined by multiplying the vector number by 4, and the offset is added to the contents of the VBR to determine displacement into the exception vector table. The exception vector is loaded into the PC. If no other exception is pending, the processor will resume normal execution at the new address in the PC.

5.5.1.3 EXCEPTION STACK FRAME. During exception processing, the most volatile portion of the current context is saved on the top of the supervisor stack. This context is organized in a format called the exception stack frame.

The exception stack frame always includes the contents of SR and PC at the time the exception occurred. To support generic handlers, the processor also places the vector offset in the exception stack frame and marks the frame with a format code. The format field allows an RTE instruction to identify stack information so that it can be properly restored.

The general form of the exception stack frame is illustrated in Figure 5-10. Although some formats are peculiar to a particular M68000 family processor, format 0000 is always legal and always indicates that only the first four words of a frame are present. See 5.5.4 CPU32+ Stack Frames for a complete discussion of exception stack frames.

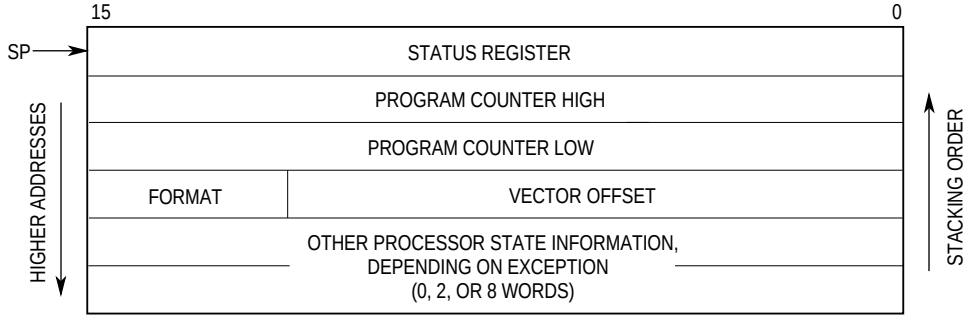


Figure 5-10. Exception Stack Frame

5.5.1.4 MULTIPLE EXCEPTIONS. Each exception has been assigned a priority based on its relative importance to system operation. Priority assignments are shown in Table 5-17. Group 0 exceptions have the highest priorities; group 4 exceptions have the lowest priorities. Exception processing for exceptions that occur simultaneously is done by priority, from highest to lowest.

It is important to be aware of the difference between exception processing mode and execution of an exception handler. Each exception has an assigned vector that points to an associated handler routine. Exception processing includes steps described in 5.5.1.2 Exception Processing Sequence, but does not include execution of handler routines, which is done in normal mode.

When the CPU32+ completes exception processing, it is ready to begin either exception processing for a pending exception or execution of a handler routine. Priority assignment governs the order in which exception processing occurs, not the order in which exception handlers are executed.

Table 5-17. Exception Priority Groups

Group/ Priority	Exception and Relative Priority	Characteristics
0	Reset	Aborts all processing (instruction or exception); does not save old context.
1.1 1.2	Address Error Bus Error	Suspends processing (instruction or exception); saves internal context.
2	BKPT#n, CHK, CHK2, Division by Zero, RTE, TRAP#n, TRAPcc, TRAPV	Exception processing is a part of instruction execution.
3	Illegal Instruction, Line A, Unimplemented Line F, Privilege Violation	Exception processing begins before instruction execution.
4.1 4.2 4.3	Trace Hardware Breakpoint Interrupt	Exception processing begins when current instruction or previous exception processing is complete.

As a general rule, when simultaneous exceptions occur, the handler routines for lower priority exceptions are executed before the handler routines for higher priority exceptions. For example, consider the arrival of an interrupt during execution of a TRAP instruction while tracing is enabled. Trap exception processing (2) is done first, followed immediately by exception processing for the trace (4.1), and then by exception processing for the interrupt (4.3). Each exception places a new context on the stack. When the processor resumes normal instruction execution, it is vectored to the interrupt handler, which returns to the trace handler that returns to the trap handler.

There are special cases to which the general rule does not apply. The reset exception will always be the first exception handled since reset clears all other exceptions. It is also possible for high-priority exception processing to begin before low-priority exception processing is complete. For example, if a bus error occurs during trace exception processing, the bus error will be processed and handled before trace exception processing has completed.

5.5.2 Processing of Specific Exceptions

The following paragraphs provide details concerning sources of specific exceptions, how each arises, and how each is processed.

5.5.2.1 RESET. Assertion of $\overline{\text{RESET}}$ by external hardware or assertion of the internal $\overline{\text{RESET}}$ signal by an internal module causes a reset exception. The reset exception has the highest priority of any exception. Reset is used for system initialization and for recovery from catastrophic failure. When the reset exception is recognized, it aborts any processing in progress, and that processing cannot be recovered. Reset performs the following operations:

1. Clears T0 and T1 in the SR to disable tracing
2. Sets the S-bit in the SR to establish supervisor privilege
3. Sets the interrupt priority mask to the highest priority level (%111)
4. Initializes the VBR to zero ($00000000)
5. Generates a vector number to reference the reset exception vector
6. Loads the first long word of the vector into the interrupt SP
7. Loads the second long word of the vector into the PC
8. Fetches and initiates decode of the first instruction to be executed

Figure 5-11 is a flowchart of the reset exception

After initial instruction prefetches, normal program execution begins at the address in the PC. The reset exception does not save the value of either the PC or the SR.

If a bus error or address error occurs during reset exception processing, a double bus fault occurs, the processor halts, and the $\overline{\text{HALT}}$ signal is asserted to indicate the halted condition.

Execution of the RESET instruction does not cause a reset exception nor does it affect any internal CPU register. The SIM60 registers and the module control register in each internal peripheral module (DMA, timers, and serial modules) are not affected. All other internal peripheral module registers are reset the same as for a hardware reset. The external devices connected to the $\overline{\text{RESET}}$ signal are reset at the completion of the reset instruction

5.5.2.2 BUS ERROR. A bus error exception occurs when an assertion of the $\overline{\text{BERR}}$ signal is acknowledged. The $\overline{\text{BERR}}$ signal can be asserted by one of three sources:

1. External logic by assertion of the $\overline{\text{BERR}}$ input pin
2. Direct assertion of the internal $\overline{\text{BERR}}$ signal by an internal module
3. Direct assertion of the internal $\overline{\text{BERR}}$ signal by the on-chip hardware watchdog after detecting a no-response condition

Bus error exception processing begins when the processor attempts to use information from an aborted bus cycle.

.

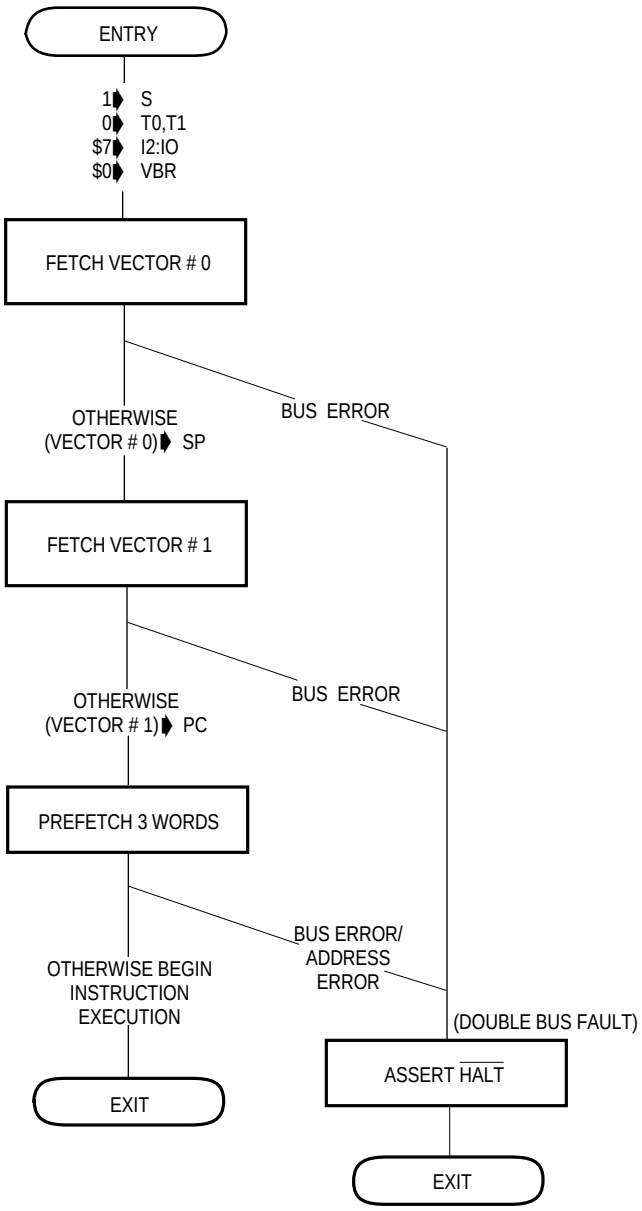


Figure 5-11. Reset Operation Flowchart

When the aborted bus cycle is an instruction prefetch, the processor will not initiate exception processing unless the prefetched information is used. For example, if a branch instruction flushes an aborted prefetch, that word is not accessed, and no exception occurs.

When the aborted bus cycle is a data access, the processor initiates exception processing immediately, except in the case of released operand writes. Released write bus errors are delayed until the next instruction boundary or until another operand access is attempted.

Exception processing for bus error exceptions follows the regular sequence, but context preservation is more involved than for other exceptions because a bus exception can be ini-

tiated while an instruction is executing. Several bus error stack format organizations are utilized to provide additional information regarding the nature of the fault.

First, any register altered by a faulted-instruction EA calculation is restored to its initial value. Then a special status word (SSW) is placed on the stack. The SSW contains specific information about the aborted access—size, type of access (read or write), bus cycle type, and function code. Finally, fault address, bus error exception vector number, PC value, and a copy of the SR are saved.

If a bus error occurs during exception processing for a bus error, an address error, a reset, or while the processor is loading stack information during RTE execution, the processor halts. This simplifies isolation of catastrophic system failure by preventing processor interaction with stacks and memory. Only assertion of $\overline{\text{RESET}}$ can restart a halted processor.

5.5.2.3 ADDRESS ERROR. Address error exceptions occur when the processor attempts to access an instruction, word operand, or long-word operand at an odd address. The effect is much the same as an internally generated bus error. The exception processing sequence is the same as that for bus error, except that the vector number refers to the address error exception vector.

Address error exception processing begins when the processor attempts to use information from the aborted bus cycle. If the aborted cycle is a data space access, exception processing begins when the processor attempts to use the data, except in the case of a released operand write. Released write exceptions are delayed until the next instruction boundary or attempted operand access.

An address exception on a branch to an odd address is delayed until the PC is changed. No exception occurs if the branch is not taken. In this case, the fault address and return PC value placed in the exception stack frame are the odd address, and the current instruction PC points to the instruction that caused the exception.

If an address error occurs during exception processing for a bus error, another address error, or a reset, the processor halts.

5.5.2.4 INSTRUCTION TRAPS. Traps are exceptions caused by instructions. They arise from either processor recognition of abnormal conditions during instruction execution or from use of specific trapping instructions. Traps are generally used to handle abnormal conditions that arise in control routines.

The TRAP instruction, which always forces an exception, is useful for implementing system calls for user programs. The TRAPcc, TRAPV, CHK, and CHK2 instructions force exceptions when a program detects a run-time error. The DIVS and DIVU instructions force an exception if a division operation is attempted with a divisor of zero.

Exception processing for traps follows the regular sequence. If tracing is enabled when an instruction that causes a trap begins execution, a trace exception will be generated by the instruction, but the trap handler routine will not be traced. (The trap exception will be processed first, then the trace exception.)

The vector number for the TRAP instruction is internally generated—part of the number comes from the instruction itself. The trap vector number, PC value, and a copy of the SR are saved on the supervisor stack. The saved PC value is the address of the instruction that follows the instruction that generated the trap. For all instruction traps other than TRAP, a pointer to the instruction causing the trap is also saved in the fifth and sixth words of the exception stack frame.

5.5.2.5 SOFTWARE BREAKPOINTS. To support hardware emulation, the CPU32+ must provide a means of inserting breakpoints into target code and of announcing when a breakpoint is reached.

The MC68000 and MC68008 can detect an illegal instruction inserted at a breakpoint when the processor fetches from the illegal instruction exception vector location. Since the VBR on the CPU32+ allows relocation of exception vectors, the exception vector address is not a reliable indication of a breakpoint. CPU32+ breakpoint support is provided by extending the function of a set of illegal instructions ($4848–$484F).

When a breakpoint instruction is executed, the CPU32+ performs a read from CPU space $0, at a location corresponding to the breakpoint number. If this bus cycle is terminated by $\overline{\text{BERR}}$, the processor performs illegal instruction exception processing. If the bus cycle is terminated by $\overline{\text{DSACK}}$x, the processor uses the data returned to replace the breakpoint in the instruction pipeline and begins execution of that instruction. See Section 4 Bus Operation for a description of CPU space operations.

5.5.2.6 HARDWARE BREAKPOINTS. The CPU32+ recognizes hardware breakpoint requests. Hardware breakpoint requests do not force immediate exception processing, but are left pending. An instruction breakpoint is not made pending until the instruction corresponding to the request is executed.

A pending breakpoint can be acknowledged between instructions or at the end of exception processing. To acknowledge a breakpoint, the CPU performs a read from CPU space $0 at location $1E (see Section 4 Bus Operation).

If the bus cycle terminates normally, instruction execution continues with the next instruction as if no breakpoint request occurred. If the bus cycle is terminated by $\overline{\text{BERR}}$, the CPU begins exception processing. Data returned during this bus cycle is ignored.

Exception processing follows the regular sequence. Vector number 12 (offset $30) is internally generated. The PC of the executing instruction, the PC of the next instruction to be executed, and a copy of the SR are saved on the supervisor stack.

5.5.2.7 FORMAT ERROR. The processor checks certain data values for control operations. The validity of the stack format code and, in the case of a bus cycle fault format, the version number of the processor that generated the frame are checked during execution of the RTE instruction. This check ensures that the program does not make erroneous assumptions about information in the stack frame.

If the format of the control data is improper, the processor generates a format error exception. This exception saves a four-word format exception frame and then vectors through vec-

tor table entry number 14. The stacked PC is the address of the RTE instruction that discovered the format error.

5.5.2.8 ILLEGAL OR UNIMPLEMENTED INSTRUCTIONS. An instruction is illegal if it contains a word bit pattern that does not correspond to the bit pattern of the first word of a legal CPU32+ instruction, if it is a MOVEC instruction that contains an undefined register specification field in the first extension word, or if it contains an indexed addressing mode extension word with bits 5–4 = 00 or bits 3–0 ≠ 0000.

If an illegal instruction is fetched during instruction execution, an illegal instruction exception occurs. This facility allows the operating system to detect program errors or to emulate instructions in software.

Word patterns with bits 15–12 = 1010 (referred to as A-line opcodes) are unimplemented instructions. A separate exception vector (vector 10, offset $28) is given to unimplemented instructions to permit efficient emulation.

Word patterns with bits 15–12 = 1111 (referred to as F-line opcodes) are used for M68000 family instruction set extensions. They can generate an unimplemented instruction exception caused by the first extension word of the instruction or by the addressing mode extension word. A separate F-line emulation vector (vector 11, offset $2C) is used for the exception vector.

All unimplemented instructions are reserved for use by Motorola for enhancements and extensions to the basic M68000 architecture. Opcode pattern $4AFC is defined to be illegal on all M68000 family members. Those customers requiring the use of an unimplemented opcode for synthesis of "custom instructions," operating system calls, etc., should use this opcode.

Exception processing for illegal and unimplemented instructions is similar to that for traps. The instruction is fetched and decoding is attempted. When the processor determines that execution of an illegal instruction is being attempted, exception processing begins. No registers are altered.

Exception processing follows the regular sequence. The vector number is generated to refer to the illegal instruction vector or in the case of an unimplemented instruction, to the corresponding emulation vector. The illegal instruction vector number, current PC, and a copy of the SR are saved on the supervisor stack, with the saved value of the PC being the address of the illegal or unimplemented instruction.

5.5.2.9 PRIVILEGE VIOLATIONS. To provide system security, certain instructions can be executed only at the supervisor access level. An attempt to execute one of these instructions at the user level will cause an exception. The privileged exceptions are as follows:

- AND Immediate to SR
- EOR Immediate to SR
- LPSTOP
- MOVE from SR

- MOVE to SR
- MOVE USP
- MOVEC
- MOVES
- OR Immediate to SR
- RESET
- RTE
- STOP

Exception processing for privilege violations is nearly identical to that for illegal instructions. The instruction is fetched and decoded. If the processor determines that a privilege violation has occurred, exception processing begins before instruction execution.

Exception processing follows the regular sequence. The vector number (8) is generated to reference the privilege violation vector. Privilege violation vector offset, current PC, and SR are saved on the supervisor stack. The saved PC value is the address of the first word of the instruction causing the privilege violation.

5.5.2.10 TRACING. To aid in program development, M68000 processors include a facility to allow tracing of instruction execution. CPU32+ tracing also has the ability to trap on changes in program flow. In trace mode, a trace exception is generated after each instruction executes, allowing a debugging program to monitor the execution of a program under test. The T1 and T0 bits in the supervisor portion of the SR are used to control tracing.

When T1–T0 = 00, tracing is disabled, and instruction execution proceeds normally (see Table 5-18).

Table 5-18. Tracing Control

T1	T0	Tracing Function
0	0	No tracing
0	1	Trace on change of flow
1	0	Trace on instruction execution
1	1	Undefined; reserved

When T1–T0 = 01 at the beginning of instruction execution, a trace exception will be generated if the PC changes sequence during execution. All branches, jumps, subroutine calls, returns, and SR manipulations can be traced in this way. No exception occurs if a branch is not taken.

When T1–T0 = 10 at the beginning of instruction execution, a trace exception will be generated when execution is complete. If the instruction is not executed, either because an interrupt is taken or because the instruction is illegal, unimplemented, or privileged, an exception is not generated.

At the present time, T1–T0 = 11 is an undefined condition. It is reserved by Motorola for future use.

Exception processing for trace starts at the end of normal processing for the traced instruction and before the start of the next instruction. Exception processing follows the regular sequence; tracing is disabled so that the trace exception itself is not traced. A vector number is generated to reference the trace exception vector. The address of the instruction that caused the trace exception, the trace exception vector offset, the current PC, and a copy of the SR are saved on the supervisor stack. The saved value of the PC is the address of the next instruction to be executed.

A trace exception can be viewed as an extension to the function of any instruction. If a trace exception is generated by an instruction, the execution of that instruction is not complete until the trace exception processing associated with it is also complete.

If an instruction is aborted by a bus error or address error exception, trace exception processing is deferred until the suspended instruction is restarted and completed normally. An RTE from a bus error or address error will not be traced because of the possibility of continuing the instruction from the fault.

If an instruction is executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception.

If an instruction forces an exception, the forced exception is processed before the trace exception.

If an instruction is executed and a breakpoint is pending upon completion of the instruction, the trace exception is processed before the breakpoint.

If an attempt is made to execute an illegal, unimplemented, or privileged instruction while tracing is enabled, no trace exception will occur because the instruction is not executed. This is particularly important to an emulation routine that performs an instruction function, adjusts the stacked PC to beyond the unimplemented instruction, and then returns. The SR on the stack must be checked to determine if tracing is on before the return is executed. If tracing is on, trace exception processing must be emulated so that the trace exception handler can account for the emulated instruction.

Tracing also affects normal operation of the STOP and LPSTOP instructions. If either instruction begins execution with T1 set, a trace exception will be taken after the instruction loads the SR. Upon return from the trace handler routine, execution will continue with the instruction following STOP (LPSTOP), and the processor will not enter the stopped condition.

5.5.2.11 INTERRUPTS. There are seven levels of interrupt priority and 192 assignable interrupt vectors within each exception vector table. Careful use of multiple vector tables and hardware chaining will permit a virtually unlimited number of peripherals to interrupt the processor.

Interrupt recognition and subsequent processing are based on internal interrupt request signals ($\overline{IRQ7}$–$\overline{IRQ1}$) and the current priority set in SR priority mask I2–I0. Interrupt request level 0 ($\overline{IRQ7}$–$\overline{IRQ1}$ negated) indicates that no service is requested. When an interrupt of level 1 through 6 is requested via $\overline{IRQ6}$–$\overline{IRQ1}$, the processor compares the request level with the interrupt mask to determine whether the interrupt should be processed. Interrupt requests are inhibited for all priority levels less than or equal to the current priority. Level 7 interrupts are nonmaskable.

$\overline{IRQ7}$–$\overline{IRQ1}$ are synchronized and debounced by input circuitry on two consecutive rising edges of the processor clock.

Interrupt requests do not force immediate exception processing, but are left pending. A pending interrupt is detected between instructions or at the end of exception processing—all interrupt requests must be held asserted until they are acknowledged by the CPU. If the priority of the interrupt is greater than the current priority level, exception processing begins.

Exception processing occurs as follows. First, the processor makes an internal copy of the SR. After the copy is made, the processor state bits in the SR are changed—the S-bit is set, establishing supervisor access level, and bits T1 and T0 are cleared, disabling

tracing. Priority level is then set to the level of the interrupt, and the processor fetches a vector number from the interrupting device (CPU space $F). The fetch bus cycle is classified as an interrupt acknowledge, and the encoded level number of the interrupt is placed on the address bus.

If an interrupting device requests automatic vectoring, the processor generates a vector number (25 to 31) determined by the interrupt level number.

If the response to the interrupt acknowledge bus cycle is a bus error, the interrupt is taken to be spurious, and the spurious interrupt vector number (24) is generated.

The exception vector number, PC, and SR are saved on the supervisor stack. The saved value of the PC is the address of the instruction that would have executed if the interrupt had not occurred.

Priority level 7 interrupt is a special case. Level 7 interrupts are nonmaskable interrupts (NMI). $\overline{IRQ7}$ is a level sensitive input and must remain low until CPU32+ returns a n interrupt acknowledge cycle for level 7 interrupt.

Many M68000 peripherals provide for programmable interrupt vector numbers to be used in the system interrupt request/acknowledge mechanism. If the vector number is not initialized after reset and if the peripheral must acknowledge an interrupt request, the peripheral should return the uninitialized interrupt vector number (15).

See Section 4 Bus Operation for detailed information on interrupt acknowledge cycles.

5.5.2.12 RETURN FROM EXCEPTION. When exception stacking operations for all pending exceptions are complete, the processor begins execution of the handler for the last exception processed. After the exception handler has executed, the processor must restore

the system context in existence prior to the exception. The RTE instruction is designed to accomplish this task.

When RTE is executed, the processor examines the stack frame on top of the supervisor stack to determine if it is valid and determines what type of context restoration must be performed. See 5.5.4 CPU32+ Stack Frames for a description of stack frames.

For a normal four-word frame, the processor updates the SR and PC with data pulled from the stack, increments the SSP by 8, and resumes normal instruction execution. For a six-word frame, the SR and PC are updated from the stack, the active SSP is incremented by 12, and normal instruction execution resumes.

For a bus fault frame, the format value on the stack is first checked for validity. In addition, the version number on the stack must match the version number of the processor that is attempting to read the stack frame. The version number is located in the most significant byte (bits 15–8) of the internal register word at location SP + $14 in the stack frame. The validity check ensures that stack frame data will be properly interpreted in multiprocessor systems.

If a frame is invalid, a format error exception is taken. If it is inaccessible, a bus error exception is taken. Otherwise, the processor reads the entire frame into the proper internal registers, de-allocates the stack (12 words), and resumes normal processing. Bus error frames for faults during exception processing require the RTE instruction to rewrite the faulted stack frame. If an error occurs during any of the bus cycles required by rewrite, the processor halts.

If a format error occurs during RTE execution, the processor creates a normal four-word fault stack frame below the frame that it was attempting to use. If a bus error occurs, a bus-error stack frame will be created. The faulty stack frame remains intact, so that it may be examined and repaired by an exception handler or used by a different type of processor (e.g., MC68010, MC68020, or future M68000 processor) in a multiprocessor system.

5.5.3 Fault Recovery

There are four phases of recovery from a fault: recognizing the fault, saving the processor state, repairing the fault (if possible), and restoring the processor state. Saving and restoring the processor state are described in the following paragraphs.

The stack contents are identified by the special status word (SSW). In addition to identifying the fault type represented by the stack frame, the SSW contains the internal processor state corresponding to the fault.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TP	MV	SZC1	TR	B1	B0	RR	RM	IN	RW	SZC0	SIZ		FUNC		

TP—BERR Frame Type

The TP field defines the class of the faulted bus operation. Two bus error exception frame types are defined. One is for faults on prefetch and operand accesses, and the other is for faults during exception frame stacking.

0 = Operand or prefetch bus fault
1 = Exception processing bus fault

MV—MOVEM in Progress

MV is set when the operand transfer portion of the MOVEM instruction is in progress at the time of a bus fault. If a prefetch bus fault occurs while prefetching the MOVEM opcode and extension word, both the MV and IN bits will be set.

0 = MOVEM was not in progress when fault occurred
1 = MOVEM was in progress when fault occurred

SZC1,SCZ0—Original Operand Size

The SZC1,SZC0 field specifies the size of the original bus cycle (i.e., the size bits of the first cycle, when a transaction is divided into two or three cycles due to bus size or operand address).

00 = Original operand size was long word
01 = Original operand size was byte
10 = Original operand size was word
11 = Unused, reserved

TR—Trace Pending

TR indicates that a trace exception was pending when a bus error exception was processed. The instruction that generated the trace will not be restarted upon return from the exception handler. This includes MOVEM and released write bus errors indicated by the assertion of either MV or RR in the SSW.

0 = Trace not pending
1 = Trace pending

B1—Breakpoint Channel 1 Pending

B1 indicates that a breakpoint exception was pending on channel 1 (external breakpoint source) when a bus error exception was processed. Pending breakpoint status is stacked, regardless of the type of bus error exception.

0 = Breakpoint not pending
1 = Breakpoint pending

B0—Breakpoint Channel 0 Pending

B0 indicates that a breakpoint exception was pending on channel 0 (internal breakpoint source) when the bus error exception was processed. Pending breakpoint status is stacked, regardless of the type of bus error exception.

0 = Breakpoint not pending
1 = Breakpoint pending

RR—Rerun Write Cycle after RTE

RR will be set if the faulted bus cycle was a released write. A released write is one that is overlapped. If the write is completed (rerun) in the exception handler, the RR bit should be cleared before executing RTE. The bus cycle will be rerun if the RR bit is set upon return from the exception handler.

0 = Faulted cycle was read, RMW, or unreleased write
1 = Faulted cycle was a released write

RM—Faulted Cycle Was Read-Modify-Write

Faulted RMW bus cycles set the RM bit. RM is ignored during unstacking.

0 = Faulted cycle was non-RMW cycle
1 = Faulted cycle was either the read or write of an RMW cycle

IN—Instruction/Other

Instruction prefetch faults are distinguished from operand (both read and write) faults by the IN bit. If IN is cleared, the error was on an operand cycle; if IN is set, the error was on an instruction prefetch. IN is ignored during unstacking.

0 = Operand
1 = Prefetch

RW—Read/Write of Faulted Bus Cycle

Read and write bus cycles are distinguished by the RW bit. Read bus cycles will set this bit, and write bus cycles will clear it. RW is reloaded into the bus controller if the RR bit is set during unstacking.

0 = Faulted cycle was an operand write
1 = Faulted cycle was a prefetch or operand read

SIZ—Remaining Size of Faulted Bus Cycle

The SIZ field shows operand size remaining when a fault was detected. This field does not indicate the initial size of the operand, nor does it necessarily indicate the proper status of a dynamically sized bus cycle. Dynamic sizing occurs on the external bus and is transparent to the CPU. Byte size is shown only when the original operand was a byte. The field is reloaded into the bus controller if the RR bit is set during unstacking. The SIZ field is encoded as follows:

00 = Long word
01 = Byte
10 = Word
11 = Unused, reserved

FUNC—Function Code of Faulted Bus Cycle

The function code for the faulted cycle is stacked in the FUNC field of the SSW, which is a copy of FC2–FC0 for the faulted bus cycle. This field is reloaded into the bus controller if the RR bit is set during unstacking. All unused bits are stacked as zeros and are ignored during unstacking. Further discussion of the SSW is included in 5.5.3.1 Types of Faults.

5.5.3.1 TYPES OF FAULTS. An efficient implementation of instruction restart dictates that faults on some bus cycles be treated differently than faults on other bus cycles. The CPU32+ defines four fault types: released write faults, faults during exception processing, faults during MOVEM operand transfer, and faults on any other bus cycle.

5.5.3.1.1 Type I—Released Write Faults. CPU32+ instruction pipelining can cause a final instruction write to overlap the execution of a following instruction. A write that is overlapped is called a released write. A released write fault occurs when a bus error or some other fault occurs on the released write.

Released write faults are taken at the next instruction boundary. The stacked PC is that of the next unexecuted instruction. If a subsequent instruction attempts an operand access while a released write fault is pending, the instruction is aborted and the write fault is acknowledged. This action prevents the instruction from using stale data.

The SSW for a released write fault contains the following bit pattern:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	0
0	0	SZC1	TR	B1	B0	1	0	0	0	SZC0	SIZ		FUNC	

TR , B1, and B0 are set if the corresponding exception is pending when the bus error exception is taken. Status regarding the faulted bus cycle is reflected in the SZCx, SIZ, and FUNC fields.

The remainder of the stack contains the PC of the next unexecuted instruction, the current SR, the address of the faulted memory location, and the contents of the data buffer that was to be written to memory. This data is written on the stack in the format depicted in Figure 5-15. When a released write fault exception handler executes, the machine will complete the faulted write and then continue executing instructions wherever the PC indicates.

5.5.3.1.2 Type II—Prefetch, Operand, RMW, and MOVEP Faults. The majority of bus error exceptions are included in this category—all instruction prefetches, all operand reads, all RMW cycles, and all operand accesses resulting from execution of MOVEP (except the last write of a MOVEP Rn,⟨ea⟩ or the last write of MOVEM, which are type I faults). The TAS, MOVEP, and MOVEM instructions account for all operand writes not considered released write faults.

All type II faults cause an immediate exception that aborts the current instruction. Any registers that were altered as the result of an EA calculation (i.e., postincrement or predecrement) are restored prior to processing the bus cycle fault.

The SSW for faults in this category contains the following bit pattern:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	0
0	0	SZC1	0	B1	B0	0	RM	IN	RW	SZC0	SIZ		FUNC	

The trace pending bit is always cleared since the instruction will be restarted upon return from the handler. Saving a pending exception on the stack causes a trace exception to be

taken prior to restarting the instruction. If the exception handler does not alter the stacked SR trace bits, the trace is requeued when the instruction is started.

The breakpoint pending bits are stacked in the SSW, even though the instruction is restarted upon return from the handler. This avoids problems with bus state analyzer equipment that has been programmed to breakpoint only the first access to a specific location or to count accesses to that location. If this response is not desired, the exception handler can clear the bits before return. The RM, IN, RW, SZCx, FUNC, and SIZ fields reflect the type of bus cycle that caused the fault. If the bus cycle was an RMW, the RM bit will be set, and the RW bit will show whether the fault was on a read or write.

5.5.3.1.3 Type III—Faults During MOVEM Operand Transfer. Bus faults that occur as a result of MOVEM operand transfer are classified as type III faults. MOVEM instruction prefetch faults are type II faults.

Type III faults cause an immediate exception that aborts the current instruction. Registers altered during execution of the faulted instruction are not restored prior to execution of the fault handler. This includes any register predecremented as a result of the effective address calculation or any register overwritten during instruction execution. Since postincremented registers are not updated until the end of an instruction, the register retains its pre-instruction value unless overwritten by operand movement.

The SSW for faults in this category contains the following bit pattern:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	0
0	1	SZC1	TR	B1	B0	RR	0	IN	RW	SZC0	SIZ		FUNC	

MV is set, indicating that MOVEM should be continued from the point where the fault occurred upon return from the exception handler. TR, B1, and B0 are set if a corresponding exception is pending when the bus error exception is taken. IN is set if a bus fault occurs while prefetching an opcode or an extension word during instruction restart. RW, SZCx, SIZ, and FUNC all reflect the type of bus cycle that caused the fault. All write faults have the RR bit set to indicate that the write should be rerun upon return from the exception handler.

The remainder of the stack frame contains sufficient information to continue MOVEM with operand transfer following a faulted transfer. The address of the next operand to be transferred, incremented or decremented by operand size, is stored in the faulted address location ($08). The stacked transfer counter is set to 16 minus the number of transfers attempted (including the faulted cycle). Refer to Figure 5-12 for the stacking format.

5.5.3.1.4 Type IV—Faults During Exception Processing. The fourth type of fault occurs during exception processing. If this exception is a second address or bus error, the machine halts in the double bus fault condition. However, if the exception is one that causes a four- or six-word stack frame to be written, a bus cycle fault frame is written below the faulted exception stack frame.

The SSW for a fault within an exception contains the following bit pattern:

15	14	13	12	11	10	9	8	7	6	5	4 3	2 0
1	0	SZC1	TR	B1	B0	0	0	0	1	SZC0	SIZ	FUNC

TR, B1, and B0 are set if a corresponding exception is pending when the bus error exception is taken.

The contents of the faulted exception stack frame are included in the bus fault stack frame. The pre-exception SR and the format/vector word of the faulted frame are stacked. The type of exception can be determined from the format/vector word. If the faulted exception stack frame contains six words, the PC of the instruction that caused the initial exception is also stacked. This data is placed on the stack in the format shown in Figure 5-13. The return address from the initial exception is stacked for RTE .

5.5.3.2 CORRECTING A FAULT. There are two ways to complete a faulted released write bus cycle. The first is to use a software handler. The second is to rerun the bus cycle via RTE.

Type II fault handlers must terminate with RTE, but specific requirements must also be met before an instruction is restarted.

There are three varieties of type III operand fault recovery. The first is completion of an instruction in software. The second is conversion to type II with restart via RTE. The third is continuation from the fault via RTE.

5.5.3.2.1 Type I—Completing Released Writes via Software. To complete a bus cycle in software, a handler must first read the SSW function code field to determine the appropriate address space, access the fault address pointer on the stack, and then transfer data from the stacked image of the output buffer to the fault address.

If the CPU32+ is configured to 16-bit operation, rather than 32-bit operation, on the internal data bus, long operands require two bus accesses. A fault during the second access of a long operand causes the SZCx bits in the SSW to be set to long word. The SIZ field indicates remaining operand size. If operand coherency is important, the complete operand must be rewritten. After a long operand is rewritten, the RR bit must be cleared. Failure to clear the RR bit can cause the RTE instruction to rerun the bus cycle. Following rewrite, it is not necessary to adjust the PC (or other stack contents) before executing RTE.

5.5.3.2.2 Type I—Completing Released Writes via RTE. An exception handler can use the RTE instruction to complete a faulted bus cycle. When RTE executes, the fault address, data output buffer, PC, and SR are restored from the stack. Any pending breakpoint or trace exceptions, as indicated by TR, B1, and B0 in the stacked SSW, are requeued during SSW restoration. The RR bit in the SSW is checked during the unstacking operation; if it is set, the RW, FUNC, and SIZ fields are restored and the released write cycle is rerun.

To maintain long-word operand coherence, stack contents must be adjusted prior to the RTE execution. The fault address must be decremented by 2 if the SZCx bits are set to long

word and SIZ indicates a remaining byte or word. SIZ must be set to long. All other fields should be left unchanged. The bus controller uses the modified fault address and SIZ field to rerun the complete released write cycle.

Manipulating the stacked SSW can cause unpredictable results because RTE checks only the RR bit to determine if a bus cycle must be rerun. Inadvertent alteration of the control bits could cause the bus cycle to be a read instead of a write or could cause access to a different address space than the original bus cycle. If the rerun bus cycle is a read, returned data will be ignored.

5.5.3.2.3 Type II—Correcting Faults via RTE. Instructions aborted because of a type II fault are restarted upon return from the exception handler. A fault handler must establish safe restart conditions. If a fault is caused by a nonresident page in a demand-paged virtual memory configuration, the fault address must be read from the stack, and the appropriate page retrieved. An RTE instruction terminates the exception handler. After unstacking the machine state, the instruction is refetched and restarted.

5.5.3.2.4 Type III—Correcting Faults via Software. Sufficient information is contained in the stack frame to complete MOVEM in software. After the cause of the fault is corrected, the faulted bus cycle must be rerun. Perform the following procedures to complete an instruction through software:

A. Set Up for Rerun

1. Read the MOVEM opcode and extension from locations pointed to by stack frame PC and PC + 2. The EA need not be recalculated since the next operand address is saved in the stack frame. However, the opcode EA field must be examined to determine how to update the address register and PC when the instruction is complete.
2. Adjust the mask to account for operands already transferred. Subtract the stacked operand transfer count from 16 to obtain the number of operands transferred. Scan the mask using this count value. Each time a set bit is found, clear it and decrement the counter. When the count is zero, the mask is ready for use.
3. Adjust the operand address. If the predecrement addressing mode is in effect, subtract the operand size from the stacked value; otherwise, add the operand size to the stacked value.

B. Rerun Instruction

1. Scan the mask for set bits. Read/write the selected register from/to the operand address as each bit is found.
2. As each operand is transferred, clear the mask bit and increment (decrement) the operand address. When all bits in the mask are cleared, all operands have been transferred.
3. If the addressing mode is predecrement or postincrement, update the register to complete the execution of the instruction.
4. If TR is set in the stacked SSW, create a six-word stack frame and execute the trace handler. If either B1 or B0 is set in the SSW, create another six-word stack frame and execute the hardware breakpoint handler.

5. De-allocate the stack and return control to the faulted program.

5.5.3.2.5 Type III—Correcting Faults by Conversion and Restart. In some situations it may be necessary to rerun all the operand transfers for a faulted instruction rather than continue from a faulted operand. Clearing the MV bit in the stacked SSW converts a type III fault into a type II fault. Consequently, MOVEM, like all other type II exceptions, will be restarted upon return from the exception handler. When a fault occurs after an operand has transferred, that transfer is not "undone". However, these memory locations are accessed a second time when the instruction is restarted. If a register used in an EA calculation is overwritten before a fault occurs, an incorrect EA is calculated upon instruction restart.

5.5.3.2.6 Type III—Correcting Faults via RTE. The preferred method of MOVEM bus fault recovery is to correct the cause of the fault and then execute an RTE instruction without altering the stack contents.

The RTE recognizes that MOVEM was in progress when a fault occurred, restores the appropriate machine state, refetches the instruction, repeats the faulted transfer, and continues the instruction.

MOVEM is the only instruction continued upon return from an exception handler. Although the instruction is refetched, the EA is not recalculated, and the mask is rescanned the same number of times as before the fault. Modifying the code prior to RTE can cause unexpected results.

5.5.3.2.7 Type IV—Correcting Faults via Software. Bus error exceptions can occur during exception processing while the processor is fetching an exception vector or while it is stacking. The same stack frame and SSW are used in both cases, but each has a distinct fault address. The stacked faulted exception format/vector word identifies the type of faulted exception and the contents of the remainder of the frame. A fault address corresponding to the vector specified in the stacked format/vector word indicates that the processor could not obtain the address of the exception handler.

A bus error exception handler should execute RTE after correcting a fault. RTE restores the internal machine state, fetches the address of the original exception handler, recreates the original exception stack frame, and resumes execution at the exception handler address.

If the fault is intractable, the exception handler should rewrite the faulted exception stack frame at SP + $14 + $06 and then jump directly to the original exception handler. The stack frame can be generated from the information in the bus error frame: the pre-exception SR (SP + $0C), the format/vector word (SP + $0E), and, if the frame being written is a six-word frame, the PC of the instruction causing the exception (SP + $10). The return PC value is available at SP + $02.

A stacked fault address equal to the current SP may indicate that, although the first exception received a bus error while stacking, the bus error exception stacking successfully completed. This occurrence is extremely improbable, but the CPU32+ supports recovery from it. Once the exception handler determines that the fault has been corrected, recovery can proceed as described previously. If the fault cannot be corrected, move the supervisor stack to another area of memory, copy all valid stack frames to the new stack, create a faulted

exception frame on top of the stack, and resume execution at the exception handler address.

5.5.4 CPU32+ Stack Frames

The CPU32+ generates three different stack frames: four-word frames, six-word frames, and twelve-word bus error frames.

5.5.4.1 FOUR-WORD STACK FRAME. This stack frame is created by interrupt, format error, TRAP #n, illegal instruction, A-line and F-line emulator trap, and privilege violation exceptions. Depending on the exception type, the PC value is either the address of the next instruction to be executed or the address of the instruction that caused the exception (see Figure 5-12).

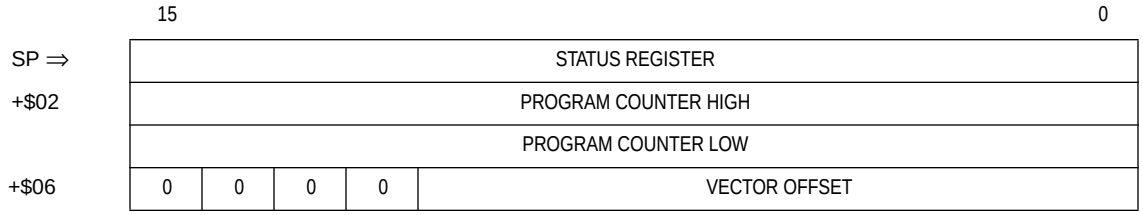


Figure 5-12. Format $0—Four-Word Stack Frame

5.5.4.2 SIX-WORD STACK FRAME. This stack frame (see Figure 5-13) is created by instruction-related traps, which include CHK, CHK2, TRAPcc, TRAPV, and divide-by-zero, and by trace exceptions. The faulted instruction PC value is the address of the instruction that caused the exception. The next PC value (the address to which RTE returns) is the address of the next instruction to be executed.

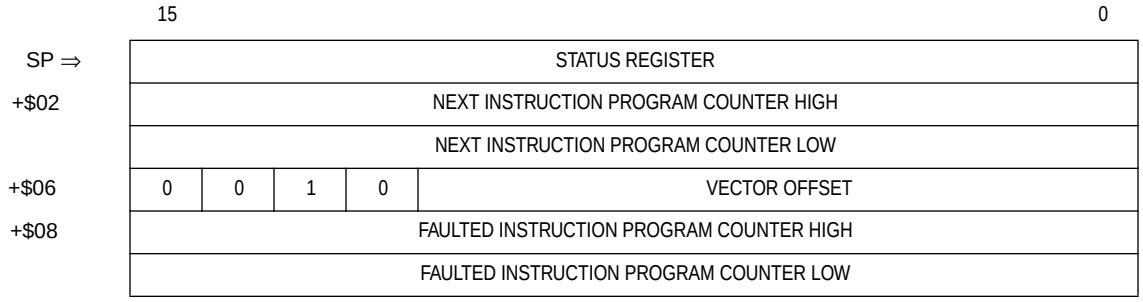


Figure 5-13. Format $2—Six-Word Stack Frame

Hardware breakpoints also utilize this format. The faulted instruction PC value is the address of the instruction executing when the breakpoint was sensed. Usually this is the address of the instruction that caused the breakpoint, but, because released writes can overlap following instructions, the faulted instruction PC may point to an instruction following the instruction that caused the breakpoint. The address to which RTE returns is the address of the next instruction to be executed.

5.5.4.3 BUS ERROR STACK FRAME. This stack frame is created when a bus cycle fault is detected. The CPU32+ bus error stack frame differs significantly from the equivalent stack

frames of other M68000 family members. The only internal machine state required in the CPU32+ stack frame is the bus controller state at the time of the error and a single register.

Bus operation in progress at the time of a fault is conveyed by the SSW.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TP	MV	SZC1	TR	B1	B0	RR	RM	IN	RW	SZC0	SIZ		FUNC		

The bus error stack frame is 12 words in length. There are three variations of the frame, each distinguished by different values in the SSW TP and MV fields.

An internal transfer count register appears at location SP + $14 in all bus error stack frames. The register contains an 8-bit microcode revision number and, for type III faults, an 8-bit transfer count. Register format is shown in Figure 5-14.

15	8	7	0
MICROCODE REVISION NUMBER		TRANSFER COUNT	

Figure 5-14. Internal Transfer Count Register

The microcode revision number is checked before a bus error stack frame is restored via RTE. In a multiprocessor system, this check ensures that a processor using stacked information is at the same revision level as the processor that created it.

The transfer count is ignored unless the MV bit in the stacked SSW is set. If the MV bit is set, the least significant byte of the internal register is reloaded into the MOVEM transfer counter during RTE execution.

For faults occurring during normal instruction execution (both prefetches and non-MOVEM operand accesses), SSW TP,MV = 00. Stack frame format is shown in Figure 5-15.

Faults that occur during the operand portion of the MOVEM instruction are identified by SSW TP,MV = 01. Stack frame format is shown in Figure 5-16.

When a bus error occurs during exception processing, SSW TP,MV = 10. The frame shown in Figure 5-17 is written below the faulting frame. Stacking begins at the address pointed to by SP – 6 (SP value is the value before initial stacking on the faulted frame).

The frame can have either four or six words, depending on the type of error. Four-word stack frames do not include the faulted instruction PC. (The internal transfer count register is located at SP + $10 and the SSW is located at SP + $12.)

The fault address of a dynamically sized bus cycle is the address of the upper byte, regardless of the byte that caused the error.

	15					0
SP ⇒	STATUS REGISTER					
+$02	RETURN PROGRAM COUNTER HIGH					
	RETURN PROGRAM COUNTER LOW					
+$06	1	1	0	0	VECTOR OFFSET	
+$08	FAULTED ADDRESS HIGH					
	FAULTED ADDRESS LOW					
+$0C	DBUF HIGH					
	DBUF LOW					
+$10	CURRENT INSTRUCTION PROGRAM COUNTER HIGH					
	CURRENT INSTRUCTION PROGRAM COUNTER LOW					
+$14	INTERNAL TRANSFER COUNT REGISTER					
+$16	0	0	SPECIAL STATUS WORD			

Figure 5-15. Format $C—BERR Stack for Prefetches and Operands

	15					0
SP ⇒	STATUS REGISTER					
+$02	RETURN PROGRAM COUNTER HIGH					
	RETURN PROGRAM COUNTER LOW					
+$06	1	1	0	0	VECTOR OFFSET	
+$08	FAULTED ADDRESS HIGH					
	FAULTED ADDRESS LOW					
+$0C	DBUF HIGH					
	DBUF LOW					
+$10	CURRENT INSTRUCTION PROGRAM COUNTER HIGH					
	CURRENT INSTRUCTION PROGRAM COUNTER LOW					
+$14	INTERNAL TRANSFER COUNT REGISTER					
+$16	0	1	SPECIAL STATUS WORD			

Figure 5-16. Format $C—BERR Stack on MOVEM Operand

	15					0
SP ⇒	STATUS REGISTER					
+$02	NEXT INSTRUCTION PROGRAM COUNTER HIGH					
	NEXT INSTRUCTION PROGRAM COUNTER LOW					
+$06	1	1	0	0	VECTOR OFFSET	
+$08	FAULTED ADDRESS HIGH					
	FAULTED ADDRESS LOW					
+$0C	PRE-EXCEPTION STATUS REGISTER					
	FAULTED EXCEPTION FORMAT/VECTOR WORD					
+$10	FAULTED INSTRUCTION PROGRAM COUNTER HIGH (SIX WORD FRAME ONLY)					
	FAULTED INSTRUCTION PROGRAM COUNTER LOW (SIX WORD FRAME ONLY)					
+$14	INTERNAL TRANSFER COUNT REGISTER					
+$16	1	0	SPECIAL STATUS WORD			

Figure 5-17. Format $C—Four- and Six-Word BERR Stack

5.6 DEVELOPMENT SUPPORT

All M68000 family members have the following special features that facilitate applications development.

Trace on Instruction Execution—All M68000 processors include an instruction-by-instruction tracing facility to aid in program development. The MC68020, MC68030, and CPU32+ can also trace those instructions that change program flow. In trace mode, an exception is generated after each instruction is executed, allowing a debugger program to monitor execution of a program under test. See 5.5.2.10 Tracing for more information.

Breakpoint Instruction—An emulator can insert software breakpoints into target code to indicate when a breakpoint occurs. On the MC68010, MC68020, MC68030, and CPU32+, this function is provided via illegal instructions ($4848–$484F) that serve as breakpoint instructions. See 5.5.2.5 Software Breakpoints for more information.

Unimplemented Instruction Emulation—When an attempt is made to execute an illegal instruction, an illegal instruction exception occurs. Unimplemented instructions (F-line, A-line) utilize separate exception vectors to permit efficient emulation of unimplemented instructions in software. See 5.5.2.8 Illegal or Unimplemented Instructions for more information.

5.6.1 CPU32+ Integrated Development Support

In addition to standard MC68000 family capabilities, the CPU32+ has features to support advanced integrated system development. These features include background debug mode, deterministic opcode tracking, hardware breakpoints, and internal visibility in a single-chip environment.

5.6.1.1 BACKGROUND DEBUG MODE (BDM) OVERVIEW. Microprocessor systems generally provide a debugger, implemented in software, for system analysis at the lowest level. The BDM on the CPU32+ is unique because the debugger is implemented in CPU microcode.

BDM incorporates a full set of debug options—registers can be viewed and/or altered, memory can be read or written, and test features can be invoked.

A resident debugger simplifies implementation of an in-circuit emulator. In a common setup (see Figure 5-18), emulator hardware replaces the target system processor. A complex, expensive pod-and-cable interface provides a communication path between target system and emulator.

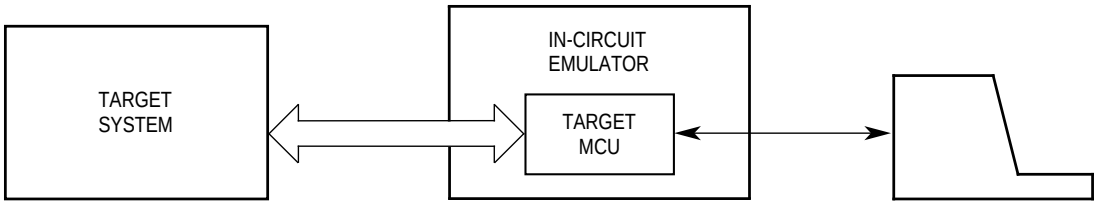


Figure 5-18. In-Circuit Emulator Configuration

By contrast, an integrated debugger supports use of a bus state analyzer (BSA) for in-circuit emulation. The processor remains in the target system (see Figure 5-19), and the interface is simplified. The BSA monitors target processor operation and the on-chip debugger controls the operating environment. Emulation is much closer to target hardware; thus, many interfacing problems (i.e., limitations on high-frequency operation, AC and DC parametric mismatches, and restrictions on cable length) are minimized.

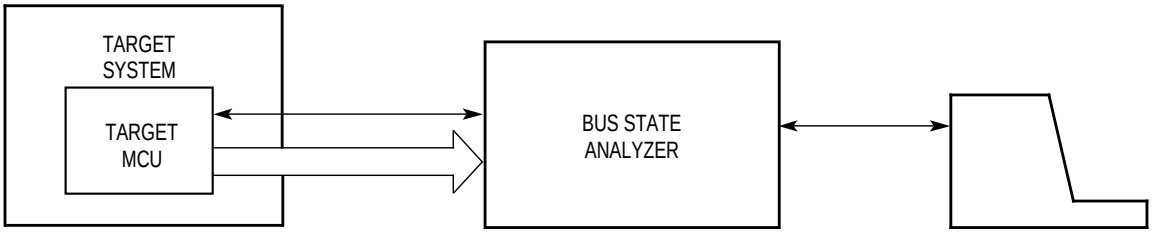


Figure 5-19. Bus State Analyzer Configuration

5.6.1.2 DETERMINISTIC OPCODE TRACKING OVERVIEW. CPU32+ function code outputs are augmented by three supplementary signals that monitor the instruction pipeline. The $\overline{\text{IFETCH}}$ output signal identifies bus cycles in which data is loaded into the pipeline and signals pipeline flushes. The $\overline{\text{IPIPE1}}$, $\overline{\text{IPIPE0}}$ output signals indicate when each mid-instruction pipeline advance occurs and when instruction execution begins. These signals allow a BSA to synchronize with instruction stream activity. Refer to 5.6.3 Deterministic Opcode Tracking for complete information.

5.6.1.3 ON-CHIP HARDWARE BREAKPOINT OVERVIEW. An external breakpoint input and an on-chip hardware breakpoint capability permit breakpoint trap on any memory access. Off-chip address comparators will not detect breakpoints on internal accesses unless show cycles are enabled. Breakpoints on prefetched instructions, which are flushed from the pipeline before execution, are not acknowledged, but operand breakpoints are always acknowledged. Acknowledged breakpoints can initiate either exception processing or BDM. See 5.5.2.6 Hardware Breakpoints for more information.

5.6.2 Background Debug Mode

BDM is an alternate CPU32+ operating mode. During BDM, normal instruction execution is suspended, and special microcode performs debugging functions under external control. Figure 5-20 is a BDM block diagram.

BDM can be initiated in several ways—by externally generated breakpoints, by internal peripheral breakpoints, by the background instruction (BGND), or by catastrophic exception conditions. While in BDM, the CPU32+ ceases to fetch instructions via the parallel bus and communicates with the development system via a dedicated, high-speed, SPI-type serial command interface.

5.6.2.1 ENABLING BDM. Accidentally entering BDM in a nondevelopment environment could lock up the CPU32+ since the serial command interface would probably not be available. For this reason, BDM is enabled during reset via the $\overline{\text{BKPT}}$ signal.

BDM operation is enabled when $\overline{\text{BKPT}}$ is asserted (low) at the rising edge of $\overline{\text{RESET}}$. BDM remains enabled until the next system reset. A high $\overline{\text{BKPT}}$ on the trailing edge of $\overline{\text{RESET}}$ disables BDM. $\overline{\text{BKPT}}$ is relatched on each rising transition of $\overline{\text{RESET}}$. $\overline{\text{BKPT}}$ is synchronized internally and must be held low for at least two clock(four clocks for $\overline{\text{RESETS}}$)cycles prior to negation of $\overline{\text{RESETH}}$.

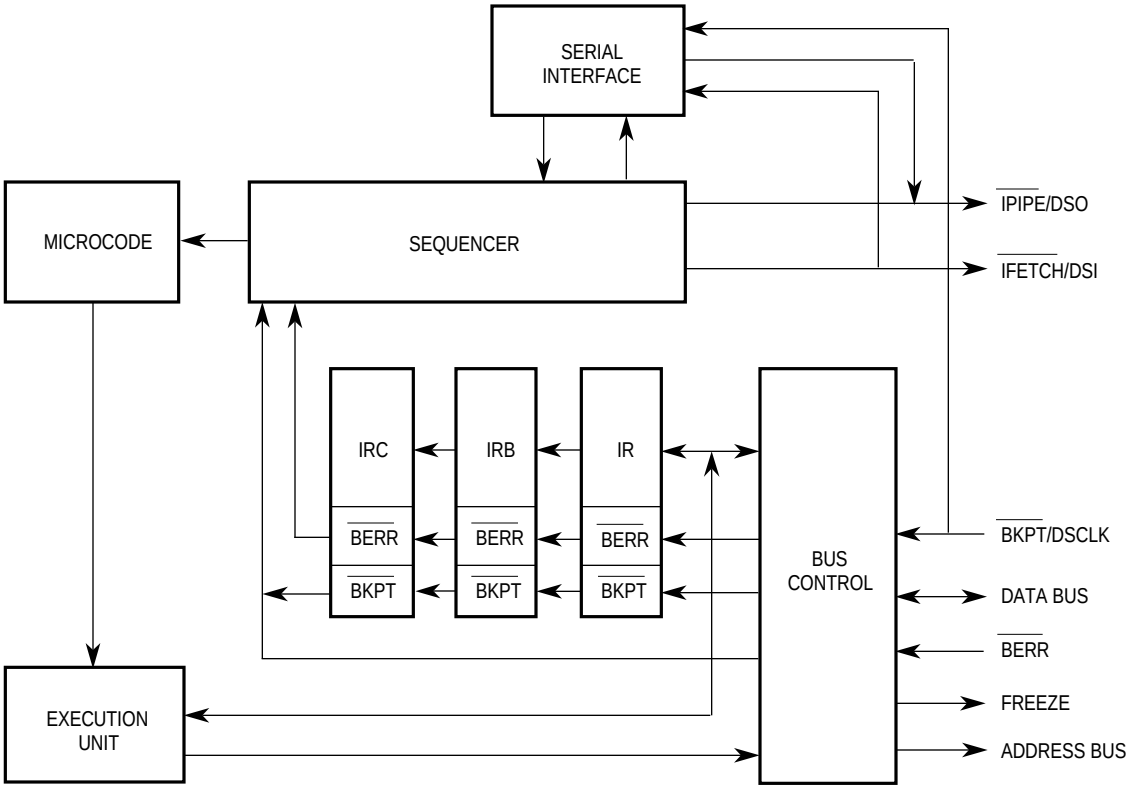


Figure 5-20. BDM Block Diagram

BDM enable logic must be designed with special care. If hold time on $\overline{\text{BKPT}}$ (after the trailing edge of $\overline{\text{RESET}}$) extends into the first bus cycle following reset, this bus cycle could be tagged with a breakpoint. Refer to Section 4 Bus Operation for timing information.

5.6.2.2 BDM SOURCES. When BDM is enabled, any of several sources can cause the transition from normal mode to BDM. These sources include external $\overline{\text{BKPT}}$ hardware, the BGND instruction, a double bus fault, and internal peripheral breakpoints. If BDM is not enabled when an exception condition occurs, the exception is processed normally. Table 5-19 summarizes the processing of each source for both enabled and disabled cases. Note that the BKPT instruction never causes a transition into BDM.

Table 5-19. BDM Source Summary

Source	BDM Enabled	BDM Disabled
BKPT	Background	Breakpoint Exception
Double Bus Fault	Background	Halted
BGND Instruction	Background	Illegal Instruction
BKPT Instruction	Opcode Substitution/ Illegal Instruction	Opcode Substitution/ Illegal Instruction

5.6.2.2.1 External $\overline{\text{BKPT}}$ Signal. Once enabled, BDM is initiated whenever assertion of $\overline{\text{BKPT}}$ is acknowledged. If BDM is disabled, a breakpoint exception (vector $0C) is acknowledged. The $\overline{\text{BKPT}}$ input has the same timing relationship to the data strobe trailing edge as read cycle data. There is no breakpoint acknowledge bus cycle when BDM is entered.

5.6.2.2.2 BGND Instruction. An illegal instruction, $4AFA, is reserved for use by development tools. The CPU32+ defines $4AFA (BGND) to be a BDM entry point when BDM is enabled. If BDM is disabled, an illegal instruction trap is acknowledged. Illegal instruction traps are discussed in 5.5.2.8 Illegal or Unimplemented Instructions.

5.6.2.2.3 Double Bus Fault. The CPU32+ normally treats a double bus fault (two bus faults in succession) as a catastrophic system error and halts. When this condition occurs during initial system debug (a fault in the reset logic), further debugging is impossible until the problem is corrected. In BDM, the fault can be temporarily bypassed so that its origin can be isolated and eliminated.

5.6.2.3 ENTERING BDM. When the processor detects a $\overline{\text{BKPT}}$ or a double bus fault or decodes a BGND instruction, it suspends instruction execution and asserts the FREEZE output. FREEZE assertion is the first indication that the processor has entered BDM. Once FREEZE has been asserted, the CPU enables the serial communication hardware and awaits a command.

The CPU writes a unique value indicating the source of BDM transition into temporary register A (ATEMP) as part of the process of entering BDM. A user can poll ATEMP and determine the source (see Table 5-20) by issuing a read system register command (RSREG). ATEMP is used in most debugger commands for temporary storage—it is imperative that the RSREG command be the first command issued after transition into BDM.

Table 5-20. Polling the BDM Entry Source

Source	ATEMP 31–16	ATEMP 15–0
Double Bus Fault	SSW*	$FFFF
BGND Instruction	$0000	$0001
Hardware Breakpoint	$0000	$0000

*SSW is described in detail in 5.5.3 Fault Recovery.

A double bus fault during initial SP/PC fetch sequence is distinguished by a value of $FFFFFFFF in the current instruction PC. At no other time will the processor write an odd value into this register.

5.6.2.4 COMMAND EXECUTION. Figure 5-21 summarizes BDM command execution. Commands consist of one 16-bit operation word and can include one or more 16-bit extension words. Each incoming word is read as it is assembled by the serial interface. The microcode routine corresponding to a command is executed as soon as the command is complete. Result operands are loaded into the output shift register to be shifted out as the next command is read. This process is repeated for each command until the CPU returns to normal operating mode.

5.6.2.5 BDM REGISTERS. BDM processing uses three special-purpose registers to track program context during development. A description of each register follows.

5.6.2.5.1 Fault Address Register (FAR). The FAR contains the address of the faulting bus cycle immediately following a bus or address error. This address remains available until overwritten by a subsequent bus cycle. Following a double bus fault, the FAR contains the address of the last bus cycle. The address of the first fault (if one occurred) is not visible to the user.

5.6.2.5.2 Return Program Counter (RPC). The RPC points to the location where fetching will commence after transition from BDM to normal mode. This register should be accessed to change the flow of a program under development. Changing the RPC to an odd value will cause an address error when normal mode prefetching begins.

5.6.2.5.3 Current Instruction Program Counter (PCC). The PCC holds a pointer to the first word of the last instruction executed prior to transition into BDM. Due to instruction pipelining, the instruction pointed to may not be the instruction that caused the transition. An example is a breakpoint on a released write. The bus cycle may overlap as many as two subsequent instructions before stalling the instruction sequencer. A $\overline{\text{BKPT}}$ asserted during this cycle will not be acknowledged until the end of the instruction executing at completion of the bus cycle. PCC will contain $00000001 if BDM is entered via a double bus fault immediately out of reset.

5.6.2.6 RETURNING FROM BDM. BDM is terminated when a resume execution (GO) or call user code (CALL) command is received. Both GO and CALL flush the instruction pipeline and prefetch instructions from the location pointed to by the RPC.

The return PC and the memory space referred to by the SR SUPV bit reflect any changes made during BDM. FREEZE is negated prior to initiating the first prefetch. Upon negation of FREEZE, the serial subsystem is disabled, and the signals revert to $\overline{\text{IPIPE}}$ and $\overline{\text{IFETCH}}$ functionality.

5.6.2.7 SERIAL INTERFACE. Communication with the CPU32+ during BDM occurs via a dedicated serial interface, which shares pins with other development features. The $\overline{\text{BKPT}}$ signal becomes the DSCLK; DSI is received on $\overline{\text{IFETCH}}$, and DSO is transmitted on $\overline{\text{IPIPE}}$.

The serial interface uses a full-duplex synchronous protocol similar to the serial peripheral interface (SPI) protocol. The development system serves as the master of the serial link since it is responsible for the generation of DSCLK. If DSCLK is derived from the CPU32+ system clock, development system serial logic is unhindered by the operating frequency of the target processor. Operable frequency range of the serial clock is from DC to one-half the processor system clock frequency

The serial interface operates in full-duplex mode—i.e., data is transmitted and received simultaneously by both master and slave devices. In general, data transitions occur on the falling edge of DSCLK and are stable by the following rising edge of DSCLK. Data is transmitted MSB first and is latched on the rising edge of DSCLK.

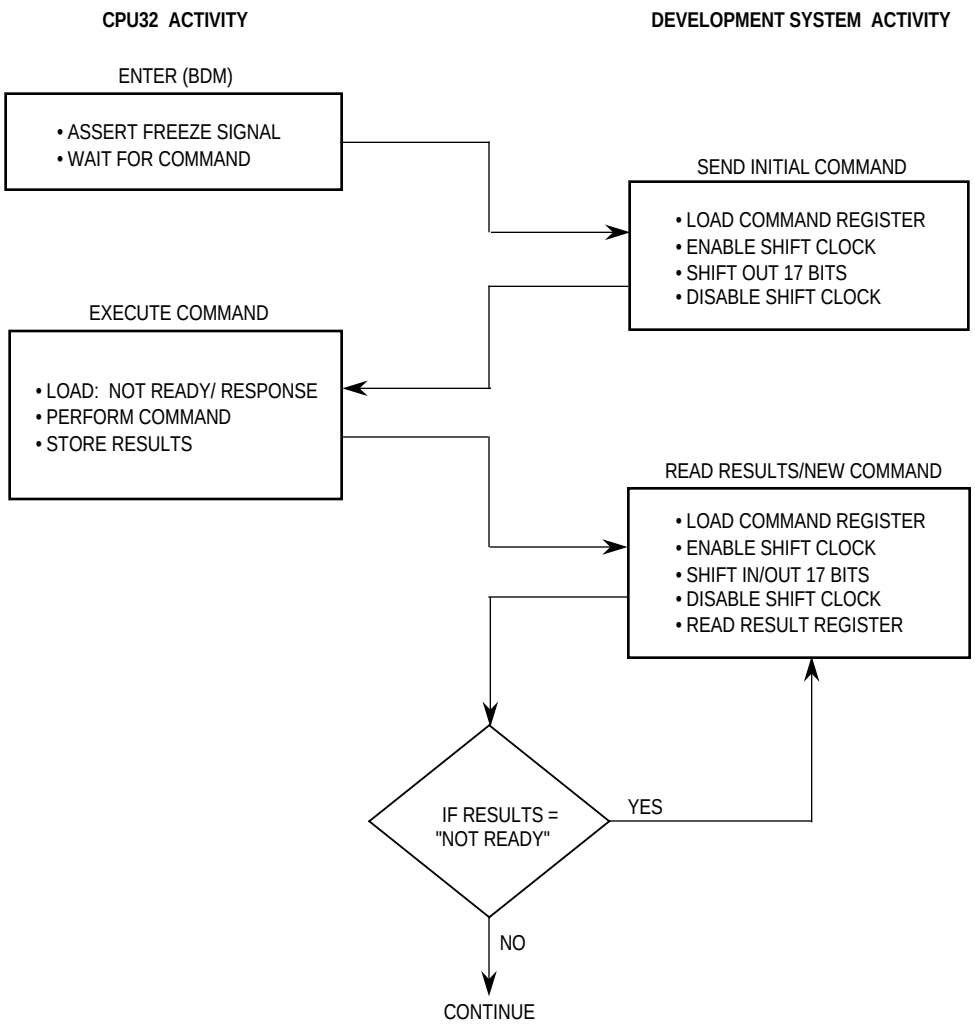
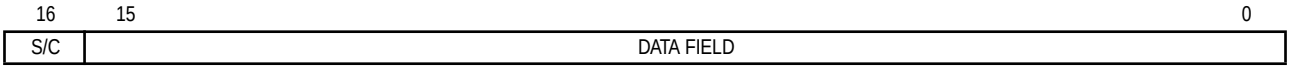


Figure 5-21. BDM Command Execution Flowchart

The serial data word is 17 bits wide—16 data bits and a status/control (S/C) bit.



Bit 16 indicates the status of CPU-generated messages as listed in Table 5-21

Command and data transfers initiated by the development system should clear bit 16. The current implementation ignores this bit; however, Motorola reserves the right to use this bit for future enhancements.

Table 5-21. CPU Generated Message Encoding

Encoding	Data	Message Type
0	xxxx	Valid Data Transfer
0	FFFF	Command Complete; Status OK
1	0000	Not Ready with Response; Come Again
1	0001	$\overline{\text{BERR}}$ Terminated Bus Cycle; Data Invalid
1	FFFF	Illegal Command

5.6.2.7.1 CPU Serial Logic. CPU32+ serial logic, shown in the left-hand portion of Figure 5-22, consists of transmit and receive shift registers and of control logic that includes synchronization, serial clock generation circuitry, and a received bit counter.

Both DSCLK and DSI are synchronized to on-chip clocks, thereby minimizing the chance of propagating metastable states into the serial state machine. Data is sampled during the high phase of CLKOUT. At the falling edge of CLKOUT, the sampled value is made available to internal logic. If there is no synchronization between CPU32+ and development system hardware, the minimum hold time on DSI with respect to DSCLK is one full period of CLKOUT.

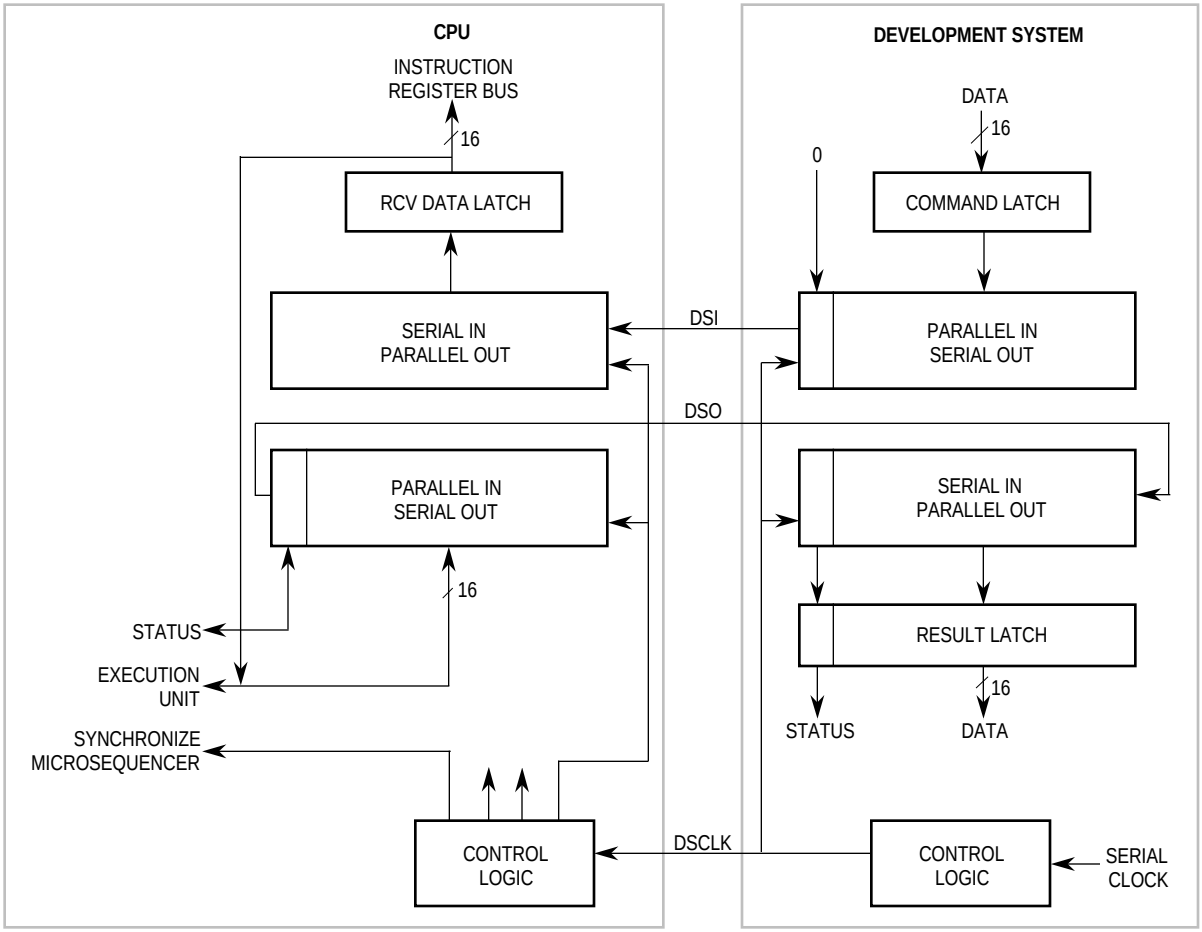


Figure 5-22. Debug Serial I/O Block Diagram

The serial state machine begins a sequence of events based on the rising edge of the synchronized DSCLK (see Figure 5-23). Synchronized serial data is transferred to the input shift register, and the received bit counter is decremented. One-half clock period later, the output shift register is updated, bringing the next output bit to the DSO signal. DSO changes relative to the rising edge of DSCLK and does not necessarily remain stable until the falling edge of DSCLK.

One clock period after the synchronized DSCLK has been seen internally, the updated counter value is checked. If the counter has reached zero, the receive data latch is updated from the input shift register. At this same time, the output shift register is reloaded with the

“not ready/come again” response. Once the receive data latch has been loaded, the CPU is released to act on the new data. Response data overwrites the “not ready” response when the CPU has completed the current operation.

Data written into the output shift register appears immediately on the DSO signal. In general, this action changes the state of the signal from a high (“not ready” response status bit) to a low (valid data status bit) logic level. However, this level change only occurs if the command completes successfully. Error conditions overwrite the “not ready” response with the appropriate response that also has the status bit set.

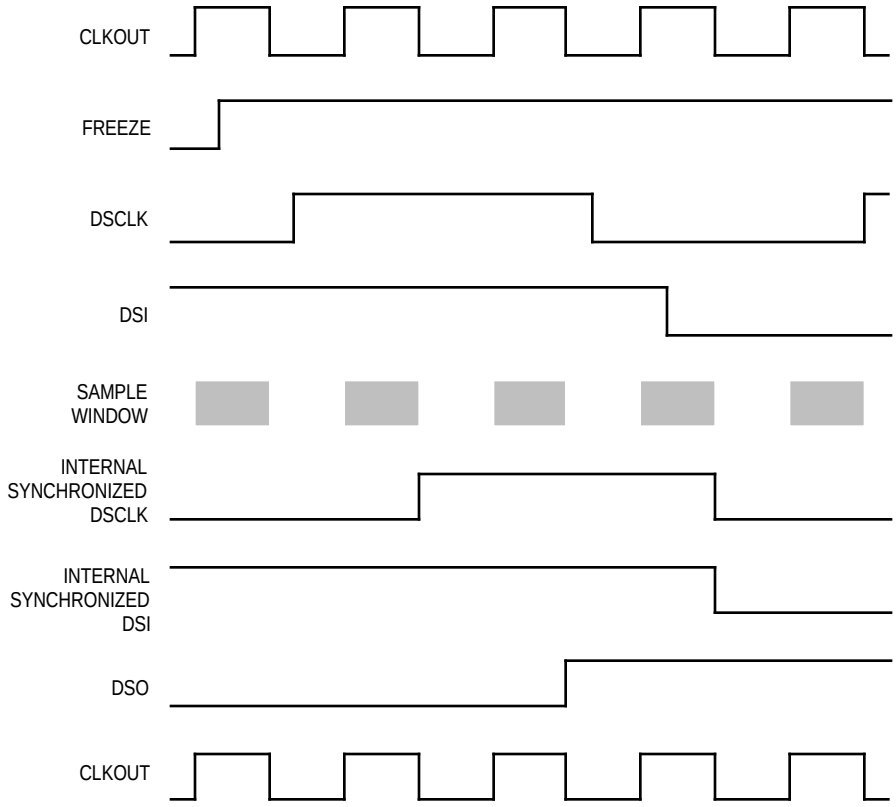


Figure 5-23. Serial InterfaceTiming Diagram

A user can use the state change on DSO to signal hardware that the next serial transfer may begin. A timeout of sufficient length to trap error conditions that do not change the state of DSO should also be incorporated into the design. Hardware interlocks in the CPU prevent result data from corrupting serial transfers in progress.

5.6.2.7.2 Development System Serial Logic. The development system, as the master of the serial data link, must supply the serial clock. However, normal and BDM operations could interact if the clock generator is not properly designed.

Breakpoint requests are made by asserting $\overline{\text{BKPT}}$ to the low state in either of two ways. The primary method is to assert $\overline{\text{BKPT}}$ during a single bus cycle for which an exception is desired. Another method is to assert $\overline{\text{BKPT}}$, then continue to assert it until the CPU32+ responds by asserting FREEZE. This method is useful for forcing a transition into BDM when

the bus is not being monitored. Each method requires a slightly different serial logic design to avoid spurious serial clocks.

Figure 5-24 represents the timing required for asserting $\overline{\text{BKPT}}$ during a single bus cycle.

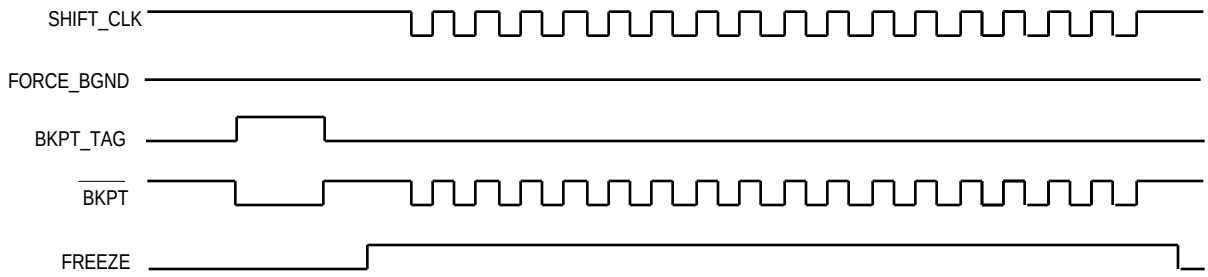


Figure 5-24. $\overline{\text{BKPT}}$ Timing for Single Bus Cycle

Figure 5-25 depicts the timing of the $\overline{\text{BKPT}}$/FREEZE method. In both cases, the serial clock is left high after the final shift of each transfer. This technique eliminates the possibility of accidentally tagging the prefetch initiated at the conclusion of a BDM session. As mentioned previously, all timing within the CPU is derived from the rising edge of the clock; the falling edge is effectively ignored.

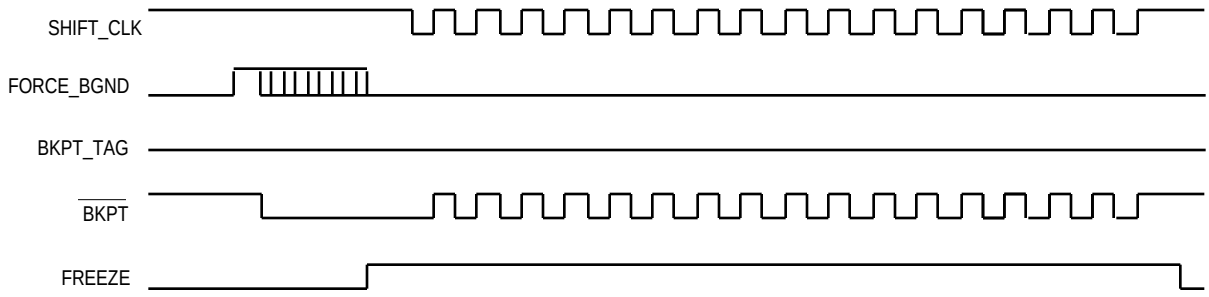


Figure 5-25. $\overline{\text{BKPT}}$ Timing for Forcing BDM

Figure 5-26 represents a sample circuit providing for both $\overline{\text{BKPT}}$ assertion methods. As the name implies, FORCE_BGND is used to force a transition into BDM by the assertion of $\overline{\text{BKPT}}$. FORCE_BGND can be a short pulse or can remain asserted until FREEZE is asserted. Once asserted, the set-reset latch holds $\overline{\text{BKPT}}$ low until the first SHIFT_CLK is applied.

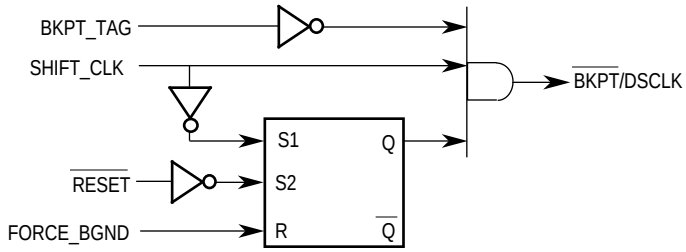


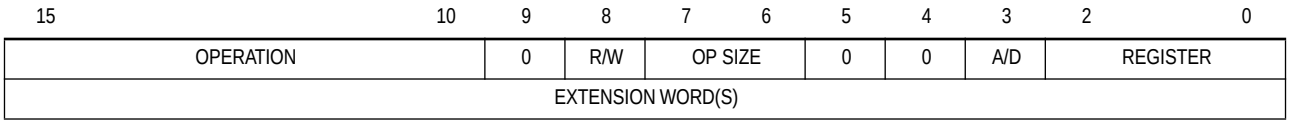
Figure 5-26. $\overline{\text{BKPT}}$/DSCLK Logic Diagram

BKPT_TAG should be timed to the bus cycles since it is not latched. If extended past the assertion of FREEZE, the negation of BKPT_TAG appears to the CPU32+ as the first DSCLK.

DSCLK, the gated serial clock, is normally high, but it pulses low for each bit to be transferred. At the end of the seventeenth clock period, it remains high until the start of the next transmission. Clock frequency is implementation dependent and may range from DC to the maximum specified frequency. Although performance considerations might dictate a hardware implementation, software solutions can be used provided serial bus timing is maintained.

5.6.2.8 COMMAND SET. The following paragraphs describe the command set available in BDM.

5.6.2.8.1 Command Format. The following standard bit .command format is utilized by all BDM commands.



Bits 15–10—Operation Field

The operation field specifies the commands. This 6-bit field provides for a maximum of 64 unique commands.

R/W Field

The R/W field specifies the direction of operand transfer. When the bit is set, the transfer is from theCPU to the development system. When the bit is cleared, data is written to the CPU or to memory from the development system.

Operand Size

For sized operations, this field specifies the operand data size. All addresses are expressed as 32-bit absolute values. The size field is encoded as listed in Table 5-22..

Table 5-22. Size Field Encoding

Encoding	Operand Size
00	Byte
01	Word
10	Long
11	Reserved

Address/Data (A/D) Field

The A/D field is used by commands that operate on address and data registers. It determines whether the register field specifies a data or address register. One indicates an address register; zero indicates a data register. For other commands, this field may be interpreted differently.

Register Field:

In most commands, this field specifies the register number for operations performed on an address or data register.

Extension Word(s) (as required):

At this time, no command requires an extension word to specify fully the operation to be performed, but some commands require extension words for addresses or immediate data. Addresses require two extension words because only absolute long addressing is permitted. Immediate data can be either one or two words in length—byte and word data each require a single extension word; long-word data requires two words. Both operands and addresses are transferred most significant word first.

5.6.2.8.2 Command Sequence Diagram. A command sequence diagram (see Figure 5-27) illustrates the serial bus traffic for each command. Each bubble in the diagram represents a single 17-bit transfer across the bus. The top half in each diagram corresponds to the data transmitted by the development system to the CPU; the bottom half corresponds to the data returned by the CPU in response to the development system commands. Command and result transactions are overlapped to minimize latency.

The cycle in which the command is issued contains the development system command mnemonic (in this example, "read memory location"). During the same cycle, the CPU responds with either the lowest order results of the previous command or with a command complete status (if no results were required).

During the second cycle, the development system supplies the high-order 16 bits of the memory address. The CPU returns a "not ready" response unless the received command was decoded as unimplemented, in which case the response data is the illegal command encoding. If an illegal command response occurs, the development system should retransmit the command.

NOTE

The “not ready” response can be ignored unless a memory bus cycle is in progress. Otherwise, the CPU can accept a new serial transfer with eight system clock periods.

In the third cycle, the development system supplies the low-order 16 bits of a memory address. The CPU always returns the “not ready” response in this cycle. At the completion of the third cycle, the CPU initiates a memory read operation. Any serial transfers that begin while the memory access is in progress return the “not ready” response.

Results are returned in the two serial transfer cycles following the completion of memory access. The data transmitted to the CPU during the final transfer is the opcode for the following command. Should a memory access generate either a bus or address error, an error status is returned in place of the result data.

5.6.2.8.3 Command Set Summary. The BDM command set is summarized in Table 5-23. Subsequent paragraphs contain detailed descriptions of each command.

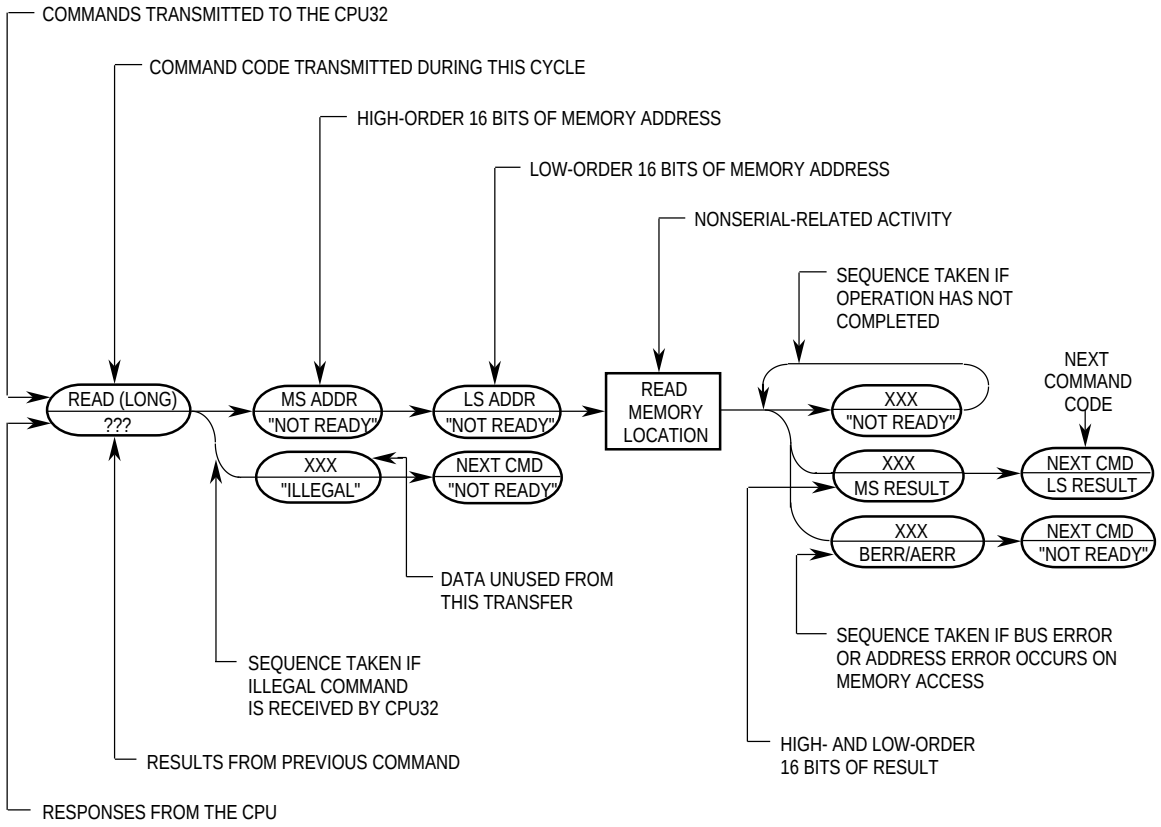


Figure 5-27. Command Sequence Diagram

Table 5-23. BDM Command Summary

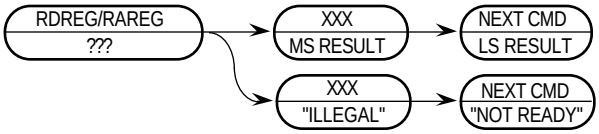
Command	Mnemonic	Description
Read A/D Register	RAREG/RDREG	Read the selected address or data register and return the results via the serial interface.
Write A/D Register	WAREG/WDREG	The data operand is written to the specified address or data register.
Read System Register		The specified system control register is read. All registers that can be read in supervisor mode can be read in BDM.
Write System Register	WSREG	The operand data is written into the specified system control register.
Read Memory Location	READ	Read the sized data at the memory location specified by the long-word address. The SFC register determines the address space accessed.
Write Memory Location	WRITE	Write the operand data to the memory location specified by the long-word address. The DFC register determines the address space accessed.
Dump Memory Block	DUMP	Used in conjunction with the READ command to dump large blocks of memory. An initial READ is executed to set up the starting address of the block and to retrieve the first result. Subsequent operands are retrieved with the DUMP command.
Fill Memory Block	FILL	Used in conjunction with the WRITE command to fill large blocks of memory. An initial WRITE is executed to set up the starting address of the block and to supply the first operand. Subsequent operands are written with the FILL command.
Resume Execution	GO	The pipeline is flushed and refilled before resuming instruction execution at the return PC.
Call User Code	CALL	Current PC is stacked at the location of the current SP. Instruction execution begins at user patch code.
Reset Peripherals	RST	Asserts RESET for 512 clock cycles. The CPU is not reset by this command. Synonymous with the CPU RESET instruction.
No Operation	NOP	NOP performs no operation and may be used as a null command.

5.6.2.8.4 Read A/D Register (RAREG/RDREG). Read the selected address or data register and return the results via the serial interface.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	0	1	1	0	0	0	A/D	REGISTER		

Command Sequence:



Operand Data:

None

Result Data:

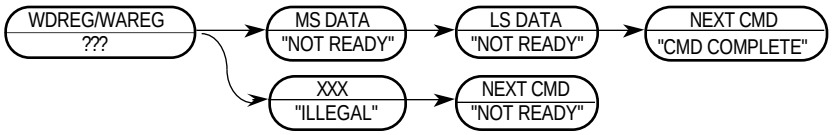
The contents of the selected register are returned as a long-word value. The data is returned most significant word first.

5.6.2.8.5 Write A/D Register (WAREG/WDREG). The operand (long-word) data is written to the specified address or data register. All 32 bits of the register are altered by the write.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	0	0	0	1	0	0	0	A/D	REGISTER		

Command Sequence:



Operand Data:

Long-word data is written into the specified address or data register. The data is supplied most significant word first.

Result Data:

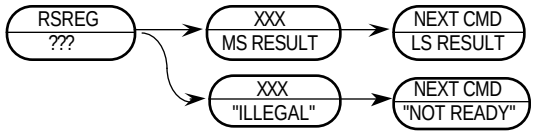
Command complete status ($0FFFF) is returned when register write is complete.

5.6.2.8.6 Read System Register (RSREG). The specified system control register is read. All registers that can be read in supervisor mode can be read in BDM. Several internal temporary registers are also accessible.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	0
0	0	1	0	0	1	0	0	1	0	0	0	REGISTER	

Command Sequence:



Operand Data:

None

Result Data:

Always returns 32 bits of data, regardless of the size of the register being read. If the register is less than 32 bits, the result is returned zero extended.

Register Field:

The system control register is specified by the register field (see Table 5-24).

Table 5-24. Register Field for RSREG and WSREG

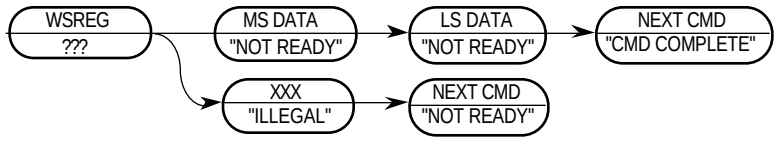
System Register	Select Code
Return Program Counter (RPC)	0000
Current Instruction Program Counter (PCC)	0001
Status Register (SR)	1011
User Stack Pointer (USP)	1100
Supervisor Stack Pointer (SSP)	1101
Source Function Code Register (SFC)	1110
Destination Function Code Register (DFC)	1111
Temporary Register A (ATEMP)	1000
Fault Address Register (FAR)	1001
Vector Base Register (VBR)	1010

5.6.2.8.7 Write System Register (WSREG). Operand data is written into the specified system control register. All registers that can be written in supervisor mode can be written in BDM. Several internal temporary registers are also accessible.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3			0
0	0	1	0	0	1	0	0	1	0	0	0	REGISTER			

Command Sequence:



Operand Data:

The data to be written into the register is always supplied as a 32-bit long word. If the register is less than 32 bits, the least significant word is used.

Result Data:

“Command complete” status is returned when register write is complete.

Register Field:

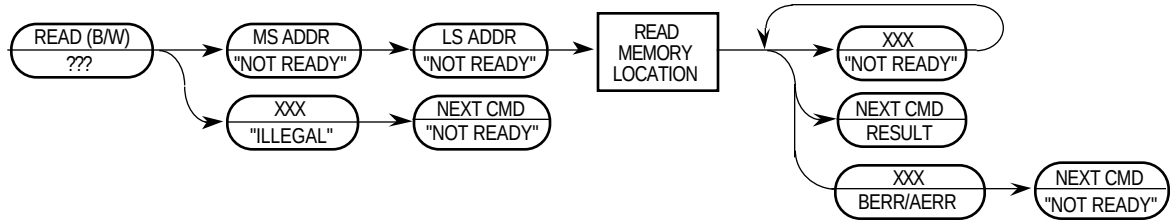
The system control register is specified by the register field (see Table 5-24). The FAR is a read-only register—any write to it is ignored.

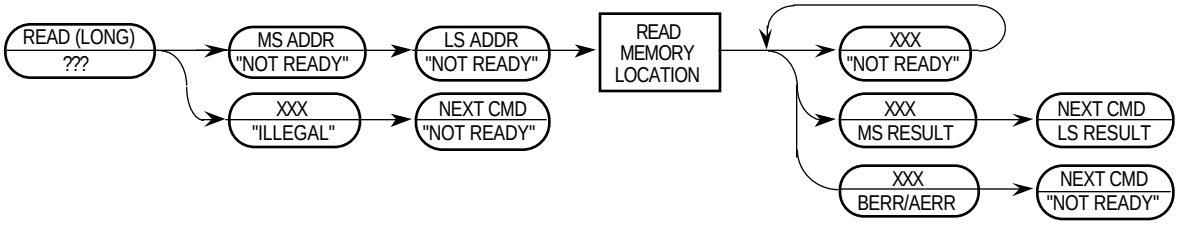
5.6.2.8.8 Read Memory Location (READ). Read the sized data at the memory location specified by the long-word address. Only absolute addressing is supported. The SFC register determines the address space accessed. Valid data sizes include byte, word, or long word.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	0	1	OP SIZE		0	0	0	0	0	0

Command Sequence:





Operand Data:

The single operand is the long-word address of the requested memory location.

Result Data:

The requested data is returned as either a word or long word. Byte data is returned in the least significant byte of a word result, with the upper byte cleared. Word results return 16 bits of significant data; long-word results return 32 bits.

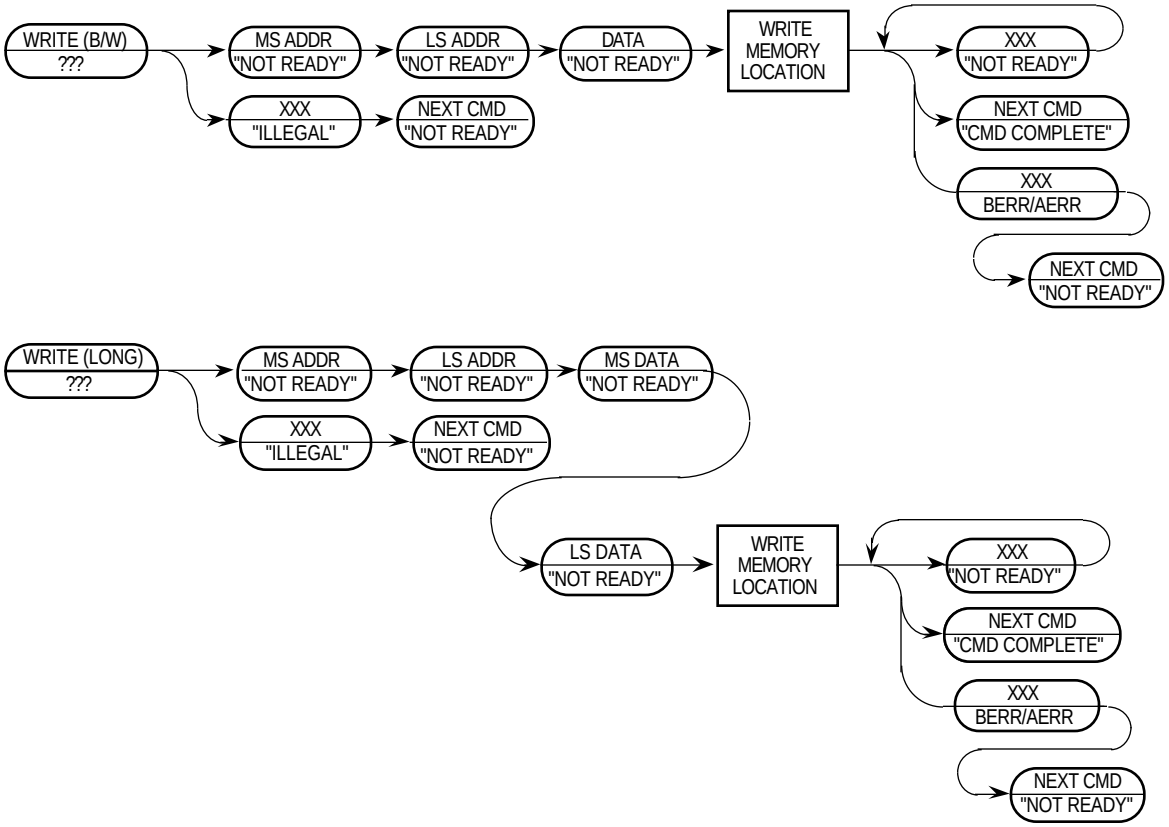
A successful read operation returns data bit 16 cleared. If a bus or address error is encountered, the returned data is $10001.

5.6.2.8.9 Write Memory Location (WRITE). Write the operand data to the memory location specified by the long-word address. The DFC register determines the address space accessed. Only absolute addressing is supported. Valid data sizes include byte, word, and long word.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	0	0	OP SIZE		0	0	0	0	0	0

Command Sequence:



Operand Data:

Two operands are required for this instruction. The first operand is a long-word absolute address that specifies a location to which the operand data is to be written. The second operand is the data. Byte data is transmitted as a 16-bit word, justified in the least significant byte; 16- and 32-bit operands are transmitted as 16 and 32 bits, respectively.

Result Data:

Successful write operations return a status of $0FFFF. Bus or address errors on the write cycle are indicated by the assertion of bit 16 in the status message and by a data pattern of $0001.

5.6.2.8.10 Dump Memory Block (DUMP). DUMP is used in conjunction with the READ command to dump large blocks of memory. An initial READ is executed to set up the starting address of the block and to retrieve the first result. Subsequent operands are retrieved with the DUMP command. The initial address is incremented by the operand size (1, 2, or 4) and saved in a temporary register. Subsequent DUMP commands use this address, increment it by the current operand size, and store the updated address back in the temporary register.

NOTE

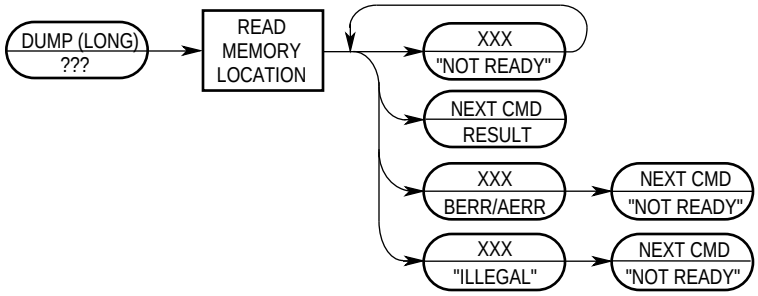
The DUMP command does not check for a valid address in the temporary register—DUMP is a valid command only when preceded by another DUMP or by a READ command. Otherwise, the results are undefined. The NOP command can be used for intercommand padding without corrupting the address pointer.

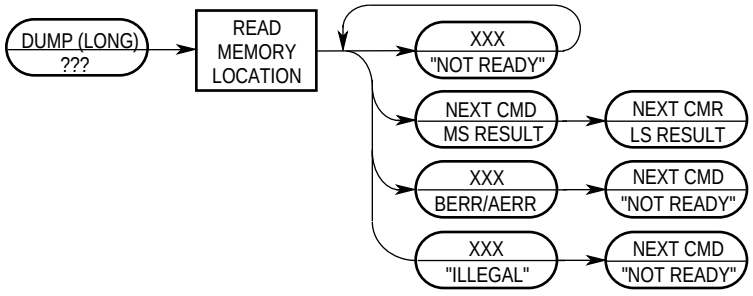
The size field is examined each time a DUMP command is given, allowing the operand size to be altered dynamically.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	1	OP SIZE		0	0	0	0	0	0

Command Sequence:





Operand Data:

None

Result Data:

Requested data is returned as either a word or long word. Byte data is returned in the least significant byte of a word result. Word results return 16 bits of significant data; long-word results return 32 bits. Status of the read operation is returned as in the READ command: $0xxxx for success, $10001 for bus or address errors.

5.6.2.8.11 Fill Memory Block (FILL). FILL is used in conjunction with the WRITE command to fill large blocks of memory. An initial WRITE is executed to set up the starting address of the block and to supply the first operand. Subsequent operands are written with the FILL command. The initial address is incremented by the operand size (1, 2, or 4) and is saved in a temporary register. Subsequent FILL commands use this address, increment it by the current operand size, and store the updated address back in the temporary register.

NOTE

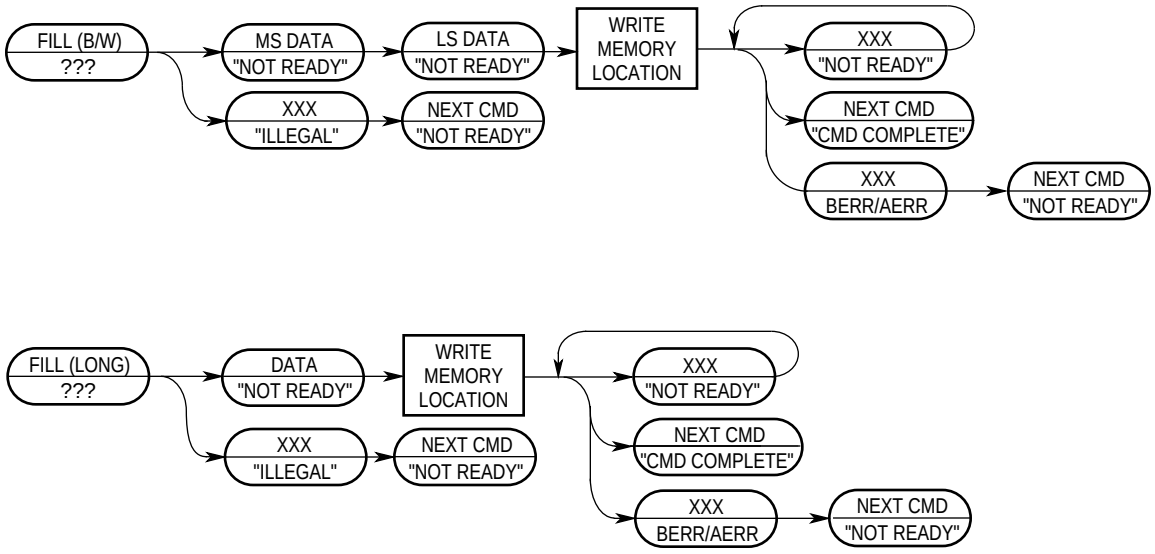
The FILL command does not check for a valid address in the temporary register—FILL is a valid command only when preceded by another FILL or by a WRITE command. Otherwise, the results are undefined. The NOP command can be used for intercommand padding without corrupting the address pointer.

The size field is examined each time a FILL command is given, allowing the operand size to be altered dynamically.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	0	OP SIZE		0	0	0	0	0	0

Command Sequence:



Operand Data:

A single operand is data to be written to the memory location. Byte data is transmitted as a 16-bit word, justified in the least significant byte; 16- and 32-bit operands are transmitted as 16 and 32 bits, respectively.

Result Data:

Status is returned as in the WRITE command: $0FFFF for a successful operation and $10001 for a bus or address error during write.

5.6.2.8.12 Resume Execution (GO). The pipeline is flushed and refilled before normal instruction execution is resumed. Prefetching begins at the return PC and current privilege level. If either the PC or SR is altered during BDM, the updated value of these registers is used when prefetching commences.

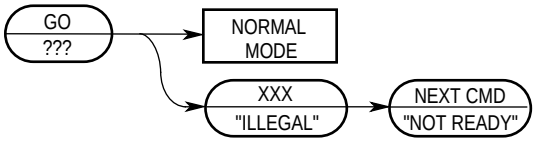
NOTE

The processor exits BDM when a bus error or address error occurs on the first instruction prefetch from the new PC—the error is trapped as a normal mode exception. The stacked value of the current PC may not be valid in this case, depending on the state of the machine prior to entering BDM. For address error, the PC does not reflect the true return PC. Instead, the stacked fault address is the (odd) return PC.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

Command Sequence:



Operand Data:

None

Result Data:

None

5.6.2.8.13 Call User Code (CALL). This instruction provides a convenient way to patch user code. The return PC is stacked at the location pointed to by the current SP. The stacked PC serves as a return address to be restored by the RTS command that terminates the patch routine. After stacking is complete, the 32-bit operand data is loaded into the PC. The pipeline is flushed and refilled from the location pointed to by the new PC, BDM is exited, and normal mode instruction execution begins.

NOTE

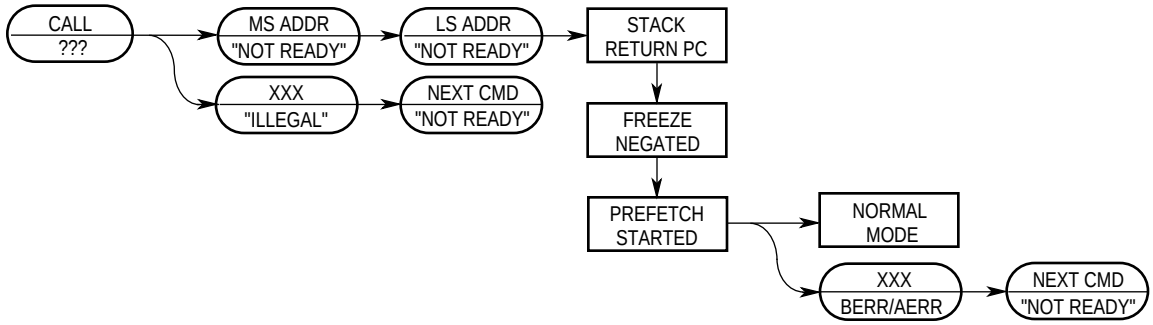
If a bus error or address error occurs during return address stacking, the CPU returns an error status via the serial interface and remains in BDM.

If a bus error or address error occurs on the first instruction prefetch from the new PC, the processor exits BDM and the error is trapped as a normal mode exception. The stacked value of the current PC may not be valid in this case, depending on the state of the machine prior to entering BDM. For address error, the PC does not reflect the true return PC. Instead, the stacked fault address is the (odd) return PC.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

Command Sequence:



Operand Data:

The 32-bit operand data is the starting location of the patch routine, which is the initial PC upon exiting BDM.

Result Data:

None

As an example, consider the following code segment. It outputs a character from the MC68340 serial module channel A.

```
CHKSTAT:    MOVE.B    SRA,D0Move serial status to D0
            BNE.B     CHKSTATLoop until condition true
            MOVE.B    TBA,OUTPUTTransmit character
MISSING:    ANDI.B    #3,D0Check for TxEMP flag
            RTS
```

BDM and the CALL command can be used to patch the code as follows:

1. Breakpoint user program at CHKSTAT
2. Enter BDM

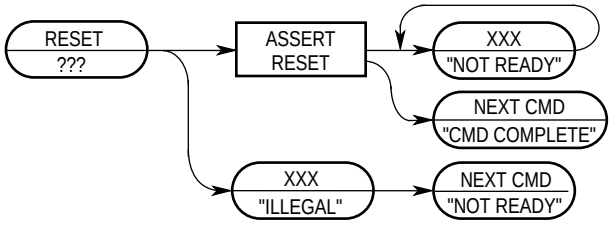
3. Execute CALL command to MISSING
4. Exit BDM
5. Execute MISSING code
6. Return to user program

5.6.2.8.14 Reset Peripherals (RST). RST asserts $\overline{\text{RESET}}$ for 512 clock cycles. The CPU is not reset by this command. This command is synonymous with the CPU RESET instruction.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Command Sequence:



Operand Data:

None

Result Data:

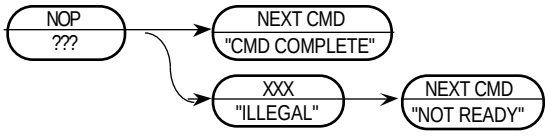
The “command complete” response ($0FFFF) is loaded into the serial shifter after negation of RESET.

5.6.2.8.15 No Operation (NOP). NOP performs no operation and may be used as a null command where required.

Command Format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Command Sequence:



Operand Data:

None

Result Data:

The “command complete” response ($0FFFF) is returned during the next shift operation.

5.6.2.8.16 Future Commands. Unassigned command opcodes are reserved by Motorola for future expansion. All unused formats within any revision level will perform a NOP and return the ILLEGAL command response.

5.6.3 Deterministic Opcode Tracking

The CPU32+ utilizes deterministic opcode tracking to trace program execution. Two signals, IPIPE and IFETCH, provide all information required to analyze instruction pipeline operation.

5.6.3.1 INSTRUCTION FETCH (IFETCH). IFETCH indicates which bus cycles are accessing data to fill the instruction pipeline. IFETCH is pulse-width modulated to multiplex two indications on a single pin. Asserted for a single clock cycle, IFETCH indicates that the data from the current bus cycle is to be routed to the instruction pipeline. IFETCH held low for two clock cycles indicates that the instruction pipeline has been flushed. The data from the bus cycle is used to begin filling the empty pipeline. Both user and supervisor mode fetches are signaled by IFETCH.

Proper tracking of bus cycles via IFETCH on a fast bus requires a simple state machine. On a two-clock bus, IFETCH may signal a pipeline flush with associated prefetch followed immediately by a second prefetch. That is, IFETCH remains asserted for three clocks, two clocks indicating the flush/fetch and a third clock signaling the second fetch. These two operations are easily discerned if the tracking logic samples IFETCH on the two rising edges of CLKO1, which follow the AS (DS during show cycles) falling edge. Three-clock and slower bus cycles allow time for negation of the signal between consecutive indications and do not experience this operation.

5.6.3.2 INSTRUCTION PIPE (IPIPE1–IPIPE0). The internal instruction pipeline can be modeled as a three-stage FIFO (see Figure 5-28). Stage A is an input buffer—data can be used out of stages B and C. The IPIPE1–IPIPE0 signals indicate the advance of instructions in the pipeline.

The 16-bit instruction register A (IRA) and 16-bit instruction register L (IRL) hold incoming words as they are prefetched. No decoding occurs in IRA or IRL. Instruction register B (IRB) provides initial decoding of the opcode and decoding of extension words; it is a source of immediate data. Instruction register C (IRC) supplies residual opcode decoding during instruction execution.

IRA is of higher priority than IRL. IRL is only loaded from the IMB when a 32-bit instruction fetch is performed. IRA is loaded during every instruction fetch.

IRB is loaded from the contents of IRA or IRL, depending on which one is currently valid. If both IRA and IRL are valid, then IRA is loaded into IRB before IRL is loaded into IRB.

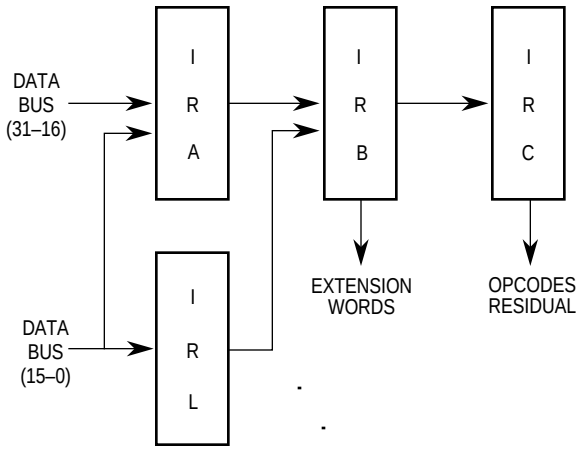


Figure 5-28. Functional Model of Instruction Pipeline

When $\overline{\text{IPIPE1}}$ is low during a clock cycle, it indicates the use of data from IRB on that clock cycle. $\overline{\text{IPIPE1}}$ should be sampled by the user on the falling edge of CLKO1. Regardless of the presence of valid data in IRA or IRL, the contents of IRB are invalidated when $\overline{\text{IPIPE1}}$ is asserted. If IRA or IRL contain valid data, the data is copied into IRB (IRA/IRL ⇒ IRB), and the IRB stage is revalidated.

When $\overline{\text{IPIPE0}}$ is low during a clock cycle, it indicates the start of a new instruction and subsequent replacement of data in IRC. This action causes a full advance of the pipeline (IRB ⇒ IRC and IRA/IRL ⇒ IRB). IRA and/or IRL is refilled during the next instruction fetch bus cycle.

Data loaded into IRA and IRL propagates automatically through subsequent empty pipeline stages. Signals that show the progress of instructions through IRB and IRC are necessary to accurately monitor pipeline operation. These signals are provided by IRA, IRL and IRB validity bits. When a pipeline advance occurs, the validity bit of the stage being loaded is set, and the validity bit of the stage supplying the data is negated.

Because instruction execution is not timed to bus activity, $\overline{\text{IPIPE1}}$–$\overline{\text{IPIPE0}}$ are synchronized with the system clock and not the bus. Figure 5-29 illustrates the timing in relation to the system clock.

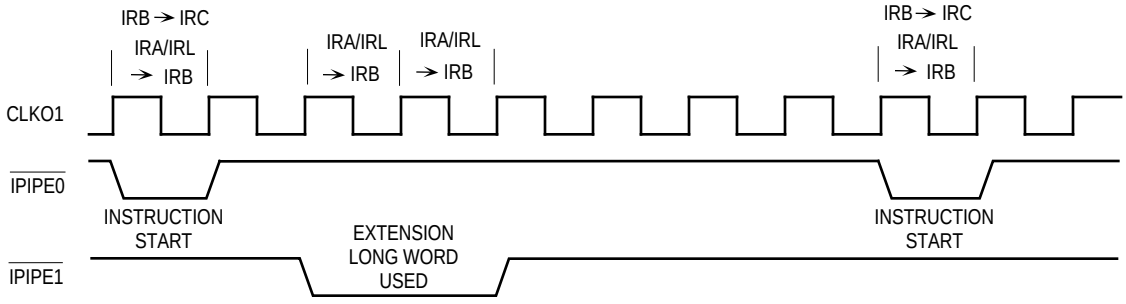


Figure 5-29. Instruction Pipeline Timing Diagram

$\overline{\text{IPIPE1}}$–$\overline{\text{IPIPE0}}$ should be sampled on the falling edge of the clock. Loading IRC always indicates that an instruction is beginning execution — the opcode is loaded into IRC by the transfer. In BDM mode, the data output DSO is connected to $\overline{\text{IPIPE0}}$. The $\overline{\text{IPIPE1}}$ pin is unused in BDM mode.

5.6.3.3 OPCODE TRACKING DURING LOOP MODE. $\overline{\text{IPIPE}}$ and $\overline{\text{IFETCH}}$ continue to work normally during loop mode. $\overline{\text{IFETCH}}$ indicates all instruction fetches up through the point that data begins recirculating within the instruction pipeline. $\overline{\text{IPIPE}}$ continues to signal the start of instructions and the use of extension words even though data is being recirculated internally. $\overline{\text{IFETCH}}$ returns to normal operation with the first fetch after exiting loop mode.

5.7 INSTRUCTION EXECUTION TIMING

This section describes the instruction execution timing of the CPU32+. External clock cycles are used to provide accurate execution and operation timing guidelines, but not exact timing for every possible circumstance. This approach is used because exact execution time for an instruction or operation depends on concurrence of independently scheduled resources, on memory speeds, and on other variables.

An assembly language programmer or compiler writer can use the information in this section to predict the performance of the CPU32+. Additionally, timing for exception processing is included so that designers of multitasking or real-time systems can predict task-switch overhead, maximum interrupt latency, and similar timing parameters. Instruction timing is given in clock cycles to eliminate clock frequency dependency.

Most instruction timing information in the following subsections is taken from the CPU32 documentation. It applies to the CPU32+ when it is executing in 16-bit mode. However, a summary of experiments run on the CPU32+ and the CPU32 is given in Table 5-25. The tests show general indications of performance improvement of the CPU32+ over the CPU32. Actual results on real applications may vary.

Table 5-25. CPU32+ Performance Improvement over the CPU32

		Bus Cycle Length	
Conditions	**2**	**3**	**5**
		PI/BU (see Note)	
16-Bit Data Bus	0/78	0/89	0/95
32-Bit Data Bus with 16-Bit Operands Only (e.g., MOVE.W, CLR.W, etc.)	6/52	13/65	24/76
32-Bit Data Bus with 32-Bit Operands Only (e.g., MOVE.L, MOVEA.L, MOVEM.L etc.)	13/50	40/62	58/73

NOTE:
PI = % Performance Increase over a CPU32 in the same conditions
BU = % Bus Utilization taken by the processor in the experiment

Note that the CPU32+ gains a significant performance advantage (58%) over the original CPU32 when using long operands on a slow external bus. (Most compilers generate code using long operands where possible.) Thus, the CPU32+ performance in 32-bit mode "falls off" less rapidly than does the original CPU32.

Also, note that the use of a 32-bit data bus reduces external bus utilization by 19 to 28 percentage points (e.g., 78–50 = 28%). This reduction gives more time for peripherals, such as DMA channels, to use the bus without adversely affecting overall system performance. In the best case, the CPU32+ can use as little as 50% of the bus, even though instructions execute continuously.

5.7.1 Resource Scheduling

The CPU32+ contains several independently scheduled resources. The organization of these resources within the CPU32+ is shown in Figure 5-30. Some variation in instruction execution timing results from concurrent resource utilization. Because resource scheduling is not directly related to instruction boundaries, it is impossible to make an accurate prediction of the time required to complete an instruction without knowing the entire context within which the instruction is executing.

5.7.1.1 MICROSEQUENCER. The microsequencer either executes microinstructions or awaits completion of accesses necessary to continue microcode execution. The microsequencer supervises the bus controller, instruction execution, and internal processor operations such as calculation of EA and setting of condition codes. It also initiates instruction word prefetches after a change of flow and controls validation of instruction words in the instruction pipeline.

5.7.1.2 INSTRUCTION PIPELINE. The CPU32+ contains a two-word instruction pipeline where instruction opcodes are decoded. Each stage of the pipeline is initially filled under microsequencer control and subsequently refilled by the prefetch controller as it empties.

Stage A of the instruction pipeline is a buffer. Prefetches completed on the bus before stage B empties are temporarily stored in this buffer. Instruction words (instruction operation words and all extension words) are decoded at stage B. Residual decoding and execution occur in stage C.

Each pipeline stage has an associated status bit that shows whether the word in that stage was loaded with data from a bus cycle that terminated abnormally.

5.7.1.3 BUS CONTROLLER RESOURCES. The bus controller consists of the instruction prefetch controller, the write pending buffer, and the microbus controller. These three resources transact all reads, writes, and instruction prefetches required for instruction execution.

The bus controller and microsequencer operate concurrently. The bus controller can perform a read or write or schedule a prefetch while the microsequencer controls EA calculation or sets condition codes.

The microsequencer can also request a bus cycle that the bus controller cannot perform immediately. When this happens, the bus cycle is queued, and the bus controller runs the cycle when the current cycle has completed.

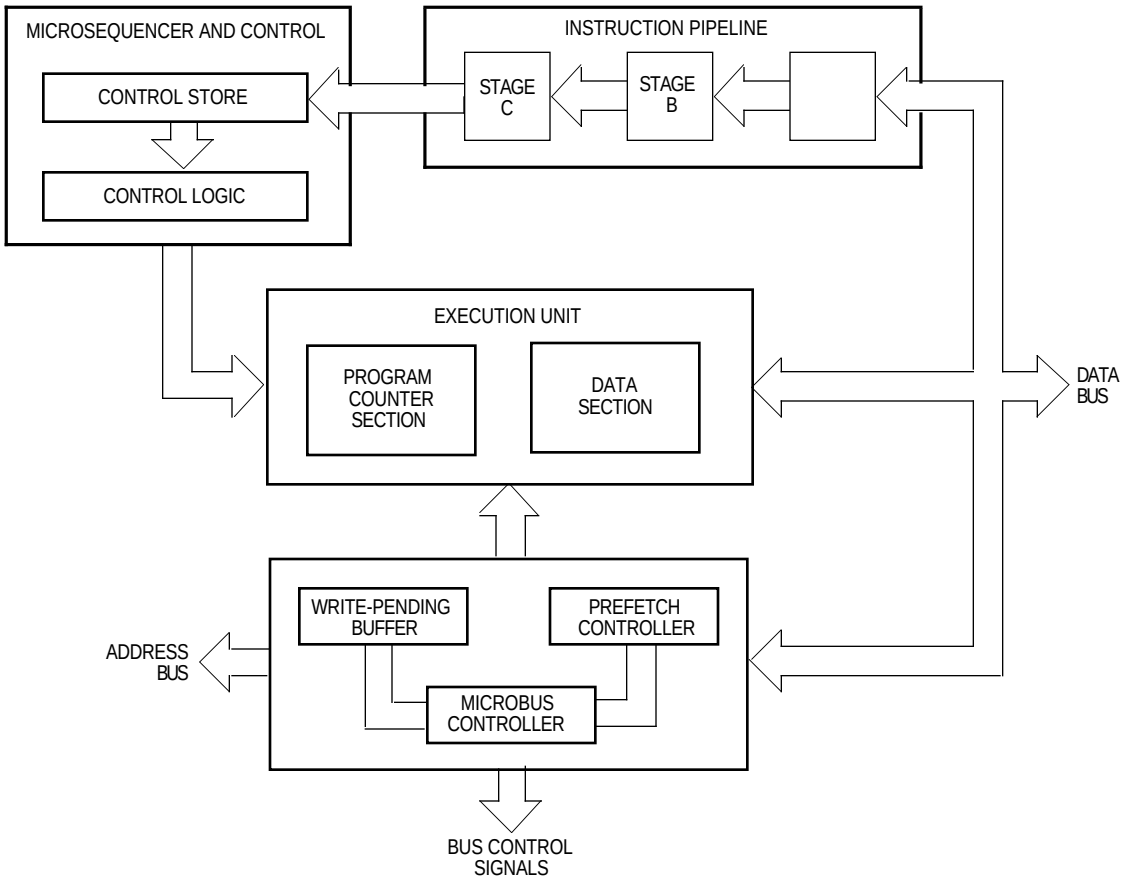


Figure 5-30. Block Diagram of Independent Resources

5.7.1.3.1 Prefetch Controller. The instruction prefetch controller receives an initial request from the microsequencer to initiate prefetching at a given address. Subsequent prefetches are initiated by the prefetch controller whenever a pipeline stage is invalidated, either through instruction completion or through use of extension words. Prefetch occurs as soon as the bus is free of operand accesses previously requested by the microsequencer. Additional state information permits the controller to inhibit prefetch requests when a change in instruction flow (e.g., a jump or branch instruction) is anticipated.

In a typical program, 10 to 25 percent of the instructions cause a change of flow. Each time a change occurs, the instruction pipeline must be flushed and refilled from the new instruction stream. If instruction prefetches, rather than operand accesses, were given priority, many instruction words would be flushed unused, and necessary operand cycles would be delayed. To maximize available bus bandwidth, the CPU32+ will schedule a prefetch only when the next instruction is not a change-of-flow instruction and when there is room in the pipeline for the prefetch.

5.7.1.3.2 Write-Pending Buffer. The CPU32+ incorporates a single-operand write-pending buffer. The buffer permits the microsequencer to continue execution after a request for a write cycle is queued in the bus controller. The time needed for a write at the end of an instruction can overlap the head cycle time for the following instruction, thus reducing overall execution time. Interlocks prevent the microsequencer from overwriting the buffer.

5.7.1.3.3 Microbus Controller. The microbus controller performs bus cycles issued by the microsequencer. Operand accesses always have priority over instruction prefetches. Word and byte operands are accessed in a single CPU-initiated bus cycle, although the external bus interface may be required to initiate a second cycle when a word operand is sent to a byte-sized external port. If long operands are accessed from a 16-bit port, they are accessed in two bus cycles, most significant word first.

The instruction pipeline is capable of recognizing instructions that cause a change of flow. It informs the bus controller when a change of flow is imminent, and the bus controller refrains from starting prefetches that would be discarded due to the change of flow.

5.7.1.4 INSTRUCTION EXECUTION OVERLAP. Overlap is the time, measured in clock cycles, that an instruction executes concurrently with the previous instruction. As shown in Figure 5-31, portions of instructions A and B execute simultaneously, reducing total execution time. Because portions of instructions B and C also overlap, overall execution time for all three instructions is also reduced.

Each instruction contributes to the total overlap time. The portion of execution time at the end of instruction A that can overlap the beginning of instruction B is called the tail of instruction A. The portion of execution time at the beginning of instruction B that can overlap the end of instruction A is called the head of instruction B. The total overlap time between instructions A and B is the smaller tail of A and the head of B.

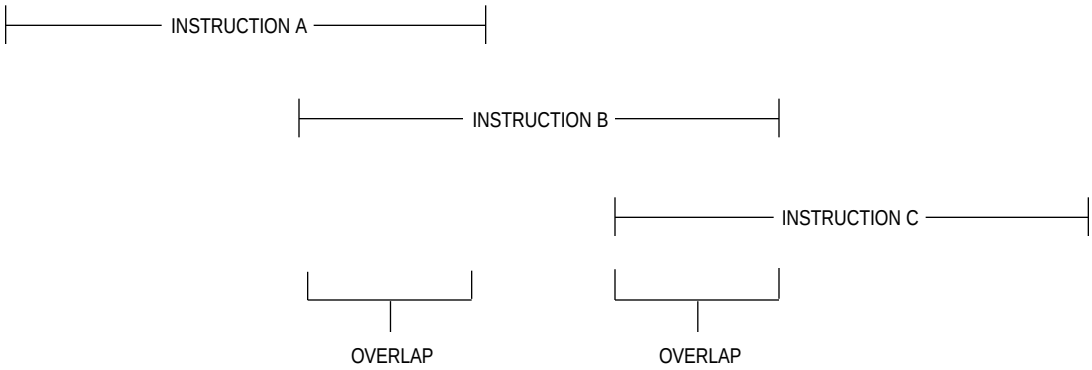


Figure 5-31. Simultaneous Instruction Execution

The execution time attributed to instructions A, B, and C after considering the overlap is illustrated in Figure 5-32. The overlap time is attributed to the execution time of the completing instruction. The following equation shows the method for calculating the overlap time:

$$\text{Overlap} = \min (\text{Tail}_N, \text{Head}_{N+1})$$

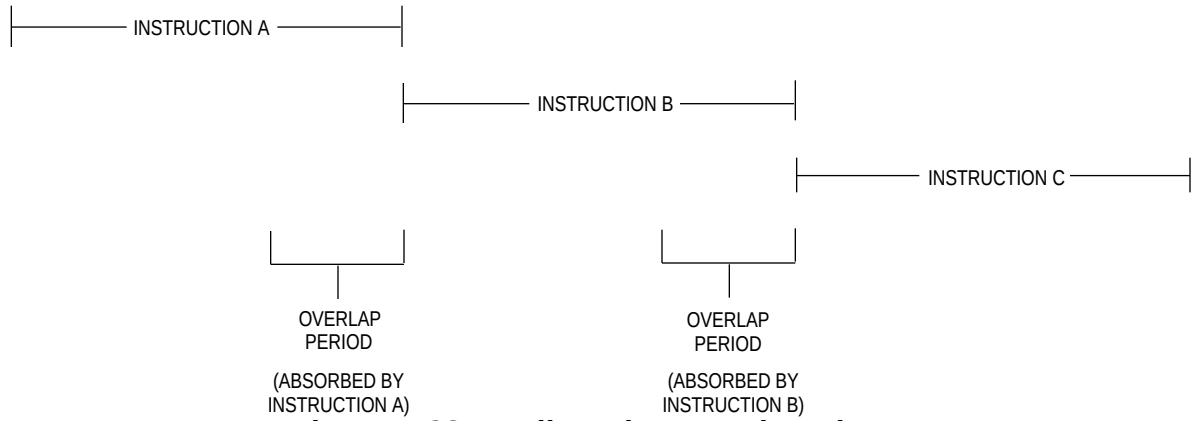


Figure 5-32. Attributed Instruction Times

5.7.1.5 EFFECTS OF WAIT STATES. The CPU32+ access time for on-chip peripherals is two clocks. While two-clock external accesses are possible when the bus is operated in a synchronous mode, a typical external memory speed is three or more clocks.

All instruction times listed in this section are for word access only (unless an explicit exception is given), and are based on the assumption that both instruction fetches and operand cycles are to a two-clock memory. Wait states due to slow external memory must be added to the access time for each bus cycle.

A typical application has a mixture of bus speeds—program execution from an off-chip ROM, accesses to on-chip peripherals, storage of variables in slow off-chip RAM, and accesses to external peripherals with speeds ranging from moderate to very slow. To arrive at an accurate instruction time calculation, each bus access must be individually considered. Many instructions have a head cycle count, which can overlap the cycles of an operand fetch to slower memory started by a previous instruction. In these cases, an increase in access time has no effect on the total execution time of the pair of instructions.

To trace instruction execution time by monitoring the external bus, note that the order of operand accesses for a particular instruction sequence is always the same provided bus speed is unchanged and the interleaving of instruction prefetches with operands within each sequence is identical.

5.7.1.6 INSTRUCTION EXECUTION TIME CALCULATION. The overall execution time for an instruction depends on the amount of overlap with previous and subsequent instructions. To calculate an instruction time estimate, the entire code sequence must be analyzed. To derive the actual instruction execution times for an instruction sequence, the instruction times listed in the tables must be adjusted to account for overlap.

The formula for this calculation is as follows:

$C_1 - \min(T_1, H_2) + C_2 - \min(T_2, H_3) + C_3 - \min(T_3, H_4) + \ldots$

where:

C_N is the number of cycles listed for instruction N

T_N is the tail time for instruction N

H_N is the head time for instruction N

min (T_N, H_M) is the minimum of parameters T_N and H_M

The number of cycles for the instruction (C_N) can include one or two EA calculations in addition to the raw number in the cycles column. In these cases, calculate overall instruction time as if it were for multiple instructions, using the following equation:

$$\langle CEA \rangle - \min (T_{ea}, H_{op}) + C_{op}$$

where:

$\langle CEA \rangle$ is the instruction's EA time

C_{op} is the instruction's operation time

T_{ea} is the EA's tail time

H_{op} is the instruction operation's head time

min (T_n, H_m) is the minimum of parameters T_n and H_m

The overall head for the instruction is the head for the EA, and the overall tail for the instruction is the tail for the operation. Therefore, the actual equation for execution time becomes:

$$C_{op1} - \min (T_{op1}, H_{ea2}) + \langle CEA \rangle_2 - \min (T_{ea2}, H_{op2}) + C_{op2} - \min (T_{op2}, H_{ea3}) + \ldots$$

Every instruction must prefetch to replace itself in the instruction pipe. Usually, these prefetches occur during or after an instruction. A prefetch is permitted to begin in the first clock of any indexed EA mode operation.

Additionally, a prefetch for an instruction is permitted to begin two clocks before the end of an instruction provided the bus is not being used. If the bus is being used, then the prefetch occurs at the next available time when the bus would otherwise be idle.

5.7.1.7 EFFECTS OF NEGATIVE TAILS. When the CPU32+ changes instruction flow, the instruction decode pipeline must begin refilling before instruction execution can resume. Refilling forces a two-clock idle period at the end of the change-of-flow instruction. This idle period can be used to prefetch an additional word on the new instruction path. Because of the stipulation that each instruction must prefetch to replace itself, the concept of negative tails has been introduced to account for these free clocks on the bus.

On a two-clock bus, it is not necessary to adjust instruction timing to account for the potential extra prefetch. The cycle times of the microsequencer and bus are matched, and no additional benefit or penalty is obtained. In the instruction execution time equations, a zero should be used instead of a negative number.

Negative tails are used to adjust for slower fetches on slower buses. Normally, increasing the length of prefetch bus cycles directly affects the cycle count and tail values found in the tables.

In the following equations, negative tail values are used to negate the effects of a slower bus. The equations are generalized, however, so that they may be used on any speed bus with any tail value.

```
NEW_TAIL = OLD_TAIL + (NEW_CLOCK – 2)
IF ((NEW_CLOCK – 4) > 0) THEN
    NEW_CYCLE = OLD_CYCLE + (NEW_CLOCK – 2) + (NEW_CLOCK – 4)
ELSE
    NEW_CYCLE = OLD_CYCLE + (NEW _CLOCK – 2)
```

where:

NEW_TAIL/NEW_CYCLE is the adjusted tail/cycle at the slower speed

OLD_TAIL/OLD_CYCLE is the value listed in the instruction timing tables

NEW_CLOCK is the number of clocks per cycle at the slower speed

Note that many instructions listed as having negative tails are change-of-flow instructions and that the bus speed used in the calculation is that of the new instruction stream.

5.7.2 Instruction Timing Tables

The following assumptions apply to the times shown in the subsequent tables:

1. A 16-bit data bus is used for all memory accesses (CPU32+ in 16-bit mode).
2. Memory access times are based on two-clock bus cycles with no wait states.
3. The instruction pipeline is full at the beginning of the instruction and is refilled by the end of the instruction.

Three values are listed for each instruction and addressing mode:

Head: The number of cycles available at the beginning of an instruction to complete a previous instruction write or to perform a prefetch.

Tail: The number of cycles an instruction uses to complete a write.

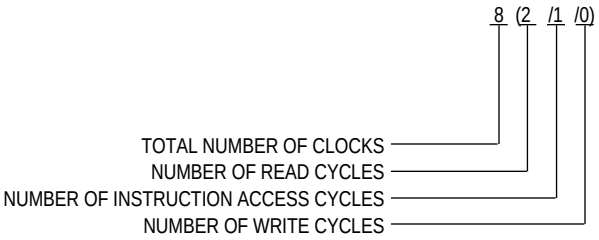
Cycles: Four numbers per entry, three contained in parentheses. The outer number is the minimum number of cycles required for the instruction to complete. Numbers within the parentheses represent the number of bus accesses performed by the instruction. The first number is the number of operand read accesses performed by the instruction. The second number is the number of instruction fetches performed by the instruction, including all prefetches that keep the instruction and the instruction pipeline filled. The third number is the number of write accesses performed by the instruction.

As an example, consider an ADD.L (12, A3, D7.W ∗ 4), D2 instruction.

Paragraph 5.7.2.5 Arithmetic/Logic Instructions shows that the instruction has a head = 0, a tail = 0, and cycles = 2 (0/1/0). However, in indexed address register indirect addressing mode, additional time is required to fetch the EA. Paragraph 5.7.2.1 Fetch Effective Address gives addressing mode data. For (d_8, An, Xn.Sz ∗ Scale), head = 4, tail = 2, cycles = 8 (2/1/0). Because this example is for a long access and the fetch EA table lists data for word

accesses, add two clocks to the tail and to the number of cycles (“X” in table notation) to obtain head = 4, tail = 4, cycles = 10 (2/1/0).

Assuming that no trailing write exists from the previous instruction, EA calculation requires six clocks. Replacement fetch for the EA occurs during these six clocks, leaving a head of four. If there is no time in the head to perform a prefetch due to a previous trailing write, then additional time to perform the prefetches must be allotted in the middle of the instruction or after the tail.



The total number of clocks for bus activity is as follows:

(2 Reads $\times$ 2 Clocks/Read) + (1 Instruction Access $\times$ 2 Clocks/Access) +
(0 Writes $\times$ 2 Clocks/Write) = 6 Clocks of Bus Activity

The number of internal clocks (not overlapped by bus activity) is as follows:

10 Clocks Total – 6 Clocks Bus Activity = 4 Internal Clocks

Memory read requires two bus cycles at two clocks each. This read time, implied in the tail figure for the EA, cannot be overlapped with the instruction because the instruction has a head of zero. An additional two clocks are required for the ADD instruction itself. The total is 6 + 4 + 2 = 12 clocks. If bus cycles take more time (i.e., the memory is off-chip), add an appropriate number of clocks to each memory access.

The instruction sequence MOVE.L D0, (A0) followed by LSL.L #7, D2 provides an example of overlapped execution. The MOVE instruction has a head of zero and a tail of four because it is a long write. The LSL instruction has a head of four. The trailing write from the MOVE overlaps the LSL head completely. Thus, the two-instruction sequence has a head of zero, a tail of zero, and a total execution of 8 rather than 12 clocks.

General observations regarding calculation of execution time are as follows:

- Any time the number of bus cycles is listed as "X", substitute a value of one for byte and word cycles and a value of two for long cycles. For long bus cycles, usually add a value of two to the tail.
- The time calculated for an instruction on a three-clock (or longer) bus is usually longer than the actual execution time. All times shown are for two-clock bus cycles.
- If the previous instruction has a negative tail, then a prefetch for the current instruction can begin during the execution of that previous instruction.
- Certain instructions requiring an immediate extension word (immediate word EA, absolute word EA, address register indirect with displacement EA, conditional branches with word offsets, bit operations, LPSTOP, TBL, MOVEM, MOVEC, MOVES, MOVEP,

MUL.L, DIV.L, CHK2, CMP2, and DBcc) are not permitted to begin until the extension word has been in the instruction pipeline for at least one cycle. This does not apply to long offsets or displacements.

5.7.2.1 FETCH EFFECTIVE ADDRESS. The fetch EA table indicates the number of clock periods needed for the processor to calculate and fetch the specified EA. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction	Head	Tail	Cycles	Notes
Dn	–	–	0(0/0/0)	–
An	–	–	0(0/0/0)	–
(An)	1	1	3(X/0/0)	1
(An)+	1	1	3(X/0/0)	1
–(An)	2	2	4(X/0/0)	1
(d_{16},An) or (d_{16},PC)	1	3	5(X/1/0)	1,3
(xxx).W	1	3	5(X/1/0)	1
(xxx).L	1	5	7(X/2/0)	1
#⟨data⟩.B	1	1	3(0/1/0)	1
#⟨data⟩.W	1	1	3(0/1/0)	1
#⟨data⟩.L	1	3	5(0/2/0)	1
(d_8,An,Xn.Sz × Sc) or (d_8,PC,Xn.Sz × Sc)	4	2	8(X/1/0)	1,2,3,4
(0) (All Suppressed)	2	2	6(X/1/0)	1,4
(d_{16})	1	3	7(X/2/0)	1,4
(d_{32})	1	5	9(X/3/0)	1,4
(An)	1	1	5(X/1/0)	1,2,4
(Xm.Sz × Sc)	4	2	8(X/1/0)	1,2,4
(An,Xm.Sz × Sc)	4	2	8(X/1/0)	1,2,3,4
(d_{16},An) or (d_{16},PC)	1	3	7(X/2/0)	1,3,4
(d_{32},An) or (d_{32},PC)	1	5	9(X/3/0)	1,3,4
(d_{16},An,Xm) or (d_{16},PC,Xm)	2	2	8(X/2/0)	1,3,4
(d_{32},An,Xm) or (d_{32},PC,Xm)	1	3	9(X/3/0)	1,3,4
(d_{16},An,Xm.Sz × Sc) or (d_{16},PC,Xm.Sz × Sc)	2	2	8(X/2/0)	1,2,3,4
(d_{32},An,Xm.Sz × Sc) or (d_{32},PC,Xm.Sz × Sc)	1	3	9(X/3/0)	1,2,3,4

X = There is one bus cycle for byte and word operands and two bus cycles for long-word operands. For long-word bus cycles, add two clocks to the tail and to the number of cycles.

NOTES:

1. The read of the EA and replacement fetches overlap the head of the operation by the amount specified in the tail.
2. Size and scale of the index register do not affect execution time.
3. The PC may be substituted for the base address register An.
4. When adjusting the prefetch time for slower buses, extra clocks may be subtracted from the head until the head reaches zero, at which time additional clocks must be added to both the tail and cycle counts.
5. Timing is calculated with the CPU32+ in 16-bit mode.

5.7.2.2 CALCULATE EFFECTIVE ADDRESS. The calculate EA table indicates the number of clock periods needed for the processor to calculate a specified EA. The timing is equivalent to fetch EA except there is no read cycle. The tail and cycle time are reduced by the amount of time the read would occupy. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction	Head	Tail	Cycles	Notes
Dn	–	–	0(0/0/0)	–
An	–	–	0(0/0/0)	–
(An)	1	0	2(0/0/0)	–
(An)+	1	0	2(0/0/0)	–
–(An)	2	0	2(0/0/0)	–
(d_{16},An) or (d_{16},PC)	1	1	3(0/1/0)	1,3
(xxx).W	1	1	3(0/1/0)	1
(xxx).L	1	3	5(0/2/0)	1
(d_8,An,Xn.Sz × Sc) or (d_8,PC,Xn.Sz × Sc)	4	0	6(0/1/0)	2,3,4
(0) (All Suppressed)	2	0	4(0/1/0)	4
(d_{16})	1	1	5(0/2/0)	1,4
(d_{32})	1	3	7(0/3/0)	1,4
(An)	1	0	4(0/1/0)	4
(Xm.Sz × Sc)	4	0	6(0/1/0)	2,4
(An,Xm.Sz × Sc)	4	0	6(0/1/0)	2,4
(d_{16},An) or (d_{16},PC)	1	1	5(0/2/0)	1,3,4
(d_{32},An) or (d_{32},PC)	1	3	7(0/3/0)	1,3,4
(d_{16},An,Xm) or (d_{16},PC,Xm)	2	0	6(0/2/0)	3,4
(d_{32},An,Xm) or (d_{32},PC,Xm)	1	1	7(0/3/0)	1,3,4
(d_{16},An,Xm.Sz × Sc) or (d_{16},PC,Xm.Sz × Sc)	2	0	6(0/2/0)	2,3,4
(d_{32},An,Xm.Sz × Sc) or (d_{32},PC,Xm.Sz × Sc)	1	1	7(0/3/0)	1,2,3,4

NOTES:
1. Replacement fetches overlap the head of the operation by the amount specified in the tail.
2. Size and scale of the index register do not affect execution time.
3. The PC may be substituted for the base address register An.
4. When adjusting the prefetch time for slower buses, extra clocks may be subtracted from the head until the head reaches zero, at which time additional clocks must be added to both the tail and cycle counts.
5. Timing is calculated with the CPU32+ in 16-bit mode

5.7.2.3 MOVE INSTRUCTION. The MOVE instruction table indicates the number of clock periods needed for the processor to calculate the destination EA and to perform a MOVE or MOVEA instruction. For entries with CEA or FEA, refer to the appropriate table to calculate that portion of the instruction time.

Destination EAs are divided by their formats (see CPU32 Reference Manual). The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

When using this table, begin at the top and move downward. Use the first entry that matches both source and destination addressing modes.

Instruction	Head	Tail	Cycles
MOVE Rn, Rn	0	0	2(0/1/0)
MOVE ⟨FEA⟩, Rn	0	0	2(0/1/0)
MOVE Rn, (Am)	0	2	4(0/1/X)
MOVE Rn, (Am)+	1	1	5(0/1/X)
MOVE Rn, –(Am)	2	2	6(0/1/X
MOVE Rn, ⟨CEA⟩	1	3	5(0/1/X)
MOVE ⟨FEA⟩, (An)	2	2	6(0/1/X
MOVE ⟨FEA⟩, (An)+	2	2	6(0/1/X)
MOVE ⟨FEA⟩, –(An)	2	2	6(0/1/X)
MOVE #, ⟨CEA⟩	2	2	6(0/1/X∗
MOVE ⟨CEA⟩, ⟨FEA⟩	2	2	6(0/1/X)

X = There is one bus cycle for byte and word operands and two bus cycles for long-word operands. For long-word bus cycles, add two clocks to the tail and to the number of cycles. Timing is calculated with the CPU32+ in 16-bit mode.

∗ = An # fetch EA time must be added for this instruction: ⟨FEA⟩ +⟨CEA⟩ + ⟨OPER⟩

NOTE: For instructions not explicitly listed, use the MOVE ⟨CEA⟩, ⟨FEA⟩ entry. The source EA is calculated by the calculate EA table, and the destination EA is calculated by the fetch EA table, even though the bus cycle is for the source EA.

5.7.2.4 SPECIAL-PURPOSE MOVE INSTRUCTION. The special-purpose MOVE instruction table indicates the number of clock periods needed for the processor to fetch, calculate, and perform the special-purpose MOVE operation on control registers or a specified EA. Footnotes indicate when to account for the appropriate EA times. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction	Head	Tail	Cycles
EXG Rn, Rm	2	0	4(0/1/0)
MOVEC Cr, Rn	10	0	14(0/2/0)
MOVEC Rn, Cr	12	0	14-16(0/1/0)
MOVE CCR, Dn	2	0	4(0/1/0)
MOVE CCR, ⟨CEA⟩	0	2	4(0/1/1)
MOVE Dn, CCR	2	0	4(0/1/0)
MOVE ⟨FEA⟩, CCR	0	0	4(0/1/0)
MOVE SR, Dn	2	0	4(0/1/0)
MOVE SR, ⟨CEA⟩	0	2	4(0/1/1)
MOVE Dn, SR	4	–2	10(0/3/0)
MOVE ⟨FEA⟩, SR	0	–2	10(0/3/0)
MOVEM.W⟨CEA⟩, RL	1	0	$8 + n \times 4(n + 1, 2, 0)^*$
MOVEM.WRL, ⟨CEA⟩	1	0	$8 + n \times 4(0, 2, n)^*$
MOVEM.L⟨CEA⟩, RL	1	0	$12 + n \times 4(2n + 2, 2, 0)$
MOVEM.LRL, ⟨CEA⟩	1	2	$10 + n \times 4\ (0, 2, 2n)$
MOVEP.WDn, (d_{16}, An)	2	0	10(0/2/2)
MOVEP.W(d_{16}, An), Dn	1	2	11(2/2/0)
MOVEP.LDn, (d_{16}, An)	2	0	14(0/2/4)
MOVEP.L(d_{16}, An), Dn	1	2	19(4/2/0)
MOVES (Save)⟨CEA⟩, Rn	1	1	3(0/1/0)
MOVES (Op)⟨CEA⟩, Rn	7	1	11(X/1/0)
MOVES (Save)Rn, ⟨CEA⟩	1	1	3(0/1/0)
MOVES (Op)Rn, ⟨CEA⟩	9	2	12(0/1/X)
MOVE USP, An	0	0	2(0/1/0)
MOVE An, USP	0	0	2(0/1/0)
SWAP Dn	4	0	6(0/1/0)

X = There is one bus cycle for byte and word operands and two bus cycles for long operands. For long bus cycles, add two clocks to the tail and to the number of cycles.

* = Each bus cycle may take up to four clocks without increasing total execution time.

Cr = Control registers USP, VBR, SFC, and DFC

n = Number of registers to transfer

RL = Register List

< = Maximum time (certain data or mode combinations may execute faster).

NOTES:

1. The MOVES instruction has an additional save step that other instructions do not have. To calculate the total instruction time, calculate the save, the EA, and the operation execution times, and combine in the order listed, using the equations given in 5.7.1 Resource Scheduling.
2. Timing is calculated with the CPU32+ in 16-bit mode.

5.7.2.5 ARITHMETIC/LOGIC INSTRUCTIONS. The arithmetic/logic instruction table indicates the number of clock periods needed to perform the specified arithmetic/logical instruction using the specified addressing mode. Footnotes indicate when to account for the appropriate EA times. The total number of clock cycles is outside the parentheses. The num-

bers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction	Head	Tail	Cycles
ADD(A) Rn, Rm	0	0	2(0/1/0)
ADD(A) ⟨FEA⟩, Rn	0	0	2(0/1/0)
ADD Dn, ⟨FEA⟩	0	3	5(0/1/x)
AND Dn, Dm	0	0	2(0/1/0)
AND ⟨FEA⟩, Dn	0	0	2(0/1/0)
AND Dn, ⟨FEA⟩	0	3	5(0/1/x)
EOR Dn, Dm	0	0	2(0/1/0)
EOR Dn, ⟨FEA⟩	0	3	5(0/1/x)
OR Dn, Dm	0	0	2(0/1/0)
OR ⟨FEA⟩, Dn	0	0	2(0/1/0)
OR Dn, ⟨FEA⟩	0	3	5(0/1/x)
SUB(A) Rn, Rm	0	0	2(0/1/0)
SUB(A) ⟨FEA⟩, Rn	0	0	2(0/1/0)
SUB Dn, ⟨FEA⟩	0	3	5(0/1/x)
CMP(A) Rn, Rm	0	0	2(0/1/0)
CMP(A) ⟨FEA⟩, Rn	0	0	2(0/1/0)
CMP2 (Save)* ⟨FEA⟩, Rn	1	1	3(0/1/0)
CMP2 (Op)⟨FEA⟩, Rn	2	0	16-18(X/1/0)
MUL(su).W⟨FEA⟩, Dn	0	0	26(0/1/0)
MUL(su).L (Save)* ⟨FEA⟩, Dn	1	1	3(0/1/0)
MUL(su).L (Op)⟨FEA⟩, Dl	2	0	46-52(0/1/0)
MUL(su).L (Op)⟨FEA⟩, Dn:Dl	2	0	46(0/1/0)
DIVU.W ⟨FEA⟩, Dn	0	0	32(0/1/0)
DIVS.W ⟨FEA⟩, Dn	0	0	42(0/1/0)
DIVU.L (Save)* ⟨FEA⟩, Dn	1	1	3(0/1/0)
DIVU.L (Op)⟨FEA⟩, Dn	2	0	<46(0/1/0)
DIVS.L (Save)* ⟨FEA⟩, Dn	1	1	3(0/1/0)
DIVS.L (Op)⟨FEA⟩, Dn	2	0	<62(0/1/0)
TBL(su) Dn:Dm, Dp	26	0	28-30(0/2/0)
TBL(su) (Save)* ⟨CEA⟩, Dn	1	1	3(0/1/0)
TBL(su) (Op)⟨CEA⟩, Dn	6	0	33-35(2X/1/0)
TBLSN Dn:Dm, Dp	30	0	30-34(0/2/0)
TBLSN (Save)* ⟨CEA⟩, Dn	1	1	3(0/1/0)
TBLSN (Op)⟨CEA⟩, Dn	6	0	35-39(2X/1/0)
TBLUN Dn:Dm, Dp	30	0	34-40(0/2/0)
TBLUN (Save)* ⟨CEA⟩, Dn	1	1	3(0/1/0)
TBLUN (Op)⟨CEA⟩, Dn	6	0	39-45(2X/1/0)

X = There is one bus cycle for byte and word operands and two bus cycles for long operands. For long bus cycles, add two clocks to the tail and to the number of cycles. Timing is calculated with the CPU32+ in 16-bit mode.
< = Maximum time (certain data or mode combinations may execute faster).
su = The execution time is identical for signed or unsigned operands.

* = These instructions have an additional save operation that other instructions do not have. To calculate total instruction time, calculate save, ⟨ea⟩, and operation execution times, then combine in the order listed, using equations in 5.7.1.6 Instruction Execution Time Calculation. A save operation is not run for long-word divide and multiply instructions when ⟨FEA⟩ = Dn.

5.7.2.6 IMMEDIATE ARITHMETIC/LOGIC INSTRUCTIONS. The immediate arithmetic/logic instruction table indicates the number of clock periods needed for the processor to fetch the source immediate data value and to perform the specified arithmetic/logic instruction using the specified addressing mode. Footnotes indicate when to account for the appropriate fetch effective or fetch immediate EA times. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction	Head	Tail	Cycles
MOVEQ#, Dn	0	0	2(0/1/0)
ADDQ #, Rn	0	0	2(0/1/0)
ADDQ #, ⟨FEA⟩	0	3	5(0/1/x)
SUBQ #, Rn	0	0	2(0/1/0)
SUBQ #, ⟨FEA⟩	0	3	5(0/1/x)
ADDI #, Rn	0	0	2(0/1/0)∗
ADDI #, ⟨FEA⟩	0	3	5(0/1/x)∗
ANDI #, Rn	0	0	2(0/1/0)∗
ANDI #, ⟨FEA⟩	0	3	5(0/1/x)∗
EORI #, Rn	0	0	2(0/1/0)∗
EORI #, ⟨FEA⟩	0	3	5(0/1/x)∗
ORI #, Rn	0	0	2(0/1/0)∗
ORI #, ⟨FEA⟩	0	3	5(0/1/x)∗
SUBI #, Rn	0	0	2(0/1/0)∗
SUBI #, ⟨FEA⟩	0	3	5(0/1/x)∗
CMPI #, Rn	0	0	2(0/1/0)∗
CMPI #, ⟨FEA⟩	0	3	5(0/1/x)∗

X = There is one bus cycle for byte and word operands and two bus cycles for long-word operands. For long-word bus cycles, add two clocks to the tail and to the number of cycles. Timing is calculated with the CPU32+ in 16-bit mode.
∗ = An # fetch EA time must be added for this instruction: ⟨FEA⟩ + ⟨FEA⟩ + ⟨OPER⟩

5.7.2.7 BINARY-CODED DECIMAL AND EXTENDED INSTRUCTIONS. The BCD and extended instruction table indicates the number of clock periods needed for the processor to perform the specified operation using the specified addressing mode. No additional tables are needed to calculate total effective execution time for these instructions. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction		Head	Tail	Cycles
ABCD	Dn, Dm	2	0	4(0/1/0)
ABCD	–(An), –(Am)	2	2	12(2/1/1)
SBCD	Dn, Dm	2	0	4(0/1/0)
SBCD	–(An), –(Am)	2	2	12(2/1/1)
ADDX	Dn, Dm	0	0	2(0/1/0)
ADDX	–(An), –(Am)	2	2	10(2/1/1)
SUBX	Dn, Dm	0	0	2(0/1/0)
SUBX	–(An), –(Am)	2	2	10(2/1/1)
CMPM	(An)+, (Am)+	1	0	8(2/1/0)

5.7.2.8 SINGLE OPERAND INSTRUCTIONS. The single operand instruction table indicates the number of clock periods needed for the processor to perform the specified operation using the specified addressing mode. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction		Head	Tail	Cycles
CLR	Dn	0	0	2(0/1/0)
CLR	⟨CEA⟩	0	2	4(0/1/X)
NEG	Dn	0	0	2(0/1/0)
NEG	⟨FEA⟩	0	3	5(0/1/X)
NEGX	Dn	0	0	2(0/1/0)
NEGX	⟨FEA⟩	0	3	5(0/1/X)
NOT	Dn	0	0	2(0/1/0)
NOT	⟨FEA⟩	0	3	5(0/1/X)
EXT	Dn	0	0	2(0/1/0)
NBCD	Dn	2	0	4(0/1/0)
NBCD	⟨FEA⟩	0	2	6(0/1/1)
Scc	Dn	2	0	4(0/1/0)
Scc	⟨CEA⟩	2	2	6(0/1/1)
TAS	Dn	4	0	6(0/1/0)
TAS	⟨CEA⟩	1	0	10(0/1/1)
TST	⟨FEA⟩	0	0	2(0/1/0)

X = There is one bus cycle for byte and word operands and two bus cycles for long-word operands. For long-word bus cycles, add two clocks to the tail and to the number of cycles.
Timing is calculated with the CPU32+ in 16-bit mode

5.7.2.9 SHIFT/ROTATE INSTRUCTIONS. The shift/rotate instruction table indicates the number of clock periods needed for the processor to perform the specified operation on the given addressing mode. Footnotes indicate when to account for the appropriate EA times. The number of bits shifted does not affect the execution time, unless noted. The total num-

ber of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction		Head	Tail	Cycles	Note
LSd	Dn, Dm	–2	0	(0/1/0)	1
LSd	#, Dm	4	0	6(0/1/0)	—
LSd	⟨FEA⟩	0	2	6(0/1/1)	—
ASd	Dn, Dm	–2	0	(0/1/0)	1
ASd	#, Dm	4	0	6(0/1/0)	—
ASd	⟨FEA⟩	0	2	6(0/1/1)	—
ROd	Dn, Dm	–2	0	(0/1/0)	1
ROd	#, Dm	4	0	6(0/1/0)	—
ROd	⟨FEA⟩	0	2	6(0/1/1)	—
ROXd	Dn, Dm	–2	0	(0/1/0)	2
ROXd	#, Dm	–2	0	(0/1/0)	3
ROXd	⟨FEA⟩	0	2	6(0/1/1)	—

d = Direction (left or right)

NOTES:

1. Head and cycle times can be derived from the following table or calculated as follows: Max (3 + (n/4) + mod(n,4) + mod (((n/4) + mod (n,4) + 1,2), 6)
2. Head and cycle times are calculated as follows: (count ≤ 63): max (3 + n + mod (n + 1,2), 6).
3. Head and cycle times are calculated as follows: (count ≤ 8): max (2 + n + mod (n,2), 6).
4. Timing is calculated with the CPU32+ in 16-bit mode.

Clocks	Shift Counts									
6	0	1	2	3	4	5	6	8	9	12
8	7	10	11	13	14	16	17	20		
10	15	18	19	21	22	24	25	28		
12	23	26	27	29	30	32	33	36		
14	31	34	35	37	38	40	41	44		
16	39	42	43	45	46	48	49	52		
18	47	50	51	53	54	56	57	60		
20	55	58	59	61	62					
22	63									

5.7.2.10 BIT MANIPULATION INSTRUCTIONS. The bit manipulation instruction table indicates the number of clock periods needed for the processor to perform the specified operation on the given addressing mode. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction		Head	Tail	Cycles
BCHG	#, Dn	2	0	6(0/2/0)∗
BCHG	Dn, Dm	4	0	6(0/1/0)
BCHG	#, ⟨FEA⟩	1	2	8(0/2/1)∗
BCHG	Dn, ⟨FEA⟩	2	2	8(0/1/1)
BCLR	#, Dn	2	0	6(0/2/0)∗
BCLR	Dn, Dm	4	0	6(0/1/0)
BCLR	#, ⟨FEA⟩	1	2	8(0/2/1)∗
BCLR	Dn, ⟨FEA⟩	2	2	8(0/1/1)
BSET	#, Dn	2	0	6(0/2/0)∗
BSET	Dn, Dm	4	0	6(0/1/0)
BSET	#, ⟨FEA⟩	1	2	8(0/2/1)∗
BSET	Dn, ⟨FEA⟩	2	2	8(0/1/1)
BTST	#, Dn	2	0	4(0/2/0)∗
BTST	Dn, Dm	2	0	4(0/1/0)
BTST	#, ⟨FEA⟩	1	0	4(0/2/0)∗
BTST	Dn, ⟨FEA⟩	2	0	8(0/1/0)

∗ = An # fetch EA time must be added for this instruction: ⟨FEA⟩ + ⟨FEA⟩ + ⟨OPER⟩
Timing is calculated with the CPU32+ in 16-bit mode.

5.7.2.11 CONDITIONAL BRANCH INSTRUCTIONS. The conditional branch instruction timing table indicates the number of clock periods needed for the processor to perform the specified branch on the given branch size, with complete execution times given. No additional tables are needed to calculate total effective execution time for these instructions. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction		Head	Tail	Cycles
Bcc	(taken)	2	–2	8(0/2/0)
Bcc.B	(not taken)	2	0	4(0/1/0)
Bcc.W	(not taken)	0	0	4(0/2/0)
Bcc.L	(not taken)	0	0	6(0/3/1)
DBcc	(T, not taken)	1	1	4(0/2/0)
DBcc	(F, –1, not taken)	2	0	6(0/2/0)
DBcc	(F, not –1, taken)	6	–2	10(0/2/0)
DBcc	(T, not taken)	4	0	6(0/1/0)∗
DBcc	(F, –1, not taken)	6	0	8(0/1/0)∗
DBcc	(F, not –1, taken)	6	0	10(0/0/0)∗

∗ = In loop mode
Timing is calculated with the CPU32+ in 16-bit mode.

5.7.2.12 CONTROL INSTRUCTIONS. The control instruction table indicates the number of clock periods needed for the processor to perform the specified operation on the given addressing mode. Footnotes indicate when to account for the appropriate EA times. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction	Head	Tail	Cycles
ANDI #, SR	0	–2	12(0/2/0)
EORI #, SR	0	–2	12(0/2/0)
ORI #, SR	0	–2	12(0/2/0)
ANDI #, CCR	2	0	6(0/2/0)
EORI #, CCR	2	0	6(0/2/0)
ORI #, CCR	2	0	6(0/2/0)
BSR.B	3	–2	13(0/2/2)
BSR.W	3	–2	13(0/2/2)
BSR.L	1	–2	13(0/2/2)
CHK ⟨FEA⟩, Dn (no ex)	2	0	8(0/1/0)
CHK ⟨FEA⟩, Dn (ex)	2	–2	42(2/2/6)
CHK2 (Save)⟨FEA⟩, Dn (no ex)	1	1	3(0/1/0)
CHK2 (Op)⟨FEA⟩, Dn (no ex)	2	0	18(X/0/0)
CHK2 (Save)⟨FEA⟩, Dn (ex)	1	1	3(0/1/0)
CHK2 (Op)⟨FEA⟩, Dn (ex)	2	–2	52(X + 2/1/6)
JMP ⟨CEA⟩	0	–2	6(0/2/0)
JSR ⟨CEA⟩	3	–2	13(0/2/2)
LEA ⟨CEA⟩, An	0	0	2(0/1/0)
LINK.W An, #	2	0	10(0/2/2)
LINK.L An, #	0	0	10(0/3/2)
NOP	0	0	2(0/1/0)
PEA ⟨CEA⟩	0	0	8(0/1/2)
RTD #	1	–2	12(2/2/0)
RTR	1	–2	14(3/2/0)
RTS	1	–2	12(2/2/0)
UNLK An	1	0	9(2/1/0)

X = There is one bus cycle for byte and word operands and two bus cycles for long-word operands. For long-word bus cycles, add two clocks to the tail and to the number of cycles.
Timing is calculated with the CPU32+ in 16-bit mode.

NOTE: The CHK2 instruction involves a save step that other instructions do not have. To calculate the total instruction time, calculate the save, the EA, and the operation execution times; then combine in the order listed using the equations given in 5.7.1 Resource Scheduling.

5.7.2.13 EXCEPTION-RELATED INSTRUCTIONS AND OPERATIONS. The exception-related instructions and operations table indicates the number of clock periods needed for the processor to perform the specified exception-related actions. No additional tables are needed to calculate total effective execution time for these instructions. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction	Head	Tail	Cycles
BKPT (Acknowledged)	0	0	14(1/0/0)
BKPT (Bus Error)	0	–2	35(3/2/4)
Breakpoint (Acknowledged)	0	0	10(1/0/0)
Breakpoint (Bus Error)	0	–2	42(3/2/6)
Interrupt	0	–2	30(3/2/4)∗
RESET	0	0	518(0/1/0)
STOP	2	0	12(0/1/0)
LPSTOP	3	–2	25(0/3/1)
Divide-by-Zero	0	–2	36(2/2/6)
Trace	0	–2	36(2/2/6)
TRAP #	4	–2	29(2/2/4)
ILLEGAL	0	–2	25(2/2/4)
A-line	0	–2	25(2/2/4)
F-line (First word illegal)	0	–2	25(2/2/4)
F-line (Second word illegal) ea = Rn	1	–2	31(2/3/4)
F-line (Second word illegal) ea ≠ Rn (Save)	1	1	3(0/1/0)
F-line (Second word illegal) ea ≠ Rn (Op)	4	–2	29(2/2/4)
Privileged	0	–2	25(2/2/4)
TRAPcc (trap)	2	–2	38(2/2/6)
TRAPcc (no trap)	2	0	4(0/1/0)
TRAPcc.W (trap)	2	–2	38(2/2/6)
TRAPcc.W (no trap)	0	0	4(0/2/0)
TRAPcc.L (trap)	0	–2	38(2/2/6)
TRAPcc.L (no trap)	0	0	6(0/3/0)
TRAPV (trap)	2	–2	38(2/2/6)
TRAPV (no trap)	2	0	4(0/1/0)

∗ = Minimum interrupt acknowledge cycle time is assumed to be three clocks.
Timing is calculated with the CPU32+ in 16-bit mode.

NOTE: The F-line (second word illegal) operation involves a save step which other operations do not have. To calculate the total operation time, calculate the save, the calculate EA, and the operation execution times, and combine in the order listed, using the equations given in 5.7.1.6 Instruction Execution Time Calculation.

5.7.2.14 SAVE AND RESTORE OPERATIONS. The save and restore operations table indicates the number of clock periods needed for the processor to perform the specified state save or return from exception. Complete execution times and stack length are given. No additional tables are needed to calculate total effective execution time for these instructions. The total number of clock cycles is outside the parentheses. The numbers inside parentheses (r/p/w) are included in the total clock cycle number. All timing data assumes two-clock reads and writes.

Instruction	Head	Tail	Cycles
BERR on instruction	0	–2	<58(2/2/12)
BERR on exception	0	–2	48(2/2/12)
RTE (four-word frame)	1	–2	24(4/2/0)
RTE (six-word frame)	1	–2	26(4/2/0)
RTE (BERR on instruction)	1	–2	50(12/12/Y)
RTE (BERR on four-word frame)	1	–2	66(10/2/4)
RTE (BERR on six-word frame)	1	–2	70(12/2/6)

Y = If a bus error occurred during a write cycle, the cycle is rerun by the RTE.

< = Maximum time is indicated (certain data or mode combinations execute faster).
Timing is calculated with the CPU32+ in 16-bit mode.

SECTION 6
SYSTEM INTEGRATION MODULE (SIM60)

The QUICC's system integration module (SIM60) consists of a number of functions that control system startup, system initialization, the external system bus, and the external system peripherals. The SIM60 functions include the following:

- Module Base Address Register (MBAR)
- System Configuration and Protection
- New Low-Power Standby Modes with Slow-Go Option
- Clock Synthesizer with Skew Elimination
- Breakpoint Logic
- Slave Mode Including MC68040 Companion Mode
- External Bus Interface (EBI) Control
- Memory Controller Supports Eight Banks of DRAM, SRAM, EPROM, or Peripherals
- Dynamic Bus Sizing
- External Master Support
- Bus Arbitration
- IEEE 1149.1 Test Access Port

6.1 MODULE OVERVIEW

The SIM60 on the QUICC device is an enhanced version of the SIM40 that is implemented on another M68300 family device called the MC68340. The italicized items show the main areas of enhancement. To a large extent, the other features are still compatible with the older SIM40.

The MBAR provides the base address for all accesses to the SIM60 and every other on-chip resource.

The system configuration and protection function controls the overall system configuration and provides various monitors and timers, including the internal bus monitor, double bus fault monitor, spurious interrupt monitor, software watchdog timer, periodic interrupt timer, low-power stop support, and freeze support.

The clock synthesizer generates the clock signals used by the SIM60 as well as other modules and external devices. This circuitry can generate the system clock from an inexpensive 32.768-khz watch crystal.

The SIM60 has additional support of low-power modes. The clock synthesizer provides system clocks to the SIM60 and other modules. This clock scheme supports low-power modes for applications that use the baud rate generators and/or serial ports during the standby mode. The main system clock can be changed dynamically (the slow-go option) while the baud rate generators and serial ports work with a fixed frequency.

The breakpoint logic provides an internal breakpoint address register that allows hardware breakpoints in a QUICC system. This function is especially useful during in-field debugging activity when it is difficult to connect an in-circuit emulator or logic analyzer to the target board.

The QUICC supports the slave mode. In this mode, the CPU32+ core on the QUICC is disabled, and the QUICC functions as an intelligent peripheral. For instance, if the application requires more serial channels than the QUICC provides, multiple QUICCs may be configured onto the same system bus, one with its CPU enabled and the rest in slave mode. Alternatively, if the application needs additional CPU performance, the QUICC may function as a companion chip to an MC68EC040 (or other M68040 family member). This is called MC68040 companion mode. In this mode, the QUICC's glueless interface to the MC68EC040 provides a two-chip MC68EC040 system solution. The MC68EC040 can also control multiple QUICCs in slave mode. Finally, the QUICC slave mode may also support an external MC68EC030 or other M68030 family member.

The EBI handles the transfer of information between the internal CPU32+ core and memory, peripherals, or other processing elements in the external address space, or between an external master and the QUICC RAM and registers. Section 4 Bus Operation describes the bus operation, but the configuration control of the EBI is contained in this section.

The following functions are physically part of the SIM60, but are described in other places in this manual.

The memory controller module provides glueless interfaces to many types of memory and peripherals. It contains up to 8 general-purpose chip selects with up to 15 wait states each and a full DRAM controller that controls up to 8 DRAM banks. See 6.10 Memory Controller for further information.

The QUICC dynamically interprets the bus port size of an addressed device during each bus cycle, allowing operand transfers to/from 8-, 16-, and 32-bit ports. The $\overline{\text{DSACK}}$ signals are used to signify the data port size. Dynamic bus sizing can result in reduced system cost. For instance, an 8-bit boot EPROM may be used with 16-bit peripherals and 32-bit DRAM. Dynamic bus sizing also allows a programmer to write code that is not bus-width specific. For a discussion on dynamic bus sizing see Section 4 Bus Operation.

The QUICC is designed to allow external bus masters the opportunity to access the intermodule bus (IMB). This design has two main purposes. First, the RAM and peripherals on the QUICC can be directly accessed, if desired, by an external master. Second, the external master can use QUICC resources, such as the chip-select generation logic and DRAM controller. See Section 4 Bus Operation for further discussion.

The QUICC also provides the ability to request and obtain mastership of the system bus. This logic is only reset during a power-on reset and is active at all other times. See Section 4 Bus Operation for further discussion.

The QUICC includes dedicated user-accessible test logic that is fully compliant with the IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture. This standard was developed under the sponsorship of the IEEE Test Technology Committee and Joint Test Action Group (JTAG). The QUICC implementation supports circuit-board test strategies based on this standard. Refer to Section 8 Scan Chain Test Access Port for additional information.

The following paragraphs describe the operation of the MBAR, system configuration and protection, clock synthesizer, breakpoint logic, slave mode, and EBI control.

6.2 MODULE BASE ADDRESS REGISTER (MBAR)

The MBAR controls the location of all module registers (see 6.9.1 Module Base Address Register (MBAR)). The address stored in this register is the base address (starting location) for the internal module registers. All internal module registers and RAM occupy a single 8-kbyte memory block (see Figure 6-1) that is relocatable along 8-Kbyte boundaries. The location of the internal registers is fixed by writing the desired base address of the 8-Kbyte block to the MBAR using the MOVES instruction to address $0003FF00 in CPU space. The source function code (SFC) and destination function code (DFC) registers contain the address space values (FC2–FC0) for the read or write operand of the MOVES instruction (see Section 5 CPU32+ or M68000PM/AD, *Programmer's Reference Manual*). Therefore, the SFC or DFC register must indicate CPU space (FC2–FC0 = $7), using the MOVEC instruction, before accessing the MBAR.

6.3 SYSTEM CONFIGURATION AND PROTECTION

The SIM60 allows the user to control certain features of system configuration by writing bits in the module configuration register (MCR).

All M68000 family members are designed to provide maximum system safeguards. As an extension of the family, the QUICC promotes the same basic concepts of safeguarded design present in all M68000 members. In addition, many functions that normally must be provided in external circuits are incorporated in this device. The following features are provided in the system configuration and protection sub-module:

SIM60 Configuration

The SIM60 allows the user to configure the system according to the particular requirements. The functions include control of slave mode (disable CPU32+) operation, freeze and show cycle operation, the access privilege of the supervisor/user registers, the level of interrupt arbitration, and automatic autovectoring for external interrupts.

Reset Status

The reset status register provides the user with information on the cause of the most recent reset. The possible causes include external, power-up, software watchdog, double bus fault, loss of clock, and the RESET instruction.

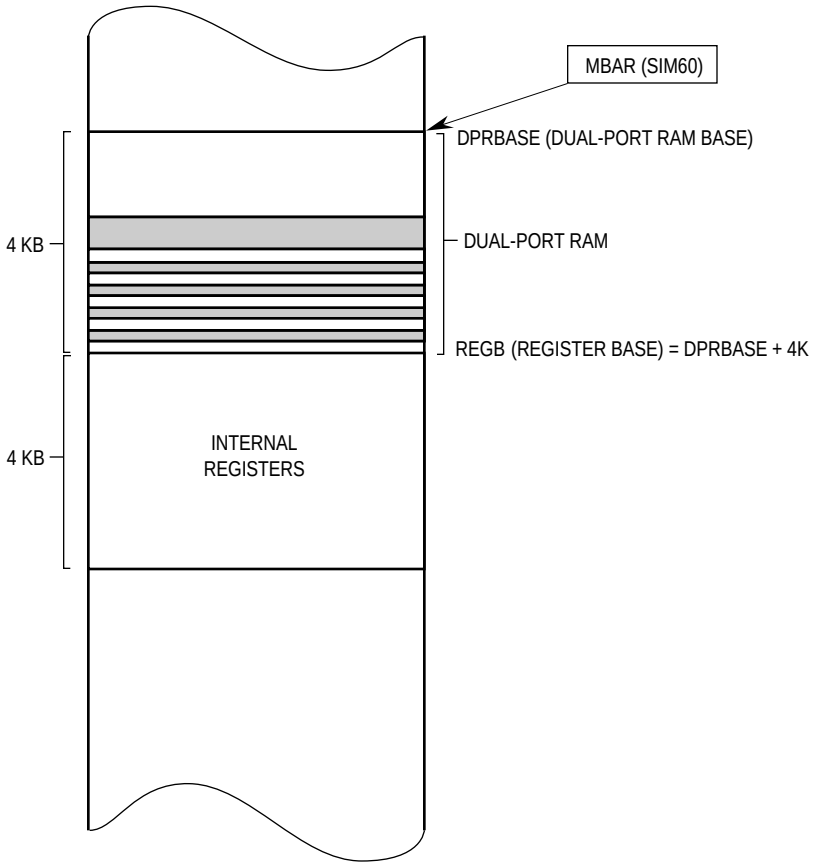


Figure 6-1. QUICC Memory Map

Bus Monitor

The SIM60 provides a bus monitor to monitor the data and size acknowledge ($\overline{\text{DSACK}}$) response time for all bus accesses (internal-to-internal, internal-to-external, external-to-internal, and external-to-external). Four selectable response times allow for variations in response speed of memory and peripherals used in the system. A bus error signal is asserted if the $\overline{\text{DSACK}}$ response limit is exceeded. This function can be disabled.

NOTE

On the MC68302, this function is called the hardware watchdog.

Double Bus Fault Monitor

The double bus fault monitor causes a reset to occur if the internal $\overline{\text{HALT}}$ signal is asserted by the CPU32+, indicating a double bus fault. A double bus fault results when a bus or address error occurs during the exception processing sequence for a previous bus or address error, a reset, or while the CPU32+ is loading information from a bus error stack frame during an RTE instruction. This function can be disabled. See Section 4 Bus Operation for more information.

Spurious Interrupt Monitor

If no interrupt arbitration occurs during an interrupt acknowledge cycle, the bus error signal is asserted internally.

Software Watchdog Timer (SWT)

The SWT asserts a reset or level 7 interrupt (as selected by the system protection control register (SYPCR)) if the software fails to service the SWT for a designated period of time (i.e., because the software is trapped in a loop or lost). There are eight selectable timeout periods. After a system reset, this function is enabled, selects a timeout of approximately 1 second, and asserts a system reset if the timeout is reached. The SWT may be disabled, or its timeout period may be changed in the SYPCR; however, once SYPCR is written, it cannot be written again until a system reset. This mechanism is used to ensure the proper operation of the SWT.

Periodic Interrupt Timer (PIT)

The SIM60 provides a timer to generate periodic interrupts for use with a real-time operating system or the application software. The PIT period can vary from 122 ms to 15.94 s (assuming a 32.768-kHz crystal is used to generate the general system clock). This function can be disabled.

Freeze Support

The SIM60 allows control of whether the SWT and PIT should continue to run during freeze mode.

Low-Power Stop Support

When executing the LPSTOP instruction, the QUICC can provide reduced power consumption with only the SIM60 remaining active.

Low-Power Standby Support

In addition to the low-power stop support, the QUICC can provide low power consumption while other modules or sub-modules are functioning. In this mode, the baud rate generators and serial ports run with a fixed frequency while the rest of the chip (including the SIM60) runs with a divided clock.

Figure 6-2 shows a block diagram of the system configuration and protection logic.

6.3.1 System Configuration

Many aspects of the system configuration are controlled by the MCR.

For debug purposes, accesses to internal peripherals can be shown on the external bus. This function is called show cycles. The SHEN1, SHEN0 bits in the MCR control the show cycles. External bus arbitration can be either enabled or disabled during show cycles.

The SIM60 provides eight bus arbitration levels for determining the priority of bus access (0–7). The SIM60 is fixed at the highest level (level 7). The CPU32+ is fixed at the lowest level (level 0). Only the SIM60, the CPU32+, the two-channel independent direct memory access (IDMA), and the serial direct memory access (SDMA) can be bus masters and arbitrate for

the bus. (The IDMA and SDMA have the ability to configure their bus arbitration level as described in Section 7 Communication Processor Module (CPM)).

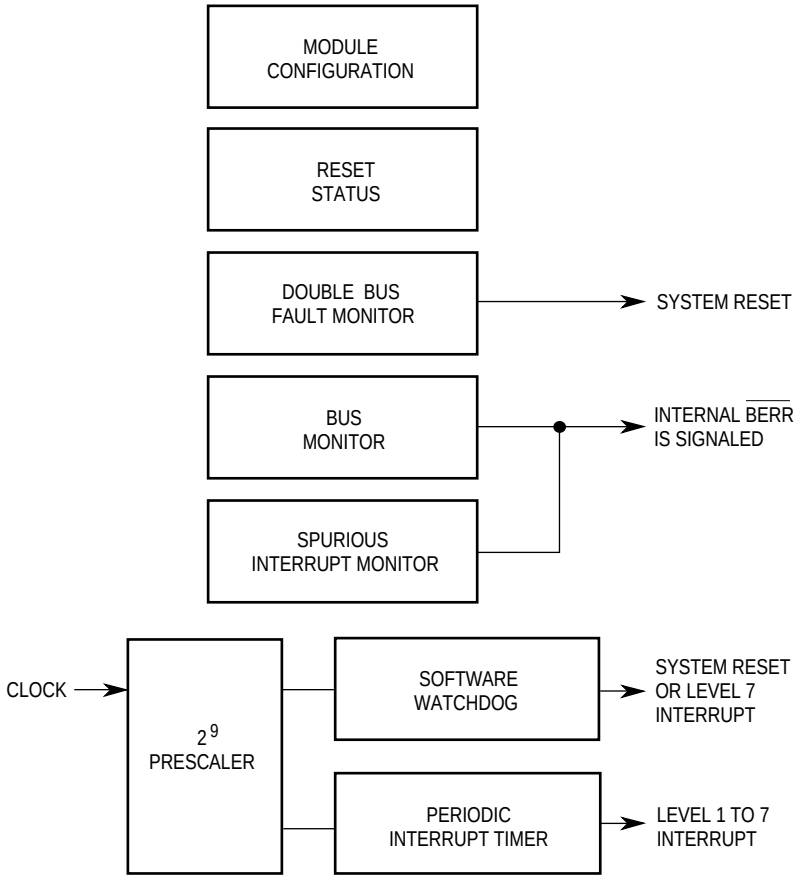


Figure 6-2. System Configuration and Protection Logic

6.3.1.1 SIM60 INTERRUPT GENERATION. An overview of the QUICC interrupt structure is shown in Figure 6-3. The lower half of the figure shows the SIM60. The SIM60 receives interrupts from internal sources, such as the SWT and PIT, and external sources, such as the $\overline{\text{IRQ7}}$–$\overline{\text{IRQ1}}$ lines.

If it generates an interrupt, the SWT always uses level 7; the PIT may use any level. The $\overline{\text{IRQx}}$ pins choose the interrupt level associated with the pin (i.e., $\overline{\text{IRQ1}}$ generates a level 1 interrupt, etc.). In addition, the CPM block may choose any level (1–7) for its interrupts.

The IMB architecture allows multiple interrupt sources to safely exist at the same level, a process called interrupt arbitration. Once an interrupt acknowledge cycle occurs at the interrupt level that matches a pending interrupt request, interrupt arbitration begins on the IMB. The interrupt arbitration process is designed to choose between multiple requests at the same level. For instance, if the PIT request is at level 4 but the CPM simultaneously is requesting an interrupt at level 4, an interrupt arbitration process is required to decide who wins the interrupt. (The interrupt arbitration process does not affect users who assign all interrupt sources in the system to a unique interrupt level (1–7)).

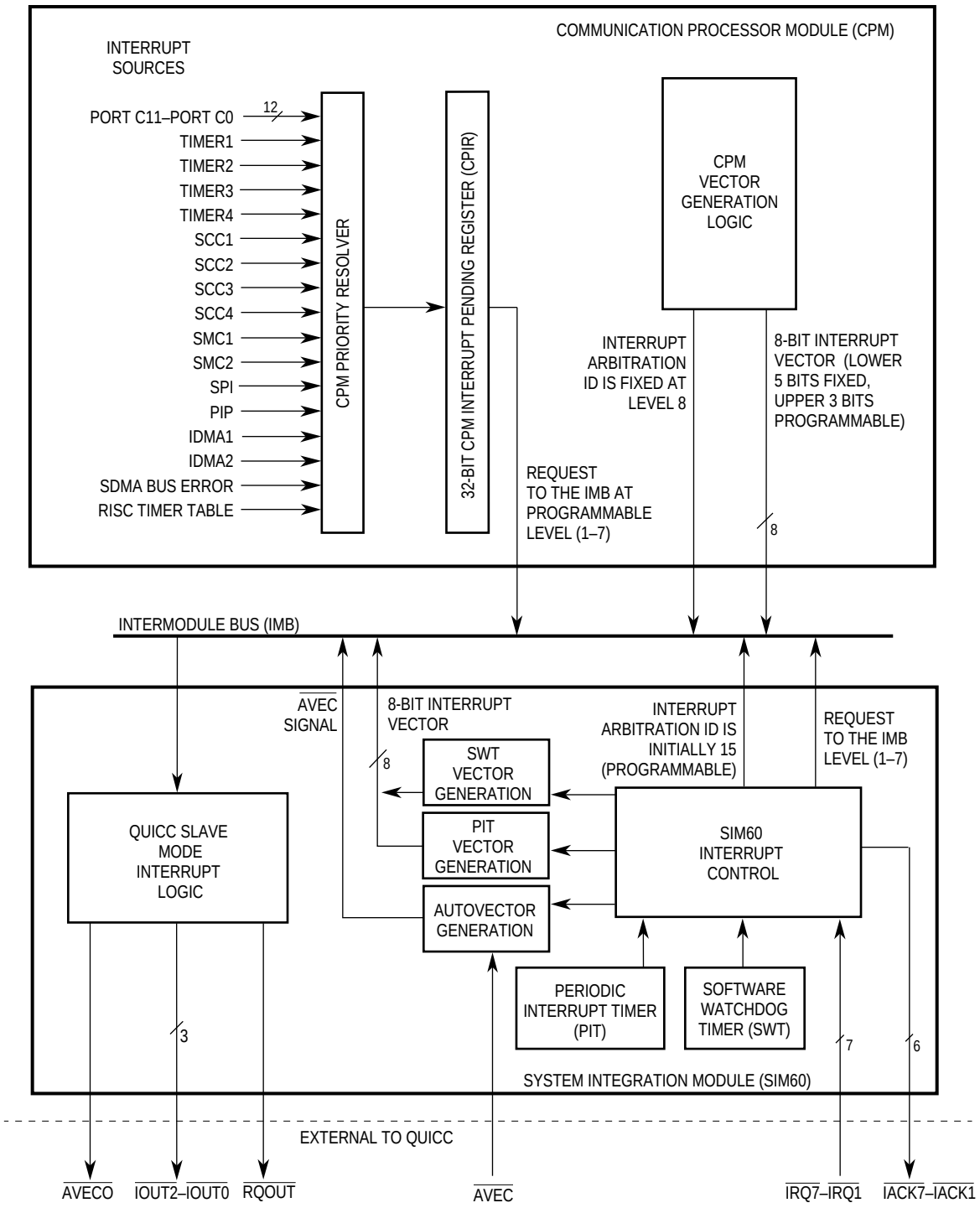


Figure 6-3. QUICC Interrupt Structure

In the interrupt arbitration process, the module places its arbitration ID on the IMB. The arbitration ID ranges in value from 0 to 15. The SIM60 interrupt controller arbitration ID is initialized to 15 (the highest value), but may be lowered if desired. The higher arbitration value always wins.

NOTES

At system reset, the SIM60 has a higher priority (at the same interrupt level) than the CPM. This priority can be changed if the SIM60 value is written to be less than 8, the level of the CPM.

No two modules are allowed to have the same interrupt arbitration value.

Assuming that the PIT wins the arbitration process, the SIM60 places the PIT 8-bit vector on the bus. The SWT also has a user-defined interrupt vector. The PIT and SWT interrupts do not allow autovectors. The IRQx lines can be vectored (externally supplied) or autovectored.

Arbitration for servicing interrupts is controlled by the value programmed into the interrupt arbitration (IARB) field of the MCR. Because no two modules are allowed to share the same IARB value and the only other module that generates interrupts (the CPM) has a fixed IARB value (IARB = 8), the SIM60 IARB value should be programmed to a value between 1 and 7 or between 9 and 15.

The autovector register (AVR) contains bits that correspond to external interrupt levels that require an autovector response. The SIM60 supports up to seven discrete external interrupt requests. If the bit corresponding to an interrupt level is set in the AVR, the SIM60 returns an internal autovector in response to the interrupt acknowledge cycle servicing that external interrupt request. Otherwise, external circuitry must either return an interrupt vector or externally assert the external $\overline{\text{AVEC}}$ signal.

See 6.8.4 Interrupts in Slave Mode for more information.

6.3.1.2 SIMULTANEOUS SIM60 INTERRUPT SOURCES. If the possible level 7 interrupt sources in the SIM60 are simultaneously asserted, the SIM60 will prioritize and service the interrupts in the following order: 1) SWT, 2) PIT, and 3) external interrupts. At level 6 or less, the PIT is higher than an external interrupt request asserted at the same level as the PIT.

6.3.1.2.1 Bus Monitor. The bus monitor ensures that each bus cycle is terminated within a reasonable period of time. It continually checks the duration of the internal/external $\overline{\text{AS}}$ line ($\overline{\text{TS}}$ for 68040). $\overline{\text{AS}}$ is normally negated by $\overline{\text{DSACK}}$x, $\overline{\text{BERR}}$, ($\overline{\text{TA}}$ or $\overline{\text{TEA}}$ for 68040).or $\overline{\text{HALT}}$ (or $\overline{\text{AVEC}}$ during an interrupt acknowledge cycle) The bus monitor asserts $\overline{\text{BERR}}$ if the response time is excessive on any bus cycle including interrupt acknowledge cycles. The BME bit in the SYPCR enables the bus monitor.

The bus cycle termination response time is measured in clock cycles, and the maximum-allowable response time is programmable. The bus monitor response time period ranges from 128 to 1K system clocks (see Table 6-5). The value chosen by the user should be larger than the longest possible response time of the slowest peripheral in the system.

6.3.1.2.2 Spurious Interrupt Monitor. In normal interrupt handling, one or more internal sub-modules recognize the CPU32+ interrupt acknowledge cycle as a signal that the CPU32+ is responding to their interrupt requests. The sub-modules then arbitrate for the privilege of returning a vector or asserting $\overline{\text{AVEC}}$ to the CPU32+. (The SIM60 also performs

internal interrupt arbitration on behalf of any $\overline{IRQx}$ pins that are asserted externally.) If, however, no internal sub-module participates in the internal interrupt arbitration process, the spurious interrupt monitor takes action by issuing the $\overline{BERR}$ signal internally. This causes the CPU32+ to terminate the cycle with a spurious interrupt vector. This feature cannot be disabled.

6.3.1.2.3 Double Bus Fault Monitor. A double bus fault is caused by a bus error or address error during the exception processing sequence. The double bus fault monitor responds to an assertion of $\overline{HALT}$ on the internal bus by initiating a system reset. Refer to Section 4 Bus Operation for more information. The DBF bit in the reset status register indicates that the last reset was caused by the double bus fault monitor. The double bus fault monitor reset can be enabled by the DBFE bit in the SYPCR.

6.3.1.2.4 Software Watchdog Timer (SWT). The SIM60 provides the SWT option to prevent system lockup in case the software becomes trapped in loops with no controlled exit. The SWT is enabled after system reset to cause a system reset if it times out. If SWT is not desired, the user must clear the SWE bit in the SYPCR to disable it. If used, the SWT requires a special service sequence to be executed on a periodic basis. If this periodic servicing action does not occur, the SWT times out and issues a reset or a level 7 interrupt (as programmed by the SWRI bit in the SYPCR). Once the SYPCR is written by software, the state of the SWT (enabled or disabled) cannot be changed. The address of the interrupt service routine for the SWT interrupt is stored in the software interrupt vector register (SWIV). Figure 6-4 shows a block diagram of the SWT as well as the clock control circuits for the PIT.

The SWT clock rate is determined by the SWP bit in the periodic interrupt timer register (PITR) and the SWT bits in the SYPCR. When MODCK1 is low (an external oscillator is used), the 512 (2^9) prescaler is enabled, and the SWP and PTP bits in the PITR are set. See Table 6-4 for a list of SWT timeout periods.

The SWT service sequence consists of the following two steps: write $55 to the software service register (SWSR) and write $AA to the SWSR. Both writes must occur in the order listed prior to the SWT timeout, but any number of instructions or accesses to the SWSR can be executed between the two writes. This allows interrupts and exceptions to occur, if necessary, between the two writes.

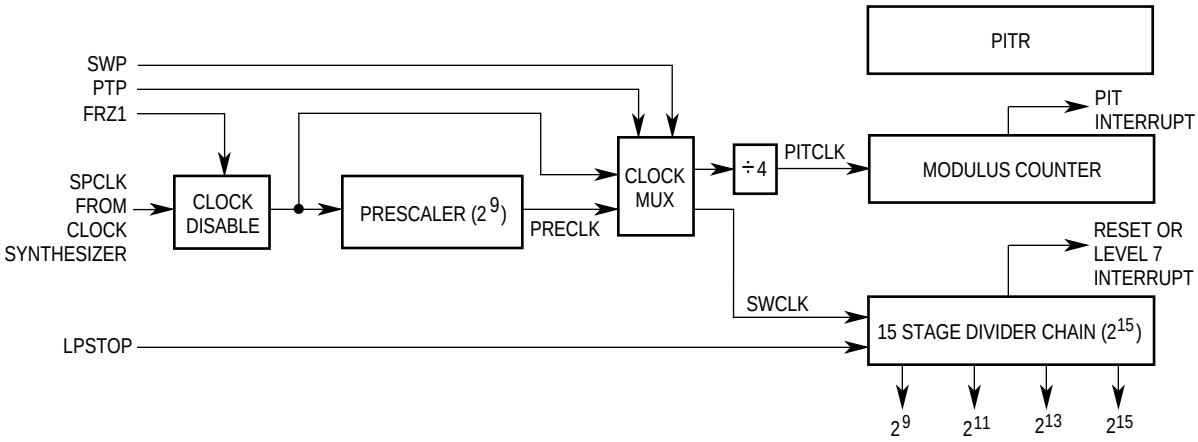


Figure 6-4. SWT and PIT Block Diagram

6.3.2 Periodic Interrupt Timer (PIT)

The PIT consists of an 8-bit modulus counter that is loaded with the value contained in the PITR (see Table 6-4). The modulus counter is clocked by the SPCLK signal derived from the EXTAL pin (either EXTAL or EXTAL divided by 128 as determined by the MODCK1–MODCK0 pins). When MODCK1 is low (an external oscillator is used), the 512 (2^9) prescaler is enabled, and the SWP and PTP bits in the PITR are set.

The clock source is divided by 4 before driving the modulus counter (PITCLK). When the modulus counter value reaches zero, an interrupt is generated. The level of the generated interrupt is programmed into the PIRQL bits in the periodic interrupt control register (PICR). During the interrupt acknowledge cycle, the SIM60 places the periodic interrupt vector, programmed into the PIV bits in the PICR, onto the bus. The value of bits 7–0 in the PITR is then loaded again into the modulus counter, and the counting process starts over. If a new value is written to the PITR, this value is loaded into the modulus counter when the current count is complete.

6.3.2.1 PIT PERIOD CALCULATION. The period of the PIT can be calculated using the following equation:

$$\text{PIT period} = \frac{\dfrac{\text{PITR count value}}{\text{SPCLK / prescaler value}}}{2^2}$$

Solving the equation using a crystal frequency of 32.768-kHz with the prescaler disabled gives:

$$\text{PIT period} = \frac{\dfrac{\text{PITR count value}}{32768/1}}{2^2}$$

$$\text{PIT period} = \frac{\text{PITR count value}}{8192}$$

This gives a range from 122 μs, with a PITR value of $01 (00000001 binary), to 31.128 ms, with a PITR value of $FF (11111111 binary).

Solving the equation with the prescaler enabled (PTP = 1) gives the following values:

$$\text{PIT period} = \frac{\dfrac{\text{PITR count value}}{32768/512}}{2^2}$$

$$\text{PIT period} = \frac{\text{PITR count value}}{16}$$

This gives a range from 62.5 ms, with a PITR value of $01, to 15.94 s, with a PITR value of $FF.

For a fast calculation of PIT period using a 32.768-kHz crystal, the following equations can be used:

With prescaler disabled:

PIT period = PITR (122 μs)

With prescaler enabled:

PIT period = PITR (62.5 ms)

6.3.2.2 USING THE PIT AS A REAL-TIME CLOCK. The PIT can be used as a real-time clock interrupt by setting it up to generate an interrupt with a 1-second period. When using a 32.768-kHz (or 4.192-MHz) crystal, the PITR should be loaded with a value of 16 decimal ($10) with the prescaler enabled to generate interrupts at a 1-sec rate.

6.3.3 Freeze Support

FREEZE is asserted by the CPU32+ if a breakpoint is encountered with background mode enabled. Refer to Section 5 CPU32+ for more information on the background mode. When FREEZE is asserted, the double bus fault monitor and spurious interrupt monitor continue to operate normally. However, the SWT, the bus monitor, and the PIT may be affected. Setting the FRZ1 bit in the MCR disables the SWT and the PIT when FREEZE is asserted. Setting the FRZ0 bit in the MCR disables the bus monitor when FREEZE is asserted.

If the CONFIG pins are configured with the CPU32+ core enabled, then one clock after reset is complete, the CONFIG2 pin will become the FREEZE output. Thus, the pin will start driving low one clock after reset. It will then be asserted (high) if the freeze condition occurs. If the CONFIG pins configure the QUICC to slave mode, then the FREEZE output is not available.

6.3.4 Low-Power Stop Support

Executing the LPSTOP instruction provides reduced power consumption when the QUICC is idle, with only the SIM remaining active. Operation of the SIM60 is controlled by the PLLCR. LPSTOP disables the clock to the SWT in the low state. The SWT, which remains stopped until the LPSTOP mode is ended, begins to run again on the next rising clock edge.

NOTE

When the CPU32+ executes the STOP instruction (as opposed to LPSTOP), the SWT continues to run. If the SWT is enabled, it issues a reset or interrupt when its timeout occurs.

The PIT does not respond to an LPSTOP instruction; thus, it can be used to exit LPSTOP as long as the interrupt request level is higher than the CPU32+ interrupt mask level. To stop the PIT while in LPSTOP, the PITR must be loaded with a zero value before LPSTOP is executed. The bus monitor, double bus fault monitor, and spurious interrupt monitor are all inactive during LPSTOP.

If an external device requires additional time to prepare for entry into LPSTOP mode, entry can be delayed by asserting $\overline{\text{HALT}}$ (see 4.4.2 LPSTOP Broadcast Cycle).

NOTE

The IDMA channels should be disabled prior to issuing the LPSTOP instruction.

6.4 LOW POWER IN NORMAL OPERATION

In addition to the LPSTOP mode, the SIM60 supports methods to minimize power consumption in normal operation. In normal operation, the QUICC provides several options to reduce power consumption:

- The sub-block clock generators are automatically disabled when the sub-block is not active. For example, when the RISC controller is idle (no pending request is present from the serial channels), its clock generator is automatically stopped.
- In most of the IMB sub-modules, there is a bit (e.g., STOP or RESET), that can disable the clock generator in that module (except for its IMB interface unit).

The SIM60 also supports methods to reduce power by dividing the clocks internally. This is called slow-go mode and is described in 6.5 SIM60 System Clock Generation.

6.5 SIM60 SYSTEM CLOCK GENERATION

The QUICC has an on-chip oscillator, a clock synthesizer, and a low-power divider, which allow a comprehensive set of choices in generating system clocks (see Figure 6-5). The choices offer many opportunities to save power and system cost, without sacrificing flexibility and control.

The operation of the clocks is determined by three registers: the clock out control register (CLKOCR), the phase-locked loop control register (PLLCR), and the clock divider control register (CDVCR). Each register has a protection mechanism to prevent accidental writing.

The clock generation features are discussed in the following paragraphs.

6.5.1 Clock Generation Methods

The first method drives the system clock at the desired system frequency (10–25 MHz), directly onto the EXTAL pin. A second method drives the EXTAL pin with a selected frequency which is pre-scaled and multiplied by the PLL. With these two methods, the XTAL pin should be left floating. A third method uses a reference crystal frequency which can also be pre-scaled and multiplied. This can be any frequency from 25 kHz to 6.0 MHz. Figure 6-6 shows external connections required for the on-chip oscillator as well as the other clock-related V_{CC} and GND connections.

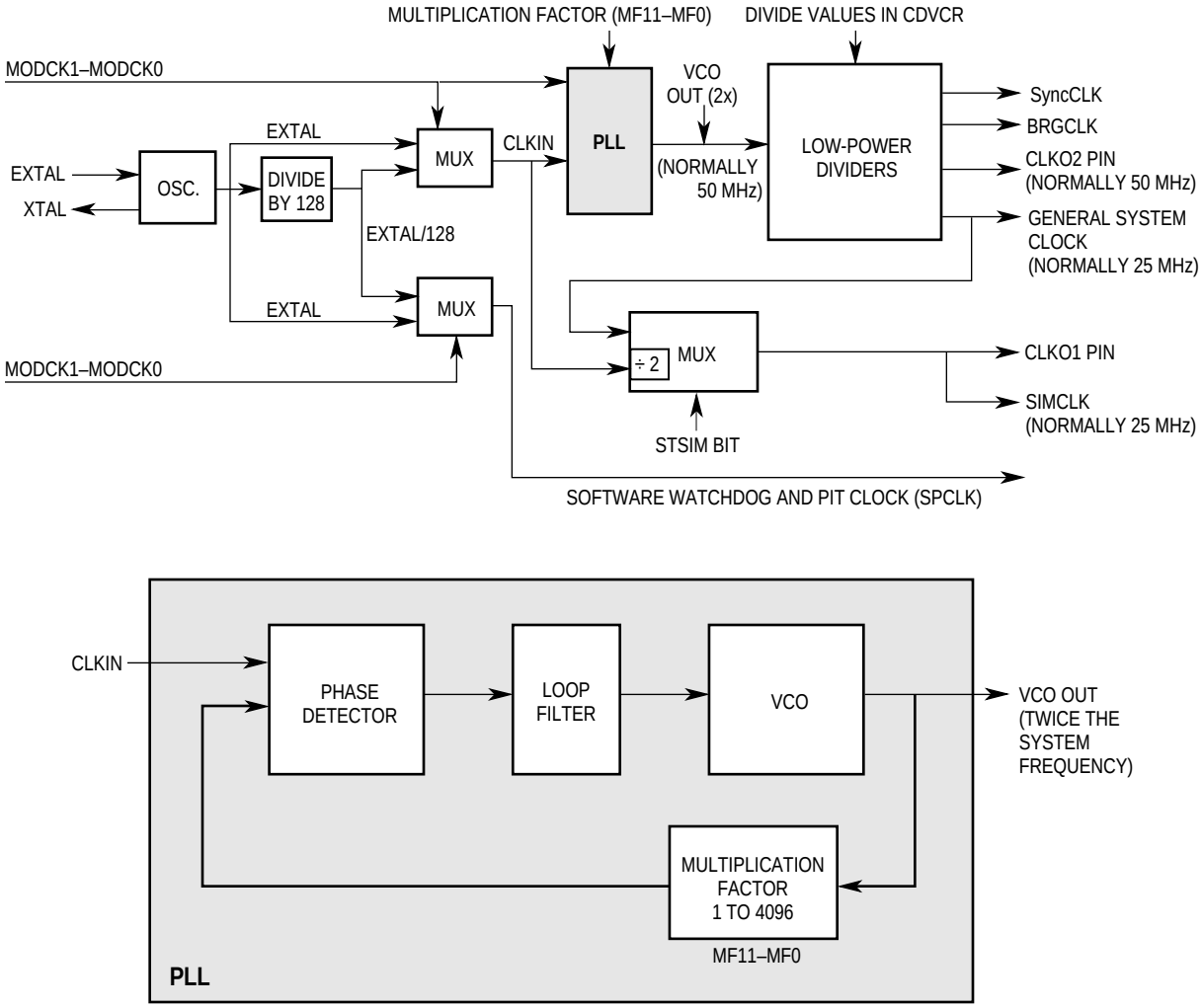


Figure 6-5. System Clocks Schematic

NOTE

User must select the proper MODCK configuration as well as correct XFC capacitor value when selecting the input clock frequency and desired system frequency.

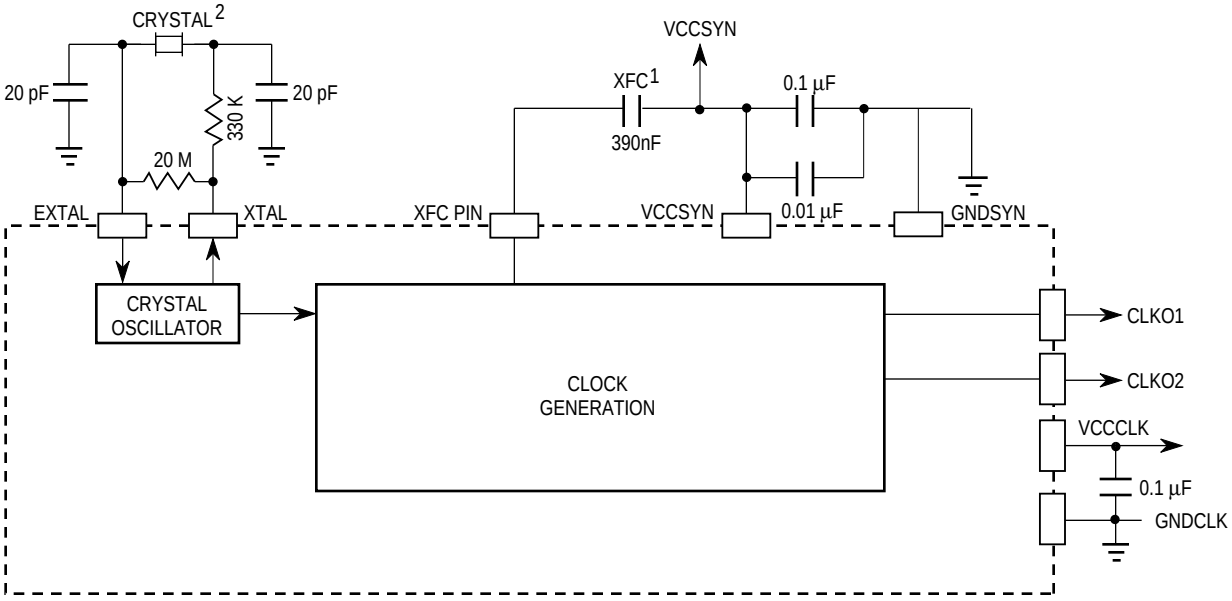
For more information on MODCK see 6.5.8 Configuration Pins (MODCK1–MODCK0). For more information on XFC see 10.8 External Capacitor for PLL.

6.5.2 Oscillator Prescaler (Divide by 128)

In some applications, the use of a ~ 32-kHz crystal is attractive because of cost and low power, but is not attractive due to the extra startup time required for such a slow frequency crystal. Therefore, the SIM60 has an option for the user to provide a higher frequency crystal (for instance, in the ~ 4-MHz range) and divide it by 128 (back down to the 32-kHz range) before it is used by the QUICC. This results in much faster startup time than a 32-kHz crystal, plus low cost (if a common 4.192-MHz frequency is chosen), with only a small impact on power consumption during low-power modes (since the ~4-MHz frequency is immediately

divided prior to being used by any QUICC on-chip module). Furthermore, the divide-by-128 function allows the value of the final system frequency to be chosen with much greater precision, since it is a multiple of ~32 kHz rather than a multiple of ~4 MHz.

The choice of whether to use the divide-by-128 function is made with the MODCK1–MODCK0 pins. This resulting frequency is called CLKIN.



NOTE:
1. Must be low-leakage capacitor. See Section 10 Electrical Characteristics for recommended values.
2. Values are for 32 kHz crystal and may vary due to capacitance on PCB.

Figure 6-6. External Components

6.5.3 Phase-Locked Loop (PLL)

The PLL takes the CLKIN frequency and outputs a high-frequency source used to derive the general system frequency of the QUICC. The PLL is comprised of a phase detector, loop filter, voltage-controlled oscillator (VCO), and multiplication block. The VCO output can be as high as 50 MHz for a 25-MHz QUICC.

The PLL's main functions are frequency multiplication and skew elimination.

6.5.3.1 FREQUENCY MULTIPLICATION. The PLL can multiply the CLKIN input frequency by any integer between 1 and 4096. The output of the VCO is twice the QUICC system frequency after reset.

If a low frequency crystal is chosen (e.g., ~32 kHz), the multiplier defaults to 401, giving a 2× VCO output of ~26 MHz and an initial general system clock of ~13 MHz. The multiplication factor may then be changed to the desired value by writing the MF11–MF0 bits in the PLLCR. When the PLL multiplier is modified in software, the PLL will lose lock, and the clocking to the QUICC will stop until lock is regained (worst case is 2500 clocks; typical case is 500 clocks). See 6.5.4 Low-Power Divider for methods of reducing clock rates without losing lock.

If the actual system clock is placed on the EXTAL pin (rather than an external crystal), the multiplier defaults to 2, giving a 2× VCO output of 2× the EXTAL frequency. The multiplier bits should not be modified by the user in this case. See 6.5.4 Low-Power Divider for methods of reducing clock rates.

NOTE

If the PLL is used, the resulting 2× VCO output should be a minimum of 20 MHz, meaning that the minimum QUICC system frequency should be 10 MHz (i.e., EXTAL × (MF + 1) >= 10 MHz.) Use the clock divider control register (CDVCR) to divide the system clock by more than 1 if fully functional operation at less than 10 MHz is desired.

6.5.3.2 SKEW ELIMINATION. The PLL is capable of eliminating the skew between the external clock entering the chip (EXTAL) and the internal clock phases. The PLL also eliminates the skew between EXTAL and the CLKO2–CLKO1 pins, providing advantages in generating low-skew clocking outputs.

The skew is less than 2 ns. Without the PLL enabled, the clock skew could be much larger. This significant reduction of the clock skew is useful for synchronous clocking of multiple system components. For instance, a 25-MHz QUICC may generate clocks for the MC68040—both the 25-MHz BCLK (EXTAL) and the 50-MHz PCLK (CLKO2) may be obtained from a single 25-MHz system clock input to the QUICC.

6.5.4 Low-Power Divider

The output of the PLL is sent to a low-power divider block. This block generates all other clocks in normal operation, but has the ability to divide the output frequency of the VCO before it generates the SyncCLK, BRGCLK, and general system clock to the rest of the QUICC.

The purpose of the low-power divider block is to allow the user to reduce and restore the operating frequencies of different sections of the QUICC without losing the PLL lock. Using the low-power divider block, the user can still obtain full chip operation, but at a slower frequency. This configuration is called slow-go mode. The selection and speed of the slow-go mode may be changed at any time, with changes occurring immediately.

The low-power divider block is controlled in the CDVCR. The default state of the low-power divider is to divide all clocks by 1. Thus, for a 25-MHz system, the SyncCLK, BRGCLK, and general system clock are each 25 MHz.

If the low-power divider block is not used and the user is concerned that errant software could accidentally write the CDVCR, the user may set a write protection bit in CDVCR to prevent further writes to the register.

6.5.5 QUICC Internal Clock Signals

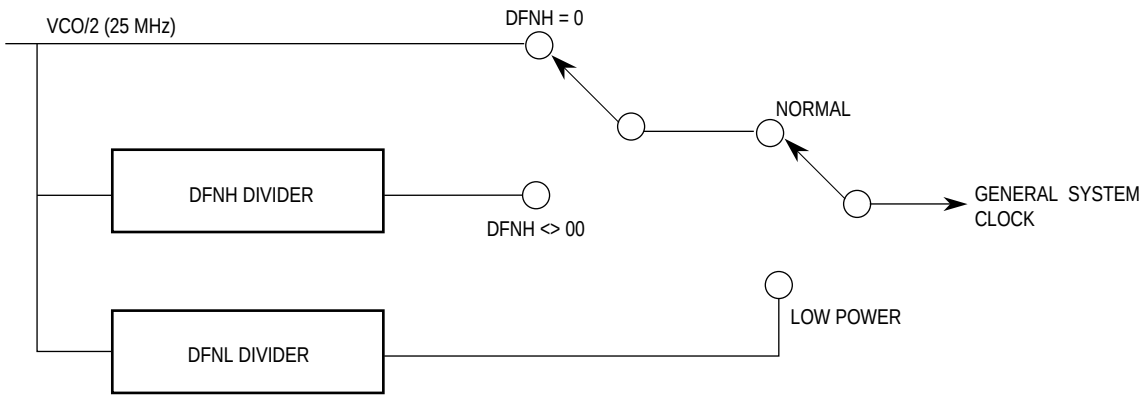
The internal logic of the QUICC uses five internal clock lines: general system clock, BRGCLK, SyncCLK, SIMCLK, and SPCLK. The QUICC also generates two external clock lines

(CLKO1 and CLKO2). The PLL synchronizes these clock signals to each other. These clock signals are discussed in the following paragraphs.

6.5.5.1 SPCLK. SPCLK is supplied to the PIT and SWT sub-modules in the SIM60. SPCLK is always the EXTAL frequency or EXTAL/128, depending on the configuration of the divide-by-128 prescaler. When EXTAL frequency > 10 MHz is selected by configuration pins (MODCK1-MODCK0 = 01), then the PLL is clocked with the EXTAL frequency and SPCLK is EXTAL/128. (i1616.e. When MODCK is 11 --> SPCLK is equal to EXTAL).

6.5.5.2 GENERAL SYSTEM CLOCK. This basic clock is supplied to all other modules and sub-modules on the QUICC, including the CPU32+, the RISC controller, and most other features in the communication processor module (CPM). The general system clock also supplies the SIMCLK to the SIM60 in normal device operation. The general system clock is the same as the CLKO1 frequency, and the CLKO2 signal is 2× the general system clock in normal device operation. The general system clock defaults to VCO/2 = 25 MHz (assuming a 25-MHz system frequency).

The frequency of the general system clock can be changed dynamically with the CDVCR, as shown in Figure 6-7. This configuration is called slow-go mode.



NOTES:
1. NORMAL = (CSRC = 0) OR (INTEN2–INTEN0 < INTERRUPT) OR (RRQEN & RISC NOT IDLE)
2. LOW POWER = $\overline{\text{NORMAL}}$

Figure 6-7. General System Clock Select

The general system clock can be operated at three frequencies. Normal operation is the highest frequency (25 MHz in a 25-MHz system). The general system clock can also be operated at a low frequency and a high frequency. The definition of low is made in the DFNL value in CDVCR; the definition of high is made in the DFNH value in CDVCR.

The frequency of the general system clock can be changed dynamically by software. The user may simply cause the general system clock to switch to its low frequency. However, in some applications, there is a need for high frequency during certain periods. An example is in interrupt routines, etc., that need more performance than the low frequency operation, but must consume less power than in normal operation. The SIM60 allows a method to automatically switch between low and high frequency operation.

The general system clock can switch automatically from low to high frequency whenever one of the following conditions exists:

- The level of the pending or current interrupt is higher than the INTEN bits in CDVCR.
- The CPM RISC controller has a pending request or is currently executing a routine (i.e., it is not idle). This option is maskable by the RRQEN bit in CDVCR.

When neither of these conditions exists, the general system clock automatically switches back to the low frequency.

When the general system clock is divided, its duty cycle is changed. One phase remains the same (e.g., 20 ns @ 25 MHz); the other becomes longer. Note that the CLKO1 and CLKO2 pins no longer have a 50% duty cycle when the general system clock is divided (see Figure 6-8).

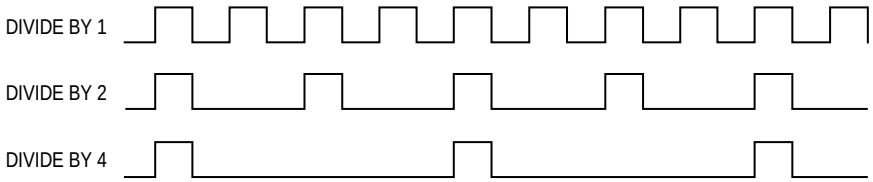


Figure 6-8. Divided Clocks

6.5.5.3 BRGCLK. The BRGCLK is used by the five CPM baud rate generators. There are four SCC/SCM baud rate generators and one SPI baud rate generator. BRGCLK defaults to VCO/2 = 25 MHz (assuming a 25-MHz system frequency). The purpose of BRGCLK is to allow the five baud rate generators to continue to operate at a fixed frequency, even when the rest of the QUICC is operating at a reduced frequency (i.e., the general system clock is divided). See 7.9 Baud Rate Generators (BRGs) for more information on how to save power using the BRGCLK.

NOTES

During early board prototyping, the user should leave BRGCLK at its standard frequency (e.g., 25 MHz) for the sake of simplicity.

Within the four SCC/SMC baud rate generators, the user should not use a baud rate generator divider equal to 1, unless the BRGCLK is at the maximum frequency.

6.5.5.4 SYNCCLK. The SyncCLK is used by the serial synchronization circuitry in the serial ports of the CPM, including the SI, SCCs, and SMCs. The SyncCLK performs the function of synchronizing externally generated clocks before they are used internally. SyncCLK defaults to VCO/2 = 25 MHz (assuming a 25-MHz system frequency).

The purpose of SyncCLK is to allow the SI, SCCs, and SMCs to continue to operate at a fixed frequency, even when the rest of the QUICC is operating at a reduced frequency. Thus, SyncCLK allows the user to maintain the serial synchronization circuitry at the desired rate, while lowering the general system clock to the lowest possible rate. However, the SyncCLK frequency must always be at least as high as the general system clock frequency.

The SyncCLK must always be at least 2× the desired serial clock rate, and at least 2.5× the desired serial clock rate if the time slot assigner (TSA) in the SI is used. See 7.8 Serial Interface with Time Slot Assigner for more information on how to select an appropriate frequency for the SyncCLK.

NOTE

Since SyncCLK does not clock very much logic on the QUICC, SyncCLK is normally left at its full frequency (25 MHz). However, to temporarily lower the value of SyncCLK during an application to save more power, SyncCLK must remain at its highest frequency (e.g., 25 MHz) until the general system clock is reduced. Only then can SyncCLK be lowered, and it must never be lowered to a frequency less than the general system clock frequency.

6.5.5.5 SIMCLK. SIMCLK is supplied to the SIM60 module. SIMCLK defaults to VCO/2 = 25 MHz (assuming a 25-MHz system frequency). The SIMCLK is the same as the general system clock when slow-go mode is programmed in the CDVCR, but can operate differently from the general system clock when the LPSTOP instruction is executed. The SIMCLK is controlled in the PLLCR.

During the LPSTOP instruction, the PLL can be left enabled or can be disabled to conserve power. This option is determined by the STSIM bit in PLLCR. If the PLL is disabled, the SIMCLK is either the EXTAL/2 or the EXTAL/128/2 frequency, depending on the divide-by-128 option.

NOTE

The SIMCLK is always the same frequency as CLKO1.

6.5.5.6 CLKO1. CLKO1 is the same as the general system clock frequency. CLKO1 defaults to VCO/2 = 25 MHz (assuming a 25-MHz system frequency). CLKO1 can drive full strength, 2/3 strength, 1/3 strength, or be disabled. This option is controlled in the CLKOCR. Disabling or decreasing the strength of CLKO1 can reduce power consumption, noise, and electromagnetic interference on the printed circuit board.

During the LPSTOP instruction, the PLL can be left enabled or can be disabled to conserve power. This option is determined by the STSIM bit in PLLCR. If the PLL is disabled, CLKO1 is either the EXTAL/2 or EXTAL/128/2 frequency, depending on the divide-by-128 option.

NOTE

CLKO1 is always the same frequency as the SIMCLK.

6.5.5.7 CLKO2. CLKO2 is 2× general system clock frequency in normal operation. The CLKO2 VCO normally equals 50 MHz (assuming a 25-MHz system frequency). CLKO2 can drive full strength, 2/3 strength, 1/3 strength, or be disabled. This option is controlled in the CLKOCR. Disabling or decreasing the strength of CLKO2 can reduce power consumption, noise, and electromagnetic interference on the printed circuit board.

CLKO2 is always 2× CLKO1 except when the PLL is acquiring lock. When the PLL is acquiring lock, the CLKO2 signal is the EXTAL or EXTAL/128 frequency, as determined by the divide-by-128 option.

For more information see 6.9.3.9 CLKO Control Register (CLKOCR).

6.5.6 PLL Power Pins

The following pins are dedicated to the PLL operation.

6.5.6.1 VCCSYN. This pin is the V_{CC} dedicated to the analog PLL circuits. The voltage should be well regulated, and the pin should be provided with an extremely low-impedance path to the V_{CC} power rail. VCCSYN should be bypassed to GNDSYN by a 0.1-µF capacitor located as close as possible to the chip package.

6.5.6.2 GNDSYN. This pin is the GND dedicated to the analog PLL circuits. The pin should be provided with an extremely low-impedance path to ground. GNDSYN should be bypassed to VCCSYN by a 0.1-µF capacitor located as close as possible to the chip package. The user should also bypass GNDSYN to VCCSYN with a 0.01-µF capacitor as close as possible to the chip package.

6.5.6.3 XFC. This pin connects to the off-chip capacitor for the PLL filter. One terminal of the capacitor is connected to XFC; the other terminal is connected to VCCSYN.

6.5.7 CLKO Power Pins

The following pins are dedicated to the CLKO operation.

6.5.7.1 VCCCLK. This pin is the V_{CC} for the CLKO1 and CLKO2 output pins. The voltage should be well regulated and the pin should be provided with an extremely low-impedance path to the V_{CC} power rail. VCCCLK should be bypassed to GNDCLK by a 0.1-µF capacitor located as close as possible to the chip package.

6.5.7.2 GNDCLK. This pin is the GND for the CLKO1 and CLKO2 output pins. The pin should be provided with an extremely low-impedance path to ground. GNDCLK should be bypassed to VCCCLK by a 0.1-µF capacitor located as close as possible to the chip package.

6.5.8 Configuration Pins (MODCK1–MODCK0)

MODCK1–MODCK0 specifies whether the PLL is enabled and what the initial VCO frequency is after a hardware reset. During the assertion of $\overline{\text{RESET}}$, the value of the MODCK1–MODCK0 input pins causes the PLLEN bit and the MF bits of the PLLCR to be appropriately written. MODCK1–MODCK0 also determines if the oscillator's prescaler is used. After $\overline{\text{RESET}}$ is negated, the MODCK1–MODCK0 pins are ignored. Table 6-1 lists the default values of the PLL. These pins have an internal pullup during a hardware reset.

Table 6-1. Default Operation Mode of the PLL

MODCK 1–0	PLL	Prescaled by 128	Multi. Factor (MF + 1)	EXTAL Freq. (Examples)	CLKIN to the PLL	Initial Freq. (VCO/2)
00	Disabled	Reserved	Reserved	Reserved	Reserved	Reserved
01	Enabled	No	1	>10 MHz	=EXTAL	=EXTAL
10	Enabled	Yes	401	4.192 MHz	32.75 kHz	13.14 MHz
11	Enabled	No	401	32.768 kHz	32.768 kHz	13.14 MHz

NOTE: If the PLL is enabled and the multiplication factor is less than or equal to 4, then CLKO2–CLKO1 is synchronized to EXTAL.

6.6 BREAKPOINT LOGIC

The breakpoint logic provides an internal breakpoint address register (BKAR) and a breakpoint control register (BKCR) that allow hardware breakpoints in a QUICC system. This function is especially useful during in-field debugging activity when it is difficult to connect an in-circuit emulator or logic analyzer to the target board. The use of the background mode of the CPU32+, in combination with the breakpoint logic, provides a convenient and powerful debugging capability.

NOTE

Emulator manufacturers use the QUICC breakpoint logic in their QUICC emulator designs. Customers using emulators should leave the breakpoint logic available for use by the emulator manufacturer, and should not configure the breakpoint logic in their application programs.

When a breakpoint match occurs, the $\overline{\text{BKPT}}$ line is asserted. This can cause a BKPT exception to the CPU32+, and will set a status bit in the IDMA or SDMA that can be used to generate a maskable interrupt. The maskable interrupt may or may not terminate IDMA or SDMA activity, depending on the bus arbitration priority of the IDMA or SDMA as compared to the interrupt level asserted.

NOTE

When the QUICC is configured for a 32-bit bus, the CPU32+ can fetch two instructions simultaneously. Since there is only one $\overline{\text{BKPT}}$ pin, the user cannot break on each instruction, but rather must break on both, causing the BKPT exception to be taken after the first instruction and before the second. The internal breakpoint logic, however, can assert a breakpoint for either instruction individually.

The breakpoint logic allows a great deal of flexibility in what constitutes a breakpoint match. If more than one hardware breakpoint is required, then additional breakpoints may be generated externally in hardware and assert the $\overline{\text{BKPT}}$ pin.

32-Bit Address Decode

The BKAR allows a full 32-bit address to be loaded. This address is qualified in various ways using the BKCR. If no address is desired, the V-bit in the BKCR may be cleared.

Address Space Checking

The BKCR allows the user to check many combinations of function codes before causing a breakpoint match. Nine bits in the BKCR allow such possibilities as excluding the IDMA and SDMA function codes and the user and supervisor programs, but including user and supervisor data.

Read/Write Checking

The breakpoint logic can cause a breakpoint match for read accesses only, write accesses only, or both read and write accesses.

8, 16, 24 and 32 Bit Sizes

The breakpoint logic can cause a breakpoint match for accesses to the specified address with a size of byte, word, three byte, long word, or any size.

Variable Block Sizes

If desired, the breakpoint match can occur in a region larger than just one address. Block sizes may be defined to be the block of memory in which the address resides. The blocks sizes may be 2K, 8K, or 32K bytes.

Additionally, the breakpoint match can be defined to occur at every address except the address or address block that is specified. This feature allows the user to single step all his program code. The breakpoint logic is then used to mask off the user's monitor/debugger. The monitor/debugger thus resides within the programmable block specified by the breakpoint logic.

External Masters

The breakpoint logic can also work with external masters, such as an external MC68040, MC68030, or QUICC. The $\overline{\text{BKPT}}$ pin is asserted when a match is detected.

In the case of an external MC68040, the user may have to set the TSS40 bit in the GMR to allow enough setup time for the address comparison logic. Also, in the case of an external MC68040, the comparison is only made on the first accesses of an MC68040 burst access (i.e., the address comparison of the breakpoint logic is performed only when the MC68040 $\overline{\text{TS}}$ pin is asserted).

6.7 EXTERNAL BUS INTERFACE CONTROL

This subsection describes the method by which the EBI is configured, which includes the data bus size (either 32 or 16 bits), port D, and port E. Refer to Section 4 Bus Operation for more information about the EBI.

NOTE

All accesses to the QUICC internal RAM and registers (including MBAR) by an external master are asynchronous to the QUICC clock. Read and write accesses are with three wait states, and DSACK is asserted by the QUICC assuming three-wait-state accesses.

6.7.1 Initial Configuration

The QUICC has three configuration (CONFIG) pins that are sampled during system (or power-up) reset to select the initial size of the global chip select and whether the QUICC is in the normal (CPU32+ enabled) mode or the slave (CPU32+ disabled) mode (see Table 6-2).

See 6.10 Memory Controller for a description of the global chip select and 6.8 Slave (Disable CPU32+) Mode for a description of slave mode. In normal mode, the global chip select can initially assume the boot ROM port size to be either 8, 16, or 32 bits. In the slave mode, the global chip select can be enabled with 8, 16, or 32 bits, or the global chip select can be disabled. The global chip select would normally be disabled if another QUICC or processor was providing the boot ROM chip select function.

Table 6-2. QUICC Initial Configuration

Configuration Pins			Result
CONFIG2 /FREEZE	CONFIG1 /BCLRO	CONFIG0 /RMC	
0	0	0	Slave mode; global chip select 8-bit size; MBAR at $003FF00.
0	0	1	Slave mode; global chip select 32-bit size; MBAR at $003FF00; not MC68040 companion mode; BR output, BG input.
0	1	0	Slave mode; global chip select 16-bit size; MBAR at $003FF00.
0	1	1	MC68040 companion mode; global chip select 32-bit size; MBAR at $003FF00; BR input, BG output.
1	0	0	CPU32+ enabled; global chip select 32-bit size; MBAR at $003FF00.
1	0	1	CPU32+ enabled; global chip select 16-bit size; MBAR at $003FF00.
1	1	0	Slave mode; global chip select disabled; MBAR at $003FF04.
1	1	1	CPU32+ enabled; global chip select 8-bit size; MBAR at $003FF00.

6.7.2 Port D

If the user configures a 16-bit data bus by driving a zero voltage on the PRTY3 pin during system reset, then the D0–D15 pins are not used as a data bus, but are referred to as port D. At this time, port D is not available for general-purpose I/O or any other alternate function on the QUICC. In the future, these pins may be defined to have an additional function in 16-bit data bus mode.

NOTES

The 16-bit data bus is available on the D16–D31 pins. Dynamic bus sizing for 8- and 16-bit ports is possible with a 16-bit data bus.

PRTY3 has a small internal pullup to pull a floating PRTY3 signal high. Thus, the default condition of the QUICC provides a full 32-bit data bus, with 8-, 16-, and 32-bit dynamic bus sizing possible.

6.7.3 Port E

Port E pins can be independently programmed to select a number of system bus signal alternatives, including $\overline{CAS}$ lines, $\overline{WE}$ lines, $\overline{OE}$ lines, $\overline{IACK}$ lines, etc. The port E pin assignment register controls the function of the port E pins. See 6.9.4 Port E Pin Assignment Register (PEPAR).

6.8 SLAVE (DISABLE CPU32+) MODE

In this mode, the CPU32+ core on the QUICC is disabled, and the QUICC functions as an intelligent peripheral. Slave mode is enabled during system (or power-up) reset by the configuration of the CONFIG pins. In slave mode, the IDMAs and SDMAs on a QUICC can still obtain ownership of the system bus, even through the CPU32+ core is disabled. The slave mode has several common uses:

1. A multiple QUICC system. One QUICC in the system works normally with its CPU32+ enabled. It is called the system master. The rest of the QUICCs are used in slave mode as peripherals, with their CPUs disabled. The slaves would have their CONFIG pins configured to 110.
2. MC68040 companion mode. The QUICC operates solely as a peripheral to the MC68EC040 processor (or other M68040 family member). In this case, the QUICC provides a two-chip MC68EC040 system solution. One benefit of this configuration is that the QUICC memory controller can provide DRAM control for the MC68EC040 that includes MC68EC040 bursting support. In an MC68EC040+QUICC system, the QUICC’s CONFIG pins would normally be set to 011 for a 32-bit boot ROM.
3. MC68040 companion mode with multiple QUICCs. In this case, multiple QUICCs can be slaves to a single MC68EC040. The user then chooses one of the QUICCs to provide the DRAM control for the MC68EC040 as well as the other QUICCs. In this case, the MC68EC040 access to the DRAM is not slowed down by the relatively slower QUICC accesses. The first QUICC in the system would have its CONFIG pins set to 011, and the other QUICCs added to that system would have their CONFIG pins set to 110.
4. QUICC is a slave to the MC68030. The QUICC operates as a peripheral to the MC68EC030 processor (or other MC68030 family member). The QUICC's standard slave mode is used since its bus is an MC68030-type bus. The QUICC does not support MC68030 burst accesses. In an MC68EC030+QUICC system, the QUICC's CONFIG pins could be set to 000, 001, or 010, depending on the boot ROM size.

NOTES

When used in slave mode, the QUICC must be configured with a 32-bit data bus.

Even without the use of the slave mode, another processor can be granted access to the QUICC's on-chip peripherals by requesting the bus with the $\overline{BR}$ pin.

6.8.1 MBAR in a Multiple QUICC System

The module base address register (MBAR) is used to configure the location of the QUICC's block of on-chip RAM and registers. In a multiple QUICC system, a technique must be provided to allow multiple MBARs on multiple QUICCs to be programmed with unique values. The QUICC has several provisions to support this.

First, any QUICC that is configured into slave mode with its global chip select disabled (CONFIG pins = 110) automatically has its MBAR location changed from $0003FF00 to $0003FF04. Second, the MBAR, newly located at address $0003FF04, can only be enabled for access after a keyed write operation is performed (see Figure 6-9). The keyed write allows the user to program the MBARs of multiple QUICC slaves without adding any external glue logic.

NOTES

If the QUICC is configured into slave mode with its global chip select enabled, the MBAR location does not change, and the keyed write is not required. Thus, a single QUICC configured as a slave to an MC68EC040 or MC68EC030 does not require a keyed write for its MBAR.

If there are N QUICCs sharing a bus, N–1 QUICCs would normally have their CONFIG pins configured as 110.

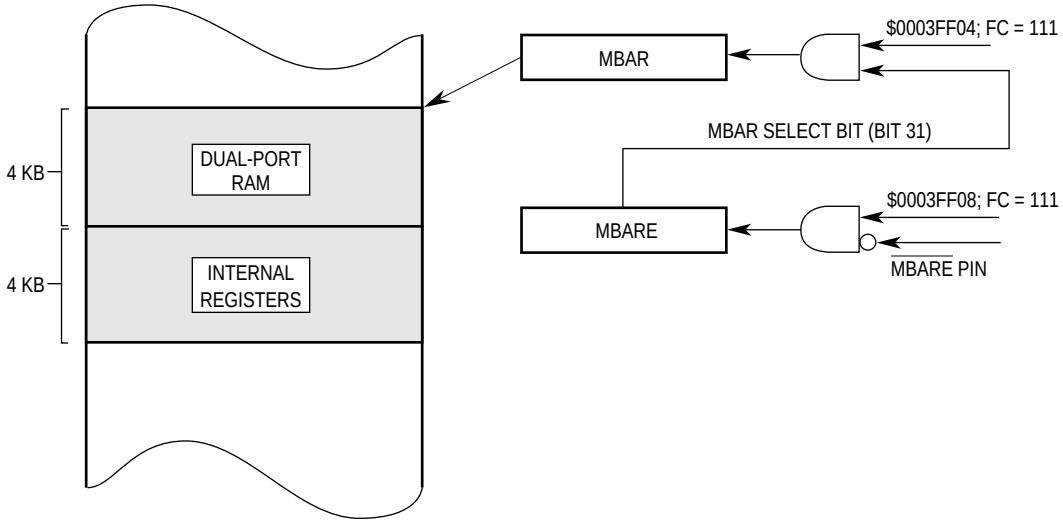


Figure 6-9. MBAR Access to a Multiple QUICC Slave System

The keyed write uses the MBAR enable (MBARE) register at address $0003FF08 and the $\overline{\text{MBARE}}$ pin. Both the newly located MBAR and the MBARE are located in the CPU address space FC = 111.

With multiple QUICCs configured in slave mode, the following keyed write is used to enable the MBAR programming: the user writes the MBAR select bit of MBARE with a 1 while the $\overline{\text{MBARE}}$ pin is a logic zero. Once this is accomplished, the MBAR may be written at its new location (using the standard MBAR writing techniques). Once MBAR is written (in particular, the low-order byte of MBAR), then the MBAR select bit in MBARE is cleared, and further accesses to MBAR are impossible until the keyed write technique is used again. There is no time limit imposed between the keyed write and the MBAR write; however, once the keyed write for a particular QUICC slave has occurred, the MBAR of that slave should be written before performing another keyed write to another QUICC slave.

The keyed write may be performed gluelessly to multiple QUICC slaves in the following way. Connect (in hardware) the $\overline{\text{MBARE}}$ pin of QUICC slave #1 to bit zero of the data bus(D0). Connect the $\overline{\text{MBARE}}$ pin of QUICC slave #2 to D1, etc. Then perform the following operations in software:

1. Write the MBARE of QUICC slave #1 at $0003FF08 with value $FFFFFFFE. This sets the MBAR select bit (bit 31) and places a low voltage on only the $\overline{\text{MBARE}}$ pin of QUICC slave #1.
2. Now the MBAR of QUICC slave #1 can be accessed at $0003FF04 and written using normal MBAR writing procedures.
3. Write the MBARE of QUICC slave #2 with the value $FFFFFFFD.
4. Now the MBAR of QUICC slave #2 can be written.

This technique will work for up to 31 QUICC slaves in the system, using no glue or parallel I/O pins.

6.8.2 Global Chip Select ($\overline{\text{CS0}}$) in Slave Mode

When the QUICC is in slave mode, the user may choose whether to enable the global chip-select operation of $\overline{\text{CS0}}$. (The global chip select is used for boot ROMs and is described in 6.10 Memory Controller.) The global chip select can be either enabled or disabled by the configuration on the CONFIG pins during power-up reset and system reset ($\overline{\text{RESETH}}$ asserted). When the global chip select function is disabled, $\overline{\text{CS0}}$ can still be enabled later and used as a normal chip select, if desired.

6.8.3 Bus Clear in Slave Mode

The bus clear out ($\overline{\text{BCLRO}}$) pin can be selected to signify to the external logic that the DRAM refresh controller, IDMA channels, or SDMA channels are requesting the bus. However, in slave mode, the $\overline{\text{BCLRO}}$ pin may become the $\overline{\text{RAS2DD}}$ double drive pin, and a new pin called bus clear in ($\overline{\text{BCLRI}}$) is defined (at another location in the pinout). $\overline{\text{BCLRI}}$ indicates to that QUICC that a request is being made for the QUICC to release the system bus. The EBI will then clear all internal bus masters with an arbitration ID smaller than the programmed value of the bus clear in ID (BCLRIID) in the MCR.

6.8.4 Interrupts in Slave Mode

The SIM60 interrupt controller continues to function in slave mode and can present interrupts from the PIT, SWT, and external interrupt sources on the IRQx pins to the processor. The highest priority request is output as an encoded value on the IOUTx pins or is output on a single RQOUT pin.

The CPM will also generate interrupts in slave mode. The CPM always generates unique vectors for its sub-modules in slave mode. The CPM also offers a number of individual interrupt request inputs (port C pins) that may be used in slave mode.

When the SIM60 is in slave mode, the PIT and SWT must also generate interrupt vectors. For the IRQx pins, no vector is output in slave mode; rather, the AVECO pin is asserted if the corresponding bit in the AVR is set.

One important restriction must be adhered to in slave mode. The user should not utilize an IRQx pin that is on the same level as the CPM, PIT, or SWT. The level of the CPM is programmed in the CICR. The level of the PIT is programmed in the PICR. The level of the SWT is 7 if it generates interrupts. Note that CPM port C pins operate similarly to the IRQx pins and may still be used.

6.8.5 Pin Differences in Slave Mode

A number of signals change functionality in slave mode. See Section 2 Signal Descriptions for a full listing. A partial list of functionality changes is as follows:

1. BR will be an output from the QUICC (refresh cycles, IDMA, and SDMA) to the external bus.

   NOTE

   BR is still an input in MC68040 companion mode.

2. BG will be an input to the QUICC (refresh cycles, IDMA, and SDMA) from the external bus.

   NOTE

   BG is still an output in MC68040 companion mode.

3. BGACK will be asserted during the QUICC external bus cycles.
4. The QUICC interrupt controller will output its interrupt requests on the IOUT2–IOUT0 pins, which normally would be sent to the CPU32+ core, and will reflect internally the interrupt acknowledge cycle. The three IOUTx pins reflect the seven interrupt request levels. The IOUTx pins can be output on the IRQ1, IRQ4, and IRQ6 pins or on the parity PRTY2–PRTY0 pins as programmed in the port E pin assignment register. Additionally, the QUICC interrupt controller can output its interrupt requests on one interrupt request pin (RQOUT) instead of three pins.
5. An AVEC output (AVECO) function is supported instead of the AVEC input pin.
6. The breakpoint logic may monitor the external bus instead of the internal bus and assert the BKPTO pin.
7. The three-state (TRIS) pin becomes the transfer start (TS) pin in slave mode. Anytime

the QUICC is in slave mode, assertion of the $\overline{TS}$ pin notifies the QUICC that an external MC68040 cycle is beginning. Although the user typically configures the CONFIGx pins for MC68040 companion mode, this configuration is not required. It is possible for the QUICC to recognize an MC68040 cycle in any of the slave mode variations. (The reason for the MC68040 companion mode configuration of the CONFIGx pins is to allow the bus arbitration pins to have their directions reversed while still in slave mode.)

6.8.6 Other Functionality in Slave Mode

Although the slave mode does enable a number of different pins on the system bus and causes functional activities such as bus arbitration and interrupt handling to occur differently, if a feature is not cited as changing its behavior in slave mode (i.e., 98% of the features on QUICC), then it is not impacted by slave mode and continues to operate normally.

6.9 PROGRAMMER'S MODEL

The SIM60 contains a number of registers, described in the following paragraphs. Their locations and initial values may be found in Section 3 QUICC Memory Map.

6.9.1 Module Base Address Register (MBAR)

The MBAR is a 32-bit, memory-mapped, read-write register consisting of the high address bits. Upon a total system reset, its value may be read as $0. The address of this register is fixed at $03FF00 in CPU space (except in slave mode where it is located at $03FF04). See 6.8 Slave (Disable CPU32+) Mode for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA15	BA14	BA13	0	0	0	AS8	AS7	AS6	AS5	AS4	AS3	AS2	AS1	AS0	V
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPU SPACE ONLY

BA31–BA13—Base Address

The base address field is the upper 19 bits of the MBAR, providing for block starting locations in increments of 8 Kbytes.

AS8–AS0—Address Space

The address space field allows particular address spaces to be masked, placing the 8K module block into a particular address space(s). If an address space is masked, an access to the register block location in that address space becomes an external access. The module block is not accessed. The address space bits for non-040 type master are:

1. AS8—mask DMA space address space (FC3–FC0=1xxx)
2. AS7—mask CPU space address space (FC3–FC0=0111)
3. AS6—mask supervisor program address space (FC3–FC0=0110)

4. AS5—mask supervisor data address space (FC3–FC0=0101)
5. AS4—mask Motorola reserved address space (FC3–FC0=0100)
6. AS3—mask user reserved address space (FC3–FC0=0011)
7. AS2—mask user program address space (FC3–FC0=0010)
8. AS1—mask user data address space (FC3–FC0=0001)
9. AS0—mask Motorola reserved address space (FC3–FC0=0000)

The address space bits for 040 type MPU are:

1. AS8—no relevance for 040 cycles
2. AS7—acknowledge access (TT1-TT0=11)
3. AS6—supervisor code access (TT1-TT0=00, TM2-TM0=110)
4. AS5—supervisor data access (TT1-TT0=00, TM2-TM0=101)
5. AS4—MMU table search code access (TT1-TT0=00, TM2-TM0=100)
6. AS3—MMU table search data access (TT1-TT0=00, TM2-TM0=011)
7. AS2—user code access (TT1-TT0=00, TM2-TM0=010)
8. AS1—user data access (TT1-TT0=00, TM2-TM0=001)
9. AS0—data cache push access (TT1-TT0=00, TM2-TM0=000)

NOTE

The user should mask off AS7, AS4, AS3 and AS1 to prevent unwanted access to the QUICC internal dual port RAM (DPR). For example, AS7 should be masked out so that the IACK cycle will not cause an access to the DPR.

For each address space bit:

1 = Mask this address space from the internal module selection. The bus cycle goes external.
0 = Decode for the internal module block.

V—Valid

This bit indicates when the contents of the MBAR are valid. The base address value is not used; therefore, all internal module registers are not accessible until the V-bit is set.

0 = Contents not valid.
1 = Contents valid.

NOTE

When working in the CPU enable mode, an access to this register does not affect external space since the cycle is not run externally.

MBAR can be read using the following code. Register D0 will contain the value of MBAR.

```
MOVE        #7,D0           load D0 with the CPU space function code
MOVEC       D0,SFC          load SFC to indicate CPU space
MOVEC       D0,DFC          load DFC to indicate CPU space
LEA         $3FF00,A0       load A0 with the address of MBAR
MOVES.L     (A0),D0         load D0 with the contents of MBAR
```

MBAR can be written to using the following code. Address $0003FF00 in CPU space (MBAR) will be loaded with the value $FFFF F001. This will set the base address of the internal registers to $FFFFF.

```
MOVE        #7,D0           load D0 with the CPU space function code
MOVEC       D0,SFC          load SFC to indicate CPU space
MOVEC       D0,DFC          load DFC to indicate CPU space
LEA         $3FF00,A0       load A0 with the address of MBAR
MOVE.L      #$FFFFF001,D0   load D0 with the value to be written into MBAR
MOVES.L     D0,(A0)         write the value contained in D0 into MBAR
```

6.9.2 Module Base Address Register Enable (MBARE)

The MBARE is a 32-bit, memory-mapped, read-write register. Upon a total system reset, its value may be read as $0. The address of this register is fixed at $03FF08 in CPU space. It is used to enable the MBAR to be programmed when multiple QUICCs are in slave mode. (See 6.8.1 MBAR in a Multiple QUICC System for details.)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MBS	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPU SPACE ONLY

MBS—MBAR Select

0 = No operation.

1 = The MBAR is now ready to be programmed on this slave QUICC device if the $\overline{\text{MBARE}}$ pin was low during the write to this bit.

6.9.3 System Configuration and Protection Registers

The following paragraphs provide descriptions of the system configuration and protection registers.

6.9.3.1 MODULE CONFIGURATION REGISTER (MCR). The MCR, which controls the SIM60 configuration, can be read or written at any time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR040ID2–BR040ID0			—	—	—	—	—	—	—	—	—	—	—	—	BSTM
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASTM	FRZ1–FRZ0		BCLROID2–BCLROID0			SHEN1–SHEN0		SUPV	BCLRISM2–BCLRISM0 or BCLRIID2–BCLRIID0			IARB3–IARB0			
0	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1

BR040ID2–BR040ID0—Bus Request MC68040 Arbitration ID

These bits contain the arbitration priority level for the MC68040 $\overline{BR}$ signal when the QUICC is in MC68040 companion mode; otherwise, this value is ignored. The MC68040 $\overline{BR}$ signal in companion mode) is reflected on the IMB with the bus arbitration level corresponding to these bits. This method gives the user a choice of where to place the arbitration level of the MC68040 (and other external masters in this system) relative to the IDMA, SDMA, or DRAM refresh cycles generated by the QIUCC.

NOTE

In a typical configuration, the user would program this value to a 3 to give the MC68040 priority over the IDMAs, but not over the SDMAs and the DRAM refresh cycle. If the SDMAs, however, are not of extremely high priority, the user may choose this value to be 5. User should never program this field to be 7.

Bits 28–17—Reserved

BSTM—Bus Synchronous Timing Mode

This bit determines whether the EBI will synchronize the $\overline{AS}$ and $\overline{DS}$ bus signals used for an external master's access into the QUICC peripherals and for $\overline{CS}$ and $\overline{RAS}$ generation by the QUICC. The synchronization will add a one-clock delay to the $\overline{RAS}$/$\overline{CS}$ assertion for an external master. The MC68EC040 signals must always be synchronized to the QUICC clock, regardless of the setting of this bit. See 6.10 Memory Controller for recommendations on the setting of BSTM in certain situations.

0 = Asynchronous timing on the bus signals may be used. The bus signals are synchronized internally by the QUICC and do not have to meet any timings relative to the system clock.

1 = Synchronous timing on the bus signals must be used. The bus control signals will not be synchronized internally and therefore must meet the system clock setup and hold timings.

NOTE

$\overline{BCLRI}$, Address, Data, $\overline{DSACK}$, $\overline{BERR}$, $\overline{HALT}$, $\overline{RESETH}$, and $\overline{RESETS}$ are always asynchronous.

ASTM—Arbitration Synchronous Timing Mode

This bit determines whether the EBI will synchronize the arbitration signals: $\overline{BR}$, $\overline{BG}$, and $\overline{BGACK}$. The synchronization will add a one-clock delay to the external bus arbitration.

0 = Asynchronous timing on the arbitration signals may be used. The arbitration signals will be synchronized internally by the QUICC and do not have to meet any timings relative to the system clock.

1 = Synchronous timing on the arbitration signals must be used. The arbitration control signals will not be synchronized internally and therefore must meet the system clock setup and hold timings.

FRZ1—Freeze SWT and PIT Enable

0 = When FREEZE is asserted, the SWT and the PIT counters continue to run. See 6.3.3 Freeze Support for more information.

1 = When FREEZE is asserted, the SWT and the PIT counters are disabled, preventing interrupts from occurring during software debugging.

FRZ0—Freeze Bus Monitor Enable

0 = When FREEZE is asserted, the bus monitor continues to operate as programmed.

1 = When FREEZE is asserted, both the internal and external bus monitors are disabled.

BCLROID2–BCLROID0—Bus Clear Out Arbitration ID

These bits contain the arbitration priority level for the assertion of the $\overline{BCLRO}$ signal. When internal masters (IDMA, SDMA, or DRAM refresh cycle) request the bus and the arbitration level on the IMB is greater than the bus clear out arbitration ID, the $\overline{BCLRO}$ signal will be asserted until the arbitration level is less than or equal to the bus clear out arbitration ID. $\overline{BCLRO}$ can be used to clear an external master from the external bus when a refresh cycle is pending. It may also be used to clear an external master from the bus when an SDMA or IDMA channel requests the external bus.

NOTE

Program this value to 3 in a normal system to allow the SDMA and DRAM refresh controller to clear other bus masters off the external bus.

SHEN1–SHEN0—Show Cycle Enable

These two control bits determine what the EBI does with the external bus during internal transfer operations (see Table 6-3). A show cycle allows internal transfers to be externally monitored. The address, data, and control signals (except for $\overline{AS}$) are driven externally. $\overline{DS}$ is used to signal address strobe timing for show cycles. Data is valid on the next falling clock edge after $\overline{DS}$ is negated. However, data is not driven externally and $\overline{AS}$ and $\overline{DS}$ are not asserted externally for internal accesses unless show cycles are enabled.

If external bus arbitration is disabled, the EBI will not recognize an external bus request until arbitration is enabled again. When SHEN1 is set, an external bus request causes an internal master to stop its current cycle and relinquish the internal bus. The internal master resumes running cycles on the bus after $\overline{BR}$ and $\overline{BGACK}$ are negated. To prevent bus

conflicts, external peripherals must not attempt to initiate cycles during show cycles with arbitration disabled.

Table 6-3. Show Cycle Control Bits

SHEN1	SHEN0	Action
0	0	Normal operation. Split buses mode. Show cycles is disabled and external arbitration is enabled.
0	1	Show cycles enabled. External arbitration is disabled and $\overline{BG}$ is never asserted.
1	x	Show cycles enabled. External arbitration is enabled and internal activity is halted when $\overline{BG}$ is asserted by the QUICC.

NOTE

During normal operation, $\overline{BERR}$ and $\overline{DSACKx}$ for internal cycles will not appear as an external cycle in master mode.

For fast termination cycles, $\overline{DSACKx}$ is never asserted externally regardless of the show cycle bit settings.

In slave mode, these bits default to 00, and writes by the user have no effect on operation.

In case 00 (show cycles disabled), if the external bus is available when an internal-to-internal access occurs, the address and function code pins will reflect the internal access.

Case 01 may be used as a debugging aid to eliminate the external bus master as a possible cause of the problem or to prevent interference in a user debug session.

Although case 00 is recommended for normal operation, case 1x may be used during initial development for visibility on the internal bus, at the expense of performance. Moving from 1x to 00 increases performance for two reasons: 1) both the internal and external buses may be used simultaneously and 2) the external bus master will obtain the $\overline{BG}$ signal assertion more quickly after asserting $\overline{BR}$.

SUPV—Supervisor/User Data Space

The SUPV bit defines the SIM60 global registers as either supervisor data space or user (unrestricted) data space. It is a don't care on the SIM60 and is reserved for future expansion.

0 = The SIM60 registers defined as supervisor/user data are unrestricted (FC2 is a don't care).

1 = The SIM60 registers defined as supervisor/user are restricted to supervisor data access (FC3–FC0 = $5). Any attempted user space write is ignored and returns $\overline{BERR}$.

NOTE

This bit is “don’t care” in the SIM60 since no user space registers exist. It is reserved for future expansions.

BCLRISM2–BCLRISM0—Bus Clear Interrupt Service Mask (Normal Mode Only)

These bits contain the interrupt service mask. When the interrupt service level on the IMB is greater than the interrupt service mask, the $\overline{\text{BCLRO}}$ signal will be asserted until the interrupt level is less than or equal to the interrupt service mask. This feature can be used to clear an external master from the external bus to reduce the interrupt latency for a certain interrupt level and above.

NOTES

This value should be programmed to 7 in a typical system unless the user needs to give certain interrupt routines priority over external bus masters.

In slave mode (disable CPU32+), these bits are not used and have a different definition.

BCLRIID2–BCLRIID0—Bus Clear In Arbitration ID (Slave Mode Only)

These bits contain the arbitration priority level for the $\overline{\text{BCLRI}}$ signal. If $\overline{\text{BCLRI}}$ is asserted when the internal master (IDMA, SDMA, or DRAM refresh cycle) is requesting or using the bus, and if the arbitration level on the IMB is lower than the bus clear in arbitration ID bits, the internal master will release the bus until the $\overline{\text{BCLRI}}$ signal is negated. Thus, $\overline{\text{BCLRI}}$ can be used to clear an internal master from the external bus when the bus is needed for a higher priority task.

NOTES

Program the arbitration IDs of the QUICC as follows: SDMA = 4, IDMAx = 2, IDMAy = 0. The DRAM refresh controller is always 6. Thus, the user may choose 3 for this value to give the external master priority over the IDMA channels only.

In the case of the MC68040 companion mode, the $\overline{\text{BG}}$ pin is also negated by the QUICC when an internal master has released the bus.

In normal operation (CPU32+ enabled), these bits are not used and have a different definition.

This field should never be programmed to be 7.

IARB3–IARB0—Interrupt Arbitration

The reset value of IARB is $F, allowing the SIM60 to win interrupt arbitrations during an interrupt acknowledge cycle immediately after reset. The system software should initialize the IARB field to a value from $F (highest priority) to $1 (lowest priority).

NOTE

If, a SIM60 interrupt source shares a level with the CPM, write either $F or $1 to this register. Since the CPM interrupt arbitration ID is always 8, the $F gives the SIM60 source higher priority than the CPM source(s), and a $1 gives the interrupt source lower priority than the CPM source(s). This field should never be programed to 0.

6.9.3.2 AUTOVECTOR REGISTER (AVR). The AVR contains bits that correspond to external interrupt levels that require an autovector response. Setting a bit allows the SIM60 to assert an internal AVEC during the interrupt acknowledge cycle in response to the specified interrupt request level. This register can be read and written at any time.

7	6	5	4	3	2	1	0
AV7	AV6	AV5	AV4	AV3	AV2	AV1	-
RESET:							
0	0	0	0	0	0	0	0

SUPERVISOR ONLY

NOTE

The IARB field in the MCR must contain a value other than $0 for the SIM60 to produce an autovector for external interrupts.

6.9.3.3 RESET STATUS REGISTER (RSR). The RSR contains a bit for each reset source to the SIM60. A set bit indicates the last type of reset that occurred. The RSR is updated by the reset control logic when the reset is complete. After power-up reset, the POW bit and the EXT bit are set. Other bits may be set after different kinds of reset occur. Since this register is only cleared upon a power-up reset, the user should clear this register after every reset so that the cause of the most recent reset may be easily determined.

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. The register may be read at any time. For more information, see Section 4 Bus Operation.

7	6	5	4	3	2	1	0
EXT	POW	SW	DBF	—	LOC	SRST	SRSTP

SUPERVISOR ONLY

EXT—External Total System Reset (Hard Reset)

1 = The last reset was caused by an external signal driving $\overline{\text{RESETH}}$. This will reset all the QUICC's peripherals to the state they had at power-up reset. This reset, which is also referred to as system reset or hardware reset, has the same effect in the system as a power-up reset.

POW—Power-Up Reset

1 = The last reset was caused by the power-up reset circuit.

SW—Software Watchdog Reset

1 = The last reset was caused by the software watchdog circuit.

DBF—Double Bus Fault Monitor Reset

1 = The last reset was caused by the double bus fault monitor.

Bit 3—Reserved

LOC—Loss of Clock Reset

1 = The last reset was caused by a loss of frequency reference to the clock sub-module. This reset can only occur if the RSTEN bit in the clock sub-module is set and the VCO is enabled.

SRST—Soft Reset

1 = The last reset was caused by the CPU32+ executing a RESET instruction. The RESET instruction does not load a reset vector or affect any internal CPU32+ registers or SIM60 configuration registers, but does reset external devices and other internal modules. See Section 3 QUICC Memory Map for a listing of registers affected by the hard reset. This bit is not valid in CPU disable mode.

SRSTP—Soft Reset Pin

1 = The last reset was caused by an external signal driving $\overline{\text{RESETS}}$. See Section 3 QUICC Memory Map for a listing of registers affected by the soft reset.

6.9.3.4 SOFTWARE WATCHDOG INTERRUPT VECTOR REGISTER (SWIV). The SWIV contains the 8-bit vector that is returned by the SIM60 during an interrupt acknowledge cycle in response to an interrupt generated by the SWT. This register can be read or written at any time. This register is set to the uninitialized vector, $0F, at reset.

7	6	5	4	3	2	1	0
SWIV7	SWIV6	SWIV5	SWIV4	SWIV3	SWIV2	SWIV1	SWIV0
RESET:							
0	0	0	0	1	1	1	1

SUPERVISOR ONLY

6.9.3.5 SYSTEM PROTECTION CONTROL REGISTER (SYPCR). The SYPCR controls the system monitors, the prescaler for the SWT, and the bus monitor timing. This register can be read at any time, but can be written only once after system reset.

7	6	5	4	3	2	1	0
SWE	SWRI	SWT1	SWT0	DBFE	BME	BMT1	BMT0
RESET:		MODCK	MODCK				
1	1	1	1	0	0	0	0

SUPERVISOR ONLY

SWE—Software Watchdog Enable

This bit should be cleared by software after a system reset to disable the SWT. See 6.3.1.2.4 Software Watchdog Timer (SWT) for more information.

0 = SWT is disabled.
1 = SWT is enabled. (This is the default value after system reset.)

SWRI—Software Watchdog Reset/Interrupt Select

0 = SWT causes a level 7 interrupt to the CPU32+.
1 = SWT causes a system reset. (This is the default value after system reset.)

NOTE

For more information on reset see 4.7 Reset Operation.

SWT1–SWT0—Software Watchdog Timing

These bits, along with the SWP bit in the PITR, control the divide ratio used to establish the timeout period for the SWT. The default value (11) yields the maximum timeout period. The SWT timeout period is given by the following formula:

$$\frac{1}{(\text{EXTALDIV})/(\text{divide count})}$$

or

$$\frac{\text{divide count}}{\text{EXTALDIV}}$$

The SWT timeout period listed in Table 6-4 gives the formula to derive the SWT timeout for any clock frequency. The timeout periods are listed for various input frequencies. Note that the input frequency to the SWT is called EXTALDIV in the formulas and is the EXTAL frequency divided by 1 or by 128, depending on the MODCK1–MODCK0 pins.

Table 6-4. Deriving SWT Timeout

SWP	SWT1–SWT0	Software Timeout Period	32.768 kHz[1]	16.677 MHz	25 MHz	33.354 MHz
0	00	2^9/Input frequency[2]	15.6 ms	3.9 ms	2.6 ms	1.9 ms
0	01	2^{11}/Input frequency	62.5 ms	15.7 ms	10.5 ms	7.9 ms
0	10	2^{13}/Input frequency	250 ms	62.9 ms	41.9 ms	31.4 ms
0	11	2^{15}/Input frequency	1 s	251.5 ms	167.7 ms	125.7 ms
1	00	2^{18}/Input frequency	8 s	2.0 s	1.3 s	1.0 s
1	01	2^{20}/Input frequency	32 s	8.0 s	5.4 s	4.0 s
1	10	2^{22}/Input frequency	128 s	32.2 s	21.5 s	16.1 s
1	11	2^{24}/Input frequency	512 s	128.8 s	85.9 s	64.4 s

NOTES:
1. Note that a 4.192-MHz oscillator produces the same 32.768 input frequency (the 4.192 MHz is divided by 128 in the oscillator circuit).
2. Programming for this timeout period must be done after the programming of the PLL. See also 6.9.3.10 PLL Control Register (PLLCR).

NOTE

When the SWP and SWT bits are modified to select a software timeout other than the default, the software service sequence ($55 followed by $AA written to the software service register) must be performed before the new timeout period takes effect.

DBFE—Double Bus Fault Monitor Enable

1 = Enable the double bus fault monitor function. (Default)
0 = Disable the double bus fault monitor function.
For more information, see 6.3.1.2.3 Double Bus Fault Monitor and Section 5 CPU32+.

BME—Bus Monitor External Enable

0 = Enable bus monitor function for the external bus cycles.
1 = Disable bus monitor function for the external bus cycles. (Default)
For more information see 6.3.1.2.1 Bus Monitor.

BMT1–BMT0—Bus Monitor Timing

These bits select the timeout period for the bus monitor (see Table 6-5).

Table 6-5. BMT Encoding

BMT1	BMT0	Bus Monitor Timeout Period
0	0	1K System Clocks (CLKO1 Clocks)
0	1	512 System Clocks
1	0	256 System Clocks
1	1	128 System Clocks

6.9.3.6 PERIODIC INTERRUPT CONTROL REGISTER (PICR). The PICR contains the interrupt level and the vector number for the periodic interrupt request. This register can be read or written at any time. Bits 15–11 are unimplemented and always return zero; a write to these bits has no effect.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	PIRQL2	PIRQL1	PIRQL0	PIV7	PIV6	PIV5	PIV4	PIV3	PIV2	PIV1	PIV0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

SUPERVISOR ONLY

PIRQL2–PIRQL0—Periodic Interrupt Request Level

These bits contain the periodic interrupt request level. Table 6-6 lists which interrupt request level is asserted during an interrupt acknowledge cycle when a periodic interrupt is generated. The PIT continues to run when the interrupt is disabled.

NOTE

Use caution with a level 7 interrupt encoding due to the SIM60 interrupt servicing order. See 6.3.1.2 Simultaneous SIM60 Interrupt Sources for the servicing order.

Table 6-6. Periodic Interrupt Request Level Encoding

PIRQL2	PIRQL1	PIRQL0	Interrupt Request Level
0	0	0	PIT Disabled
0	0	1	Interrupt Request Level 1
0	1	0	Interrupt Request Level 2
0	1	1	Interrupt Request Level 3
1	0	0	Interrupt Request Level 4
1	0	1	Interrupt Request Level 5
1	1	0	Interrupt Request Level 6
1	1	1	Interrupt Request Level 7

PIV7–PIV0—Periodic Interrupt Vector

These bits contain the value of the vector generated during an interrupt acknowledge cycle in response to an interrupt from the PIT. When the SIM60 responds to the interrupt acknowledge cycle, the periodic interrupt vector from the PICR is placed on the bus. This vector number is multiplied by 4 to form the vector offset, which is added to the VBR in the CPU32+ to obtain the address of the vector.

6.9.3.7 PERIODIC INTERRUPT TIMER REGISTER (PITR). The PITR contains control for prescaling the SWT and PIT as well as the count value for the PIT. This register can be read or written at any time. Bits 15–10 are not implemented and always return zero when read. A write does not affect these bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	SWP	PTP	PITR7	PITR6	PITR5	PITR4	PITR3	PITR2	PITR1	PITR0
RESET:						MODCK	MODCK								
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

SUPERVISOR ONLY

SWP—Software Watchdog Prescaler Control

This bit controls the SWT clock source as shown in 6.9.3.5 System Protection Control Register (SYPCR). The SWP reset value is the inverse of the MODCK1 pin state on the rising edge of reset.

0 = SWT clock is not prescaled.
1 = SWT clock is prescaled by a value of 512.

PTP—Periodic Timer Prescaler Control

This bit contains the prescaler control for the PIT. The PTP reset value is the inverse of the MODCK1 pin state on the rising edge of reset.

0 = PIT clock is not prescaled.
1 = PIT clock is prescaled by a value of 512.

PITR7–PITR0—Periodic Interrupt Timer Register

These bits contain the remaining bits of the PITR count value for the PIT. A zero value turns off the PIT. These bits may be written at any time to modify the PIT count value.

NOTE

If the PIT is enabled with the PTP bit set, then the first interrupt can be up to 512 clocks early.

6.9.3.8 SOFTWARE SERVICE REGISTER (SWSR). The SWSR is the location to which the SWT servicing sequence is written. To prevent an SWT timeout, the user should write a $55 followed by a $AA to this register. The SWT can be disabled by clearing the SWE bit in the SYPCR. The SWSR can be written at any time, but returns all zeros when read.

7	6	5	4	3	2	1	0
SWSR7	SWSR6	SWSR5	SWSR4	SWSR3	SWSR2	SWSR1	SWSR0

SUPERVISOR ONLY

6.9.3.9 CLKO CONTROL REGISTER (CLKOCR). The CLKOCR controls the operation of the CLKO2-1 pins. CLKOWP bit in CLKOCR is used as a protect mechanism to prevent erroneous writing. CLKOCR can be read or written only in supervisor mode.

7	6	5	4	3	2	1	0
CLKOWP	—	—	—	COM2		COM1	
RESET:							
0	0	0	0	0/1	0/1	0/1	0/1

CLKOWP—CLKOCR Write Protect

This bit protects accidental writing into the CLKOCR. After reset, this bit defaults to zero to enable writing. Setting this bit prevents further writing (excluding the first write that sets this bit).

Bits 6 -4—Reserved

COM2—Clock Output 2 Mode

The COM2 bits control the output buffer strength of the CLKO2 pin. When both bits are set, the CLKO2 pin is held in the high (1) state. These bits can be dynamically changed without generating spikes on the CLKO2 pin. If the CLKO2 pin is not connected to external circuits, set both bits (disabling the clock output) to minimize noise and power dissipation.

The COM2 bits are set to ones at system reset, unless MODCK = 01, in which case they are cleared. This causes CLKO2 to be disabled at system reset, unless MODCK = 01. (This causes CLKO2 to default to a quiet state, unless it is needed in an MC68040 companion mode system.)

00 = Clock Out Enabled, Full-Strength Output Buffer
01 = Clock Out Enabled, 2/3-Strength Output Buffer
10 = Clock Out Enabled, 1/3-Strength Output Buffer
11 = Clock Out Disabled (Driving 1)

COM1—Clock Output 1 Mode

The COM1 bits control the output buffer strength of the CLKO1 pin. When both bits are set, the CLKO1 pin is held in the high (1) state. These bits can be dynamically changed

without generating spikes on the CLKO1 pin. If the CLKO1 pin is not connected to external circuits, set both bits (disabling the clock output) to minimize noise and power dissipation.

The COM1 bits are cleared at system reset, unless MODCK = 01, in which case they are ones. This prevents CLKO1 and CLKO2 from both defaulting to an active state after reset, for all four combinations of the MODCK1-0 pins. This reduces the potential for system noise at reset. CLKO1 may be enabled later, if desired.

00 = Clock Out Enabled, Full-Strength Output Buffer
01 = Clock Out Enabled, 2/3-Strength Output Buffer
10 = Clock Out Enabled, 1/3-Strength Output Buffer
11 = Clock Out Disabled (Driving 1).

NOTE

If a continuous clock source is needed by the user when MODCK = 01, then the user should use the output of the external oscillator instead of the CLKO1 pin.

The sum of strength of CLKO1 and CLKO2 should not exceed 1. (If COM2 is set to 2/3 drive configuration, then COM1 cannot be greater than 1/3 drive configuration)

When MODCK is set to 01, CLOCKO1 is disabled at reset until the COM1 bit is changed.

The CLKO1 logic is as follows:

when COM1 bits in the CLKOCR = 11, CLKO1 is driven high;

when COM1 bits in the CLKOCR ≠11, CLKO1 is driven according to the following conditions:

a. Driven low if the PLL is NOT locked AND $\overline{\text{RESETH}}$ is asserted.

b. Driven with the same frequency as EXTAL clock if the PLL is locked.

c. Driven low if the PLL unlocked due to MF change.

6.9.3.10 PLL CONTROL REGISTER (PLLCR). The PLLCR controls the operation of the PLL. It can be read or written only in supervisor mode. Writing into this register is allowed only if the PLLWP bit is zero. The reset state of PLLCR produces an operating frequency of 13.14 MHz when the PLL is referenced to a 32.768-kHz crystal or to 4.192 MHz. Two pins (MODCK1–MODCK0) are sampled during hardware reset (see Table 6-1).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLLEN	PLLWP	PREEN	STSIM	MF11	MF10	MF9	MF8	MF7	MF6	MF5	MF4	MF3	MF2	MF1	MF0
RESET:															
1*	0	0	0	0	0	0	MODCK1	MODCK1	0	0	MODCK1	0	0	0	0

Note:
The default value is one unless MODCK1-MODCK0 pins are tdriven with 00 during reset.

PLLEN—PLL Enable Bit

The QUICC does not support disabled PLL. The bit is always set to one on reset unless the MODCK1-MODCK0 pins are driven with 00 during reset. This mode of MODCK (00) is reserved as indicated in 2.1.10.4 Clock Mode Select (MODCK1–MODCK0).

PLLWP—PLLCR Write Protect

This bit protects accidental writing of the PLLCR. After reset, this bit defaults to zero to enable writing. Setting this bit prevents further writing (excluding the first write that sets this bit).

PREEN—Prescaler Enable

This bit controls the divide-by-128 prescaler on the EXTAL signal. This bit is set during hardware reset only if the MODCK1–MODCK0 pins specify that the divide-by-128 prescaler is used. It may be read thereafter as a status. If it is ever modified by software, it should be changed at the same time that the corresponding change in the MF bits is performed.

0 = The divide-by-128 prescaler is disabled. CLKIN = EXTAL—the PLL input clock frequency is the EXTAL frequency.
1 = The divide-by-128 prescaler is enabled. CLKIN = EXTAL/128—the PLL input clock frequency is the EXTAL frequency divided by 128.

STSIM—Stop Mode SIMCLK

0 = When the LPSTOP instruction is executed, the SIMCLK is driven from the crystal. The frequency is either EXTAL/2 or EXTAL/256, depending on the divide-by-128 option. The PLL is disabled to conserve power.
1 = When the LPSTOP instruction is executed, the SIM60 clock is driven from the VCO.

MF11–MF0—Multiplication Factor

These bits define the multiplication factor that will be applied to the PLL input frequency. The multiplication factor can be any integer from 1 to 4096. The system frequency is ((MF bits + 1) $\times$ EXTALDIV), where EXTALDIV is either EXTAL or EXTAL/128, depending on the MODCK bit configuration. The multiplication factor must be chosen to ensure that the resulting VCO output frequency will be in the range from 10 MHz to the maximum allowed clock input frequency (e.g., 25 MHz for a 25-MHz device). In addition, the VCO outputs a 2$\times$ frequency signal, which is 2$\times$ the multiplied value configured in the MF bits. This frequency is not used in any of the MF calculations.

The value 000 results in a multiplier value of 1; the value $FFF results in a multiplier value of 4096.

Anytime a new value is written into the MF11–MF0 bits, the PLL will lose the lock condition and, after a delay, will relock. When the PLL loses its lock condition, all clocks generated by the PLL are disabled. After a hardware reset, the MF11–MF0 bits default to either 0 or 400 ($190 hex), depending on the MODCK1–MODCK0 pins (giving a multiplication factor of 1 or 401). If the multiplication factor is 401, then a standard 32.768-kHz crystal generates an ini-

tial general system clock of 13.14 MHz. The user would then write the MF bits to raise this frequency to the desired frequency.

NOTE

SWT clocking does not stop when the PLL is in the process of acquiring a lock. Therefore, the user should service the SWT (reset its count) before and after changing the MF bits.

6.9.3.11 CLOCK DIVIDER CONTROL REGISTER (CDVCR). The CDVCR controls the operation of the low-power divider for the various clocks on the QUICC. It can be read or written only in supervisor mode. Writing this register is allowed only if the CDVWP bit is zero. The reset state of CDVCR produces the maximum frequency for all the clocks that it affects.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CDVWP	DFSY		DFTM		INTEN			RRQEN	DFNL			DFNH			CSRC
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CDVWP—CDVCR Write Protect

This bit protects accidental writing of the CDVCR. After reset, this bit defaults to zero to enable writing. Setting this bit prevents further writing (excluding the first write that sets this bit).

DFSY—Division Factor for the SyncCLK

These bits define the SyncCLK frequency. Changing the value of the these bits will not result in a loss-of-lock condition. These bits are cleared by a hardware reset. The default value is divide by 1 (VCO/2) which is 25 MHz in a 25-MHz system.

00 = Divide by 1 (normal operation)
01 = Divide by 4
10 = Divide by 16
11 = Divide by 64

DFTM—Division Factor for the BRGCLK

These bits define the BRGCLK frequency. Changing the value of the these bits will not result in a loss-of-lock condition. These bits are cleared by a hardware reset. The default value is divide by 1 (VCO/2) which is 25 MHz in a 25-MHz system.

00 = Divide by 1 (normal operation)
01 = Divide by 4
10 = Divide by 16
11 = Divide by 64

INTEN—Interrupt Enable

These bits specify if the general system clock returns to high frequency (defined by the DFNH bits) *while* the CPU32+ either has a pending interrupt or an interrupt routine in process, either of which has a level higher than INTEN2–INTEN0. To prevent interrupts from causing the general system clock to automatically switch to high frequency, write INTEN with 111.

RRQEN—RISC Request Enable

This bit specifies if the general system clock returns to high frequency (defined by the DFNH bits) *while* the CPM RISC controller is not idle.

0 = Remain in lower frequency (defined by DFNL) even if the RISC controller is not idle.
1 = Switch to the high frequency (defined by DFNH) when the RISC controller needs to execute a routine.

DFNL—Division Factor Lowest Frequency

These bits are required in two cases: 1) to reduce the general system clock to a frequency lower than that which can be obtained in DFNH and 2) to automatically switch between the DFNL rate and the DFNH rate. See 6.5.5 QUICC Internal Clock Signals for details on how to automatically switch between the DFNH rate and the DFNL rate.

The user may load these bits with the desired divide value, and then set the CSRC bit to change the frequency. Changing the value of the these bits will never cause a loss-of-lock condition. These bits are cleared by a hardware reset.

000 = Divide by 2
001 = Divide by 4
010 = Divide by 8
011 = Divide by 16
100 = Divide by 32
101 = Divide by 64
110 = Reserved
111 = Divide by 256

DFNH—Division Factor High Frequency

Changing the value of these bits will never cause a loss-of-lock condition. These bits are cleared (divide by 1) by a hardware reset. The default value is divide by 1 (VCO/2), which is 25 MHz in a 25-MHz system. The user may write the DFNH bits at any time to change the general system clock rate.

See 6.5.5 QUICC Internal Clock Signals for details on how to automatically switch between the DFNH rate and the DFNL rate.

000 = Divide by 1 (normal operation of general system clock when CSRC = 0)
001 = Divide by 2
010 = Divide by 4
011 = Divide by 8
100 = Divide by 16
101 = Divide by 32
110 = Divide by 64
111 = Reserved

CSRC—Clock Source Bit

The CSRC bit specifies whether the general system clock is determined by the DFNH or the DFNL bits. Setting this bit switches the general system clock to the DFNL value (i.e.,

for entering into low-power mode). Clearing this bit switches the general system clock to the DFNH value. CSRC is cleared at hardware reset.

0 = General system clock is determined by the DFNH value.
1 = General system clock is determined by the DFNL value.

6.9.3.12 BREAKPOINT ADDRESS REGISTER (BKAR). This register contains the 32-bit breakpoint address used in the breakpoint address match function. Its contents are only valid if the valid bit is set in the BKCR. BKAR is undefined at reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA15	BA14	BA13	BA12	BA11	BA10	BA9	BA8	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

6.9.3.13 BREAKPOINT CONTROL REGISTER (BKCR). This register contains miscellaneous bits required for the breakpoint address match function. BKCR is cleared at reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
—	—	—	—	—	—	—	—	—	—	—	—	BAS	BUSS	RW1	RW0
												0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIZM	SIZ1	SIZ0	NEG	MA1	MA0	AS8	AS7	AS6	AS5	AS4	AS3	AS2	AS1	AS0	V
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits 31–20—Reserved

BAS—Breakpoint Acknowledge Support

This bit determines whether to support the CPU32+ breakpoint acknowledge cycle by asserting $\overline{\text{BERR}}$ or ignore the breakpoint acknowledge cycle by allowing it to be handled by the external bus.

0 = No action taken during CPU32+ breakpoint acknowledge cycles.
1 = Assert $\overline{\text{BERR}}$ during CPU32+ breakpoint acknowledge cycles.

NOTE

Do not assert this bit if the QUICC is in slave mode.

BUSS—Bus Select

This bit determines whether the breakpoint logic will use the IMB value or the external bus value to detect breakpoint match.

0 = Use the IMB.
1 = Use the external bus. A0 and A1 are masked from the comparison.

NOTES

This mode is used in QUICC slave operation to assert either the $\overline{\text{BKPTO}}$ line for the external CPU or the internal IMB $\overline{\text{BKPT}}$ line for an internal-to-internal IDMA/SDMA access. When the external bus is used, the breakpoint line will be asserted as if the SIZM bit is set.

In the case of an external MC68040 burst, only the first address of the burst is checked.

When the QUICC is in master mode this bit should be zero to prevent external breakpoint from being ignored.

RW1–RW0—Read/Write Selection

Assert a breakpoint match on read cycles only, write cycles only, or on both.

00 = Assert breakpoint on read cycles.
01 = Assert breakpoint on write cycles.
10 = Assert breakpoint on read or write cycles.
11 = Reserved.

SIZM—Size Mask

This bit determines whether the breakpoint logic will use the SIZ bits to determine whether a breakpoint match has occurred.

0 = Compare the size lines as programmed in the SIZ bits to determine whether a breakpoint match has occurred.

NOTE

This mode would normally be used to break on an access to a location that contains data.

1 = Mask the size lines. The size of the access is not used in determining whether a breakpoint match has occurred. The breakpoint logic will assert the break signal when the address and size overlaps the programmable value. For example if the programmable address is xxx2, the breakpoint line for the low word will be asserted when the access address is xxx2 with a word size or when the address is xxx0 with a long-word size.

NOTE

This mode would normally be used to break on an instruction fetch.

SIZ1–SIZ10—Size Bits

The breakpoint logic can cause a breakpoint match for accesses that correspond to the size of the access. Set the SIZM bit to disable this feature.

NOTES

The breakpoint logic will assert the break signal only when the address and size match the programmable value. For example, if the programmable address is xxx2 with word size, the breakpoint will be asserted only when the access address is xxx2 with word size, not when the address is xxx0 with long-word size.

The MA bits must be 00 for the size comparison to occur.

00 = Long Word
01 = Byte
10 = Word
11 = 3 Byte

Table 6-7. Breakpoint and Size Pin

Programmed BA1-BA0 (BKAR Reg)	IMB/EXT A1-A0	IMB/EXT SIZ1-SIZ0	Assert $\overline{\text{BKPT}}$
00	00	x	Yes
	01	x	No
	10	x	No
	11	x	No
01	00	00	Yes
	00	01	No
	00	1x	Yes
	01	x	Yes
	1x	x	Yes
10	00	00	Yes
	00	01	No
	00	10	No
	00	11	Yes
	01	00	Yes
	01	01	No
	01	1x	Yes
	10	x	Yes
	11	x	No
11	00	00	Yes
	00	01	No
	00	1x	No
	01	00	Yes
	01	01	No
	01	10	No
	01	11	Yes
	10	00	Yes
	10	01	No
	10	1x	Yes
	11	x	Yes

NOTE

The table is true for the case SIZM=1.

Breakpoint will be asserted ONLY if the programmed address is actually accessed.

Table 6-7 shows which combinations of A0-A1 and SIZ1-SIZ0, on eitherthe external bus or the IMB bus, assert the $\overline{\text{BKPT}}$ pin.

NEG—Negative Breakpoint Match

This bit allows the breakpoint match to occur, using negative address matching logic, when a block address is selected. If this bit is set, the rest of the address and address match logic define when a breakpoint match is *not* to occur. The R/$\overline{\text{W}}$, size, and FC compare logic are not affected by the NEG bit.

0 = Assert a breakpoint when the memory cycle matches the programmed values.
1 = Assert a breakpoint when the memory cycle does *not* match the programmed block address. NEG is ignored if the MA bits are 00.

MA1–MA0—Mask Address

The address mask bits allow the breakpoint logic to assert the breakpoint on a memory block boundary.

00 = No address bits are masked, 32 address bits are compared.
01 = Mask address bits 10–0; the block size is 2K.
10 = Mask address bits 12–0; the block size is 8K.
11 = Mask address bits 14–0; the block size is 32K.

NOTE

Using the NEG bit, the breakpoint can be asserted for accesses that fall into the block range or for those that fall out of the block range.

AS8–AS0—Address Space Bits

The address space field allows particular address spaces (function code combinations) to be masked during the breakpoint match decision. If an address space is masked, an access to this space will NOT assert the $\overline{\text{BKPT}}$ pin. To ignore function codes in the breakpoint match decision, program these bits to zero. The address space bits are:

AS8—Mask DMA space address space (FC3–FC0 = 1xxx)

AS7—Mask CPU space address space (FC3–FC0 = 0111)

AS6—Mask supervisor program address space (FC3–FC0 = 0110)

AS5—Mask supervisor data address space (FC3–FC0 = 0101)

AS4—Mask [Motorola reserved] address space (FC3–FC0 = 0100)

AS3—Mask [user reserved] address space (FC3–FC0 = 0011)

AS2—Mask user program address space (FC3–FC0 = 0010)

AS1—Mask user data address space (FC3–FC0 = 0001)

AS0—Mask [Motorola reserved] address space (FC3–FC0 = 0000)

The address space bits for 040 type MPU are:

AS8—Not Relevant for 040 Cycles

AS7—Acknowledge Access(TT1-TT0=11)

AS6—Supervisor Code Access(TT1-TT0=00, TM2-TM0=110)

AS5—Supervisor Data Access(TT1-TT0=00, TM2-TM0=101)

AS4—MMU Table search Code Access (TT1-TT0=00, TM2-TM0=100)

AS3—MMU Table search Data Access(TT1-TT0=00, TM2-TM0=011)

AS2—User Code Access(TT1-TT0=00, TM2-TM0=010)

AS1—User Data Access(TT1-TT0=00, TM2-TM0=001)

AS0—Data Cache Push Access(TT1-TT0=00, TM2-TM0=000)

For each address space bit:

0 = A breakpoint match can occur for this address space.

1 = Mask this address space from the breakpoint match logic. No breakpoint match will occur if this address space is used on a bus access.

V—Valid

This bit indicates when the contents of the breakpoint address register and breakpoint control register pair are valid. $\overline{\text{BKPT}}$ signal will not be asserted unless the valid bit is set.

0 = Contents not valid.

1 = Contents valid.

6.9.4 Port E Pin Assignment Register (PEPAR)

The PEPAR controls the I/O pins associated with the EBI. Refer to Section 4 Bus Operation for more information about the EBI. Port E pins can be independently programmed to be either CAS3–CAS0 or $\overline{\text{IACK6}}$ and $\overline{\text{IACK3}}$–$\overline{\text{IACK1}}$; $\overline{\text{AVEC}}$ (or $\overline{\text{AVECO}}$) or $\overline{\text{IACK5}}$; $\overline{\text{CS3}}$ or $\overline{\text{IACK7}}$; AMUX or $\overline{\text{OE}}$; A31–A28 or $\overline{\text{WE3}}$–$\overline{\text{WE0}}$.

Until the low byte of PEPAR is written, the $\overline{\text{WE3–WE0}}$/A31–A28 pins are three-stated. The PWW bit indicates whether the low byte of PEPAR was written. PEPAR may be read or written at any time.

15	14	13	12	11	10	9	8
—	SINTOUT			—	CF1MODE		IPIPE1/ RAS1DD
0	0	0	0	0	0	0	0
7	6	5	4	3	2	1	0
A28–A31 WE0–WE3	OE/ AMUX	PWW	CAS2, 3 IACK3, 6	—	CAS0, 1 IACK1, 2	CS7 IACK7	AVEC or (AVECO)/ IACK5
0	0	0	0	0	0	0	0

Bits 15, 11, and 3—Reserved

Bits 14–12—SINTOUT

These bits should only be modified from its default when the QUICC is configured in slave (disable CPU32+) mode. They are used to program the way the interrupt controller will assert its interrupt requests to the external logic.

000 = Default (Used only in CPU enable mode).
001 = Reserved.
010 = The QUICC interrupt request is the $\overline{\text{RQOUT}}$ output function on the $\overline{\text{IRQ1}}$ pin.
011 = The QUICC interrupt request is the $\overline{\text{IOUT2}}$–$\overline{\text{IOUT0}}$ outputs with the standard M68000 family interrupt level encoding on the $\overline{\text{IRQ6}}$, $\overline{\text{IRQ4}}$, and $\overline{\text{IRQ1}}$ pin, respectively.
100 = The QUICC interrupt request is the $\overline{\text{RQOUT}}$ output function on the $\overline{\text{PRTY2}}$ pin.
101 = The QUICC interrupt request is the $\overline{\text{IOUT2}}$–$\overline{\text{IOUT0}}$ outputs with the standard M68000 family interrupt level encoding on the PRTY0–PRTY2 pins, respectively.
110 = Reserved.
111 = Reserved.

NOTE

Until the low byte of PEPAR is written, the parity lines will be three-stated. The user should write the high byte of PEPAR at the same time that the low byte is written to avoid selecting a reserved combination of the SINTOUT bits.

Bits 10–9—CF1MODE

These bits are used to control the CONFIG1/$\overline{\text{BCLRO}}$/$\overline{\text{RAS2DD}}$ pin functionality.

00 = CONFIG1 input pin function is chosen.
01 = CONFIG1 input pin function is chosen.
10 = The $\overline{\text{BCLRO}}$ output function is chosen instead of the CONFIG1 pin.
11 = $\overline{\text{RAS2DD}}$ output function (RAS2 double-drive) is chosen instead of the CONFIG1 pin.

Bit 8—$\overline{\text{IPIPE1}}$/$\overline{\text{RAS1DD}}$

0 = If the QUICC is in normal mode, the $\overline{\text{IPIPE1}}$ output function is selected. If the QUICC is in slave mode, the $\overline{\text{BCLRI}}$ input function is selected.
1 = The $\overline{\text{RAS1DD}}$ output function (RAS1 double-drive) is selected.

Bit 7—A31–A28/$\overline{\text{WE0}}$–$\overline{\text{WE3}}$

0 = The A31–A28 input/output functions are selected.
1 = The $\overline{\text{WE0}}$–$\overline{\text{WE3}}$ output functions are selected.

NOTE

Until the low byte of PEPAR is written, the $\overline{\text{WE3}}$–$\overline{\text{WE0}}$/A31-28 pins are three-stated.

Bit 6—$\overline{\text{OE}}$/AMUX

0 = The $\overline{\text{OE}}$ output function is selected.
1 = The AMUX output function is selected.

Bit 5—PWW

This read-only bit is used to indicate if the $\overline{WE}$/ADDR and the PRTY lines have been programmed by the user or are still in the three-state condition because the PEPAR register has not been written.

0 = PEPAR has not been written. The $\overline{WE}$/ADDR and the PRTY lines are still being three-stated.
1 = PEPAR was written. The $\overline{WE}$/ADDR and the PRTY lines have been programmed in the PEPAR, so the configuration choices of these pins in the PEPAR are valid.

Bit 4—$\overline{CAS2}$, $\overline{CAS3}$/$\overline{IACK3}$, $\overline{IACK6}$

0 = The $\overline{CAS2}$ and $\overline{CAS3}$ output functions are selected.
1 = The $\overline{IACK3}$ and $\overline{IACK6}$ output functions are selected.

Bit 2—$\overline{CAS0}$, $\overline{CAS1}$/$\overline{IACK1}$, $\overline{IACK2}$

0 = The $\overline{CAS0}$ and $\overline{CAS1}$ output functions are selected.
1 = The $\overline{IACK1}$ and $\overline{IACK2}$ output functions are selected.

Bit 1—$\overline{CS7}$/$\overline{IACK7}$

0 = The $\overline{CS7}$ output function is selected.
1 = The $\overline{IACK7}$ output function is selected.

Bit 0—AVEC ($\overline{AVECO}$)/$\overline{IACK5}$

0 = The $\overline{AVEC}$ input function is selected in normal operation, or $\overline{AVECO}$ is selected in slave mode.
1 = The $\overline{IACK5}$ output function is selected.

6.10 MEMORY CONTROLLER

The memory controller is a sub-block of the SIM60 that is responsible for up to eight general-purpose chip-select lines and the DRAM controller. The DRAM controller itself can control up to eight memory banks.

6.10.1 Memory Controller Key Features

The key features of the memory controller are as follows:

- All Eight Memory Banks Support the Following:
  - —32-Bit Address Decode with 17 Bits of Address Masking
  - —Various Block Sizes—2 Kbytes up to 256 Mbytes
  - —From 0 to 15 Wait States Programmable with $\overline{DSACK}$ Generation
  - —Memory Bank Can Be Used by an External Master
  - —Supports Burst Accesses of the MC68040
  - —Byte Parity Generation/Checking
  - —Write-Protect Capability
  - —Four Byte-Write Enable ($\overline{WE}$) Signals
  - —Output Enable ($\overline{OE}$) Signal
  - —Special Options for Interfacing to Slow Peripherals
  - —Function Code Match with Mask Can Qualify Memory Bank Accesses

- General-Purpose Chip Selects (SRAM Banks)
  - —May Be Used with SRAM, EPROM, FEPROM, and Peripherals
  - —Global (Boot) Chip Select Available at System Reset
  - —Two-Clock Accesses to External SRAM
  - —Programmable Port Size of 8, 16, and 32 Bits for Each Chip Select
- DRAM Controller (DRAM Banks)
  - —Supports up to Eight Banks of DRAM of Size 128K × X, 256K × X, 512K × X, 1M × X, 2M × X, 4M × X, 8M × X or 16M × X
  - —Supports a DRAM Port Size of 16 or 32 Bits
  - —Internal Address Multiplexing for 16- and 32-Bit DRAM Systems Available for all On-Chip Bus Masters
  - —Glueless Interface to One Bank of DRAM SIMMs (Only External Buffers Are Required for Additional SIMM Banks)
  - —Four $\overline{\text{CAS}}$ Lines
  - —Two of the Eight $\overline{\text{RAS}}$ Lines May Be Output on Two Pins Each for Double-Drive Capability
  - —Page Mode with Page Switch Detection Logic
  - —Page Mode Supports 128K, 256K, 512K, 1M, 2M, 4M, 8M, and 16M Page Banks
  - —Supports Page Mode Normal, Page Hit, and Page Miss
  - —Burst Support for the MC68040 Accesses to DRAM
- DRAM Controller Also Contains a Refresh Unit with:
  - —$\overline{\text{CAS}}$ Before $\overline{\text{RAS}}$ Refresh Support
  - —A Programmable Refresh Timer
  - —Refresh Active During External Reset
  - —Disable Refresh Mode
  - —Stacking of up to Seven Refresh Cycles
- DRAM Controller Also Supports External Masters
  - —Supports MC68EC040 with 3,2,2,2 Line Fill (60-ns DRAMs)
  - —Supports DRAM for External QUICC or MC68030-Type Accesses (Page Support Available in this Mode)
  - —Supports DRAM Control for System Bus Containing External MC68EC040 and Multiple QUICCs
  - —Synchronous and Asynchronous External Masters Possible
  - —Special Options for External Master to Improve DRAM Performance

6.10.2 Memory Controller Overview

The block diagram of the QUICC memory controller is shown in Figure 6-10. The general-purpose chip selects provide a glueless interface to EPROM, SRAM, flash EPROM (FEPROM), and other peripherals. The general-purpose chip selects are available on lines $\overline{\text{CS0}}$–$\overline{\text{CS7}}$. $\overline{\text{CS0}}$ also functions as the global (boot) chip select for accessing the boot EPROM. The chip selects allow 0 to 15 wait states.

The flexible memory controller allows a glueless DRAM interface to single in-line memory modules (SIMMs) as well as a grid array of DRAMs on a board. The DRAM controller controls the address multiplexing, access mode, refresh operation, and the timing generation

for up to eight banks of DRAMs. The DRAM controller provides eight $\overline{RAS}$ lines for up to eight DRAM banks, four $\overline{CAS}$ lines and four parity (PRTY) lines (one for each data byte on the QUICC system bus), and a parity error signal ($\overline{PERR}$). The DRAM controller also provides multiplexed address lines for on-chip bus masters and an address mux signal (AMUX) to support an external address muxing for external masters that wish to use the QUICC DRAM controller for their accesses to DRAM. The DRAM controller also fully supports an external MC68EC040 (or other MC68040 family variations) with the signals BADD2, BADD3, $\overline{TA}$, $\overline{TS}$, and $\overline{TBI}$.

Alternatively, a general-purpose chip select may be used instead of any DRAM bank.

NOTE

When one of the eight banks of memory is configured to control DRAM, it is referred to as a DRAM bank. When one of the eight banks of memory is configured to control standard memory (such as SRAM and EPROM) or a peripheral, it is referred to as an SRAM bank. Thus, the term “SRAM bank” is used to mean “non-DRAM” in this description.

Some features are common to all eight memory banks. First, a full 32-bit address decode for each memory bank is possible, with 17 bits having address masking. The full 32-bit decode is available, even if all 32 address bits are not brought outside the QUICC. Each memory bank includes a variable block size from 2 Kbytes up to 256 Mbytes). From 0 to 15 wait states may be programmed with $\overline{DSACK}$ generation. The memory bank can be used by an external master, including the MC68EC040, in which case burst accesses are also supported. Parity may be generated and checked for any memory bank (SRAM, DRAM, etc.). Each memory bank may be selected for read-only or read/write operation. Byte-write enable ($\overline{WE}$) signals are available for each byte that is written to memory. Also, an output enable ($\overline{OE}$) signal is provided to eliminate external glue logic. Finally, the access to a memory bank may be restricted to only certain function codes for system protection. The function code comparison occurs with a mask option also.

The memory controller functionality allows QUICC-based systems to be built very easily. For instance, a minimal QUICC system may require no glue logic as shown in Figure 6-11. In this example, $\overline{CS0}$ is used for the boot EPROM, and $\overline{RAS1}$ is used for the DRAM SIMM. The $\overline{WE}$ signals are used to simplify the interface to the DRAM SIMM, and the $\overline{OE}$ signal is used to simplify the interface to the EPROM. Byte parity is supported in this configuration.

NOTE:

If the $\overline{WE}$ signal of the QUICC is used to control memory write operation, unless the memory width is 32 bit, the user must specify the correct memory width with the SPS0-1 bits of the option register. For example, if the SPS is programed for a 16 bit port size, only WE0 and WE1 will be asserted. If external assertion of $\overline{DSACK}$ is used due to the algorithm of dynamic bus sizing, the first bus cycle assumes a 32 bit port size and will output $\overline{WE}$ for 32bit regardless of external $\overline{DSACK}$ encoding.

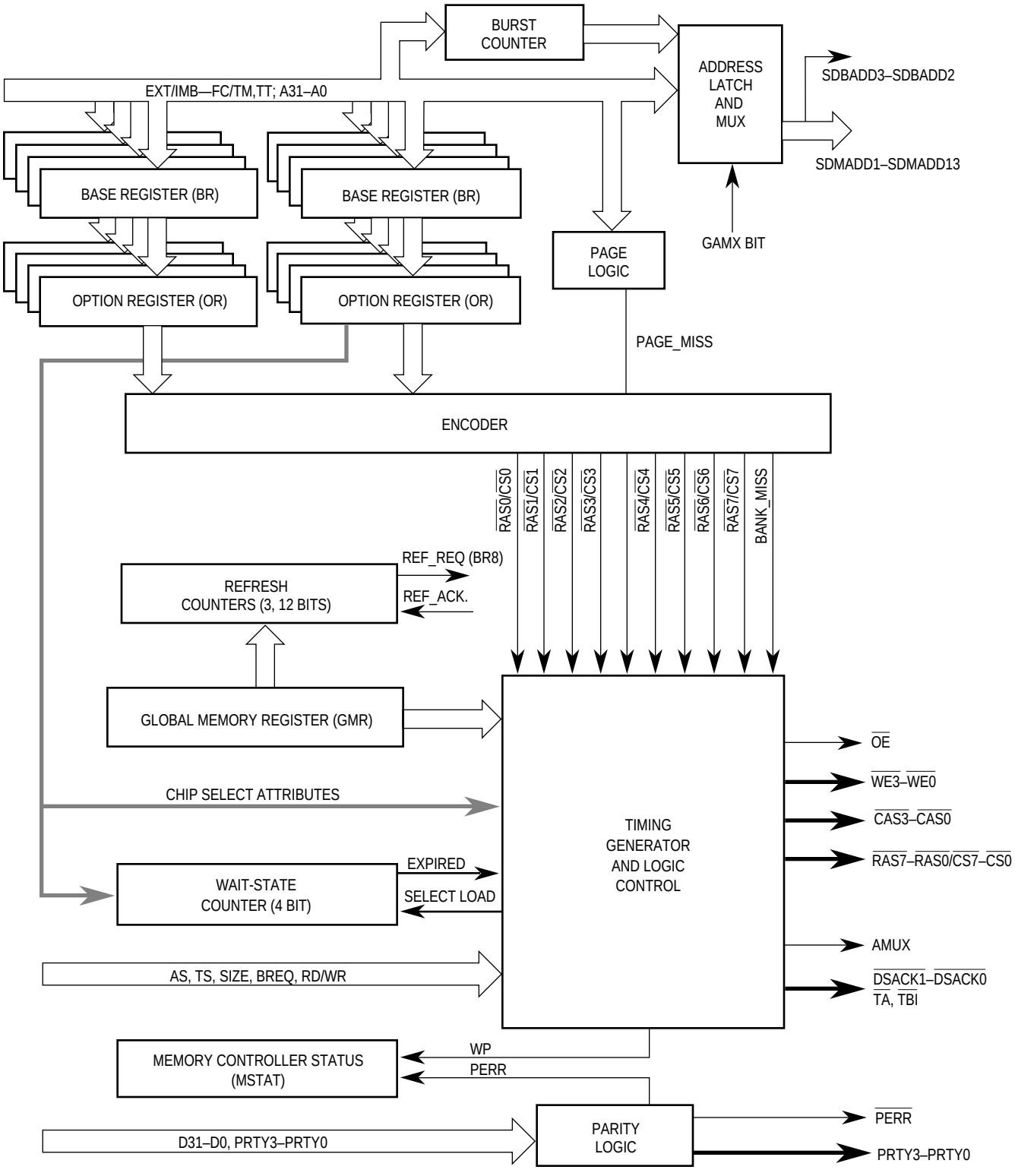


Figure 6-10. Memory Controller Block Diagram

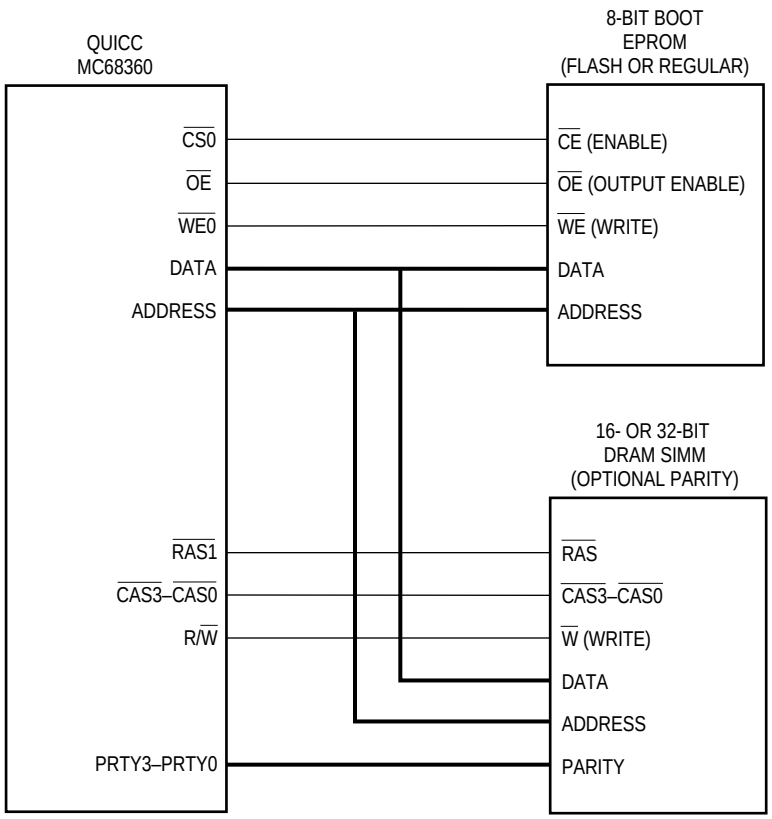


Figure 6-11. Minimum QUICC System Configuration

If a larger system is required, the only additional glue logic that may be needed is external buffers (see Figure 6-12). In this case, a boot EPROM and a flash EPROM are supported. Also, two DRAM SIMMs are supported using $\overline{\text{RAS1}}$ and $\overline{\text{RAS2}}$.

Each of the eight memory banks may be used by an external master such as an MC68EC040, MC68030, or even another QUICC. Whenever an external master accesses DRAM, SRAM, or a peripheral within one of the regions of the memory banks, the memory controller will control the access for that external master.

If DRAM is accessed by an external master, an external multiplexer must be provided. In that case, the QUICC AMUX signal can be used to control the multiplexing. The DRAM controller supports use by an MC68EC040 and another QUICC or MC68030-type device. In such a case, the MC68EC040 and QUICC/MC68030-type device can access the DRAM in different modes and at different rates. For instance, the MC68EC040 can access the DRAM using two-clock bursts, while an external QUICC accesses the DRAM using page mode with three-clock page hits, four-clock page normal, and five-clock page miss accesses. Thus, the MC68EC040 access to DRAM is not slowed by the presence of other slower masters on the system bus. In addition, the MC68EC040 is not slowed by the performance of the DRAM accesses by the QUICC's internal bus masters (CPU32+, IDMAs, SDMAs, etc.) All accesses may occur at different rates, with the MC68EC040 parameters being programmed independently and the external QUICC/MC68030-type master being up to one wait state

slower than the QUICC's internal bus masters. The external master may be either synchronous or asynchronous with respect to the QUICC system clock, with the exception of the MC68EC040, which must always be synchronous with respect to the QUICC system clock. Thus, if a 25-MHz QUICC is used, a 25-MHz MC68EC040 should also be used.

If an external MC68040 master does not use the memory controller at all, then the QUICC can operate asynchronously to the MC68040, but the QUICC MC68040 companion mode signals cannot be used, and the MC68040 bus signals must be converted to MC68030-type bus signals before the MC68040 accesses the QUICC's internal RAM and peripherals.

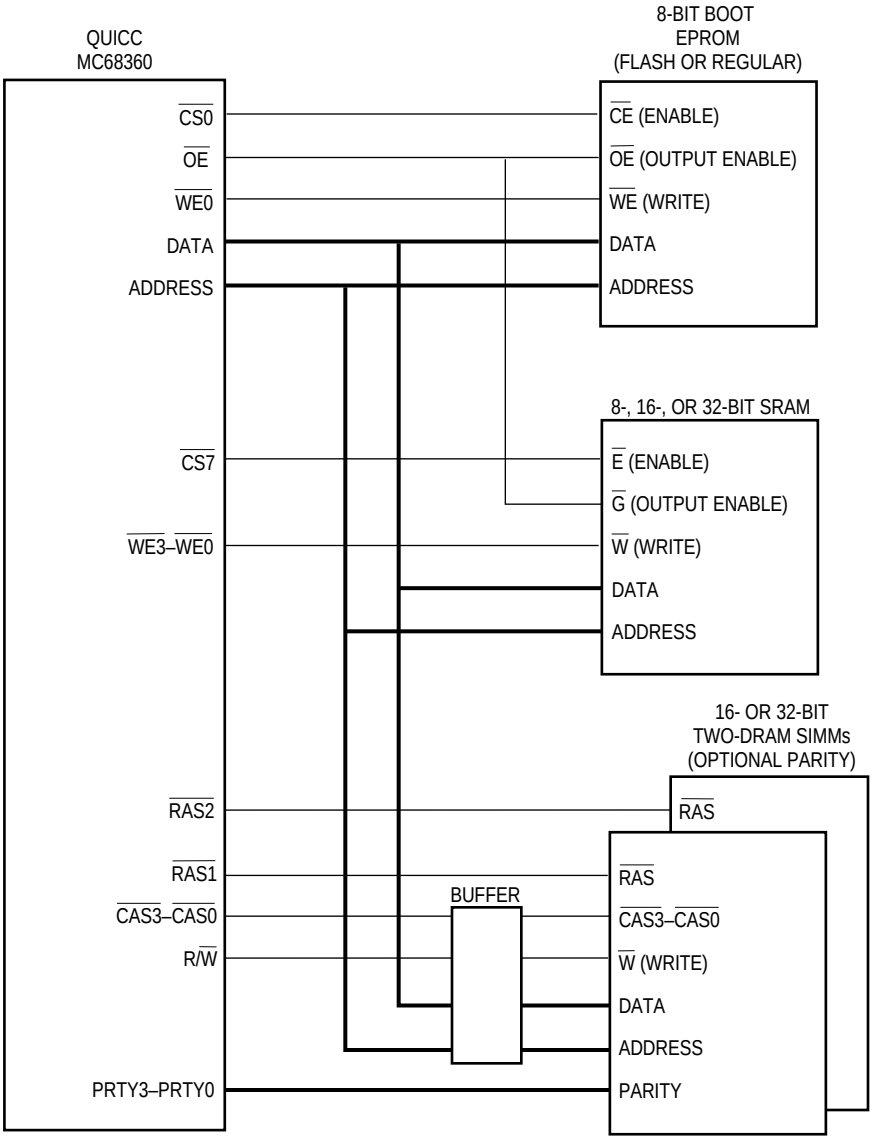


Figure 6-12. Larger QUICC System Configuration

6.11 GENERAL-PURPOSE CHIP-SELECT OVERVIEW (SRAM BANKS)

Any memory bank that is not used to control DRAM may be used as a general-purpose chip select, including pins $\overline{CS0}$–$\overline{CS7}$. This bank is called an SRAM bank. These pins may be used to support external memory such as SRAM, EPROM, flash EPROM, EEPROM, and peripherals.

The SRAM banks also have some unique features not available in the DRAM banks. First, upon system reset, a global (boot) chip select is available. This provides a boot ROM chip select before the system is fully configured. Second, the SRAM banks offer two-clock accesses to external SRAM. Finally, each SRAM bank supports a choice of the port size of its memory or peripheral to be 8, 16, or 32 bits with proper $\overline{DSACK}$ generation for those port sizes. Thus, an 8-bit EPROM may be used with a 32-bit SRAM, etc.

6.11.1 Associated Registers

The general-purpose chip selects are controlled by the global memory register (GMR) and the memory controller status register (MSTAT). There is one GMR and MSTAT in the memory controller. Additionally, each SRAM bank has a base register (BR) and an option register (OR).

The GMR is used to control global parameters for both SRAM and DRAM banks.

The MSTAT reports write protect violations and parity errors for both SRAM and DRAM banks.

The BR and the OR for each of the general-purpose chip selects program most of the features. The BR contains a valid (V) bit to indicate that the register information for that chip select is valid.

6.11.2 8-, 16-, and 32-Bit Port Size Configuration

The general-purpose chip selects support dynamic bus sizing. Defined 8-bit ports are accessible on both odd and even addresses when connected to data bus bits 31–24; defined 16-bit ports can be accessed as odd bytes, even bytes, or even words when connected to data bus bits 31–16; and defined 32-bit ports can be accessed as odd bytes, even bytes, odd words, and even words or long words on long-word boundaries. The port size is specified by the SPS bits in the OR.

6.11.3 Write Protect Configuration

The WP bit in each BR can restrict write access to its range of addresses. Any attempt to write this area will result in the WPER bit being set in the MSTAT.

6.11.4 Programmable Wait State Configuration

The general-purpose chip selects support internal DSACKx generation. They allow fast two-clock accesses to external memory by an internal bus master; from zero-wait-state accesses (3 clocks) up to 14-wait-state accesses (17 clocks) are allowed for internal bus masters. For external bus masters, two-clock accesses are not allowed, but 14 wait states may be programmed. Additionally, if the EMWS bit is set in the GMR, the chip selects can

provide one additional wait state for external masters, giving up to 15 wait states by the chip selects. This is programmed using the TCYC bits in the OR.

6.11.5 Address and Address Space Checking

The defined base address is written to the BR. The address mask bits for that address are written to the OR. The function code access value, if desired, is written to FC bits in the BR. The FCM bits in the OR may be used to mask this selection. If the address space (function code) checking is not desired, program the FCM bits to zero. Also, the chip select can be configured not to assert during CPU space (i.e., interrupt acknowledge) cycles that have a function code value 0111. This option is decided with the NCS bit in the GMR.

6.11.6 SRAM Bank Parity

Parity can be configured for any SRAM bank. Parity is generated and checked on a per-byte basis using PRTY3–PRTY0 if the PAREN bit is set in the BR. The OPAR bit in the GMR determines the type of parity (odd or even), and the PBEE bit in the GMR determines if an internal master should generate an error as a result of a parity error. Any parity error activates the $\overline{\text{PERR}}$ pin until the associated PERx bit in the MSTAT is cleared.

NOTE

Asynchronous external masters do not have parity support.

DW40 bit in the GMR must be set to support parity with external 040 master.

Parity is not supported for bus cycles terminated with external assertion of $\overline{\text{DSACK}}$ or $\overline{\text{TA}}$.

6.11.7 External Master Support

The SRAM banks support the internal bus masters, such as the CPU32+, IDMAs, and SDMAs, as well as external bus masters, such as the QUICC, MC68030, or MC68EC040. In the case of an external master, an additional wait state may be programmed into the SRAM bank to compensate for the additional decoding time. This capability is programmed in the EMWS bit of the GMR.

The MC68EC040 must always be synchronous to the QUICC clock. The SRAM bank supports bursting by the MC68EC040 if the BACK40 bit in the BR is set. During this access, $\overline{\text{CS}}$, PRTYx, $\overline{\text{PERR}}$, $\overline{\text{DSACK}}$/$\overline{\text{TA}}$/$\overline{\text{TBI}}$, and BADDR3–BADDR2 are all valid signals. The SRAM bank waits for the MC68EC040 $\overline{\text{TS}}$ line to be asserted before starting any MC68EC040 access. Burst (line fill) transfers are also supported.

The chip-select logic supports MC68030/QUICC external masters in two modes. In the asynchronous mode, the logic asserts the $\overline{\text{CS}}$ and $\overline{\text{DSACK}}$ lines as soon as an address match is detected from the external master. The chip select in this mode is waiting for the external master's $\overline{\text{AS}}$ line to be asserted. In the synchronous mode, the $\overline{\text{CS}}$ and $\overline{\text{DSACK}}$ assertion and negation timings are synchronous. The synchronous mode is programmed in the SYNC bit of the GMR.

When more wait states are programmed into the TCYC bits of the OR, the external $\overline{AS}$ (or $\overline{TS}$ line) is synchronized internally.

The BADDR3–BADDR2 signals equal the A3–A2 signals when a burst is not in progress. This configuration allows a non-bursting master to access the same memory as a bursting external master by using the BADDR3–BADDR2 signals.

6.11.8 Global (Boot) Chip-Select Operation

Global (boot) chip-select operation allows address decoding for a boot ROM before system initialization occurs. $\overline{CS0}$ is the global chip-select output. Its operation differs from the other external chip-select outputs following a system reset. When the CPU32+ begins accessing memory after a system reset, $\overline{CS0}$ is asserted for every address, unless the MBAR is accessed or an internal peripheral on the IMB is accessed.

The global chip select provides a programmable port size at system reset using the CONFIG pins. This capability allows a boot ROM to be located anywhere in the address space (with up to 14 wait states), while still providing the stack pointer and program counter values at $00000000 and $00000004, respectively. The global chip select does not provide write protection and responds to all function codes. $\overline{CS0}$ operates in this manner until the first write to the $\overline{CS0}$ option register (OR0). $\overline{CS0}$ can be programmed to continue decoding a range of addresses after this write, provided the desired address range is first loaded into base register 0. After the first write to the OR0, the global chip select can only be restarted with a system reset.

6.11.9 SRAM Bus Error

The $\overline{BERR}$ signal may be asserted by the SRAM controller in the case of a parity error or by the bus monitor of the SIM60 as a result of a write-protect violation. In addition, if the $\overline{BERR}$ signal is asserted externally, it should not be asserted until at least S2 of the bus cycle.

6.12 DRAM CONTROLLER OVERVIEW (DRAM BANKS)

The DRAM controller supports a glueless interface to 16-bit (18 bit with parity) or 32-bit (36 bit with parity) DRAM or DRAM SIMMs from an internal QUICC master (CPU32+, IDMAs, SDMAs). Many different DRAM bank sizes are supported: 128K, 256K, 512K, 1M, 2M, 4M, 8M, and 16M; thus, DRAMs such as 128K x 8, and 16M x 4 are supported. The DRAM controller performs the address multiplexing for internal masters using the low-order address lines.

Table 6-8 lists the physical address lines of the DRAM (row and column). In the case of a 16-bit DRAM port size with a 512K DRAM device (e.g., two 512K x 8 devices for a total of 16 bits wide), the row address to the DRAM bank will be A19–A10, and the column address to the DRAM bank will be A9–A1. However, these signals will be internally multiplexed on the A10–A1 pins; thus, the user should connect A10–A1 to the address pins on each 512K DRAM device.

Table 6-8. Address Multiplexing

DRAM Size	Address Lines (32-Bit Port)			Address Lines (16-Bit Port)		
	Physical Address			Physical Address		
	Column	Row	DRAM Address	Column	Row	DRAM Address
128K	A2–9	A10–18	A2–10	A1–8	A9–17	A1–9
256K	A2–10	A11–19	A2–10	A1–9	A10–18	A1–9
512K	A2–10	A11–20	A2–11	A1–9	A10–19	A1–10
1M	A2–11	A12–21	A2–11	A1–10	A11–20	A1–10
2M	A2–11	A12–22	A2–12	A1–10	A11–21	A1–11
4M	A2–12	A13–23	A2–12	A1–11	A12–22	A1–11
8M	A2–12	A13–24	A2–13	A1–11	A12–23	A1–12
16M	A2–13	A14–25	A2–13	A1–12	A13–24	A1–12

When there are external masters on the system bus, an external multiplexer should be used for the DRAM banks that are accessed by the external masters. The DRAM controller provides this timing with the AMUX line.

The DRAM controller supports byte-level parity for any DRAM bank.

The DRAM controller use $\overline{CAS}$-before-$\overline{RAS}$ refresh cycles. The refresh cycles are timed using a dedicated refresh timer. The refresh operation can be disabled.

The DRAM controller supports normal access mode and several fast access modes:

- Normal Access Mode. In this mode, each access to DRAM is handled independently using conventional DRAM timing.
- Page Mode. In this mode, the DRAM controller first establishes a constant row address, and then strobes a series of column addresses into the DRAM. The DRAM controller strobes both a row and a column address into the DRAM on the first access, but from that point on, it strobes only column addresses into the DRAM during access periods to the same DRAM page. After each access, the $\overline{CAS}$ signal is negated. The $\overline{RAS}$ line remains asserted until a different DRAM bank is accessed.

NOTE

This mode is not supported for external MC68040 masters.

- Burst Mode. In this mode, the DRAM controller detects the MC68EC040 line transfer and strobes both a row and a column address into the DRAM on the first access, but from that point on, it strobes only column addresses into the DRAM. For this access, the DRAMC internally generates address lines 2 and 3 on the BADD3–BADD2 pins.

NOTE

Burst mode is supported for the MC68XX040 type master only.

During all DRAM accesses, $\overline{RAS}$, $\overline{CAS}$, R/$\overline{W}$ and $\overline{DSACK}$/$\overline{TA}$ are valid signals. The following paragraphs detail the operation of each DRAM controller access type.

6.12.1 DRAM Normal Access Support

When accessing a DRAM, the DRAM controller uses the $\overline{RAS}$ and $\overline{CAS}$ pins. When an access to a DRAM memory bank is made, a normal cycle occurs when DSSEL = 1 in the OR, PGME = 0 in the OR, and BACK40 = 0 in the BR. The timing of the cycle is programmable using the TCYC bits in the OR.

A normal DRAM access can also be made by an external MC68EC040. In this case, the WBT40 bit determines the $\overline{RAS}$ precharge time, and the TSS40 bit determines how $\overline{TS}$ is sampled.

A normal DRAM access can also be made by an external MC68030/QUICC. In this case, the WBTQ bit determines the $\overline{RAS}$ precharge time.

The DRAM controller initiates a transaction by driving the row address on the low address lines. After the value on the address pins is the row address, the DRAM controller asserts $\overline{RAS}$. One clock phase later, the column address is driven on the low address lines as defined by the programmed DRAM size, and a clock phase later, the $\overline{CAS}$ signal is asserted. If the cycle is a write transfer, then data is output at that point. The DRAM controller then waits for the expiration of the TCYC length attribute and completes the cycle. The next cycle will begin only after the value programmed in the WBTQ field expires.

The assertion of $\overline{RAS}$ can be delayed by one clock phase to relax the address to $\overline{RAS}$ timing by setting the TRLXQ bit in the BR. When this bit is set the column address is driven one clock later, and $\overline{CAS}$ is delayed by one clock (see Figure 6-16).

The DRAM controller may also generate and check four parity lines (PRTY3–PRTY0). The parity can be either odd or even as programmed with the OPAR bit in the GMR. During write cycles, the DRAM controller generates the parity on the four parity lines. During a read cycle, the DRAM controller checks the parity. If the PAREN bit in the BR is set when a parity error occurs, the DRAM controller asserts the $\overline{PERR}$ line and sets the PERx bit in the MSTAT. For internal cycles, the DRAM controller will assert $\overline{BERR}$ when a parity error occurs and the PBEE bit in the GMR is set.

NOTE

Read-modify-write cycles may be performed using the DRAM controller. These are implemented as a read cycle followed by a write cycle. Some DRAMs offer a special read-modify-write access using special timing. This access timing is not supported by the QUICC's DRAM controller.

6.12.2 DRAM Page Mode Support

The DRAM banks supports a page mode memory access to DRAMs for the internal masters and for an external QUICC/MC68030-type master. A memory bank is configured to page mode if its DSSEL and PGME bits in the OR are set.

Many DRAMs support a page mode operation that reduces access time if multiple accesses are performed within the same page. In this mode, the DRAM controller continues to assert

the $\overline{RAS}$ signal of the DRAM bank. This $\overline{RAS}$ signal will remain active until another DRAM bank is accessed. The page size is determined by the PGS bits in the GMR.

If a different bank of DRAM is accessed, followed by an access to a DRAM bank on which page mode is selected, then the DRAM controller negates the $\overline{RAS}$ signal to the other bank and asserts the particular $\overline{RAS}$ line for the page mode bank, followed by the rest of the DRAM access. This is called a page mode normal cycle.

On each access to a DRAM bank in which the page mode is enabled and the previous DRAM cycle was to that bank, the address of the last access to this bank is compared to the current address. If the two addresses fall within the same page, then the access cycle begins immediately with the assertion of the column address and $\overline{CAS}$ signal. This is called a page hit.

In case of a page miss (the address of the last access and current address do not fall within the same page), the $\overline{RAS}$ signal must be negated and held high for a period that matches the value programmed in the WBTQ control field of the current DRAM region, and then a full cycle (including row and column phases) is executed. This is the slowest DRAM access since the $\overline{RAS}$ signal must first be negated, followed by the precharge time.

Since it is difficult to predict the performance impact of page mode, the user may wish to try the application software with and without page mode enabled, and compare the results. The ability to concentrate the code/data accesses into the same page of the DRAM is central to achieving a performance improvement.

Some systems will need an additional wait state to perform write cycles during a page hit. To gain a wait state, set the delay write cycle for the QUICC DWQ bit in the GMR of the DRAM bank.

NOTES

Page mode is supported only for the internal QUICC cycles or external MC68030/QUICC cycles.

If any two DRAM banks overlap each other in their address space, page mode must not be selected for either of those banks.

6.12.3 DRAM Burst Access Support

The DRAM controller supports burst accesses made by an external MC68EC040 (or other MC68040 family member) if the BACK40 bit is set in the BR. The MC68EC040 requests a burst to be performed with a line-fill indication on the SIZx pins ($\overline{SIZ}$ = 11) and the TTx pins. In this case, the DRAM controller performs a normal access ($\overline{RAS}$ and $\overline{CAS}$), followed by requests to the DRAM for the next three sequential long-word operands ($\overline{CAS}$ only). The DRAM controller automatically increments the addresses to the DRAM using the BADDR3–BADDR2 pins.

The length of an MC68EC040 burst cycle can be distinguished from the length of the initial access with the BCYC bits of the OR.

6.12.4 DRAM Bank Parity

Parity can be configured for any DRAM bank. Parity is generated and checked on a per-byte basis using PRTY3–PRTY0 if the PAREN bit is set in the BR. The OPAR bit in the GMR determines the type of parity (odd or even), and the PBEE bit in the GMR determines if an internal master should generate an error as a result of a parity error. Any parity error activates the $\overline{\text{PERR}}$ pin until the associated PERx bit in the MSTAT is cleared.

NOTE

Asynchronous external masters do not have parity support.

Parity is not supported for bus cycles terminated with external assertion of $\overline{\text{DSACK}}$ or $\overline{\text{TA}}$.

6.12.5 Refresh Operation

The DRAM controller uses $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ refresh cycles. The refresh cycles are timed using a dedicated refresh timer. In the $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ method, the DRAMs have an internal refresh row address counter, so row addresses need not be supplied by the DRAM controller. These DRAMs recognize the assertion of $\overline{\text{CAS}}$ before the assertion of $\overline{\text{RAS}}$ and perform the refresh using their internal refresh row address value.

Each time the refresh timer expires, the DRAMC performs a refresh cycle. At the first opportunity after acquiring bus mastership, the DRAM controller requests the bus with the highest bus arbitration priority level 6. In addition, it asserts the $\overline{\text{BCLRO}}$ signal to minimize the delay before the refresh cycle begins, assuming the external bus master recognizes this signal and clears itself off the bus. Once the DRAM controller obtains the bus, it performs a refresh bus cycle to the DRAM bank.

If more than one bank of DRAM exists in the system, the user should program the refresh controller to request the bus more often (N times as often, where N is the number of banks). For instance, typical DRAMs require a refresh every 15.6 μs. If 2 banks of DRAM exist in the system, the DRAM controller should be programmed to refresh every 7.8 μs. In the two bank case, the DRAM controller will alternate between the banks, using the $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ technique on each bank every 7.8 μs.

The DRAM controller will automatically stack up to seven refresh requests before receiving the bus mastership. Once it receives the bus, it will perform all stacked cycles (up to seven), as sequential, back-to-back refresh bus cycles.

Refresh cycles are executed only when the RFEN bit in the GMR is set. The refresh cycle length (three to six clocks) is programmed by the RCYC bits in the GMR. The time between refreshes is programmed in the RCNT bits in the GMR (see 6.13.1 Global Memory Register (GMR)).

NOTE

DRAM banks normally need eight read cycles and some delay time after a power-on reset. After enabling the DRAM bank, the

user can either perform the reads in software or wait for the DRAM refresh controller to perform these reads.

6.12.6 DRAM Bank External Master Support

The DRAM controller supports an external MC68EC040 as well as external MC68030-type masters, including an external QUICC.

Whenever an external master is supported, external address multiplexing must be provided by the user. The DRAM controller controls the multiplexing with the AMUX pin. On a normal access, AMUX defaults high, and the upper address lines (row) should be multiplexed to the DRAM first. After the external master outputs the full address, it asserts the $\overline{AS}$/$\overline{TS}$ signal to the QUICC. The DRAM controller then performs the address comparison, detects that the access is to one of its DRAM banks, and issues the corresponding $\overline{RAS}$ signal. After the assertion of the $\overline{RAS}$ signal, the DRAM controller continues the access and negates the AMUX signal, controls the $\overline{CAS}$ and $\overline{RAS}$ timing, and generates the $\overline{DSACK}$/$\overline{TA}$ signals to terminate the access. Refer to Section 9 Applications for a description of an external master system.

NOTE

To support the MC68030 cache fill operations, the DRAM controller asserts all four $\overline{CAS}$ signals during every QUICC/ MC68030-type external master read cycle to a DRAM bank. (This includes byte or word reads by the MC68030.)

The DRAM controller supports the MC68EC040 in an optimized way. The DRAM controller supports burst accesses made by an external MC68EC040 (or other M68040 family member) if the BACK40 bit is set in the BR. The MC68EC040 requests a burst to be performed with a line-fill indication on the SIZx (SIZ = 11) and TTx pins. In this case, the DRAM controller performs a normal access ($\overline{RAS}$ and $\overline{CAS}$), followed by requests to the DRAM for the next three sequential long-word operands ($\overline{CAS}$ only). The DRAM controller automatically increments the addresses to the DRAM using the BADDR3–BADDR2 pins.

6.12.7 Double-Drive $\overline{RAS}$ Lines

$\overline{RAS1}$ and $\overline{RAS2}$ have a special capability. To increase the available drive strength of these pins, the $\overline{RAS1}$ and $\overline{RAS2}$ signals may be output simultaneously on two pins each. The extra signals, called $\overline{RAS1DD}$ and $\overline{RAS2DD}$, increase the effective drive strength of the $\overline{RAS}$ signals. This selection is made in the PEPAR.

6.12.8 DRAM Bus Error

The $\overline{BERR}$ signal may be asserted by the DRAM controller in the case of a parity error or by the bus monitor of the SIM60 as a result of a write-protect violation. In addition, if the $\overline{BERR}$ signal is asserted externally, it should not be asserted until at least S2 of the bus cycle if TSS = 0 in the GMR, and until at least S4 of the bus cycle if TSS = 1 in the GMR.

6.13 PROGRAMMING MODEL

The user interfaces with the memory controller using eight identical sets of two registers, the BR and OR. There are also two global registers in the memory controller: the GMR and the MSTAT.

6.13.1 Global Memory Register (GMR)

The 32-bit read-write GMR contains selections that are common to the entire memory controller: DRAM refresh properties, DRAM bank properties, SRAM bank properties, and some global SRAM/DRAM properties. The reserved bits (4–0) should be written with zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RCNT7	RCNT6	RCNT5	RCNT4	RCNT3	RCNT2	RCNT1	RCNT0	RFEN	RCYC1	RCYC0	PGS2	PGS1	PGS0	DPS1	DPS0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SUPERVISOR SPACE ONLY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WBT40	WBTQ	SYNC	EMWS	OPAR	PBEE	TSS40	NCS	DWQ	DW40	GAMX	—	—	—	—	—
0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

The following bits are used for DRAM refresh properties.

RCNT7–RCNT0—Refresh Counter Period

These bits determine the refresh period according to the following equation:

$$\text{Refresh period} = \frac{\text{RFCNT+1}}{\text{System clk/16}}$$

Example: For a 25-MHz system clock and a required refresh rate of 15.6 μs per row, the RFCNT value should be 24 (decimal). 24/(25 MHz/16) = 15.36 μs, which is less than the required refresh period of 15.6 μs.

RFEN—Refresh Enable

0 = DRAM refresh is disabled.
1 = DRAM refresh is enabled.

RCYC1–RCYC0—Refresh Cycle Length

These bits determine the length of a refresh cycle.

00 = The refresh cycle is 4 clocks long, and $\overline{\text{RAS}}$ is negated for 3 phases prior to being asserted.
01 = The refresh cycle is 6 clocks long, and $\overline{\text{RAS}}$ is negated for 5 phases prior to being asserted.
10 = The refresh cycle is 7 clocks long, and $\overline{\text{RAS}}$ is negated for 5 phases prior to being asserted.
11 = The refresh cycle is 8 clocks long, and $\overline{\text{RAS}}$ is negated for 5 phases prior to being asserted.

The following bits are used for DRAM bank properties:

PGS2–PGS0—Page Size

This attribute determines the page size for the DRAM controller (see Table 6-9). The page size is the smallest DRAM size the user needs to support with page mode capability.

Table 6-9. DRAM Page Size

PGS2-PGS0	Address Lines Used	# Address/Page in Page Compare
000	A10-25(32), A9-25(16)	256 Addresses
001	A11-25(32), A10-25(16)	512 Addresses
010	A11-25(32), A10-25(16)	512 Addresses
011	A12-25(32), A11-25(16)	1024 Addresses
100	A12-25(32), A11-25(16)	1024 Addresses
101	A13-25(32), A12-25(16)	2048 Addresses
110	A13-25(32), A12-25(16)	2048 Addresses
110	A14-25(32), A13-25(16)	4096 Addresses

For instance, PGS = 001 (256K) should be used for a 32-bit-wide memory composed of four 256K × 8 devices, a 16-bit-wide memory composed of two 256K × 8 devices, or sixteen 256K × 1 devices. In all cases, the width of the DRAMs is irrelevant.

DPS1–DPS0—DRAM Port Size

This attribute determines the DRAM bank port size (see Table 6-10). The DRAM controller asserts the appropriate $\overline{\text{DSACKx}}$ lines according to these bits. If an MC68EC040 access is performed using this DRAM bank and SPS = 00 or 01, the DRAM controller operates the same way, but asserts $\overline{\text{TA}}$ instead of $\overline{\text{DSACK}}$.

Table 6-10. DRAM Port Size

DPS1–DPS0	Result
00	DRAM Port Size Is 32 Bits
01	DRAM Port Size Is 16 Bits
10	Reserved
11	External $\overline{\text{DSACKx}}$ Support

NOTES

The internal DRAM address multiplexer and the page logic support only a port size of 32 bits or 16 bits. An 8-bit DRAM port size is not allowed.

The DRAM controller does not support an external $\overline{\text{DSACKx}}$ response for a bank on which page mode is used. Also, an external $\overline{\text{DSACK}}$ response may not occur before $\overline{\text{RAS}}$ is asserted.

The DRAM controller does not support an external $\overline{TA}$ response for the MC68040 burst mode. Also, for non-burst MC68040 cycles, $\overline{TA}$ cannot be externally asserted before $\overline{RAS}$ is asserted.

WBT40—Wait Between Transfers (MC68EC040)

This attribute guarantees a minimum negation time for $\overline{RAS}$ when the QUICC DRAM controller is used by an external MC68EC040 master. It is used to comply with the $\overline{RAS}$ precharge time in DRAMs.

The user would normally decide whether to set the TSS40 bit before setting this bit.

0 = $\overline{RAS}$ is negated for 4 phases of the QUICC system clock (3 phases if TSS40 = 1).
1 = $\overline{RAS}$ is negated for 6 phases (5 phases if TSS40 = 1).

NOTE

TSS40 affects the WBT40 value in order to gain back one of the two phases that was lost by setting TSS40 = 1. This “gain back” only applies to back-to-back DRAM cycles.

WBTQ—Wait Between Transfers (QUICC-Type)

This attribute guarantees a minimum negation time for $\overline{RAS}$ when the QUICC DRAM controller is used by one of the internal masters or by an external master of the MC68030-type (includes an external QUICC). It is used to comply with the $\overline{RAS}$ precharge time in DRAMs.

0 = $\overline{RAS}$ is negated for 4 phases (3 phases in page mode—PGME = 1).
1 = $\overline{RAS}$ is negated for 6 phases (5 phases in page mode—PGME = 1).

DWQ—Delay Write for QUICC (DRAM Bank Only)

This attribute is used to add a clock to the assertion and negation of the $\overline{CAS}$ signal on DRAM page hit write cycles. The write cycle lasts one additional clock in this case. This attribute is applicable to an internal QUICC master and to an external MC68030/QUICC.

0 = Reads and writes are the same length.
1 = Add one clock to write cycles for DRAM banks where TCYC is set to 01.

NOTE

This bit must be set by the user if page mode is enabled for this DRAM bank (PGME = 1), or else the DRAM may latch invalid data during writes.

The following bits are used for SRAM bank properties:

DW40—Delay Write for 040 (SRAM Bank Only)

This attribute should be set if an additional wait state is necessary for SRAM write cycles. This attribute is applicable only to an external MC68040 writing to a non-DRAM bank.

0 = Reads and writes are the same length.
1 = Insert one additional wait state to MC68040 write cycles to SRAM banks.

EMWS—External Master Wait State (SRAM Bank Only)

This attribute should be set if an additional wait state is necessary when an asynchronous external MC68030-type device or external QUICC is accessing SRAM banks (see Table 6-11). This bit is only used if SYNC = 0.

0 = Normal operation.
1 = Insert one additional wait state for external QUICC/MC68030-type masters on their accesses to all SRAM banks.)

Table 6-11. External MC68030-Type Cycle Length (SRAM Bank in Asynchronous Operation

	External QUICC/MC68030-Type Bus Cycle Length			
	Synchronous Bus Timing (BSTM = 1)		Asynchronous Bus Timing (BSTM = 0)	
TCYC =	EMWS = 0	EMWS = 1	EMWS = 0	EMWS = 1
0	3	3	3	3
1	3	4	3	5
2	4	5	5	6
3	5	6	6	7
4	6	7	7	8
5	7	8	8	9
6	8	9	9	10
…	…	…	…	…
15	17	18	18	19

NOTE: The BSTM bit is located in the MCR of the SI60.

The following bits are used for both DRAM and SRAM memory:

SYNC—Synchronous External Access MC68030-Type

This attribute applies only to an external MC68030-type device or external QUICC that uses the on-chip memory controller. It determines how the memory controller will assert its signals in response to what it sees from the external master.

0 = Asynchronous operation of the memory controller (external MC68030-type master only).

When the SRAM controller is used, $\overline{CS}$ and $\overline{DSACK}$ assertion and negation timings are asynchronous. They are asserted and negated in relation to the external master's $\overline{AS}$ line. The CSNTQ and the TRLXQ attributes are ignored. When EMWS is set, one wait state is added to the programmed TCYC.

When the DRAM controller is used, $\overline{CAS}$ and $\overline{DSACK}$ are negated asynchronously with the negation of the external master's $\overline{AS}$.

NOTE

The DRAM controller's assertion of $\overline{RAS}$ and $\overline{CAS}$ is always synchronous to the QUICC clock. When asynchronous external masters are using the DRAM controller, the BSTM bit in the MCR should be cleared.

1 = Synchronous operation of the memory controller (external MC68030-type master only).

When the SRAM controller is used, $\overline{CS}$ and $\overline{DSACK}$ assertion and negation timings are synchronous. The CSNTQ and the TRLXQ attributes may be set as desired.

When the DRAM controller is used, $\overline{CAS}$ and $\overline{DSACK}$ are negated synchronously to the QUICC clock.

Only when the SYNC bit is set, is parity support possible for an external MC68030-type master.

Table 6-12 summarizes the effects of the various combinations of the SYNC bit in the GMR and the BSTM bit in the MCR.)

Table 6-12. SYNC-BSTM Bit Combination Summary (MC68030-Type External Master)

SYNC-BSTM	Result
00	MC68030-type master and QUICC can be asynchronous. Lowest performance, since the external AS signal is synchronized prior to being used. Parity support is not available.
01	External MC68030-type master is running synchronously with the QUICC, and the user desires to make external-to-external SRAM accesses as fast as possible. The CSNTQ and TRLXQ attributes may not be used. Does not affect DRAM performance. Parity support is not available.
10	Do not use.
11	Not as fast as case 01, but CSNTQ and TRLXQ attributes may be used. Parity support is available.

NOTES:
If Synchronous bus mode is selected, glue logic is required for external MC68030-type bus master (including MC68360) ensuring that proper set up time for address strobe assertion is met

OPAR—Odd Parity

This attribute is used to program odd or even parity. It may also be used to generate parity errors for testing purposes by writing the DRAM/SRAM with OPAR = 1 and reading the DRAM/SRAM with OPAR = 0.

0 = Even parity
1 = Odd parity

PBEE—Parity Bus Error Enable

This attribute is used to enable an internal bus error if a parity error is detected. It is applicable only when the QUICC is the bus master; if in slave mode the $\overline{PERR}$ will be asserted if the parity function is enabled but it will not cause bus error regardless of the setting of this bit. The $\overline{BERR}$ signal will be internally asserted on the memory read cycle.

0 = Disable internal bus error.
1 = Enable internal bus error.

NOTE

Using the internal bus error requires a longer data setup time for read cycles.

TSS40—$\overline{TS}$ Sample (MC68EC040)

This attribute is used to control the MC68EC040 cycles. When the MC68EC040 address to clock setup timing does not meet the memory controller decoding time, the memory

controller may sample $\overline{TS}$ with a one-clock-phase delay. This will delay the assertion of the $\overline{CS}$ or $\overline{RAS}$ in the MC68EC040 memory cycle by one clock phase. It will delay the rest of the bus cycle by one clock (effectively adding one extra clock cycle per bus cycle).

NOTE

In general, the user determines whether this bit must be set before to selecting the WBT40 and TCYC bits.

0 = Do not sample $\overline{TS}$.
1 = Sample $\overline{TS}$ prior to using it.

NCS—No CPU Space

This attribute specifies whether the $\overline{CS}$/$\overline{RAS}$ signal will assert on a CPU space access cycle. If both supervisor data and program accesses are desired, while ignoring CPU space accesses, then this bit should be set. (Note that an interrupt acknowledge cycle is a CPU space access, but a user or supervisor read/write cycle is not.) A CPU space access has the function code value 0111.

0 = Assert $\overline{CS}$/$\overline{RAS}$ on CPU space accesses (default).
1 = Suppress $\overline{CS}$/$\overline{RAS}$ on CPU space accesses.

NOTE

In default state, user should program the FC3-FC0 in both the Option Registers and Base Registers so that CS/RAS will not get asserted in an undesirable address range.

GAMX—Global Address Mux Enable

This attribute determines whether the QUICC will provide internal address multiplexing for DRAM banks. If not, the address multiplexing must be provided externally, with the QUICC's AMUX pin being used to control the multiplexers. AMUX is high to signify the row, low to signify the column address, and then negated (high) at the end of the DRAM bus cycle.

There are two situations in which the user may wish to provide address multiplexing externally. First, external multiplexers are required when an external master exists in the system and that external master needs to access the DRAM. Second, using external address multiplexing causes the clock to address valid timing as slightly accelerated, which may be beneficial in certain high-performance situations.

0 = Disable internal address multiplexing for all DRAM banks.
1 = Enable internal address multiplexing for all DRAM banks.

Bits 4–0—Reserved

6.13.2 Memory Controller Status Register (MSTAT)

The MSTAT register reports memory controller error information to the user. These bits are set, regardless of whether an internal or external master originated the cycle. Bits are reset by writing a one to that bit; writing a zero has no effect. The register may be read at any time and is cleared by reset. No interrupts are generated from this register; however, an internal

master may generate a bus error as a result of this register, and for parity errors, the $\overline{\text{PERR}}$ pin may be externally connected to an interrupt input.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	WPER	PER7	PER6	PER5	PER4	PER3	PER2	PER1	PER0

Bits 15–9—Reserved

WPER—Write Protect Error

This bit is asserted when a write protect error occurs. A bus monitor ($\overline{\text{BERR}}$ assertion) will (if enabled) prompt the user to read this register if no $\overline{\text{DSACK}}$ is provided on a write cycle. The accessed address will be in the $\overline{\text{BERR}}$ exception descriptor. WPER is cleared by writing one to this bit or by performing a system reset. Writing a zero has no effect on WPER.

PERx—Parity Error

These bits indicate that a parity error was detected when reading from bank N. $\overline{\text{BERR}}$ is internally asserted if PBEE in the GMR is set and if an internal master performs this cycle. The $\overline{\text{PERR}}$ signal is continuously asserted until all PERx bits are cleared. PERx is cleared by writing one or by performing a system reset. Writing a zero has no effect on PERx.

NOTE

If external masters of the MC68030-type (including QUICCs) are chosen to be asynchronous (configured by clearing the SYNC bit in the GMR), then they have no parity support.

6.13.3 Base Register (BR)

This register is used for both DRAM and SRAM banks. Most bits are valid for both the DRAM and SRAM banks, but some bits are only valid for SRAM banks. This register is a 32-bit read-write register that may be accessed at any time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA31	BA30	BA29	BA28	BA27	BA26	BA25	BA24	BA23	BA22	BA21	BA20	BA19	BA18	BA17	BA16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA15	BA14	BA13	BA12	BA11	FC3	FC2	FC1	FC0	TRLXQ	BACK40	CSNT40	CSNTQ	PAREN	WP	V
0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0

V—Valid Bit

This bit indicates that the contents of the BR and OR pair are valid. The $\overline{\text{CS}}$/$\overline{\text{RAS}}$ signal will not assert until the V-bit is set.

NOTE

An access to a region that has no V-bit set may cause a bus monitor timeout.

0 = This DRAM/SRAM bank is invalid.
1 = This DRAM/SRAM bank is valid.

NOTE

Following a system reset, the V-bit is set in BR0 if the global chip select is enabled. See the CONFIG pins for more details.

WP—Write Protection

This bit can restrict write accesses within the address range of a BR. An attempt to write to the range of addresses specified in a BR that has this bit set can cause the $\overline{\text{BERR}}$ signal to be asserted by the bus monitor logic (if enabled), causing termination of this cycle.

0 = Both read and write accesses are allowed.
1 = Only read accesses are allowed. The $\overline{\text{RAS}}$/$\overline{\text{CS}}$ signal, $\overline{\text{TA}}$, and $\overline{\text{DSACK}}$ will not be asserted by the QUICC on write cycles to this memory bank. WPER will be set in the MSTAT register if a write to this memory bank is attempted.

PAREN—Parity Checking Enable

This bit is used to enable checking of parity on either an SRAM or DRAM bank.

0 = Parity checking is disabled.
1 = Parity checking is enabled.

NOTE

Parity checking is not possible for asynchronous external masters.

CSNTQ—$\overline{\text{CS}}$ Negate Timing QUICC (SRAM Bank Only)

This bit is used to determine when $\overline{\text{CS}}$ is negated during an internal QUICC or external QUICC/MC68030-type bus master write cycle. This is helpful to meet address/data hold time requirements for slow memories and peripherals (see Figure 6-13 and Figure 6-14).

0 = $\overline{\text{CS}}$ is negated normally (as late as possible).
1 = $\overline{\text{CS}}$ is negated one phase earlier, but the cycle length is not affected.

NOTE

CSNTQ is ignored for an SRAM cycle by an external master if the SYNC bit is cleared. CSNTQ = 1 is not valid for external DSACK assertion

CSNT40—$\overline{\text{CS}}$ Negate Timing MC68EC040 (SRAM Bank Only)

This bit is used to determine when $\overline{\text{CS}}$ is negated during an MC68EC040 write cycle. This is helpful to meet address/data hold time requirements (see Figure 6-15).

0 = $\overline{\text{CS}}$ is negated normally (as late as possible).
1 = $\overline{\text{CS}}$ is negated one phase earlier, but the cycle length is not affected.

NOTE

CSNT40 is ignored for an SRAM cycle by an external master if the SYNC bit is cleared. CSNT40 = 1 is not valid for external DSACK assertion

BACK40—Burst Acknowledge MC68EC040

This bit is used to acknowledge a burst cycle to the MC68040. If set, bursts are enabled in this bank. The QUICC generates address lines 2,3 on the BADDR3–BADDR2 pins.

0 = Do not acknowledge burst.
1 = Acknowledge burst; MC68040 bursts are handled by the memory controller for this bank.

TRLXQ—Timing Relax

This bit delays the beginning of the internal QUICC or external QUICC/MC68030-type bus master cycle to relax the timing constraints on the user. This attribute is useful for slow peripherals that require additional address setup time. Chip selects are delayed by one phase, and the cycle is delayed by one clock. For accesses to DRAM, $\overline{RAS}$ is delayed by one phase, and $\overline{CAS}$ and AMUX are delayed by two phases, giving a total cycle increase of one clock. See Figures 6-16 and 6-17 for timing diagrams of different cases.

0 = Do not relax timing.
1 = Relax timing at the beginning of the cycle. One additional clock cycle is added when this bit is set.

NOTE

TRLXQ is ignored for an SRAM cycle by an external master if the SYNC bit is cleared.

To relax the MC68EC040 cycles, use the TSS40 bit in the GMR.

User should avoid setting both TRLXQ and CSNTQ = 1, when TCYC = 0. This bit combination will result in a bus cycle without CS assertion.

FC3–FC0—Function Code

This field can be used to specify that accesses with the memory bank be limited to a certain address space type. These bits are used in conjunction with the FCM3–FCM0 bits in the OR.

BA31–BA11—Base Address

The base address field, the upper 21 bits of each BR, and the function code field are compared to the address on the address bus to determine if a DRAM/SRAM region is being accessed by an internal QUICC master.

If the SRAM/DRAM region is being accessed by an external master and the $\overline{WE}$ lines are not used, then A31–A28 address lines and the BA31–BA28 bits are also used in the comparison. If, however, the SRAM/DRAM region is being accessed by an external master

and the A31–A28 lines are configured as $\overline{WE}$ lines, then the user should write zeros to the BA31–BA28 bits so that A31–A28 will be masked by the address comparison logic.

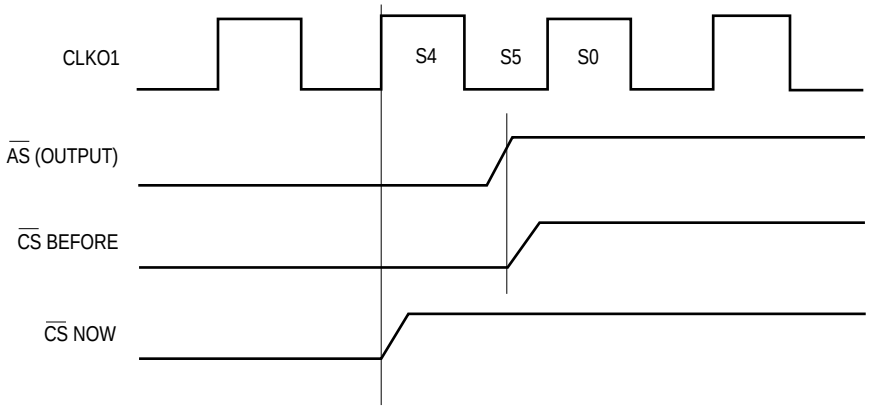


Figure 6-13. CSNTQ = 1 During an Internal Cycle

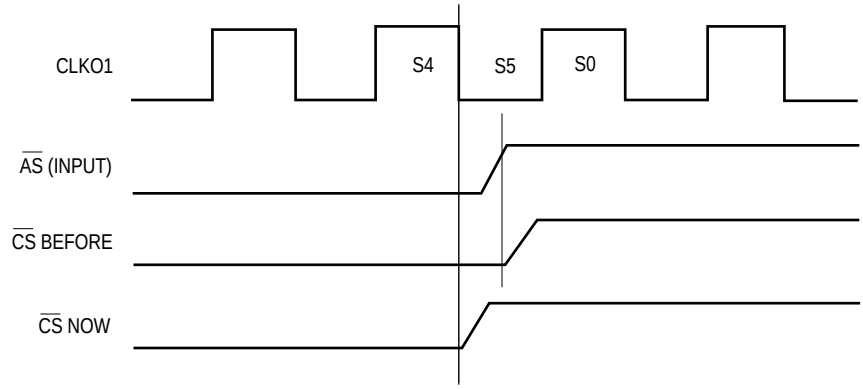


Figure 6-14. CSNTQ = 1 During an External QUICC/MC68EC030 Cycle

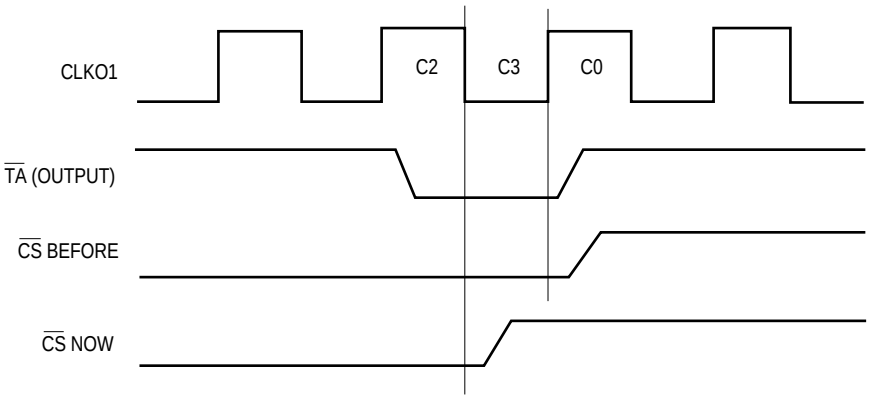


Figure 6-15. CSNT40 = 1 During an External MC68EC040 Cycle

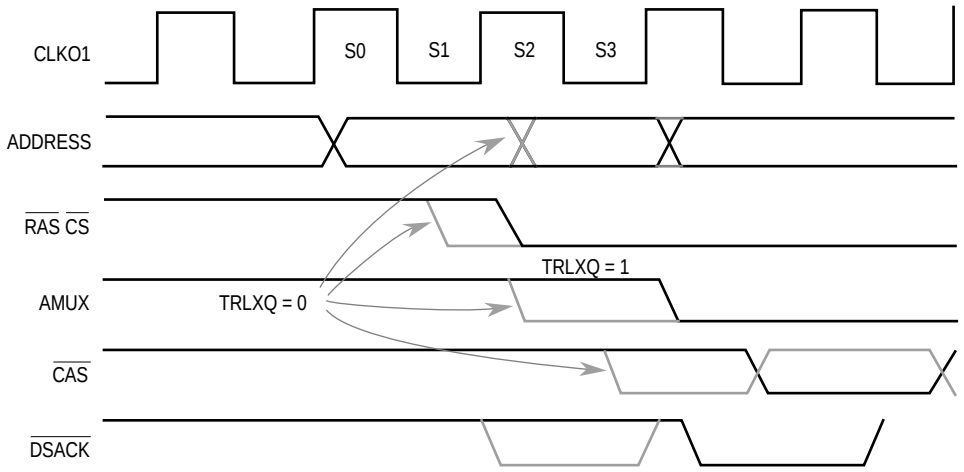


Figure 6-16. TRLXQ = 1 During an Internal Cycle

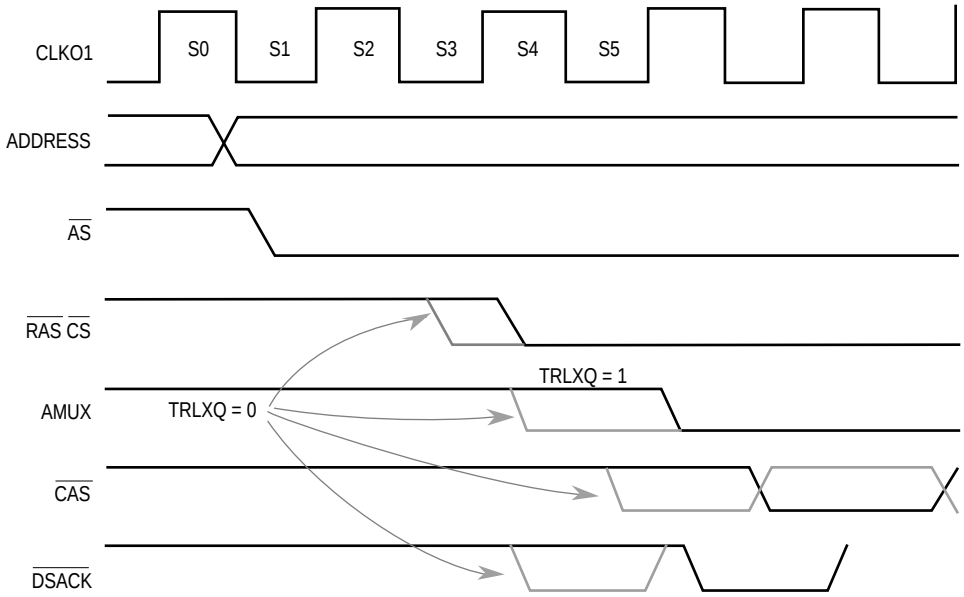


Figure 6-17. TRLXQ = 1 During an External QUICC/MC68030 Cycle

6.13.4 Option Register (OR)

This register is used for both DRAM and SRAM banks. Most bits are valid for both banks, but some bits are only valid for DRAM banks, and others are only valid for SRAM banks. This register is a 32-bit read-write register that may be accessed at any time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCYC3	TCYC2	TCYC1	TCYC0	AM27	AM26	AM25	AM24	AM23	AM22	AM21	AM20	AM19	AM18	AM17	AM16
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AM15	AM14	AM13	AM12	AM11	FCM3	FCM2	FCM1	FCM0	BCYC1	BCYC0	—	PGME	SPS1	SPS0	DSSEL
0	0	0	0	0	0	0	0	0	0	0	0	0	0/1	0/1	0

DSSEL—Dynamic RAM Select

This bit determines if the bank is a DRAM or SRAM, which impacts a number of signals: 1) the length of the cycle is different; 2) address muxing is performed if GAMX = 1; and 3) the previous $\overline{RAS}$ is negated if a page bank miss occurs and DSSEL = 1 (for the new bank).

0 = SRAM bank (i.e., SRAM, EPROM, peripherals, etc.)
1 = DRAM bank

SPS1–SPS0—SRAM Port Size (SRAM Bank Only)

This attribute determines whether a given chip select responds with $\overline{DSACKx}$ and, if so, what port size is returned (see Table 6-13).

If the cycle is terminated by using the internal wait-state attributes, the QUICC drives the $\overline{DSACKx}$ lines according to those bits. If the internal wait-state attributes are not used, the cycle should be terminated with external $\overline{DSACKx}$. In this case, the QUICC does not drive the $\overline{DSACKx}$ lines, but rather samples them at every falling edge of the clock.

If an MC68EC040 access is performed using this SRAM bank and SPS= 00, 01, or 10, the SRAM controller operates in the same way, except it asserts $\overline{TA}$ instead of $\overline{DSACKx}$. If SPS= 11, $\overline{TA}$ is sampled at every rising edge of the clock.

Table 6-13. SRAM Port Size

SPS1–SPS0	Result
00	32-Bit Port Size
01	16-Bit Port Size
10	8-Bit Port Size
11	External $\overline{DSACKx}$ Response

NOTES

If DSACK is provided internally, then the $\overline{DSACKx}$ lines are still sampled externally, and can be asserted externally to end the cycle. However, in this case of external $\overline{DSACKx}$ assertion, external $\overline{DSACKx}$ should be asserted and negated prior to when internal DSACK would have been asserted by the QUICC. This is easily accomplished on the boot chip select since the QUICC default value is 14 wait states.

The SRAM controller does not support an external $\overline{TA}$ response for MC68040 burst mode. Also, for non-burst MC68040 cycles, $\overline{TA}$ cannot be externally asserted before $\overline{CS}$ is asserted.

PGME—Page Mode Enabled (DRAM Banks Only)

This bit is used to enable page mode accesses to a DRAM bank. Page mode accesses are performed only for an internal QUICC or an external QUICC/MC68030-type master.

0 = Page mode is disabled.
1 = Page mode is enabled.

NOTE

When the DRAM controller supports MC68EC040 cycles, PGME must be cleared by the user, or erratic behavior may occur.

Bit 4—Reserved

BCYC1–BCYC0—Burst Length Cycle in Clocks

These bits determine the number of wait states inserted in an MC68040 burst cycle. This attribute is for the second, third, and fourth access of the burst cycle. Program TCYC3–TCYC0 for the first access.

00 = The burst cycles are 1 clock in length (x,1,1,1).
01 = The burst cycles are 2 clocks in length (x,2,2,2).
10 = The burst cycles are 3 clocks in length (x,3,3,3).
11 = The burst cycles are 4 clocks in length (x,4,4,4).

FCM0–FCM3—Function Code Mask

This field can be used to mask certain function code bits, allowing more than one address space type to be assigned to a chip select. Any set bit causes the corresponding function code pin to be used as part of the address comparison. Any cleared bit masks the corresponding function code bit. If both supervisor data and program accesses are desired, while ignoring CPU space accesses, then the NCS bit in the GMR should be set.

NOTE

Clear the FCM bits to ignore function codes as part of the address comparison.

Regardless of the setting in this register, an external encoding of X111 of the function code pins will be taken as a CPU space access.

AM27–AM11—Address Mask

The address mask field, bits 27–11 of each OR, provides for masking any of the corresponding bits in the associated BR. By masking the address bits independently, external devices of different address range sizes can be used. Any cleared 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. This field can be read or written at any time.

NOTES

When address lines A31–A28 are multiplexed with the $\overline{WE}$ lines, the A31–A28 lines are still used in the comparison by an internal QUICC master. See the base address bit description in 6.13.3 Base Register (BR).

If two chip selects are programmed to assert in the same address region, only the lower chip select (or $\overline{RAS}$ line) will assert. For example, $\overline{CS1}$ has priority over $\overline{CS4}$.

TCYC3–TCYC0—Cycle Length in Clocks

This field determines the length of a bus cycle (see Table 6-14). Both internal masters and external masters use this field for their accesses to a given memory bank. In addition, an external MC68040 uses this field for the first access of a burst access sequence.

Although TCYC3–TCYC0 is the main parameter for determining cycle length since it selects the number of wait states inserted in the cycle, the total cycle length may vary for other reasons, such as a DRAM page hit, DRAM page miss, or whether the bus master is internal or external to the QUICC. Besides TCYC, other bits that can affect the total cycle length in certain situations are WBT40, WBTQ, DWQ, DW40, EMWS, SYNC, and TSS40 in the GMR, and TRLXQ, PGME, and BCYC in the OR. CSNTQ and CSNT40 affect the $\overline{CS}$ timing, but do not affect the total cycle length.

If the user has selected an external $\overline{DSACKx}$ or $\overline{TA}$ response for this memory bank, with the SPS or DPS bits, then TCYC3–TCYC0 are not used.

Table 6-14. Cycle Length in Clocks

	Internal QUICC Master Memory Bus Cycle Length				
	Number of Clocks		**Number of Wait States (SRAM)**		**Number of Wait States (DRAM)**
TCYC =	**Number**	**Comments**	**Number**	**Comments**	**Numbers**
0	2	Fast Termination	*	Undefined	3
1	3	Normal	0		4
2	4		1		5
3	5		2		6
4	6		3		7
5	7		4		8
6	8		5		9
…	…		…		…
15	17		14		18

NOTES

External cycles are always three clocks or longer. SeeTable 6-11 for more details.

Normal DRAM cycles are three clocks when TCYC=0 and four clocks when TCYC=1, etc. Therefore fast termination is not possible during the initial access to DRAM. Two clock DRAM cycles are only possible when page mode is enabled for an internal master.

If an external $\overline{DSACK}$ response is selected with either DPS in the GMR or SPS in the OR, TCYC should not be set to zero.

6.13.5 DRAM-SRAM Performance Summary;

Table 6-15 lists the performance results possible when setting TCYC = 0, assuming a 25-MHz system clock, 60-ns DRAMs, and 15-ns SRAMs. The items marked with a dash are not applicable to the situation.)

Table 6-15. Maximum DRAM/SRAM Performance (25 MHz

Memory Move	Internal Master	External QUICC/ MC68030 Type	External MC68040 (TSS40 = 0)	External MC68EC040 (TSS40 = 1)
SRAM Normal	2 (See note 2)	3	2	3
Burst	—	—	2, 1, 1,1	3,1,1,1
DRAM Normal	3*	4	3*	4
DRAM Normal Back-to-Back Access	4*	4	4*	5
Page Hit	2	3	—	—
Page Miss	4*	5	—	—
Line Fill	—	—	3*, 2, 2, 2	4,2,2,2

NOTES:
1.For 70-ns DRAMs, items marked with an asterisk should have one additional clock added.
2.For the internal master case, the QUICC also supports two-clock accesses with 20-ns SRAMs.

SECTION 7
COMMUNICATION PROCESSOR MODULE (CPM)

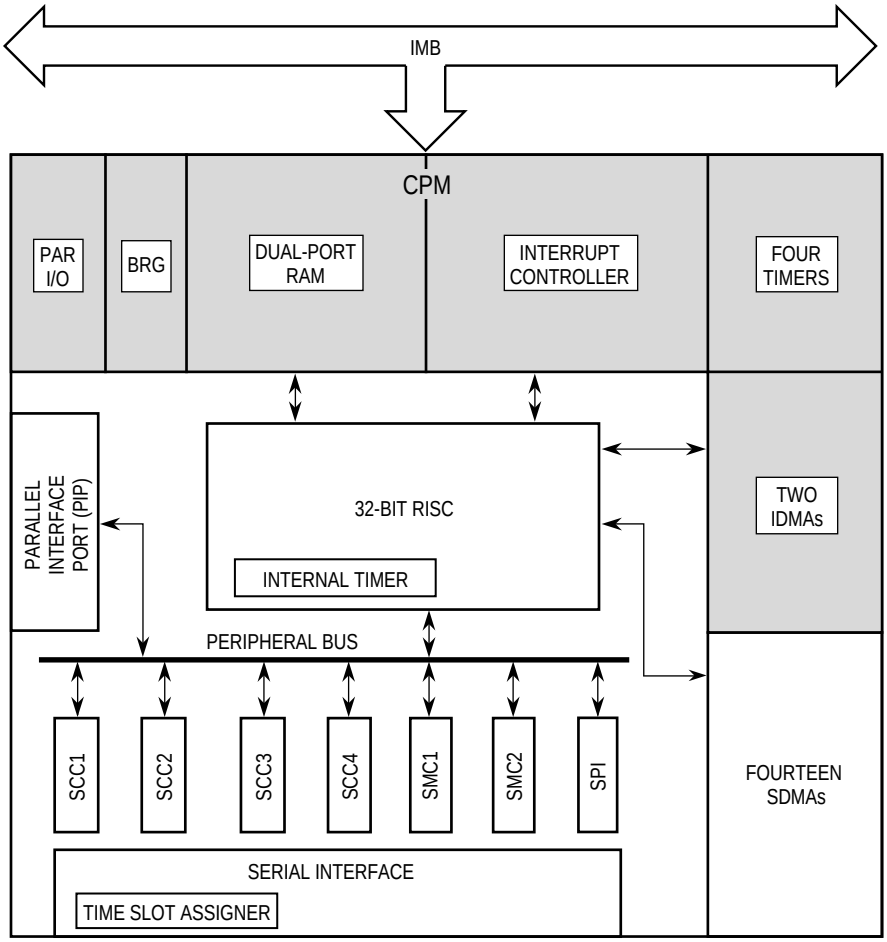
The CPM includes many blocks that work together to allow an extremely flexible and integrated approach to solving many communications problems. The CPM (see Figure 7-1) includes the following modules:

- RISC Controller
- Four Full-Duplex Serial Communication Controllers (SCCs) Support the Following Protocols:

  —IEEE 802.3/Ethernet (Optional Feature on SCC)
  —High-Level/Synchronous Data Link Control (HDLC/SDLC)
  —HDLC Bus (Multidrop Bus Configuration of HDLC)
  —AppleTalk (HDLC-Based Local Area Network (LAN) Protocol)
  —Universal Asynchronous Receiver Transmitter (UART)
  —Synchronous UART (Isochronous, 1x Clock Mode)
  —Binary Synchronous Communication (BISYNC)
  —Totally Transparent Operation
  —Signaling System #7 (HDLC-Based Protocol. RAM Microcode Option Only)
  — Profibus (RAM Microcode Option Only)
  —Asynchronous HDLC (RAM Microcode Option Only)
  —Multiple Chanel GCI (RAM Microcode Option Only)
  —ATM Framing (RAM Microcode Option Only)
  —Enhanced Ethernet Filtering (RAM Microcode Option Only)
- Four Independent Baud Rate Generators
- Two Serial Management Controllers (SMCs) Provide Additional UART and Totally Transparent Functionality or Support the GCI Channel 0 and 1 Monitor and C/I Channels in Integrated Services Digital Network (ISDN)
- Serial Interface Provides Nonmultiplexed Serial Interface (NMSI) for the Four SCCs (includes TXD, RXD, TCLK, RCLK, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ pins)
- Time Slot Assigner (TSA) Supports Multiplexing of Data from any of the Four SCCs and Two SMCs onto Two Time-Division Multiplexed (TDM) Interfaces. The TSA Supports the Following TDM Formats:

  —T1/CEPT Lines
  —Pulse Code Modulation (PCM) Highway Interface
  —ISDN Primary Rate
  —Motorola Interchip Digital Link (IDL)
  —General Circuit Interface (GCI), also known as IOM-2
  —User-Defined Interfaces

- Serial Peripheral Interface (SPI) for Synchronous Interchip Communication
- Fourteen Serial Direct Memory Access (SDMA) Channels Support the SCC, SMCs, and SPI
- Two Independent Direct Memory Access (IDMA) Channels Support External Memory and Peripherals
- A Command Set Register Supports the RISC, IDMA, SCCs, SMCs, and SPI
- Four General-Purpose 16-Bit Timers or Two 32-Bit Timers
- Internal Timers to Implement Up to 16 Additional Timers
- General-Purpose Parallel Port for Parallel Protocols such as Centronics (Can Also Be Used as Standard Parallel I/O)
- CPM Interrupt Controller
- 2.5-kbyte Dual-Port RAM
- Twelve Parallel I/O Lines with Interrupt Capability



NOTE: The term "CP" refers to the nonshaded portion of the CPM.

Figure 7-1. CPM Block Diagram

7.1 RISC CONTROLLER

The RISC controller is the 32-bit central controller of the communication processor module (CPM). Since its execution occurs on a separate bus that is hidden from the user, it does not impact CPU32+ core performance. The RISC controller works with the serial channels and parallel interface port (PIP) to implement the user-chosen protocols and to manage the SDMA channels that transfer data between the SCCs and memory. The RISC controller contains an internal timer that can be used to implement up to 16 additional timers for the user application software. These features are collectively known as the communication processor (CP), which is a subset of the overall CPM. Additionally, the RISC controller can manage the operation of the IDMA channels, if desired. The 32-bit RISC handles the lower layer tasks and DMA control activities, leaving the 32-bit CPU32+ core (or other external processor) free to handle higher layer activities. Thus, the QUICC can be thought of as a dual 32-bit processor system.

The RISC controller communicates with the host (CPU32+ core or other external processor) in several ways. First, many parameters are exchanged through the dual-port RAM. In the case of simultaneous accesses (at least one of which is a write operation), the RISC controller may be delayed by one clock in its access to the dual-port RAM. The host is never delayed. Second, the RISC controller can execute special commands issued by the host. These commands are only required to be issued in special situations. Third, the RISC controller can generate interrupts through the CPM interrupt controller. Fourth, status/event registers, which show events that have occurred within the RISC, may be read at any time by the CPU32+ or an external processor.

The RISC controller has the ability to control a set of up to 16 timers. These timers are separate and distinct from the four general-purpose timers and baud rate generators in the CPM. The 16 timers are ideally used in protocols that do not require extreme precision, but in which it is desirable to off-load the host CPU from having to scan the timer tables that are created in software. These timers are clocked from an internal timer used only by the RISC controller.

The RISC controller uses the peripheral bus to communicate with all of its peripherals. Each SCC has a separate receive and transmit FIFO. The SCC1 FIFOs are 32-bytes each; the other SCC FIFOs are 16-bytes each. The SMC and SPI FIFO sizes are double-buffered. The PIP is a single register interface.

The following priority scheme determines the processing priority of the RISC controller. It is as follows:

1. Reset in CP Command Register or System Reset
2. DMA Bus Error
3. Commands Issued to the CP Command Register
4. CC1 Rx
5. SCC1 Tx
6. SCC2 Rx
7. SCC2 Tx

8. CC3 Rx
9. SCC3 Tx
10. SCC4 Rx
11. SCC4 Tx
12. SMC1 Rx
13. SMC1 Tx
14. SMC2 Rx
15. SMC2 Tx
16. SPI Rx
17. SPI Tx
18. PIP
19. RISC Timer Tables

The RISC controller has an option to execute microcode from a portion of user RAM, located in the on-chip dual-port RAM. In this mode, either 512 bytes or 1024 bytes of the user RAM cannot be accessed by the host or another bus master and are used exclusively by the RISC. In this mode, the RISC controller can fetch instructions from both the dual-port RAM and its private ROM. This mode allows Motorola to add new protocols or enhancements to the QUICC in the form of Motorola-supplied RAM microcodes. The binary microcode is obtained from Motorola and then loaded by the user into the dual-port RAM.

The RISC controller contains one configuration register described in the following paragraph.

7.1.1 RISC Controller Configuration Register (RCCR)

The 16-bit, memory-mapped, read-write RCCR is used to configure the RISC processor and controls the RISC internal timer. This register is initialized to zero at reset. Bits 0-7 should not be modified unless the user is downloading a Motorola-supplied RAM microcode package..

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIME	—	TIMEP						RESERVED							

TIME—Timer Enable

This bit enables the RISC controller internal timer. The timer will generate a tick to the RISC based on the value programmed into the TIMEP bit. TIME may be modified at any time to start or stop the scanning of the RISC timer tables.

Bit 14—Reserved

TIMEP—Timer Period

This field controls the RISC controller timer tick. The RISC timer tables are scanned on each timer tick. The input to this timer tick generator is the general system clock divided by 1024. The formula is (TIMEP + 1) $\times$ 1024 = (general system clock period). Thus, a val-

ue of 0 stored in these bits gives a timer tick of 1 × (1024) = 1024 general system clocks. A value of 63 (decimal) stored in these bits gives a timer tick of 64 × (1024) = 65536 general system clocks.

Bits 7-0—Reserved - set to zero.

7.1.2 RISC Microcode Revision Number

The RISC controller writes a revision number stored in its ROM to a dual-port RAM location called REV_num. REV_num is located in the miscellaneous parameter RAM. The other locations are reserved for future use. The microcode rivision number only reflect the revision of the micro code. It dose not always refrect the MASK number.

Address	Name	Width	Description
Misc Base + 00	REV_num	Word	Microcode Revision Number
Misc Base + 02	RES	Word	Reserved
Misc Base + 04	RES	Long	Reserved
Misc Base + 08	RES	Long	Reserved

7.2 COMMAND SET

The host processor (CPU32+ or other external processor) issues commands to the RISC by writing to the command register (CR). The CR only needs to be accessed on rare occasions. For instance, to terminate the transmission of a frame by an SCC without waiting until the end of the frame, a STOP TX command can be issued to an SCC through the command register. The commands are described in general terms in the following paragraphs; they are described in specific terms when the protocol or feature is described in detail.

The host should set the FLG bit in the CR when it issues commands. The CP clears FLG after completing the command to indicate to the host that it is ready for the next command. Subsequent commands to the CR may be given only after FLG is cleared. The software reset command (issued by setting the RST bit) may be given regardless of the state of FLG, but the host should still set FLG when setting RST.

The CR, a 16-bit, memory-mapped, read-write register, is cleared by reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RST	—			OPCODE				CH NUM				—			FLG

RST—Software Reset Command

This bit is set by the host and cleared by the CP. On execution of this command, the RST bit and the FLG bit are cleared within two general system clocks. The RISC reset routine is approximately 60 clocks long, but the user can begin initialization of the CP immediately after this command is given. This command is useful when the host wants to reset the registers and parameters for all the channels (SCCs, SMCs, SPI, and PIP) as well as the RISC processor and RISC timer tables. This command does not affect the serial interface (SI) or the parallel I/O registers.

Bits 14–12, 3–1—Reserved

OPCODE—Operation Code

The opcodes are listed in Table 7-1.

Table 7-1. Opcodes

Opcode	SCC	SMC (UART/Trans)	SMC (GCI)	SPI	IDMA	Timer
0000	INIT RX & TX PARAMS	INIT RX & TX PARAMS	INIT RX & TX PARAMS	INIT RX & TX PARAMS		
0001	INIT RX PARAMS	INIT RX PARAMS		INIT RX PARAMS		
0010	INIT TX PARAMS	INIT TX PARAMS		INIT TX PARAMS		
0011	ENTER HUNT MODE	ENTER HUNT MODE				
0100	STOP TX[1]	STOP TX				
0101	GR STOP TX[2]				INIT IDMA	
0110	RESTART TX	RESTART TX				
0111	CLOSE RX BD	CLOSE RX BD		CLOSE RX BD		
1000	SET GROUP ADDR					SET TIMER
1001			GCI TIMEOUT			
1010	RESET BCS		GCI ABORT REQ			
1011						
1100	U	U	U	U	U	U
1101	U	U	U	U	U	U
1110	U	U	U	U	U	U
1111	U	U	U	U	U	U

NOTES:
1.STOP TX = MC68302 original STOP TRANSMIT command.
2.GR STOP TX = GRACEFUL STOP TRANSMIT command.

INIT TX and RX PARAMETERS. This command 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 serial channel.

INIT RX PARAMETERS. This command initializes the receive parameters of the serial channel.

INIT TX PARAMETERS. This command initializes the transmit parameters of the serial channel.

ENTER HUNT MODE. This command 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. This command 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 will proceed when the RESTART command is issued.

GRACEFUL STOP TX. This command stops the transmission from this channel as soon as the current frame has been fully transmitted from the transmit FIFO. Transmission will proceed once the RESTART command is issued and the R-bit is set in the next transmit buffer descriptor.

RESTART TX. When the STOP TX command has been issued, this command can be used to restart the transmission at the current buffer descriptor.

CLOSE RX BD. This command causes the receiver to simply close the current receive buffer descriptor, making the receive buffer immediately available for manipulation by the user. Reception continues normally using the next available buffer descriptor. This command may be used to access the data buffer without waiting until the data buffer is completely filled by the SCCμSET TIMER. This command activates, deactivates, or reconfigures one of the 16 timers in the RISC timer table.

SET GROUP ADDRESS. This command sets a bit in the hash table for the Ethernet logical group address recognition function.

GCI ABORT REQUEST. The GCI receiver sends an abort request on the E-bit.

GCI TIMEOUT. The GCI performs the timeout function.

RESET BCS. This command is used in BISYNC mode to reset the block check sequence calculation.

Undefined (U). Reserved for use by Motorola-supplied RAM microcodes.

CH NUM—Channel Number

These bits are set by the host to define the specific sub-block on which the command is to operate. Some sub-blocks share channel number encodings if their commands are mutually exclusive.

0000	SCC1
0001	
0010	
0011	
0100	SCC2
0101	SPI/RISC Timers
0110	
0111	
1000	SCC3
1001	SMC1/IDMA1
1010	
1011	
1100	SCC4
1101	SMC2/IDMA2
1110	
1111	

FLG—Command Semaphore Flag

The bit is set by the host and cleared by the CP.

0 = The CP is ready to receive a new command.
1 = The CR contains a command that the CP is currently processing. The CP clears this bit at the end of the command execution or after reset.

7.2.1 Command Register Examples

To perform a complete reset of the CP, the value $8001 should be written to the CR. Following this command, the CR will return the value $0000 in two clocks.

To execute an ENTER HUNT MODE command to SCC3, the value $0381 should be written to the CR. While the command is executing, the CR will return the value $0381. When the command has been completely executed, the CR will return the value $0380.

7.2.2 Command Execution Latency

The worst-case command execution latency is 120 clocks. The typical command execution latency is about 40 clocks.

7.3 DUAL-PORT RAM

The CPM has 2560 bytes of static RAM configured as dual-port memory. The dual-port RAM memory map is shown in Figure 7-2, and a block diagram is shown in Figure 7-3.

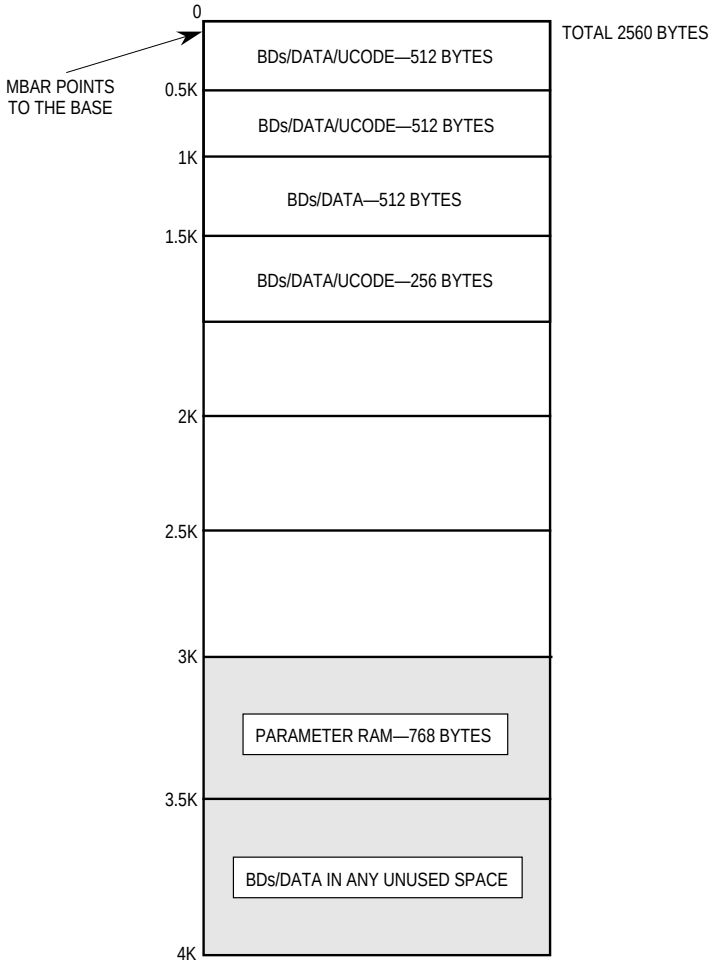


Figure 7-2. Dual-Port RAM Memory Map

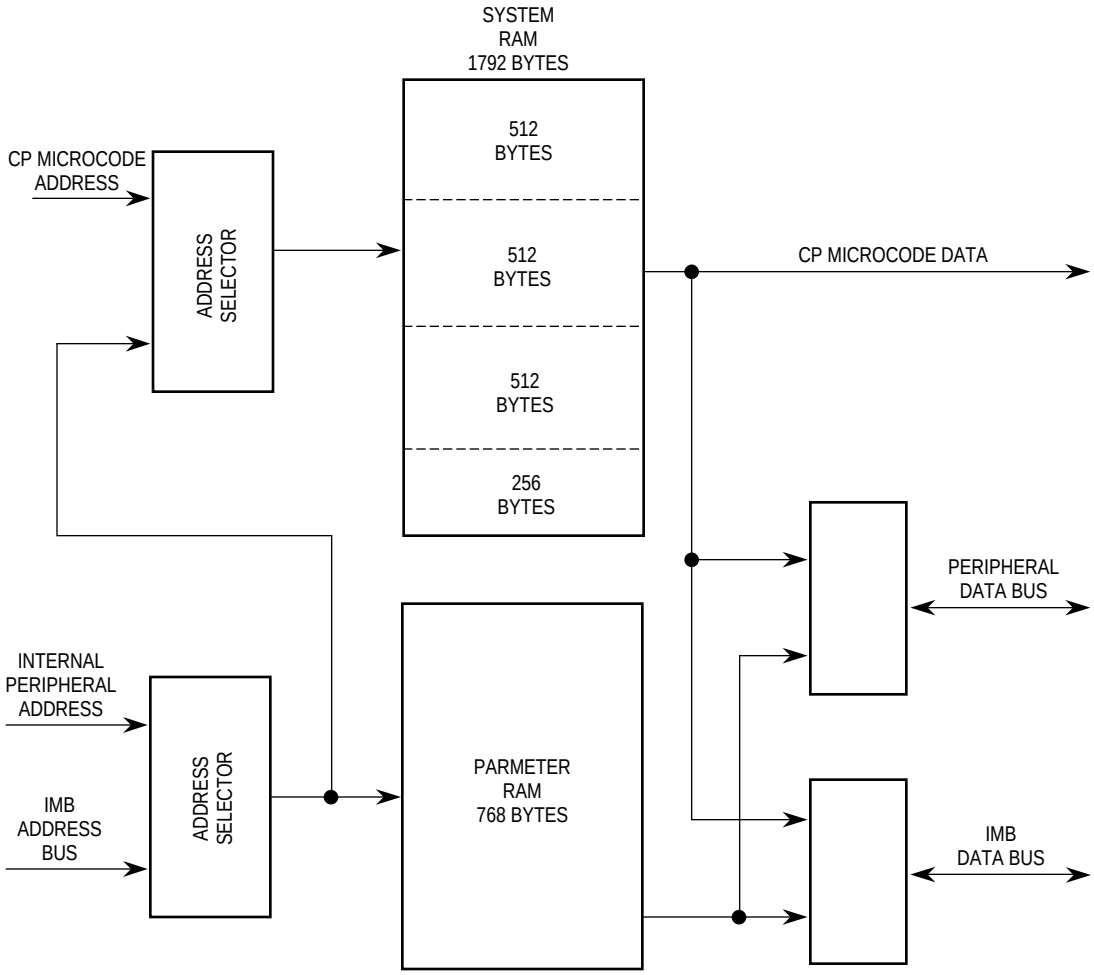


Figure 7-3. Dual-Port RAM Block Diagram

The dual-port RAM can be accessed by the RISC or one of four bus masters: CPU32+ core, IDMAs, SDMAs, or external bus master. When the dual-port RAM is accessed by an external bus master, CPU32+ core, IDMA, or SDMA channel, it is accessed in three clocks. When the dual-port RAM is accessed by the RISC, it is accessed in one clock. In the case of simultaneous access (with at least one write operation), the RISC is delayed by one clock.

When the dual-port RAM is accessed by the CPU32+ core, IDMAs, SDMAs, or external bus master, the data and address are taken from the IMB. The data is then presented on the IMB data bus. The RISC has access to the entire dual-port RAM for data fetches and portions of the system RAM for microcode instruction fetches.

The dual-port RAM is used for five possible tasks; any two tasks can occur simultaneously. The first use is to store parameters associated with the SCCs, SMCs, SPI, and IDMAs in the 768-byte parameter RAM. The second use is to store the buffer descriptors that describe where data is to be received and transmitted from. The third use is to store data from the serial channels. This usage is optional since data may also be stored externally in the system memory. The fourth use is to store RAM microcode for the RISC processor. This feature allows additional protocols to be added by Motorola in the future. The fifth use is for additional scratchpad RAM space for the user program.

Only the parameters in the parameter RAM and the microcode RAM option require fixed addresses to be used. The buffer descriptors, buffer data, and scratchpad RAM may be located in the internal system RAM or in any unused parameter RAM (for instance, in the available area when a serial channel or sub-block is not being used).

When a microcode from RAM is executed, certain portions of the system RAM are no longer available. This includes either the first 512-byte block and the last 256-byte block for a small RAM microcode, and the first two 512-byte blocks and the last 256-byte block for a large RAM microcode. The third 512-byte block is always available as system RAM.

7.3.1 Buffer Descriptors

The SCCs, SMCs, SPI always use buffer descriptors for controlling data buffers. The buffer descriptor format of the SCCs, SMCs, and SPI is identical. The buffer descriptor format for these channels is shown in the following illustration.

	15 0
OFFSET + 0	STATUS AND CONTROL
OFFSET + 2	DATA LENGTH
OFFSET + 4	HIGH-ORDER DATA BUFFER POINTER
OFFSET + 6	LOW-ORDER DATA BUFFER POINTER

If the IDMA is used in the buffer chaining or auto buffer mode, the IDMA channel also uses buffer descriptors. The buffer descriptors for the IDMA are described in 7.6.1 IDMA Key Features;.

7.3.2 Parameter RAM

The CP maintains a section of dual-port RAM called the parameter RAM. This RAM contains many parameters for the operation of the SCCs, SMCs, SPI, and the IDMA channels. An overview of the parameter RAM structure is shown in Figure 7-4. The exact definition of the parameter RAM is contained in each subsection describing a device that uses a parameter RAM.

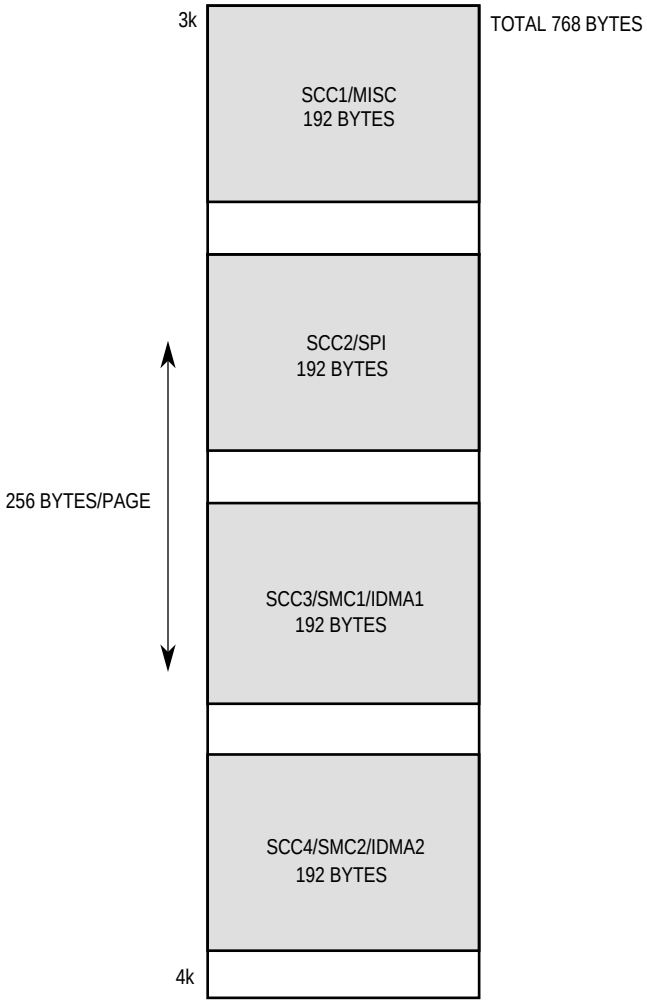


Figure 7-4. Parameter RAM Overview

7.4 RISC TIMER TABLES

The RISC controller has the ability to control up to 16 timers. These timers are separate from the four general-purpose timers and baud rate generators in the CPM. The 16 timers are ideally used in protocols that do not require extreme precision, but in which it is desirable to off-load the host CPU from having to scan the timer tables that are created in software. These timers are clocked from an internal timer used only by the RISC.

The features of the RISC timer tables are as follows:

- Up to 16 Timers Supported
- Two Timer Modes: One-Shot and Restart
- Maskable Interrupt on Timer Expiration
- Programmable Timer Resolution As Low As 41 μs at 25 MHz
- Maximum Timeout Period of 172 Sec at 25 MHz

- Continuously Updated Reference Counter

All operations on the RISC timer tables are based on a fundamental "tick" of the RISC internal timer, which is programmed in the RISC RCCR. The tick is a multiple of 1024 general system clocks. (See 7.1 RISC Controller for more details.)

The RISC timer tables have the lowest priority of all RISC operations. Therefore, if the RISC is so busy with other tasks that it does not have time to service the timer during a tick interval, one or more of the timers may not be updated during a tick.

This behavior can actually be used to estimate the worst-case loading of the RISC processor. (See Table 7-2 for more details.)

The RISC timer tables are configured in the RCCR, the RISC timer table parameter RAM, and by the SET TIMER command issued to the CP command register, the RISC timer event register, and the RISC timer mask register.

7.4.1 RISC Timer Table Parameter RAM

Two areas of internal RAM are used for the RISC timer tables: the RISC timer table parameter RAM and RISC timer table entries (see Figure 7-5). The RISC timer table parameter RAM area begins at the RISC timer base address (see Table 7-2). This area is used for the general timer parameters.

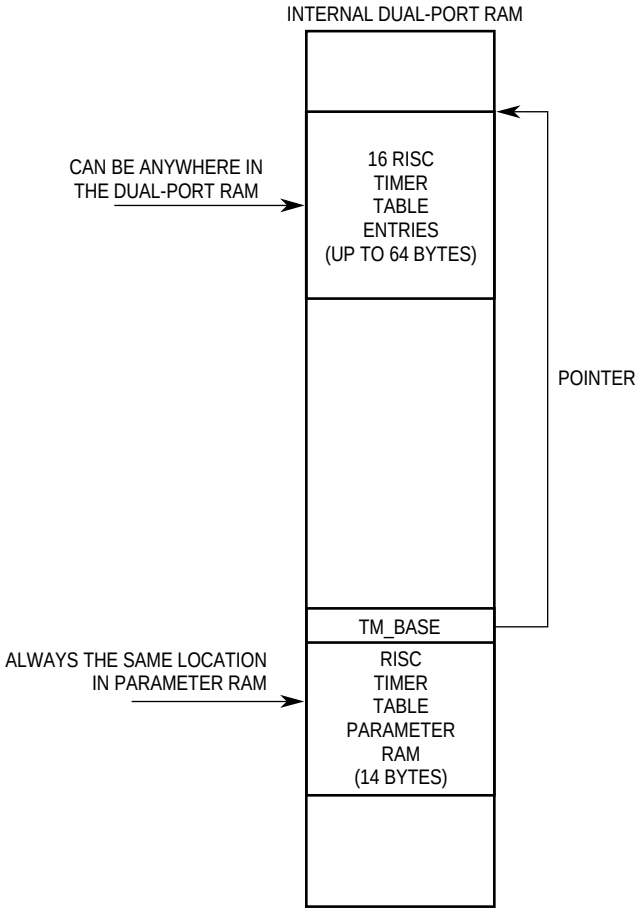


Figure 7-5. RISC Timer Table RAM Usage

Table 7-2. RISC Timer Table Parameter RAM

Address	Name	Width	Description
Timer Base + 00	TM_BASE	Word	RISC Timer Table Base Address
Timer Base + 02	TM_ptr	Word	RISC Timer Table Pointer
Timer Base + 04	R_TMR	Word	RISC Timer Mode Register
Timer Base + 06	R_TMV	Word	RISC Timer Valid Register
Timer Base + 08	TM_cmd	Long	RISC Timer Command Register
Timer Base + 0C	TM_cnt	Long	RISC Timer Internal Count

NOTE: Boldfaced items are initialized by the user.

TM_BASE. The actual RISC timers are located by the user as a small block of memory in the dual-port RAM. TM_BASE is the offset from the beginning of dual-port RAM where that block resides. The user should allocate 4 bytes at TM_BASE for each timer used (64 bytes at TM_BASE if all 16 timers are used). If less than 16 timers are used, the timers should always be allocated in ascending order (RISC timer 0, RISC timer 1, etc.) to save space. For example, if the user only needs two timers, then 8 bytes are required at location TM_BASE as long as the user only enables RISC timer 0 and RISC timer 1.

NOTE

TM_BASE should always be aligned to a long-word boundary (i.e., evenly divisible by 4).

TM_ptr. This value is used exclusively by the RISC to point to the next timer to be accessed in the timer table. It should not be modified by the user.

R_TMR. This value is used exclusively by the RISC to store the mode of the timer: one-shot (bit is zero) or restart (bit is one). R_TMR should not be modified by the user. The SET TIMER command should be used instead.

R_TMV. This value is used exclusively by the RISC to store whether a timer is currently enabled. A bit is a one if the corresponding timer is enabled. R_TMV should not be modified by the user. The SET TIMER command should be used instead.

TM_cmd. This value is used as a parameter location when the SET TIMER command is issued. The user should write this location prior to issuing the SET TIMER command. This parameter is defined as follows:

31	30	29					20	19	16	15					0
V	R	—						TIMER NUMBER		TIMER PERIOD (16 BITS)					

V—Valid

This bit should be set to enable the timer and cleared to disable the timer.

R—Restart

This bit should be set for an automatic restart or cleared for a one-shot operation of the timer.

Bits 29–20—Reserved

These bits should be written with zeros.

Bits 19–16—Timer Number

The timer number is a value from 0 to 15 that signifies the timer is configured.

Bits 15–0—Timer Period

The timer period is the 16-bit timeout value of the timer. The maximum value is 65536, which is programmed by writing $0000 to the timer period.

TM_cnt. This value is simply a tick counter that is updated by the RISC after each tick. It is updated if the RISC internal timer is enabled, regardless of whether any of the 16 timers are enabled. It can be used to track the number of ticks that the RISC has received and responded to. This value is updated only after the RISC scans the timer table.

7.4.2 RISC Timer Table Entries

The actual 16 timers themselves are located in the block of memory following the TM_BASE location. Each timer occupies 4 bytes. The first word forms the initial value of the timer written during the execution of the SET TIMER command, and the next word is the current value of the timer, which 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.

7.4.3 RISC Timer Event Register (RTER)

This 16-bit register is used to report events recognized by the 16 timers and to generate interrupts. Bit 0 corresponds to timer 0, and bit 15 corresponds to timer 15. Note that an interrupt will only be generated if the RISC timer table bit is set in the CPM interrupt mask register. RTER may be read at any time. A bit is cleared by writing a one (writing a zero does not affect a bit's value), and more than one bit may be cleared at a time. This register is cleared at reset.

7.4.4 RISC Timer Mask Register (RTMR)

This 16-bit register is used to enable interrupts that may be generated in the RISC timer event register. If a bit is set, it enables the corresponding interrupt in the RTER. If a bit is cleared, it masks the corresponding interrupt in the RTER. Note that an interrupt will only be generated if the RISC timer table bit is set in the CPM interrupt mask register. This read-write register is cleared at reset.

7.4.5 SET TIMER Command

This command is used to enable, disable, and configure the 16 timers in the RISC timer table. The SET TIMER command is issued to the CR. This means the value $0851 should be written to CR. However, before writing this value, the TM_cmd value should be set up by the user. See 7.4.1 RISC Timer Table Parameter RAM for details.

7.4.6 RISC Timer Initialization Sequence

The following sequence initializes the RISC timers:

1. Configure the RCCR to determine the desired tick interval that will be used for the en-

tire timer table. The TIME bit would normally be turned on at this time; however, it can be turned on later if it is required that all RISC timers be synchronized.

2. Determine the maximum number of timers to be located in the timer table and configure 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 (for space efficiency).
3. Clear the TM_cnt in the RISC timer table parameter RAM to show how many ticks have elapsed since the RISC internal timer was enabled. This step is optional.
4. Clear the RISC timer event register if it is not already cleared. (Ones are written to clear this register.)
5. Configure the RTMR to enable those timers that should generate interrupts. (Ones enable interrupts.)
6. Set the RISC timer table bit in the CPM interrupt mask register to generate interrupts to the system. (The CPM interrupt controller may require other initialization not mentioned here.)
7. Configure the TM_cmd field of the RISC timer table parameter RAM. At this point, determine whether a timer is to be enabled or disabled, one-shot or restart, and what its timeout period should be. If the timer is being disabled, the parameters (other than the timer number) are ignored.
8. Issue the SET TIMER command by writing $0861 to the CR.
9. Repeat the preceding two steps for each timer to be enabled or disabled.

7.4.7 RISC Timer Initialization Example

The following sequence initializes RISC timer 0 to generate an interrupt approximately every second using a 25-MHz general system clock:

1. Write the TIMEP bits of the RCCR with 111111 to generate the slowest clock. This value will generate a tick every 65536 clocks, which is every 2.6 ms at 25 MHz.
2. Configure 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 $0000 to TM_BASE.
3. Write $0000 to TM_cnt in the RISC timer table parameter RAM to see how many ticks have elapsed since the RISC internal timer was enabled. This step is optional.
4. Write $FFFF to the RTER to clear any previous events.
5. Write $0001 to the RTMR to enable RISC timer 0 to generate an interrupt.
6. Write $00020000 to the CPM interrupt mask register to allow the RISC timers to generate a system interrupt. Initialize the CPM interrupt configuration register.
7. Write $C0000EE6 to the TM_cmd field of the RISC timer table parameter RAM. This enables RISC timer 0 to time out after 3814 (decimal) ticks of the timer. The timer will automatically restart after it times out.
8. Write $0851 to the CR to issue the SET TIMER command.
9. Set the TIME bit in the RCCR to enable the RISC timer to begin operation.

7.4.8 RISC Timer Interrupt Handling

The following sequence describes what would normally occur within an interrupt handler for the RISC timer tables:

1. Once an interrupt occurs, read the RISC timer event register to see which timer or timers have caused interrupts. The RISC timer event bits would normally be cleared at this time.
2. Issue additional SET TIMER commands at this time or later, as desired. Nothing need be done if the timer is being restarted automatically for a repetitive interrupt.
3. Clear the R-TT bit in the CPM interrupt status register.
4. Execute the RTE instruction.

7.4.9 RISC Timer Table Algorithm

The RISC scans the timer table once every tick. For each valid timer in the timer table, the RISC decrements the count and checks for a timeout. If no timeout occurs, it moves to the next timer. If a timeout occurs, the RISC sets the corresponding event bit in the RISC timer event register. It checks to see if the timer is to be restarted. If so, it leaves the timer valid bit set in the R_TMV location and resets the current count to the initial count; otherwise, it clears the R_TMV bit. Once the timer table is scanned, the RISC updates the TM_cnt value in the RISC timer table parameter RAM and ceases working on the timer tables until the next tick.

If a SET TIMER command is issued, the RISC controller 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 the timer table alterations to the execution of the RISC.

7.4.10 RISC Timer Table Application: Track the RISC Loading

The RISC timers can be used to track the loading of the RISC controller. The following sequence gives a method for using the 16 RISC timers to determine if the RISC controller ever exceeds the 96% utilization level during any tick interval. Removing the timers then adds a 4% margin to the RISC utilization level. The aggressive user can use this technique to push the RISC performance to its limit in an application.

The user should use the standard initialization sequence, with the following differences:

1. Program the tick of the RISC timers to be 1024 x 16 = 16384.
2. Disable RISC timer interrupts, if desired.
3. Using the SET TIMER command, initialize all 16 RISC timers to have a timer period of $0000, which equates to 65536.
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 65536.
5. After hours of operation, compare the general-purpose timer to the current count of RISC timer 15. If RISC timer 15 is more than two ticks different from the general-purpose timer, the RISC controller has, during some tick interval, exceeded the 96% uti-

lization level.

NOTE

The general-purpose timers are up-counters, but the RISC timers are down-counters. The user should consider this fact when comparing timer counts.

7.5 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), and a timer event register (TER). The TMR contains the prescaler value programmed by the user. In addition, there is one timer global configuration register (TGCR). The timer block diagram is shown in Figure 7-6.

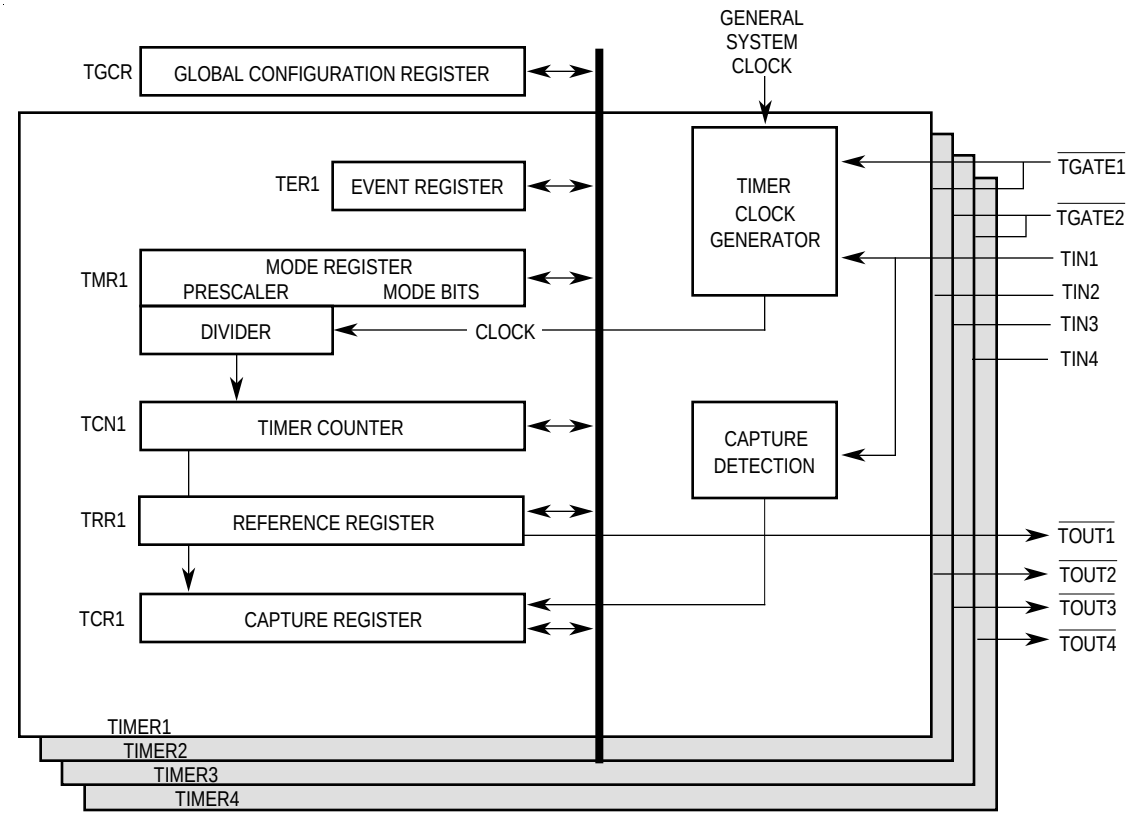


Figure 7-6. Timer Block Diagram

7.5.1 Timer Key Features

The four identical general-purpose timers have the following features:

- Maximum Period of 10.7 Sec (at 25 MHz)
- 40-ns Resolution (at 25 MHz)
- Programmable Sources for the Clock Input

- Input Capture Capability
- Output Compare with Programmable Mode for the Output Pin
- Two Timers Internally or Externally Cascadable To Form a 32-Bit Timer
- Free Run and Restart Modes
- Functionally Compatible with Timer 1 and Timer 2 on the MC68302

7.5.2 General-Purpose Timer Units

The clock input to the prescaler may be selected from three sources: the general system clock, the general system clock divided by 16, or the corresponding TINx pin. Each option is discussed in the following paragraphs.

The general system clock is generated in the clock synthesizer and defaults to the system frequency (for instance, 25 MHz). However, the general system clock has the option to be divided before it leaves the clock synthesizer. This mode, called slow go, is used to save power. Whatever the resulting frequency of the general system clock, the user may choose either that frequency or that frequency divided by 16 as the input to the prescaler of each timer.

Alternatively, the user may choose the TINx pin to be the clock source. TINx is internally synchronized to the internal clock. If the user has chosen to internally cascade two 16-bit timers to a 32-bit timer, then a timer may internally use the clock generated by the output of another timer.

The clock input source is selected by the ICLK bits of the corresponding TMR. The prescaler is programmed to divide the clock input by values from 1 to 256. 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 (40 ns at 25 MHz). The maximum period (when the reference value is all ones) is 268,435,456 cycles (10.7 sec at 25 MHz). Both values assume that the general system clock is the full 25 MHz.

Each timer may 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 the ORI bit in the TMR is set.

Each timer may output a signal on the timer output pin ($\overline{\text{TOUT1}}$, $\overline{\text{TOUT2}}$, $\overline{\text{TOUT3}}$, or $\overline{\text{TOUT4}}$) when the reference value is reached (selected by the OM bit of the corresponding TMR). This signal can be an active-low pulse or a toggle of the current output. The output can also be internally connected to the input of another timer, resulting in a 32-bit timer.

Each timer has a 16-bit TCR, which is 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 CE bits in the corresponding TMR. Upon a capture or reference event, the corresponding TER bit is set, and a maskable interrupt request is issued to the CPM interrupt controller.

The timers may be gated/restarted by an external gate signal. There are two gate pins: $\overline{\text{TGATE1}}$ controls timer 1 and/or timer 2; $\overline{\text{TGATE2}}$ controls timer 3 and/or timer 4.

Normal gate mode enables the count on a falling edge of the $\overline{\text{TGATEx}}$ pin and disables the count on the rising edge of the $\overline{\text{TGATEx}}$ pin. Normal gate mode allows the timer to count conditionally based on the state of the $\overline{\text{TGATEx}}$ pin.

Restart gate mode performs the same function as normal mode, except that it also resets the counter on the falling edge of the $\overline{\text{TGATEx}}$ pin. The restart gate mode has applications in pulse interval measurement and bus monitoring:

- Pulse Measurement—The restart gate mode can measure a low pulse on the $\overline{\text{TGATEx}}$ pin. The rising edge of the $\overline{\text{TGATEx}}$ pin completes the measurement, and if $\overline{\text{TGATEx}}$ is externally connected to TINx, 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 the $\overline{\text{TGATEx}}$ pin. 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, and the gate operating mode is selected in the TGCR.

NOTE:

TGATE is internally synchronized to the system clock. If TGATE meets the asynchronous input setup time (spec #47A) then, when working with the internal clock, the counter will begin counting after 1 system clock.

7.5.2.1 CASCADED MODE. In this mode (see Figure 7-7) two 16-bit timers can be internally cascaded to form a 32-bit counter. Timer 1 may be internally cascaded to timer 2, and timer 3 may be internally cascaded to timer 4. Since, the decision to cascade timers is made independently, the user may select such options as two 16-bit timers and one 32-bit timer. The TGCR is used to put the timers into cascaded mode.

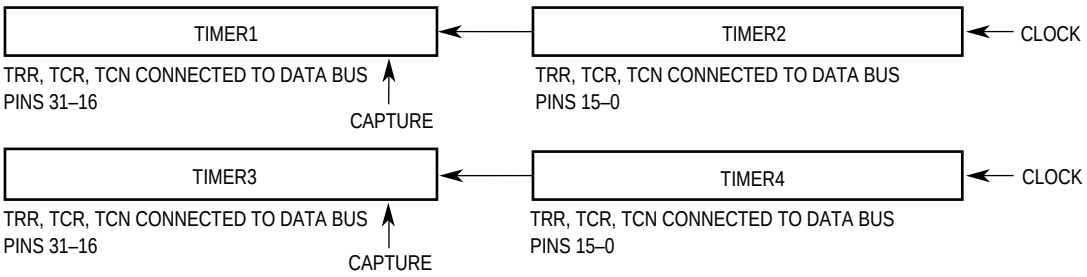


Figure 7-7. Timer Cascaded Mode Block Diagram

If the CAS bit is set in the TGCR, the two timers function as a one 32-bit timer with one 32-bit TRR, one 32-bit TCR, and one 32-bit TCN. In this case, TMR1 and/or TMR3 are ignored, and the modes are defined using TMR2 and/or TMR4. The capture will be controlled from TIN2 or TIN4. Interrupts will be generated from TER2 or TER4.

When working in the cascaded mode, the cascaded TRR, TCR, and TCN should always be referenced with 32-bit bus cycles.

7.5.2.2 TIMER GLOBAL CONFIGURATION REGISTER (TGCR). The TGCR is a 16-bit, memory-mapped, read/write register that contains configuration parameters used by all four timers. It allows starting and stopping any number of timers simultaneously if one bus cycle is used to access TGCR. The TGCR is cleared by reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAS4	FRZ4	STP4	RST4	GM2	FRZ3	STP3	RST3	CAS2	FRZ2	STP2	RST2	GM1	FRZ1	STP1	RST1

CAS4—Cascade Timers

0 = Normal Operation.
1 = Timers 3 and 4 are cascaded to form a 32-bit timer.

CAS2—Cascade Timers

0 = Normal Operation.
1 = Timers 1 and 2 are cascaded to form a 32-bit timer.

FRZ—Freeze

0 = The corresponding timer ignores the FREEZE pin.
1 = Halt the corresponding timer if the FREEZE pin is asserted. (The FREEZE pin is asserted in background debug mode when the CPU32+ is enabled.)

STP —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 IMB 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.

RST—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.

GM2—Gate Mode for Pin 2

This bit is only valid if the gate function is enabled in TMR3 or TMR4.

0 = Restart gate mode. The $\overline{\text{TGATE2}}$ pin 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.

GM1—Gate Mode for Pin 1

This bit is only valid if the gate function is enabled in TMR1 or TMR2.

0 = Restart gate mode. The $\overline{\text{TGATE1}}$ pin is used to enable/disable count. A falling $\overline{\text{TGATE1}}$ pin 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.

7.5.2.3 TIMER MODE REGISTER (TMR1, TMR2, TMR3, TMR4). TMR1–TMR4 are identical 16-bit, memory-mapped, read/write registers. These registers are cleared by reset.

NOTE

The TGCR should be initialized prior to the TMRs, or erratic behavior may occur. The only exception is the RST bit in the TGCR, which may be modified at any time.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS								CE		OM	ORI	FRR	ICLK		GE

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; the value 11111111 divides the clock by 256.

CE—Capture Edge and Enable Interrupt

00 = Disable interrupt on capture event; capture function is disabled.
01 = Capture on rising TINx edge only and enable interrupt on capture event.
10 = Capture on falling TINx edge only and enable interrupt on capture event.
11 = Capture on any TINx edge and enable interrupt on capture event.

OM—Output Mode

0 = Active-low pulse on $\overline{\text{TOUTx}}$ for one timer input clock cycle as defined by the ICLK bits. Thus, $\overline{\text{TOUTx}}$ may be low for one general system clock period, one general system clock/16 period, or one TINx pin clock cycle period. $\overline{\text{TOUTx}}$ changes occur on the rising edge of the system clock.

1 = Toggle the $\overline{\text{TOUTx}}$ pin. $\overline{\text{TOUTx}}$ changes occur on the rising edge of the system clock.

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.

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.

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 general system clock.
10 = Internal general system clock divided by 16.
11 = Corresponding TIN pin: TIN1, TIN2, TIN3, or TIN4 (falling edge).

GE—Gate Enable

0 = The $\overline{\text{TGATE}}$ signal is ignored.
1 = The $\overline{\text{TGATE}}$ signal is used to control the timer.

7.5.2.4 TIMER REFERENCE REGISTERS (TRR1, TRR2, TRR3, TRR4). Each TRR is a 16-bit, memory-mapped, read-write register containing the reference value for the timeout. TRR1–TRR4 are set to all ones by reset. The reference value is not reached until TCN increments to equal TRR.

7.5.2.5 TIMER CAPTURE REGISTERS (TCR1, TCR2, TCR3, TCR4). Each TCR is a 16-bit register used to latch the value of the counter. TCR1–TCR4 appear as memory- mapped, read-only registers to the user. TCR1–TCR4 are cleared by reset.

7.5.2.6 TIMER COUNTER (TCN1, TCN2, TCN3, TCN4). Each TCN is a 16-bit, memory-mapped, read-write up-counter. A read cycle to TCN1–TCN4 yields the current value of the timer, but does not affect the counting operation. A write cycle to TCN1–TCN4 sets the register to the written value, causing its corresponding prescaler to be reset.

NOTE

Write operation to this register while the timer is not running may not update the register correctry. User should always use timer refrence register to define desired count value.

7.5.2.7 TIMER EVENT REGISTERS (TER1, TER2, TER3, TER4). Each TER is a 16-bit register used to report events recognized by any of the timers. On recognition of an output reference event, the timer sets the REF bit in the TER, regardless of the corresponding ORI in the TMR. The capture event will be set only if enabled by the CE bits in the TMR. TER1–TER4, which appear to the user as memory-mapped registers, may be read at any time.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—														REF	CAP

A bit is reset by writing a one to that bit (writing a zero does not affect a bit's value). More than one bit may be reset at a time. Both bits must be reset before the timer will negate the interrupt to the CPM interrupt controller. This register is cleared by reset.

Bits 15–2—Reserved

REF—Output Reference Event

The counter has reached the TRR value. The ORI bit in the TMR is used to enable the interrupt request caused by this event.

CAP—Capture Event

The counter value has been latched into the TCR. The CE bits in the TMR are used to enable generation of this event.

7.5.3 Timer Examples

The following example lists the required initialization sequence of timer 2 to generate an interrupt every 10 μs, assuming a general system clock of 25 MHz. This means that an interrupt should be generated every 250 system clocks.

1. TGCR = $0000. Put timer 2 into the reset state. Do not use cascaded mode.
2. TMR2 = $001A. Enable the prescaler of the timer to divide-by-1 and the clock source to general system clock. Enable an interrupt when the reference value is reached, and restart the timer to repeatedly generate 10-μs interrupts.
3. TCN2 = $0000. Initialize the timer 2 count to zero. This is the default state of this register.
4. TRR2 = $00FA. Initialize the timer 2 reference value to 250 (decimal).
5. TER2 = $FFFF. Clear TER2 of any bits that might have been set.
6. CIMR = $00040000. Enable the timer 2 interrupt in the CPM interrupt controller. Initialize the CPM interrupt configuration register.
7. TGCR = $0010. Enable timer 2 to begin counting.

To implement the same function with a 32-bit timer using timer 1 and timer 2, the following sequence may be used:

1. TGCR = $0080. Cascade timer 1 and timer 2. Put timer 1 and timer 2 in the reset state.
2. TMR2 = $001A. Enable the prescaler of timer 2 to divide-by-1 and the clock source to general system clock. Enable an interrupt when the reference value is reached, and restart the timer to repeatedly generate 10 μs interrupts.
3. TMR1 = $0000. Enable timer 1 to use the output of timer 2 as its input, which is the default state of this register.
4. TCN1 = $0000, TCN2 = $0000. Initialize the combined timer 1 and timer 2 count to zero which is the default state of this register. (This can be accomplished with one 32-bit data move to TCN1.)
5. TRR1 = $0000, TRR2 = $00FA. Initialize the combined timer 1 and timer 2 reference value to 250 (decimal). (This can be accomplished with one 32-bit data move to TRR1.)
6. TER2 = $FFFF. Clear TER2 of any bits that might have been set.
7. CIMR = $00040000. Enable the timer 2 interrupt in the CPM interrupt controller. Initialize the CPM interrupt configuration register.
8. TGCR = $0091. Enable timer 1 and timer 2 to begin counting. Leave the timers in cas-

caded mode.

7.6 IDMA CHANNELS

The QUICC includes a number of DMA channels, including 14 SDMA channels for the four SCCs, two SMCs, and SPI and two general-purpose IDMA controllers. The SDMA channels are discussed in 7.7 SDMA Channels. The IDMA channels are discussed in the following paragraphs.

The two general-purpose IDMA controllers can operate in different modes of data transfer as programmed by the user. The IDMA can transfer data between any combination of memory and I/O. In addition, data may be transferred in either byte, word, or long-word quantities, and the source and destination addresses may be either odd or even. The most efficient packing algorithms are used in the IDMA transfers. The single address mode gives the highest performance, allowing data to be transferred between memory and a peripheral in a single bus cycle. The chip-select and wait-state generation logic on the QUICC may be used with the IDMA.

The IDMA supports three buffer handling modes: single buffer, auto buffer, and buffer chaining. Single buffer mode is that of the traditional DMA controller. 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 buffer descriptors that are similar to those used by an SCC. These buffer descriptions reside in the dual-port RAM.

If the single buffer mode of the IDMA is used, programming the IDMA is very similar (although not exactly software compatible) to that of the IDMA on the MC68302 or the DMA controller on the MC68340. The auto buffer and buffer chaining modes, however, are not available on those devices, and the single address mode is not available on the MC68302.

The maximum transfer rate of the IDMA is 50 Mbyte/sec. This assumes a 32-bit data transfer from memory to peripheral using fast termination (2 clocks per bus cycle) timing and single address mode: (4 Bytes $\times$ 25 MHz Clocks/sec)/(2 Clocks per Transfer) = 50 Mbyte/sec.

The maximum transfer rate of the IDMA in dual address mode is 25 Mbyte/sec. This assumes a 32-bit source and destination, fast termination (2 clocks per bus cycle) timing, and two bus cycles for each transfer: (4 Bytes $\times$ 25 MHz Clocks/sec)/(4 Clocks per Transfer) = 25 Mbyte/sec.

The IDMA controller block diagram is shown in Figure 7-8.

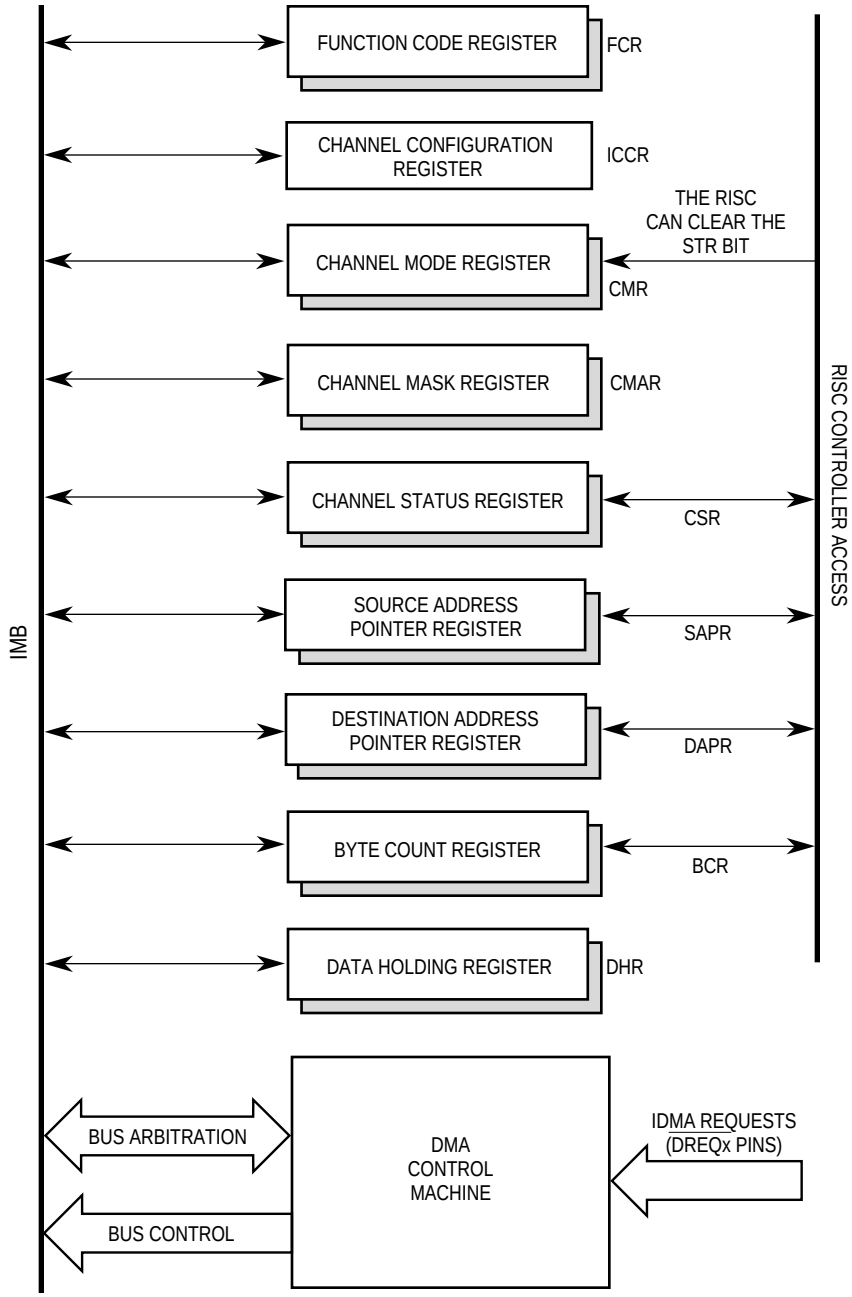


Figure 7-8. IDMA Controller Block Diagram

7.6.1 IDMA Key Features;

The IDMA contains the following features:

- Two Independent, Fully Programmable DMA Channels
- Dual Address or Single Address Transfers with 32-Bit Address and 32-Bit Data Capability

- Up to 50 Mbyte/sec Transfer Rates in Single Address Mode and 25 Mbyte/sec in Dual Address Mode (assuming a 25-MHz system clock)
- 32-Bit Byte Transfer Counters
- 32-Bit Address Pointers That Can Increment or Remain Constant
- Operand Packing and Unpacking for Dual Address Transfers using the Most Efficient Techniques
- Supports All Bus-Termination Modes
- Provides Full DMA Handshake for Cycle Steal and Burst Transfers
- Supports Fixed and Rotating Priority Between IDMA Channels
- Buffer Handling Modes: Single Buffer, Auto Buffer, and Buffer Chaining

7.6.2 IDMA Registers

Each IDMA channel has eight registers that define its specific operation. These registers include a 32-bit source address pointer register (SAPR), a 32-bit destination address pointer register (DAPR), an 8-bit function code register (FCR), a 32-bit byte count register (BCR), a 16-bit channel mode register (CMR), an 16-bit channel configuration register (ICCR), an 8-bit channel status register (CSR), and an 8-bit channel mask register (CMAR). These registers provide the addresses, transfer count, and configuration information necessary to set up a transfer. They also provide a means of controlling the IDMA channel and monitoring its status. All registers can be modified by the CPU32+ core.

For the auto buffer and buffer chaining modes, the RISC controller uses a buffer descriptor ring to automatically initialize the DAPR, SAPR, and BCR. The buffer descriptor ring resides in dual-port RAM so that it may be accessed by the RISC controller without bus overhead.

The IDMA channel also includes a 32-bit data holding register (DHR), which is not accessible to the CPU32+ core and is used by the IDMA for temporary data storage.

7.6.2.1 IDMA CHANNEL CONFIGURATION REGISTER (ICCR). The 16-bit ICCR configures both IDMA channels. It is always readable and writable in the supervisor mode, although writing is not recommended unless the module is disabled. It is initialized to $0000 at reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STP	FRZ		ARBP		ISM			—	IAID			—			

STP—Stop Bit

0 = The system clock operates normally within the IDMA.
1 = Stop the system clock to the IDMA channels. This setting is used to conserve power when both IDMAs are unused.

FRZ1–FRZ0—Freeze

These bits determine the action to be taken when the FREEZE signal is asserted. The IDMA negates its internal bus request and keeps it negated until FREEZE is negated or the IDMA is reset.

00 = The IDMA channels ignore the FREEZE signal.
01 = Reserved.
10 = The IDMA channels freeze on the next bus cycle.
11 = Reserved.

ARBP—Arbitration Priority

These two bits select the arbitration priority between the two IDMA channels.

00 = IDMA channel 1 has priority over channel 2.
01 = IDMA channel 2 has priority over channel 1.
10 = Rotating priority.
11 = Reserved.

ISM—Interrupt Service Mask

These bits contain the interrupt service mask. When the interrupt service level on the IMB is greater than the interrupt service mask, the IDMA vacates the bus and negates its bus request to the IMB until the interrupt level service is less than or equal to the interrupt service mask.

NOTE

The user should program ISM to 7 for typical user applications. This gives the IDMA priority over all interrupt handlers. These bits MUST be set to 7 if the QUICC is in slave mode.

Bits 7, 3–0—Reserved

IAID—IDMA Arbitration ID

These bits establish bus arbitration priority level among sub-blocks that have the capability of becoming bus master. In the QUICC, the IDMAs, the SDMAs, and the SIM60 DRAM refresh controller can become bus masters. An arbitration ID uses a number (0–7) to decide the priority of multiple bus masters that are requesting the IMB. A 0 is the lowest priority and a 7 is the highest priority.

The value programmed into the IAID bits is the arbitration ID of the highest priority IDMA channel. The arbitration ID of the lowest priority IDMA channel is IAID minus 2. The ARBP bits determine which IDMA channel has the higher priority. If round-robin priority is selected, then the IDMA channels alternate between the two IAID values.

Example: If ARBP = 00, selecting IDMA channel 1 to always have the highest priority, the IAID values are:

IDMA channel 1 arbitration ID = IAID

IDMA channel 2 arbitration ID = IAID – 2

NOTES

The user should program IAID to 2 in typical user applications. IAID should not be programmed to a value less than 2. This val-

ue should be less than the SDMA arbitration ID so that the SDMA channels have priority over the IDMA channels. User must program this field to 7 when the QUICC is configured in slave mode.

7.6.2.2 CHANNEL MODE REGISTER (CMR). Each IDMA channel contains a 16-bit CMR that is reset to $0000. It is used to configure most of the IDMA options.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECO	SRM	S/D	RCI	REQG		SAPI	DAPI	SSIZE		DSIZE		BT		RST	STR

ECO — External Control Option

*Dual Address Mode***:** this bit defines which device is connected to the control signals.

0 = The control signals ($\overline{\text{DREQx}}$, $\overline{\text{DACKx}}$, and $\overline{\text{DONEx}}$) are associated with the destination (write) portion of the transfer.

1 = The control signals ($\overline{\text{DREQx}}$, $\overline{\text{DACKx}}$, and $\overline{\text{DONEx}}$) are associated with the source (read) portion of the transfer.

*Single Address Mode***:** this bit defines the direction of the transfer.

0 = The device writes to memory, and the control signals ($\overline{\text{DREQx}}$, $\overline{\text{DACKx}}$, and $\overline{\text{DONEx}}$) are used by the device to provide data during the destination (write) portion of the transfer.

1 = The device reads from memory, and the control signals ($\overline{\text{DREQx}}$, $\overline{\text{DACKx}}$, and $\overline{\text{DONEx}}$) are used by the device to write data during the source (read) portion of the transfer.

NOTE

If REQG is programmed to be internal (REQG = 0X), $\overline{\text{DREQx}}$ is ignored.

SRM — Synchronous Request Mode

This bit controls how external devices may use the $\overline{\text{DREQx}}$ pin for IDMA service. This bit is only relevant for applications that use external request mode or use the external $\overline{\text{DONEx}}$ pin to terminate the IDMA operation.

0 = Asynchronous request mode is selected. The $\overline{\text{DREQx}}$ and $\overline{\text{DONEx}}$ input signals are internally synchronized to the IDMA clock before they are used by the IDMA.

1 = Synchronous request mode is selected. The $\overline{\text{DREQx}}$ and $\overline{\text{DONEx}}$ input signals are used by the IDMA without first being internally synchronized. This results in faster operation, but should only be used if setup and hold times can be met.

S/D — Single/Dual Address Transfer

0 = The IDMA channel runs standard dual address transfers. Each transfer requires at least two bus cycles. Data packing is performed using the DHR.

1 = The IDMA channel runs single address transfers from a peripheral to memory or from memory to a peripheral. The transfer requires one bus cycle. The DHR is not used for these transfers because the data is transferred directly into the destination location.

RCI — RISC Controls IDMA

0 = Single Buffer Mode. The user programs all IDMA registers for each buffer transfer.
1 = Auto buffer or buffer chaining mode. The RISC reconfigures the IDMA channel at the end of each buffer transfer according to the buffer descriptor ring. The choice between auto buffer and buffer chaining is made in the buffer descriptor itself.

REQG — Request Generation

The REQG bits define what generates the requests for IDMA activity over the bus.

00 = Internal request at limited rate (limited burst bandwidth) set by BT bits
01 = Internal request at maximum rate (one burst)
10 = External request burst transfer mode ($\overline{\text{DREQx}}$ is level sensitive)
11 = External request cycle steal ($\overline{\text{DREQx}}$ is edge sensitive)

SAPI — SAPR Increment

0 = SAPR is not incremented after each transfer.
1 = SAPR is incremented by one, two, or four after each transfer, according to the SSIZE bits. (SAPR may be incremented by an amount less than the SSIZE value at the beginning or end of a block transfer, depending on the source starting address or byte count.)

DAPI — DAPR Increment

0 = DAPR is not incremented after each transfer.
1 = DAPR is incremented by one, two, or four after each transfer, according to the DSIZE bits. (DAPR may be incremented by an amount less than the DSIZE value at the beginning or end of a block transfer, depending on the destination starting address or byte count.)

SSIZE — Source Size

The following decoding shows the definitions for the SSIZE bits. The user should set these bits to the port size of the source (e.g., choose byte for an 8-bit peripheral).

00 = Long word
01 = Byte
10 = Word
11 = Reserved

DSIZE — Destination Size

The following decoding shows the definitions for the DSIZE bits. The user should set these bits to the port size of the destination (e.g., choose byte for an 8-bit peripheral).

00 = Long word
01 = Byte
10 = Word
11 = Reserved

BT — Burst Transfer

The BT bits control the maximum percentage of the IMB that the IDMA can use during each 1024 clock cycle period after enabling the IDMA.

00 = IDMA gets up to 75% of the bus bandwidth.
01 = IDMA gets up to 50% of the bus bandwidth.
10 = IDMA gets up to 25% of the bus bandwidth.
11 = IDMA gets up to 12.5% of the bus bandwidth.

NOTE

These percentages are valid only when using internal request generation (REQG = 00).

RST—Software Reset

This bit resets the IDMA to the same state as an external reset. The IDMA clears RST when the reset is complete.

0 = Normal operation.
1 = The channel aborts any external pending or running bus cycles and terminates channel operation. Setting RST clears all bits in the CSR and CMR.

NOTE

The user should reset the IDMA channel prior to issuing the LP-STOP instruction.

STR—Start Operation

This bit starts the IDMA transfer if the REQG bits are programmed for an internal request. If the REQG bits are programmed for an external request, this bit must be set before the IDMA will recognize the first request on the $\overline{\text{DREQ}}$x input.

0 = Stop channel. Clearing this bit causes the IDMA to stop transferring data at the end of the current bus cycle. The IDMA internal state is not altered.
1 = Start channel. Setting this bit allows the IDMA to start transferring data (or continue if previously stopped).

NOTES

STR is cleared automatically when the transfer is complete.

If the STR bit is cleared by software during the middle of an IDMA operand transfer, the IDMA will continue to hold the bit in a one state until the operand transfer has completed. Thus, if the user waits for the STR bit to be cleared after clearing it in software, he is assured that the values of SAPR, DAPR, and BCR accurately show the current state of the IDMA transfer.

7.6.2.3 SOURCE ADDRESS POINTER REGISTER (SAPR). The SAPR contains 32 address bits of the source operand used by the IDMA to access memory or memory-mapped peripheral controller registers. During the IDMA read cycle, the address on the master address bus is driven from this register. The SAPR may be programmed by the SAPI bits to be incremented or remain constant after each operand transfer.

The register is incremented using unsigned arithmetic and will roll over if an overflow occurs. For example, if a register contains $FFFFFFFF and is incremented by one, it will roll over to $00000000. This register can be incremented by one, two, or four, depending on the SSIZE bits and the starting address in this register.

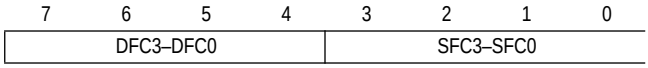
The SAPR may be initialized by the host processor or by the RISC controller via a buffer descriptor's ring structure when the RCI bit is set for special buffer handling modes.

7.6.2.4 DESTINATION ADDRESS POINTER REGISTER (DAPR). The DAPR contains 32 address bits of the destination operand used by the IDMA to access memory or memory-mapped peripheral controller registers. During the IDMA write cycle, the address on the master address bus is driven from this register. The DAPR may be programmed by the DAPI bits to be incremented or remain constant after each operand transfer.

The register is incremented using unsigned arithmetic and will roll over if overflow occurs. For example, if a register contains $FFFFFFFF and is incremented by one, it will roll over to $00000000. This register can be incremented by one, two, or four, depending on the DSIZE bit and the starting address.

The DAPR may be initialized by the host processor or by the RISC controller via a buffer descriptor's ring structure when the RCI bit is set for special buffer handling modes.

7.6.2.5 FUNCTION CODE REGISTER (FCR). Each IDMA channel has an 8-bit FCR that is initialized to $00 at reset.



During an IDMA bus cycle, the SFC and DFC bits define the source and destination function code values that are output by the IDMA and the appropriate address registers. The address space on the function code lines may be used by an external memory management unit (MMU) or other memory-protection device to translate the IDMA logical addresses to proper physical addresses. The function code value programmed into the FCR is placed on pins FC3–FC0 during a bus cycle to further qualify the address bus value.

NOTES

This register is typically set to 1xxx1xxxb to cause the IDMA to operate in the DMA function code space, as opposed to a CPU program or data space.

To keep interrupt acknowledge cycles unique in the system, do not set this register to $77.

7.6.2.6 BYTE COUNT REGISTER (BCR). This 32-bit register specifies the number of bytes of data to be transferred by the IDMA. The largest value that can be specified is 4 Gbytes (BCR = $00000000). This register is decremented once for each byte transferred successfully, for a total of 1, 2, or 4 per operand transfer. BCR may be even or odd as desired. The

IDMA channel will terminate the transfer of a block of memory if this register reaches zero during operation.

7.6.2.7 CHANNEL STATUS REGISTER (CSR). The CSR is an 8-bit register used to report events recognized by the IDMA controller. On recognition of an event, the IDMA sets its corresponding bit in the CSR, regardless of the corresponding bits in the CMAR. The CSR is a memory-mapped register that may be read at any time. A bit is reset by writing a one and is left unchanged by writing a zero. More than one bit may be reset at a time, and the register is cleared by reset.

7	6	5	4	3	2	1	0
—		AD	BRKP	OB	BES	BED	DONE

Bits 7–6—Reserved

AD—Auxiliary Done

This bit is valid in auto buffer and buffer chaining modes. It is set when the IDMA channel has completed a buffer transfer for a buffer descriptor (BD) that has its I-bit set. For AD to be set, the BCR must have been decremented to zero with no errors occurring during any IDMA transfer bus cycle. The IDMA will then move to the next BD and continue to transfer data.

BRKP—Breakpoint

This bit indicates that the breakpoint signal was asserted during an IDMA transfer. This bit is cleared by writing a one or by reset. Writing a zero has no effect on BRKP.

OB—Out of Buffers

This bit is valid only when the RISC controls the IDMA (RCI bit in the CMR is set). It is set when working with the RISC controller and there are no more valid buffers out of which to transfer data.

BES—Bus Error Source

This bit indicates that the IDMA channel terminated with an error during the read cycle. The channel terminates the IDMA operation without setting DONE. BES is cleared by writing a one or by setting RST in the CMR. Writing a zero has no effect on BES.

BED—Bus Error Destination

This bit indicates that the IDMA channel terminated with an error during the write cycle. The channel terminates the IDMA operation without setting DONE. BED is cleared by writing a one or by setting RST in the CMR. Writing a zero has no effect on BED.

DONE—Normal Channel Transfer Done

This bit indicates that the IDMA channel has terminated normally. Normal channel termination is defined as follows:

1. In single buffer mode, the BCR has decremented to zero, and no errors have occurred during any IDMA transfer bus cycle.

2. In buffer chaining or auto buffer modes, the BCR has decremented to zero, the L-bit in the BD has been set, and no errors have occurred during any IDMA transfer bus cycle.
3. An external peripheral has asserted $\overline{\text{DONEx}}$ during an access by the IDMA to that peripheral and no errors have occurred during any IDMA transfer bus cycle.

DONE will not be set if the channel terminates due to an error. DONE is cleared by writing a one or by setting RST in the CMR. Writing a zero has no effect on DONE.

7.6.2.8 CHANNEL MASK REGISTER (CMAR). The CMAR is an 8-bit, memory-mapped, read-write register that has the same bit format as the CSR. If a bit in the CMAR is a one, the corresponding interrupt in the CSR will be enabled. If the bit is a zero, the corresponding interrupt in the CSR will be masked. CMAR is cleared at reset.

7.6.2.9 DATA HOLDING REGISTER (DHR). This 7-byte register serves as a buffer register for the data being transferred during dual address IDMA cycles. No address for DHR is given since this register cannot be addressed by the programmer. The DHR allows the data to be packed and unpacked by the IDMA during the transfer. For example, if the source operand size is byte and the destination operand size is word, then two-byte read cycles occur, followed by a one-word write cycle. The two bytes of data are buffered in the DHR until the word write cycle occurs. The DHR allows for packing and unpacking of operands for all possible combinations: bytes to words, bytes to long words, words to long words, words to bytes, long words to bytes, and long words to words.

7.6.3 Interface Signals

The IDMA has three dedicated control signals per channel: DMA request ($\overline{\text{DREQx}}$), DMA acknowledge ($\overline{\text{DACKx}}$), and end of IDMA transfer ($\overline{\text{DONEx}}$). The peripheral used with these signals may be either a source or a destination of the IDMA transfers.

NOTE

$\overline{\text{DREQ}}$ must be level sensitive if IDMA uses buffer chaining mode.

7.6.3.1 $\overline{\text{DREQ}}$ AND $\overline{\text{DACK}}$. These are the handshake signals between the peripheral requiring service and the QUICC. When the peripheral requires IDMA service, it asserts $\overline{\text{DREQx}}$, and the QUICC begins the IDMA process. When the IDMA service is in progress, $\overline{\text{DACKx}}$ is asserted during accesses to the device. $\overline{\text{DREQx}}$ is ignored when the IDMA is programmed to one of the internal request modes.

7.6.3.2 $\overline{\text{DONEX}}$. This bidirectional open-drain signal is used to indicate the last IDMA transfer. $\overline{\text{DONEx}}$ is always an output of the IDMA if the transfer count is exhausted.

$\overline{\text{DONEx}}$ may also operate as an input. If $\overline{\text{DONEx}}$ is externally asserted during internal request modes, the IDMA transfer is terminated. With external request modes, $\overline{\text{DONEx}}$ may be used as an input to the IDMA controller to indicate that the device being serviced requires no more transfers and the transmission is to be terminated.

7.6.4 IDMA Operation

Every IDMA operation involves the following steps: IDMA channel initialization, data transfer, and block termination. In the initialization phase, the core (or external processor) loads the registers with control information, initializes the IDMA BDs (if auto buffer or buffer chaining is used), and then starts the channel. In the transfer phase, the IDMA accepts requests for operand transfers and provides addressing and bus control for the transfers. The termination phase occurs when the operation is complete and the IDMA interrupts the core if interrupts are enabled.

To initialize a block transfer operation, the user must initialize the IDMA registers. For the auto buffer and buffer chaining modes, the IDMA BDs must be initialized with information describing the data block, device type, request generation method, and other special control options. See 7.6.2 IDMA Registers and 7.6.4.2.3 IDMA Commands (INIT_IDMA) for further details.

7.6.4.1 SINGLE BUFFER. The single buffer mode is used to transfer only one buffer of data. When the buffer has been completely transferred (transfer count exhausted or $\overline{\text{DONE}}$x is asserted), the IDMA channel operation is terminated, STR is cleared, and a maskable interrupt is generated by the DONE bit in the CSR.

7.6.4.2 AUTO BUFFER AND BUFFER CHAINING. The auto buffer and the buffer chaining modes are supported with the RISC controller by setting the RCI bit in the CMR. The host processor should initialize the IDMA BD ring (see Figure 7-9) with the appropriate buffer handling mode, source address, destination address, and block length. The user then sets the STR bit in the CMR. All transfer modes described in 7.6.4.4.4 External Cycle Steal are still valid. The function codes for the source and destination addresses are programmed as described in 7.5.2.5 Timer Capture Registers (TCR1, TCR2, TCR3, TCR4).

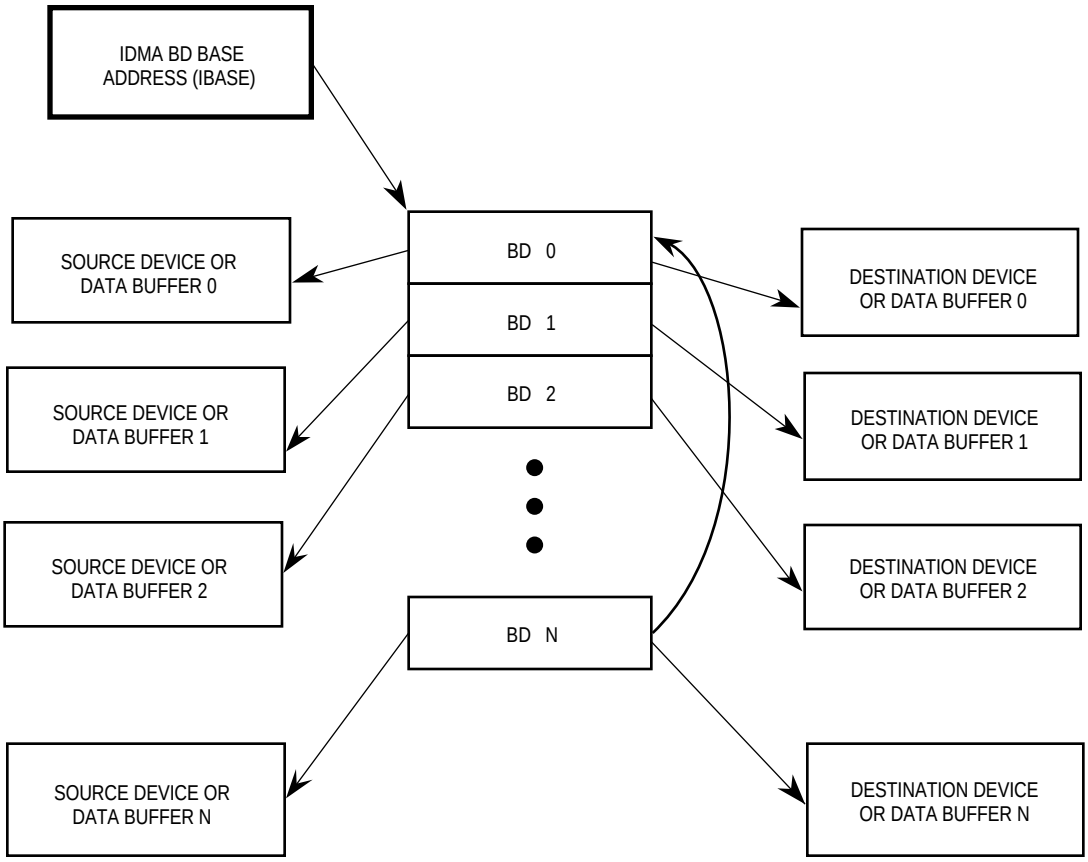


Figure 7-9. IDMA BD Ring

The data associated with each IDMA channel for the auto buffer and buffer chaining modes is stored in buffers. Each buffer is referenced by a BD. The BDs use a ring structure located in the dual-port RAM.

7.6.4.2.1 IDMA Parameter RAM. When an IDMA channel is configured to the auto buffer or buffer chaining mode, the QUICC uses the IDMA parameters listed in Table 7-2.T

Table 7-3. IDMA Parameter RAM

Address	Name	Width	Description
IDMA Base + 00	IBASE	Word	IDMA BD Base Address
IDMA Base + 02	IBPTR	Word	IDMA BD Pointer
IDMA Base + 04	ISTATE	Long	IDMA Internal State
IDMA Base + 08	ITEMP	Long	IDMA Temp

NOTE: The entry in boldface must be initialized by the user.

The IBASE entry 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 this entry before enabling the IDMA channel. Furthermore, the user should not overlap BD tables of two enabled serial channels or IDMA channels, or erratic operation will result. IBASE should contain a value that is divisible by 16.

The IBPTR entry points to the next BD that the IDMA will transfer data to when it is in IDLE state or points to the current BD during transfer processing. After a reset or when the end of an IDMA BD table is reached, the CP initializes this pointer to the value programmed in the IBASE entry.

ISTATE and ITEMP are for RISC use only.

7.6.4.2.2 IDMA Buffer Descriptors (BDs). Source addresses, destination addresses, and byte counts are presented to the RISC controller using special IDMA BDs. The RISC controller reads the BDs, programs the IDMA channel, and notifies the CPU32+ about the completion of a buffer transfer using the IDMA BDs. This concept is like that used for the serial channels on the QUICC, except that the BD is larger to contain additional information.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	V	—	W	I	L	—	CM	—	—	—	—	—	—	SE	DE	DA
OFFSET + 2																
OFFSET + 4	**DATA LENGTH**															
OFFSET + 6																
OFFSET + 8	**SOURCE DATA BUFFER POINTER**															
OFFSET + A																
OFFSET + C	**DESTINATION DATA BUFFER POINTER**															
OFFSET + E																

NOTE: Entries in boldface must be initialized by the user.

The following bits are prepared by the user before transfer and are set by the RISC controller after the buffer has been transferred.

V—Valid

0 = The data buffers associated with this BD are not currently ready for transfer. The user is free to manipulate this BD or its associated data buffer. When it is not in auto buffer mode, the RISC controller clears this bit after the buffer has been transferred (or after an error condition is encountered).

1 = The data buffers have been prepared for transfer by the user. (Note that only one data buffer needs to be prepared if the source/destination is a peripheral device.) It may be only the source data buffer when the destination is a device or the destination data buffer when the source is a device. No fields of this BD may be written by the user once this bit is set.

NOTE

The only difference between auto buffer mode and buffer chaining mode is that the V-bit is not cleared by the RISC controller in the auto buffer mode. Auto buffer mode is enabled by the CM bit.

W—Wrap (Final BD in Table)

0 = This is not the last BD in the table.
1 = This is the last BD in the table. After the associated buffer has been used, the RISC controller will transfer data from the first BD in the table (pointed to by IBASE). The number of BDs in this table is programmable and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.
1 = When this buffer has been serviced by the RISC controller the AD bit in the CSR will be set, which can cause an interrupt.

L—Last

0 = This is not the last buffer to be transferred in the buffer chaining mode. The I-bit may be used to generate an interrupt when this buffer has been serviced.
1 = This is the last buffer to be transferred in the buffer chaining mode. When the transfer count is exhausted, the START bit will be reset and an interrupt (DONE) will be generated, regardless of the I-bit.

CM—Continuous Mode

0 = Buffer chaining mode. The RISC will clear the V-bit after this BD is serviced. The buffer chaining mode is used for transferring large quantities of data into noncontiguous buffer areas. The user can initialize BDs ahead of time, if desired. The RISC controller automatically reloads the IDMA registers from the next BD's values when the transfer is terminated. If $\overline{\text{DONE}}$x is asserted by an external peripheral, the buffer will be closed, the STR bit will be reset, and the DONE bit will be set in the CSR, which can cause an interrupt.
1 = Auto buffer mode (continuous mode). The RISC will not clear the V-bit after this BD is serviced. This is the only difference between auto buffer mode and buffer chaining mode behavior. The auto buffer mode is used to transfer multiple groups of data to/from a buffer ring. This mode does not require reprogramming. The RISC controller automatically reloads the IDMA registers from the next BD values when the transfer is terminated. Either a single BD or multiple BDs may be used in this mode to create an infinite loop of repeated data moves.

NOTE

The I-bit may still be used to generate an interrupt in this mode.

The following bits are written by the RISC controller after it has finished receiving data from the associated data buffer.

SE—Source Access Bus Error

The buffer was closed due to a bus error on the source access. An interrupt (BES) will be generated, regardless of the I-bit. The RISC will clear the V-bit of this BD.

DE—Destination Access Bus Error

The buffer was closed due to a bus error on the destination access. An interrupt (BED) will be generated, regardless of the I-bit. The RISC will clear the V-bit of this BD.

DA—Done Asserted During Transfer

The buffer was closed due to the assertion of $\overline{\text{DONE}}$x. An interrupt (DONE) will be generated, regardless of the I-bit. The RISC will clear the V-bit of this BD.

Data Length

The data length is the number of bytes that the IDMA should transfer from/to this BD's data buffer. The data length should be programmed to a value greater than zero.

Source Buffer Pointer

The source buffer pointer contains the address of the associated source data buffer. The buffer may reside in either internal or external memory.

NOTE

In single address mode when the source is a device, this field is ignored. In dual address mode when the source is a device, this field should contain the device address.

Destination Buffer Pointer

The destination buffer pointer contains the address of the associated destination data buffer. The buffer may reside in either internal or external memory.

NOTE

In single address mode when the destination is a device, this field is ignored. In dual address mode when the destination is a device, this field should contain the device address.

7.6.4.2.3 IDMA Commands (INIT_IDMA). This command causes the RISC controller to reinitialize its IDMA internal state to the condition it had after a system reset. The IDMA BD pointer is reinitialized to the top of BD ring. When in the auto buffer and buffer chaining modes, the IDMA can be reset by setting the RST bit in the CMR and issuing the INIT_IDMA command. The INIT_IDMA command should only be executed in conjunction with the setting of the RST bit in the CMR.

7.6.4.3 STARTING THE IDMA. Once the channel has been initialized with all parameters required for a transfer operation, it is started by setting the STR bit in the CMR. After the channel has been started, any register that describes the current operation may be read but not modified (SAPR, DAPR, FCR, or BCR).

Once STR has been set, the channel is active and either accepts operand transfer requests in external mode or generates requests automatically in internal mode. When the first valid external request is recognized, the IDMA arbitrates for the bus. The $\overline{\text{DREQ}}$x input is ignored until STR is set.

For the single buffer mode, STR is cleared automatically when the BCR reaches zero or when DONEx is asserted externally. For the other buffer handling modes see 7.6.4.8.2 Auto Buffer Mode Termination. and 7.6.4.8.3 Buffer Chaining Mode Termination. The STR is cleared in all modes if the IDMA cycle is terminated by a bus error.

Channel transfer operation may be suspended at any time by clearing STR in software. In response, any operand transfer in progress will be completed, and the bus will be released. No further bus cycles will be started while STR remains cleared. During this time, the CPU32+ core may access IDMA internal registers to determine channel status or to alter operation. When STR is set again, if a transfer request is pending, the IDMA will arbitrate for the bus and continue normal operation.

Interrupts from the IDMA are sent to the interrupt controller. In the interrupt handler, the unmasked bits in the CSR should be cleared (by writing them with a one) to negate the interrupt request to the CPM interrupt controller.

7.6.4.4 REQUESTING IDMA TRANSFERS. Once the IDMA has been started, the transfers can be requested to the IDMA.

IDMA transfers may be initiated by either internally or externally generated requests. Internally generated requests can be initiated by setting STR in the CMR or, in auto buffer and buffer chaining modes, by also setting RCI in the CMR and preparing a data buffer to the RISC controller. Externally generated transfers are those requested by an external device using DREQx in conjunction with the activation of STR.

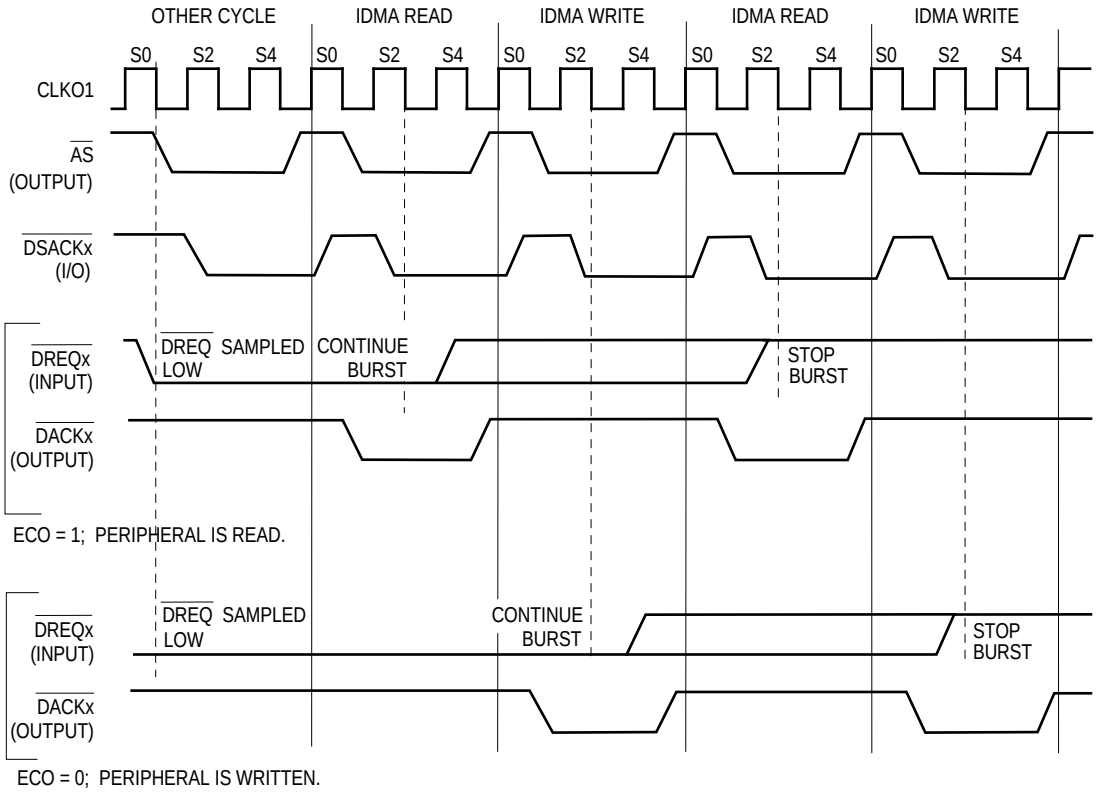
7.6.4.4.1 Internal Maximum Rate. The first method of internal request generation is a non-stop transfer until the transfer count is exhausted. If this method is chosen, the IDMA will arbitrate for the bus and begin transferring data after STR is set and the IDMA becomes the bus master. During each access to the device (determined by the ECO bit in the CMR), the IDMA will assert DACKx to indicate to the device that it is being serviced. If no exception occurs, all operands in the data block will be transferred in one burst with the IDMA using 100% of the available bus bandwidth (unless a higher priority bus master requests the bus or a higher priority interrupt requests service). See 7.6.2.2 Channel Mode Register (CMR) for more detail.

7.6.4.4.2 Internal Limited Rate. To guarantee that the IDMA will not use all the available system bus bandwidth during a transfer, internal requests can be limited to the amount of bus bandwidth allocated to the IDMA. Programming the REQG bits to internal limited rate and the BT bits to determine the percentage of bandwidth achieves this result. The options are 12.5%, 25%, 50%, or 75% of the bus. As soon as STR is set, the IDMA module arbitrates for the bus and begins to transfer data when it becomes bus master. During each access to the device (determined by the ECO bit in the CMR), the IDMA will assert DACKx to indicate that it is being serviced. If no exception occurs, transfers will continue normally, but the IDMA will not exceed the percentage of bus bandwidth programmed into the control register. The percentage is calculated over each ensuing 1024 internal clock cycle period.

For example, if 12.5% is chosen, the IDMA will attempt to use the bus for the first 128 clocks of each 1024 clock cycle period. However, because of other bus masters or higher priority

interrupts, the IDMA may not be able to take its 128 clock allotment in a single burst. If, for whatever reason, the IDMA is not able to take its full 128 clock allotment in a 1024 clock cycle period, the IDMA is still only granted a 128 clock allotment in the next 1024 clock cycle period.

7.6.4.4.3 External Burst Mode. For external devices requiring very high data transfer rates, the external burst mode allows the IDMA to use all of the bus bandwidth to service the device (see Figure 7-10). In the burst mode, the $\overline{\text{DREQx}}$ input to the IDMA is level-sensitive and is sampled at falling edges of the clock to determine when a valid request is asserted by the device. The device requests service by asserting $\overline{\text{DREQx}}$ and leaving it asserted. In response, the IDMA begins to arbitrate for the system bus. If $\overline{\text{DREQx}}$ is negated prior to the IDMA winning the bus, the IDMA will cease requesting the bus. If $\overline{\text{DREQx}}$ is negated long enough for the IDMA to win the bus, cycles will continue as long as $\overline{\text{DREQx}}$ is asserted and no higher priority bus master or interrupt occurs.



NOTES:
1. This example assumes dual address mode. In single address mode, the $\overline{\text{DREQx}}$ sample points would occur in every IDMA cycle.
2. This example assumes SRM = 1 in the CMR. If SRM = 0, $\overline{\text{DREQx}}$ would have to be asserted and negated one clock earlier that what is shown to allow it to be internally synchronized by the IDMA before it is used. Alternatively, the timing shown would be correct for the SRM = 0 case if a wait state were included (between S3 and S4) in all cycles shown above.

Figure 7-10. External Burst Requests;

Each time the IDMA issues a bus cycle to either read or write the device, the IDMA will output the $\overline{\text{DACKx}}$ signal. The device is either the source or destination of the transfers, as

determined by the ECO bit in the CMR. The $\overline{\text{DACKx}}$ timing is similar to the timing of the $\overline{\text{AS}}$ pin. Thus, $\overline{\text{DACKx}}$ is the acknowledgment of the original burst request given on the $\overline{\text{DREQx}}$ pin.

During each access to the device (i.e., $\overline{\text{DACKx}}$ is asserted), the IDMA will sample $\overline{\text{DREQx}}$ at the S3 falling edge of the bus cycle to determine whether the burst should continue. If $\overline{\text{DREQx}}$ is asserted, the burst continues. If $\overline{\text{DREQx}}$ is negated, the burst ceases, and another operand transfer to/from the device does not occur until $\overline{\text{DREQx}}$ is asserted again. If $\overline{\text{DREQx}}$ is negated, but not in time to stop the burst on this bus cycle, one additional bus cycle to the device will occur before the IDMA stops the burst.

NOTES

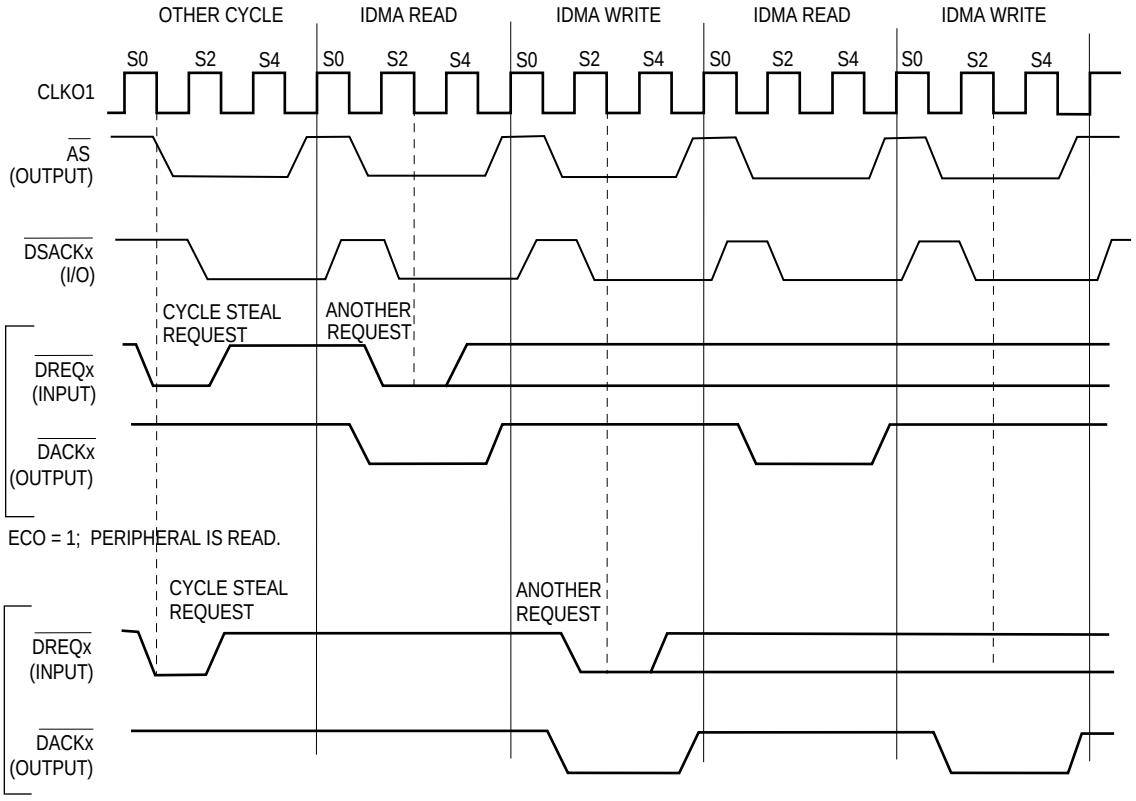
Because $\overline{\text{DACKx}}$ timing is similar to $\overline{\text{AS}}$ timing, the user typically uses the assertion of $\overline{\text{DACKx}}$ as an indication that $\overline{\text{DREQx}}$ is negated.

To meet the S3 sampling time, $\overline{\text{DREQx}}$ should be negated no later than $\overline{\text{DSACKx}}$ because $\overline{\text{DSACKx}}$ pins are also sampled at falling S3 to determine the end of the bus cycle.

The previous paragraphs discuss the general rules; however, important special cases are discussed in the following points:

1. The sample point at the S3 falling edge means the last S3 before the S4 edge that completes the cycle. Thus, if wait states are inserted in the bus cycle, the sample point is later in the cycle.
2. The sample point at S3 assumes that the required setup time is met, as defined in Section 10 Electrical Characteristics.
3. If SRM is cleared in the CMR (default condition), then $\overline{\text{DREQx}}$ is synchronized internally before it is used; therefore, $\overline{\text{DREQx}}$ must be negated one clock earlier than the S3 falling edge to be recognized on that cycle.
4. If operand packing is performed, the user does not need to negate $\overline{\text{DREQx}}$ on any particular access to the device. For instance, if the source is a 32-bit memory and the destination is an 8-bit peripheral, $\overline{\text{DREQx}}$ can be negated on the first, second, third, or fourth byte access to the peripheral. In each case, if the $\overline{\text{DREQx}}$ negation timings are met, the IDMA will stop accessing the peripheral immediately with no additional bus cycles to the peripheral. Accesses to the peripheral will resume when $\overline{\text{DREQx}}$ is asserted.
5. If operand packing is performed and the peripheral is the source and $\overline{\text{DREQx}}$ is negated to stop the burst, the IDMA will attempt to empty the contents of the DHR (by performing one additional write cycle to memory) before giving up the bus. The IDMA attempts to minimize the contents of the DHR between burst requests.
6. If the access to the device is a fast termination access, the $\overline{\text{DREQx}}$ negation timing cannot be met, and one additional bus cycle will always occur to the device before the burst stops.

7.6.4.4.4 External Cycle Steal. For external devices that generate a pulsed signal for each operand to be transferred, the external cycle steal mode should be used. In external cycle steal mode, the IDMA moves one operand for each falling edge of the DREQx input (see Figure 7-11). In this mode, DREQx is sampled at each falling edge of the clock to determine when a valid request is asserted by the device. When the IDMA detects a falling edge on DREQx, a request becomes pending and remains pending until it is serviced by the IDMA. Further falling edges on DREQx are ignored until the request begins to be serviced. The servicing of the request results in one operand being transferred. The operand will be transferred in back-to-back read and write cycles as long as no other higher priority bus master or interrupt occurs between the bus cycles.



NOTES:
1. This example assumes dual address mode. In single address mode, the DREQx sample points would occur in every IDMA cycle.
2. This example assumes SRM = 1 in the CMR. If SRM = 0, DREQx would have to be asserted one clock earlier and remain asserted for one clock longer than what is shown to allow it to be internally synchronized by the IDMA before it is used. Alternatively, the user could assert DREQx as shown and keep DREQx asserted for one additional clock in the SRM = 0 case, if a wait state were included (between S3 and S4) in all cycles shown above.
3. The sample point for "ANOTHER REQUEST" determines that another IDMA transfer will occur following the current IDMA operand transfer. During that time, if the IDMA remains the highest priority bus master of the IMB, the transfers will occur back-to-back as shown.

Figure 7-11. External Cycle Steal

Each time the IDMA issues a bus cycle to either read or write the device, the IDMA will output the DACKx signal. The device is either the source or destination of the transfers, as determined by the ECO bit in the CMR. The DACKx timing is similar to the timing of the AS

pin. Thus, $\overline{\text{DACKx}}$ is the acknowledgment of the original cycle steal request given on the $\overline{\text{DREQx}}$ pin.

It is possible to cause the IDMA to perform back-to-back cycle steal requests. To achieve this, $\overline{\text{DREQx}}$ should be asserted to generate the first request, negated, and reasserted during the access to the device. If the IDMA detects that $\overline{\text{DREQx}}$ is reasserted prior to the S3 falling edge of the bus cycle to the device (i.e., bus cycle when $\overline{\text{DACKx}}$ is asserted), then another back-to-back cycle steal request will be performed. Otherwise, the bus is relinquished. If $\overline{\text{DREQx}}$ was not reasserted soon enough, a new request will be made to the IDMA, but the bus will be relinquished and re-requested by the IDMA.

NOTE

To generate back-to-back cycle steal requests, $\overline{\text{DREQx}}$ should be reasserted after $\overline{\text{DACKx}}$ is asserted, but before the S3 falling edge. Instead of saying before the S3 falling edge, one could also say before or with the assertion of $\overline{\text{DSACKx}}$ because the $\overline{\text{DSACKx}}$ pins are also sampled at falling S3 to determine the end of the bus cycle.

The previous paragraphs discuss the general rules; however, important special cases are discussed in the following points:

1. The sample point at the S3 falling edge means the last S3 before the S4 edge that completes the cycle. Thus, if wait states are inserted in the bus cycle, the sample point is later in the cycle.
2. The sample point at S3 assumes that the required setup time is met, as defined in Section 10 Electrical Characteristics.
3. If SRM is cleared in the CMR (default condition), then $\overline{\text{DREQx}}$ is synchronized internally before it is used; therefore, $\overline{\text{DREQx}}$ must be reasserted one clock earlier than the S3 falling edge to be recognized on that cycle and generate a back-to-back request.

7.6.4.5 IDMA BUS ARBITRATION. Once the IDMA receives a request for a transfer, it begins arbitrating for the IMB. (The four request types are internal maximum rate, internal limited rate, external burst, and external cycle steal.)

On the QUICC, the IDMAs, SDMAs, and DRAM refresh controller, called internal masters, have the capability to become bus master. To determine the relative priority of these masters, each is given an arbitration ID. The user programs the arbitration ID (a value between 0 and 7) of the IDMAs into the ICCR. The arbitration IDs of the two IDMAs must be different by a value of 2 (e.g., IDMA1 ID = 2 and IDMA2 ID = 0). These values are used to determine the relative priority of the IDMA channel and the other internal bus masters.

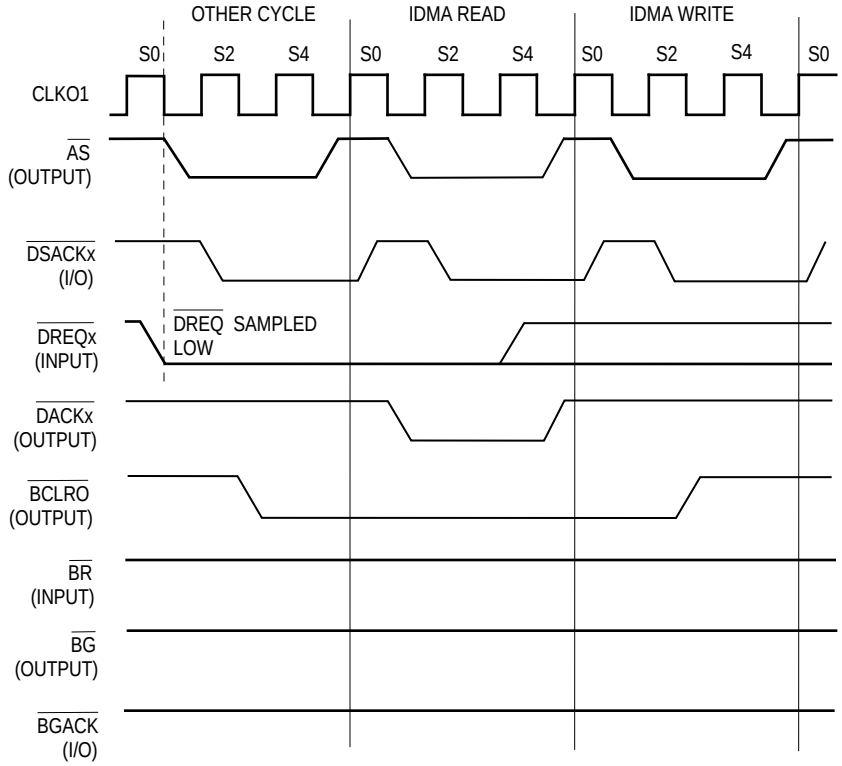
NOTE

Typically, the IDMA IDs are configured by the user to be the lowest of the internal bus masters.

IMB bus masters request bus ownership on a per-cycle basis. Thus, on each bus cycle, the IMB is given to the highest priority bus master requesting the bus. External bus masters may also request the bus and obtain priority over the internal bus masters.

In addition, on the QUICC, interrupts may take priority over bus masters. Thus, another condition for the IDMA to obtain the bus is for the interrupt service level on the IMB to be less than or equal to the interrupt service mask (ISM bits) in the ICCR.

If the CPU32+ is enabled, the IDMA bus arbitration sequence is like that shown in Figure 7-12. The $\overline{BR}$, $\overline{BG}$, and $\overline{BGACK}$ signals are not affected during the arbitration sequence. The only external indication of an IDMA bus request is the bus clear out ($\overline{BCLRO}$) pin. $\overline{BCLRO}$ is only available externally if programmed in the SIM60 port E pin assignment register. Additionally, $\overline{BCLRO}$ is only asserted if the IDMA ID for that channel is greater than the value programmed into the BCLROID2-BCLROID0 bits in the SIM60 module configuration register. $\overline{BCLRO}$ can be used to clear off an external bus master from the external bus, if desired. For instance, $\overline{BCLRO}$ can be connected through logic to the external master's $\overline{HALT}$ signal, and then negated externally when the external master's $\overline{AS}$ signal is negated. $\overline{BCLRO}$ is negated during S2 of the final IDMA bus cycle before it relinquishes the bus.



NOTES:
1. The $\overline{BCLRO}$ signal is only asserted if the IDMA bus arbitration ID is greater than the BCLROID2–BCLROID0 bits in the SIM60 module configuration register.
2. Note that the $\overline{BR}$, $\overline{BG}$, and $\overline{BGACK}$ signals are not affected by the IDMA bus arbitration process if the CPU32+ is enabled.

Figure 7-12. IDMA Bus Arbitration (Normal Operation)

The relative priority between the two IDMAs and SDMA channels is user programmable. Regardless of the system configuration, if the SDMA is a bus master when a higher priority IDMA channel needs to transfer over the bus, the IDMA will steal cycles from the SDMA with no arbitration overhead.

When the QUICC is in slave mode (CPU32+ disabled), the IDMA can steal cycles from the SDMA with no arbitration overhead. See Section 4 Bus Operation for diagrams of bus arbitration by an internal master in slave mode.

Additionally, when the QUICC is in slave mode, the $\overline{\text{BCLRI}}$ pin can be used to force the IDMA and other internal bus masters off the bus. The $\overline{\text{BCLRI}}$ pin is assigned an arbitration ID in slave mode to allow a selection of which internal bus masters are allowed to be forced off the bus. An application of this capability is to connect the $\overline{\text{BCLRO}}$ pin of a QUICC in normal operation to the $\overline{\text{BCLRI}}$ pin of a QUICC in slave mode. This configuration allows the user to implement capabilities such as giving all SDMA channels priority over all IDMA channels in the system.

7.6.4.6 IDMA OPERAND TRANSFERS. Once the IDMA successfully arbitrates for the bus, it can begin making operand transfers. The source IDMA bus cycle has timing identical to an internal master read bus cycle. The destination IDMA bus cycle has timing identical to an internal master write bus cycle.

The two-channel IDMA module supports dual and single address transfers. The dual address operand transfer consists of a source operand read and a destination operand write. Each single address operand transfer consists of one external bus cycle, which allows either a read or a write cycle to occur.

7.6.4.6.1 Dual Address Mode. The two IDMA channels can each be programmed to operate in a dual address transfer mode (see Figure 7-13). In this mode, the operand is read from the source address specified in the SAPR and placed in the DHR. The operand read may take up to four bus cycles to complete because of differences in operand sizes of the source and destination. The operand is then written to the address specified in the DAPR. This transfer may also be up to four bus cycles long. In this manner, various combinations of peripheral, memory, and operand sizes may be used.

The dual address transfers can be started either by the internal request mode or by an external device using $\overline{\text{DREQx}}$. When the external device uses $\overline{\text{DREQx}}$, the channel can be programmed to operate in either the cycle steal or burst transfer modes. See 7.6.4.4.3 External Burst Mode and 7.6.4.4.4 External Cycle Steal for information about these modes.

Dual Address Source Read. During this type of IDMA cycle, the SAPR drives the address bus, the FCR drives the source function codes, and the CMR drives the size control. Data is read from the memory or peripheral and placed in the DHR when the bus cycle is terminated. When the complete operand has been read, the SAPR is incremented by 1, 2, or 4, depending on the address and size information specified by the SAPI and SSIZE bits of the CMR. See 7.6.2.3 Source Address Pointer Register (SAPR) for more information.

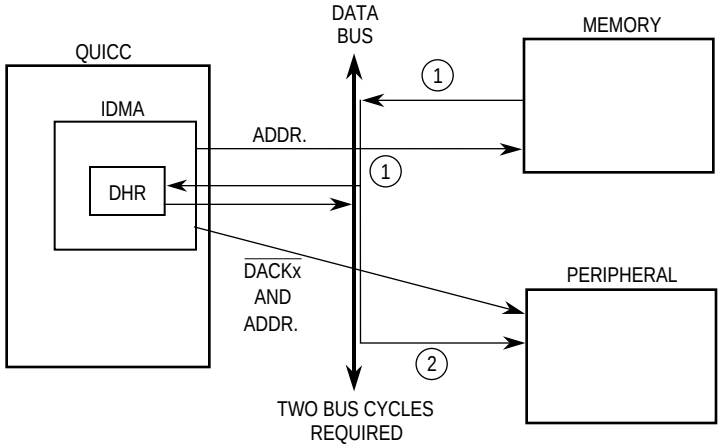


Figure 7-13. Dual Address Transfer Example

Dual Address Destination Write. During this type of IDMA cycle, the data in the DHR is written to the device or memory selected by the address in the DAPR, the destination function codes in the FCR, and the size in the CMR. The same options exist for operand size and alignment as in the dual address source read. When the complete operand is written, the DAPR is incremented by 1, 2, or 4, according to the DAPI and DSIZE bits of the CMR, and the BTC is decremented by the number of bytes transferred. If the BTC is equal to zero, the $\overline{\text{DONEx}}$ signal for the IDMA handshake is asserted, and if the transfer is completed with no errors, the DONE bit in the CSR is set. See 7.5.2.4 Timer Reference Registers (TRR1, TRR2, TRR3, TRR4) and 7.6.2.6 Byte Count Register (BCR) for more information.

Dual Address Packing. When dual address mode is selected, the IDMA can perform packing. Regardless of the source size, destination size, source starting address, or destination starting address, the IDMA will use the most efficient packing algorithm possible to perform the transfer in the fewest possible number of bus cycles.

NOTE

The packing algorithms are subject to the restriction that the IDMA never performs 3-byte transfers.

Three examples of the packing technique follow.

Example 1. This simple example shows how packing is performed when the source and destination sizes are the same—word. The source address is $00000001, and the destination address is $20000000. The number of bytes to be transferred is 4.

IDMA channel 1 initialization required for this example:

—ICCR = $0720. Recommended normal configuration.
—FCR1 = $89. Source function code is 1000; destination function code is 1001.
—SAPR1 = $00000001. Source address.
—DAPR1 = $20000000. Destination address.
—BCR1 = $00000003. Byte transfer count.

—CSR1 = $FF. Clear any CSR bits that are currently set.
—CMAR1 = $00. Disable interrupts for this example.
—CMR1 = $47A1. Internal maximum transfer rate; starts IDMA.

Bus Access #	Address (Hex)	Operation	No. Bytes	No. Bytes in DHR
1	$00000001	Read	1	1
2	$00000002	Read	2	3
3	$20000000	Write	2	1
4	$00000002	Write	1	0

Example 2. This more complicated example shows how packing is performed when the source and destination sizes are the same—long word. This example also shows the entire 7-byte DHR in use. The source address is $00000000, and the destination address is $20000003. The number of bytes to be transferred is 16.

IDMA channel 1 initialization required for this example:

- ICCR = $0720. Recommended normal configuration.
- FCR1 = $89. Source function code is 1000; destination function code is 1001.
- SAPR1 = $00000000. Source address.
- DAPR1 = $20000003. Destination address.
- BCR1 = $00000010. Byte transfer count.
- CSR1 = $FF. Clear any CSR bits that are currently set.
- CMAR1 = $00. Disable interrupts for this example.
- CMR1 = $4701. Internal maximum transfer rate; starts IDMA.

Bus Access #	Address (Hex)	Operation	No. Bytes	No. Bytes in DHR
1	$00000000	Read	4	4
2	$20000003	Write	1	3
3	$00000004	Read	4	7
4	$20000004	Write	4	3
5	$00000008	Read	4	7
6	$20000008	Write	4	3
7	$0000000C	Read	4	7
8	$2000000C	Write	4	3
9	$20000010	Write	2	1
10	$20000012	Write	1	0

Example 3. This example shows how packing operates when the source and destination sizes are different. The source address is $00000002, and the destination address is $20000002. The source size is long word, and the destination size is byte. The number of bytes to be transferred is 8.

IDMA channel 1 initialization required for this example:

- ICCR = $0720. Recommended normal configuration.
- FCR1 = $89. Source function code is 1000; destination function code is 1001.
- SAPR1 = $00000002. Source address.
- DAPR1 = $20000002. Destination address.
- BCR1 = $00000008. Byte transfer count.
- CSR1 = $FF. Clear any CSR bits that are currently set.
- CMAR1 = $00. Disable interrupts for this example.
- CMR1 = $4711. Internal maximum transfer rate; starts IDMA.

Bus Access #	Address (Hex)	Operation	No. Bytes	No. Bytes in DHR
1	$00000002	Read	2	2
2	$20000002	Write	1	1
3	$20000003	Write	1	0
4	$00000004	Read	4	4
5	$20000004	Write	1	3
6	$20000005	Write	1	2
7	$20000006	Write	1	1
8	$20000007	Write	1	0
9	$00000008	Read	2	2
10	$20000008	Write	1	1
11	$20000009	Write	1	0

7.6.4.6.2 Single Address Mode (Flyby Transfers). Each IDMA channel can be independently programmed to provide single address transfers. Figure 7-14 illustrates a transfer from memory to a peripheral. The DHR is not used by the IDMA, since the transfer occurs directly from a device to memory. This mode is often referred to as "flyby" mode because the DHR is not used.

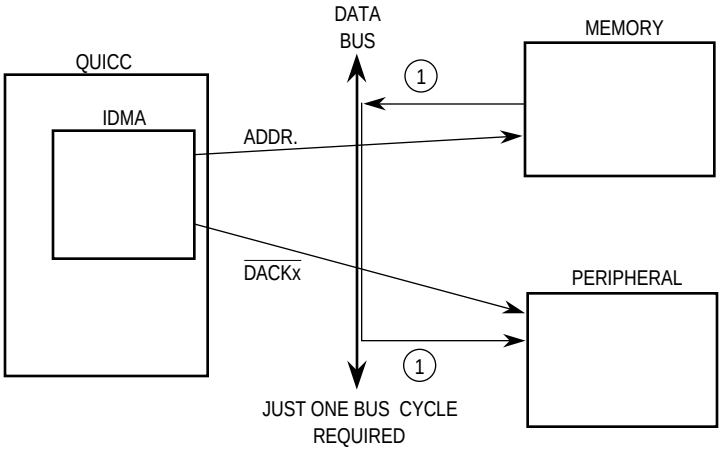
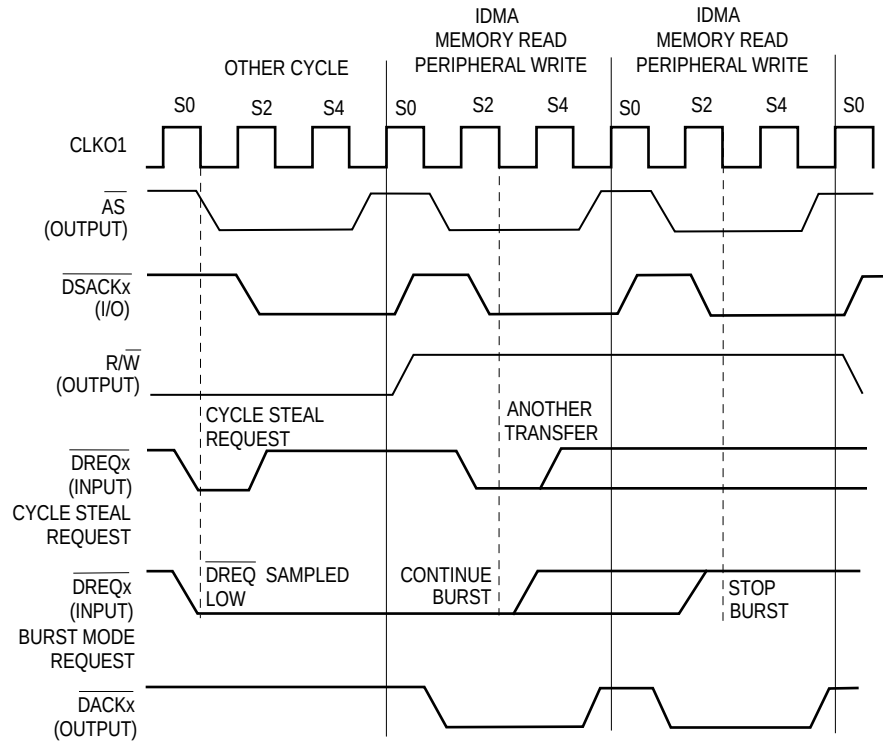


Figure 7-14. Single Address Transfer Example

Both internal and external request modes can be used to start a transfer when the single address mode is selected (see Figure 7-15). The ECO bit in the CMR controls whether a source read or a destination write cycle occurs on the data bus. If the ECO bit is set, the external handshake signals are used with the source operand, and a single address source read occurs. If the ECO bit is cleared, the external handshake signals are used with the destination operand, and a single address destination write occurs.

NOTE

Single address mode does not support access to the internal dual port ram of MC68360. In order to transfer from/to internal dual port ram, user should use dual address mode.



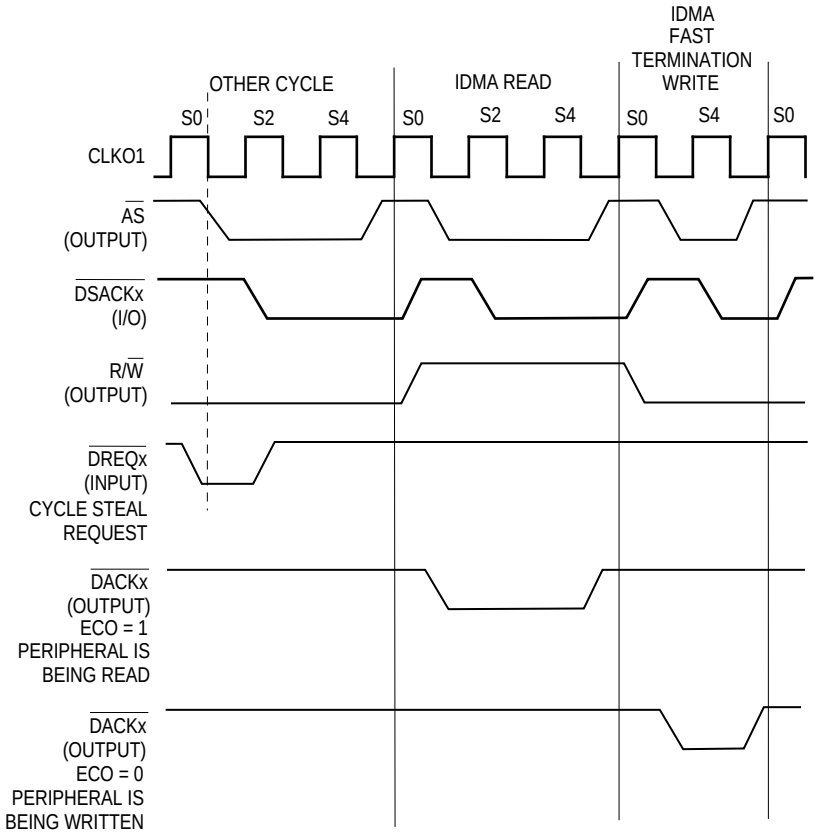
NOTE:
1. This example assumes the peripheral is being written. If the peripheral is being read, R/W would be low during the transfers.
2. This example shows the operation of DREQ in two different modes.
3. This example assumes that SRM = 0 in the CMR. Otherwise, DREQx would not be recognized by the IDMA until it had been sampled on two consecutive falling edges of the clock.

Figure 7-15. Single Address Mode Timing

Single Address Source Read. During the single address source read cycle, the device or memory selected by the address in the SAPR, the source function codes in the FCR, and the size in the CMR provides the data and control signals on the data bus. This bus cycle operates like a normal read bus cycle. The destination device is controlled by the IDMA handshake signals (DREQx, DACKx, and DONEx). The assertion of DACKx provides the write control to the destination device. For more details about the IDMA handshake signals, see 7.6.3 Interface Signals.

Single Address Destination Write. During the single address destination write cycle, the source device is controlled by the IDMA handshake signals (DREQx, DACKx, and DONEx). When the source device requests service from the IDMA channel, the IDMA asserts of DACKx to allow the source device to drive data onto the data bus. The data is written to the device or to memory selected by the address in the DAPR, the destination function codes in the FCR, and the size in the CMR. The data bus is placed in a high-impedance state for this write cycle. For more details about the IDMA handshake signals, see 7.6.3 Interface Signals.

7.6.4.6.3 Fast-Termination Option. While in the operand transfer phase, the IDMA supports an option to achieve a transfer in the shortest possible number of clocks (see Figure 7-16).



NOTE: This example shows a fast termination on the write cycle. The fast termination may occur on the read, write, or both.

Figure 7-16. Fast Termination Example

Using the SIM60 chip-select logic, the fast-termination option can be employed to give a fast bus access of two clock cycles rather than the standard three-cycle access time. The fast-termination option is described in Section 6 System Integration Module (SIM60) and in Section 4 Bus Operation.

If the fast-termination option is used with external request burst mode, an extra IDMA cycle results on every burst transfer. In the burst mode with fast termination selected, a new cycle starts even if DREQx negation and DACKx assertion occur simultaneously.

7.6.4.6.4 Externally Recognizing IDMA Operand Transfers. There are several methods to externally determine that a bus cycle is being executed by the IDMA:

1. The function code lines may be programmed to a unique function code that identifies an IDMA transfer.
2. The $\overline{\text{BCLRO}}$ pin can be used to show when the bus request is made. $\overline{\text{BCLRO}}$ is negated during the final access by the IDMA before relinquishing the bus.
3. The $\overline{\text{DACKx}}$ signal shows accesses to the peripheral device. $\overline{\text{DACKx}}$ will operate even in the internal request modes and will activate on either the source or destination bus cycles, depending on the ECO bit in the CMR.

NOTE

Items 1 and 2 may also be used by the SDMA channels.

7.6.4.7 BUS EXCEPTIONS. While the IDMA has the bus and is performing operand transfers, it is possible for bus exceptions to occur.

In any computer system, the possibility exists that an error will occur during a bus cycle due to a hardware failure, random noise, or an improper access. When an asynchronous bus structure, such as that supported by the M68000 is used, it is easy to make provisions allowing a bus master to detect and respond to errors during a bus cycle. The IDMA recognizes the same bus exceptions as the CPU32+ core: reset, bus error, halt, and retry.

7.6.4.7.1 Reset. Upon an external reset, the IDMA immediately aborts the channel operation, returns to the idle state, and clears CSR and CMR (including the STR bit). If a bus cycle is in progress when reset is detected, the cycle is terminated, the control and address/data pins are three-stated, and bus ownership is released. The IDMA can also be reset by RST in the CMR.

7.6.4.7.2 Bus Error. When a fatal error occurs during a bus cycle, a bus error exception is used to abort the cycle and systematically terminate that channel's operation. The IDMA terminates the current bus cycle, signals an error in the CSR using either the BES or BED bit, and signals an interrupt if the corresponding bit in the CMAR is set. The IDMA clears STR and waits for a restart of the channel and the negation of $\overline{\text{BERR}}$ before starting any new bus cycles. Any data that was previously read from the source into the DHR will be lost.

NOTE

Any device that is the source or destination of the operand under IDMA handshake control for single address transfers may need to monitor $\overline{\text{BERR}}$ to detect a bus exception for the current bus cycle. $\overline{\text{BERR}}$ terminates the cycle immediately and negates $\overline{\text{DACKx}}$, which is used to control the transfer to or from the device.

7.6.4.7.3 Retry. When $\overline{\text{HALT}}$ and $\overline{\text{BERR}}$ are asserted during a bus cycle, the IDMA terminates the bus cycle, releases the bus, and suspends further operation until these signals are negated. When $\overline{\text{HALT}}$ and $\overline{\text{BERR}}$ are negated, the IDMA will arbitrate for the bus, re-execute the previous bus cycle, and continue normal operation.

If the IDMA has obtained the IMB and is also waiting to obtain the external bus, but the external bus master performs an access to a location internal to the QUICC, the IDMA will relinquish the IMB and retry the cycle once it has obtained the IMB.

7.6.4.8 ENDING THE IDMA TRANSFER. If no bus exceptions occur, the IDMA eventually finishes the transfer of a block of data. These paragraphs describe normal termination in more detail. (Termination by error is discussed in 7.6.4.7.2 Bus Error.)

The IDMA channel operation experiences normal termination when the BCR is decremented to zero or the external device signals a termination of the transfer using $\overline{\text{DONE}}$x. These terminations are independent of how requests are generated to the IDMA.

Additionally, the user may stop the IDMA channel by clearing STR. However, this is considered a suspension of activity, rather than normal termination, since the transfer resumes when STR is set once again.

The user may also terminate the transfer by setting the RST bit in the CMR; however, this is not a normal termination of IDMA activity.

Further description of normal termination depends on the mode of the IDMA: single buffer mode, auto buffer mode, and buffer chaining. These modes are described in the following paragraphs.

7.6.4.8.1 Single Buffer Mode Termination. The following methods may be used to terminate an IDMA transfer in the single buffer mode. They may also be used to terminate a current BD transfer in the auto buffer and buffer chaining modes.

Transfer Count Exhausted. When the channel performs an operand transfer, it decrements the BCR for each byte transferred successfully. When the BCR is decremented to zero, the transfer is terminated. When the last bus cycle of the transfer occurs (either a byte, word, or long-word access), $\overline{\text{DONE}}$x is asserted during that bus cycle. If the device is the source, further destination accesses will take place after $\overline{\text{DONE}}$x is asserted. If the device is the destination, $\overline{\text{DONE}}$x will be asserted on the final bus cycle of the destination write.

NOTE

This behavior of $\overline{\text{DONE}}$x also applies to memory-to-memory transfers. $\overline{\text{DONE}}$x is asserted on either the last source or destination bus cycle, as determined by the ECO bit in the CMR.

When the operand transfer has completed and the BCR has been decremented to zero, the channel operation is terminated, STR is cleared, and a DONE bit interrupt is generated if the corresponding CMAR bit is set. The SAPR and/or DAPR are also incremented in the normal fashion.

NOTE

If the channel was started with the BCR value set to zero, the channel will transfer 4 Gbytes before the transfer count is exhausted.

External Device Termination. If the $\overline{\text{DONEx}}$ pin is asserted externally, a transfer may be terminated by the device even before the BCR is decremented to zero. $\overline{\text{DONEx}}$ is sampled by the IDMA on the access to the device.

NOTE

This behavior of $\overline{\text{DONEx}}$ also applies to memory-to-memory transfers. $\overline{\text{DONEx}}$ is sampled on either the source or destination bus cycles, as determined by the ECO bit in the CMR.

If $\overline{\text{DONEx}}$ is asserted on a bus cycle to a source device, the destination accesses will be performed before the IDMA terminates transfers. If $\overline{\text{DONEx}}$ is asserted during a bus cycle to a destination device, no further IDMA bus cycles occur, and the IDMA terminates transfers.

The IDMA samples $\overline{\text{DONEx}}$ on the S3 falling edge of the bus cycle. Thus, the user should assert $\overline{\text{DONEx}}$ at least one setup time before the S3 falling edge for $\overline{\text{DONEx}}$ to be recognized on that bus cycle.

NOTES

Because $\overline{\text{DACKx}}$ timing is similar to $\overline{\text{AS}}$ timing, the user uses the assertion of $\overline{\text{DACKx}}$ as an indication that $\overline{\text{DONEx}}$ is asserted.

To meet the S3 sampling time, $\overline{\text{DONEx}}$ should be asserted no later than $\overline{\text{DSACKx}}$ because the $\overline{\text{DSACKx}}$ pins are also sampled at falling S3 to determine the end of the bus cycle.

The previous paragraphs discuss the general rules; however, important special cases are discussed in the following points:

1. The sample point at the S3 falling edge means the last S3 before the S4 edge that completes the cycle. Thus, if wait states are inserted in the bus cycle, the sample point is later in the cycle.
2. The sample point at S3 assumes that the required setup time is met, as defined in Section 10 Electrical Characteristics.
3. If SRM is cleared in the CMR (default condition), then $\overline{\text{DONEx}}$ is synchronized internally before it is used; therefore, $\overline{\text{DONEx}}$ must be negated one clock earlier than the S3 falling edge to be recognized on that cycle.
4. If the device is configured to be the source and dual address mode, the sample point used by the IDMA is S5 rather than S3. This gives the user one additional clock to assert the $\overline{\text{DONEx}}$ signal.

When the operand transfer has terminated, STR is cleared, and a DONE bit interrupt is generated if the corresponding CMAR bit is set. The SAPR and/or DAPR are also incremented in the normal fashion, and the BCR is decremented.

7.6.4.8.2 Auto Buffer Mode Termination. The user can suspend a transfer in auto buffer mode by clearing the STR bit in the CMR. When STR is set once again, the transfer will continue.

The user can terminate the transfer by setting the RST bit in the CMR and then issuing the INIT_IDMA command.

The user can terminate the transfer with an "out of buffers" error if the V-bit of one of the BDs is cleared by the user. When the RISC reaches this IDMA BD, it will terminate activity. This technique is useful when the IDMA is required to stop transfers after fully completing a BD transfer.

If the BCR is decremented to zero, the transfer from this BD completes, but the RISC controller reloads the IDMA registers with the values from the next IDMA BD, and the IDMA transfer continues. Thus, the fact that the BCR is decremented to zero does not terminate a transfer in auto buffer mode; it only terminates the current BD transfer.

If DONEx is asserted externally, the transmission from this BD is terminated and the following actions are performed by the RISC controller:

1. Sets the Done Bit in the status register
2. Sets the DA bit in the BD
3. Clears the Valid bit in the BD
4. Resets the start bit in the CMR

Thus the current buffer is closed immediately and all IDMA operation ceases.

7.6.4.8.3 Buffer Chaining Mode Termination. The user can suspend a transfer in auto buffer mode by clearing the STR bit in the CMR. When STR is set once again, the transfer will continue.

The user can terminate the transfer by setting the RST bit in the CMR and then issuing the INIT_IDMA command.

The user can also terminate the transfer by setting the L-bit in the IDMA BD. When processing of this BD has completed, the transmission will terminate with the DONE bit being set in the CSR. This can cause an interrupt if the corresponding bit in the CMAR is set.

If the BCR is decremented to zero, the transfer from this BD completes, but the RISC controller reloads the IDMA registers with the values from the next IDMA BD, and the IDMA transfer continues. Thus, the fact that the BCR is decremented to zero does not terminate a transfer in buffer chaining mode; it only terminates the current BD transfer.

If $\overline{\text{DONEx}}$ is asserted externally, the transmission from this BD is terminated and the following actions are performed by the RISC controller.

1. Sets the Done Bit in the status register
2. Sets the Abort bit in the BD
3. Clears the Ready bit in the BD
4. Resets the start bit in the CMR
5. Sets the Reset bit in the CMR

7.6.5 IDMA Examples

The following paragraphs provide IDMA examples.

7.6.5.1 SINGLE BUFFER EXAMPLES. To see three examples of single buffer operation, see the 7.6.4.6.1 Dual Address Mode.

7.6.5.2 BUFFER CHAINING EXAMPLE. •The following example shows the setup required to initialize IDMA channel 1 to perform three buffer transfers using the buffer chaining mode. This example will move 16 bytes from address 0 to address $1000, then 16 bytes from address $100 to $1100, and then 16 bytes from address 200 to $1200.

1. Initialize basic IDMA channel 1 registers:
2. ICCR = $0720. Recommended normal configuration.
3. FCR1 = $89. Source function code is 1000; destination function code is 1001.
4. SAPR1 not initialized. Will be initialized later by RISC controller.
5. DAPR1 not initialized. Will be initialized later by RISC controller.
6. BCR1 not initialized. Will be initialized later by RISC controller.
7. CSR1 = $FF. Clear any CSR bits that are currently set.
8. CMAR1 = $00. Disable interrupts for this example.
9. CMR1 = $530C. The RISC controls the IDMA activity (RCI bit is set). The IDMA channel uses 12.5% of the bus bandwidth. The source and destination size are long word. Do not set the STR bit yet.
10. Issue the INIT_IDMA command to the RISC controller. This command is not required unless the IDMA was reset with the CMR RST bit while in the buffer chaining or auto buffer modes.
11. CR = $0591. Issue INIT_IDMA command to IDMA channel 1.
12. Initialize the IDMA channel 1 parameter RAM:
13. IBASE = $0000. This points the beginning of the IDMA BDs. The value of $0000 means that the first IDMA BD is located at the beginning of the internal dual-port RAM.
14. Initialize the first IDMA BD:
15. BD1_STATUS = $0000. This is offset 0 from the BD. Set up all bits except the V-bit.
16. BD1_Data_Length = $00000010. Transfer 16 bytes.
17. BD1_Source_Pointer = $00000000. Source address.
18. BD1_Destination_Pointer = $00001000. Destination address.
19. BD1_STATUS = $8000. Set the V-bit. It is good practice to set the V-bit last; however, in this example the IDMA channel is not yet enabled, so it could have been set earlier.
20. Initialize the second IDMA BD:
21. BD2_STATUS = $0000. This is offset 0 from the BD. Set up all bits except the

V-bit.

22. BD2_Data_Length = $00000010. Transfer 16 bytes.
23. BD2_Source_Pointer = $00000100. Source address.
24. BD2_Destination_Pointer = $00001100. Destination address.
25. BD2_STATUS = $8000. Set the V-bit. It is good practice to set the V-bit last; however, in this example the IDMA channel is not yet enabled, so it could have been set earlier.
26. Initialize the third IDMA BD:
27. BD3_STATUS = $2800. This is offset 0 from the BD. Set up all bits except the V-bit. In this case, set the L-bit to indicate that the IDMA should stop after this BD, and set the DONE bit in the CSR. Additionally, set the W-bit to cause the RISC to point to the first BD when done. The W-bit should always be set in the last BD of the list.
28. BD3_Data_Length = $00000010. Transfer 16 bytes.
29. BD3_Source_Pointer = $00000200. Source address.
30. BD3_Destination_Pointer = $00001200. Destination address.
31. BD3_STATUS = $A800. Set the V-bit. It is good practice to set the V-bit last; however, in this example the IDMA channel is not yet enabled, so it could have been set earlier.
32. Start the IDMA channel:
33. CMR1 = $530D. Set the STR bit of this register. The IDMA now begins transferring all three BDs.
34. Check for successful completion:
35. Read the CMR and wait for the STR bit to be cleared, indicating the end of the transfer. Read the CSR to see what status has been set. In this case, only the DONE bit should be set. The AD bit would only be set if the I-bit of the BD_STATUS field had been set.

7.6.5.3 AUTO BUFFER EXAMPLE. The previous buffer chaining example can be easily modified to show the auto buffer operation. Simply set the CM bit in the BD_STATUS words of each of the three BDs, and for the sake of clarity, clear the L-bit of the third BD. The IDMA channel will then repeatedly transfer groups of 16 bytes until the STR bit is cleared in software, the IDMA is reset, or the V-bit is cleared in one of the IDMA BDs.

NOTE

Use of the IDMA internal maximum rate option in the auto buffer mode is not recommended because the CPU32+ would only be able to execute instructions during the brief period that the RISC is configuring the IDMA channel between BDs. These bits MUST be set to 7 if the QUICC is in slave mode.

7.7 SDMA CHANNELS

Fourteen SDMA channels are present on the QUICC. Eight are associated with the four full-duplex SCCs. The other six are assigned to the service of the SPI and the two SMCs. Each channel is permanently assigned to service either the receive or transmit operation of an SCC, SMC, or SPI.

Figure 7-17 shows the paths of the data flow. Data from the SCCs, SMCs, and SPI may be routed to the external RAM (path 1) or the internal dual-port RAM (path 2). In both cases, however, the IMB is used for the data transfer. On a path 1 access, the IMB and the external system bus must be acquired by the SDMA channel. On a path 2 access, only the IMB needs to be acquired, and the access will not be seen on the external system bus unless the QUICC is configured into the "show cycles" mode of the SIM60. Thus, the transfer on the IMB can occur while other operations occur simultaneously on the external system bus.

Each SDMA channel may be programmed to output one of 16 function codes. The function codes are used to identify the channel that is currently accessing memory. Also, the SDMA channel may be assigned a big endian (Motorola) or little endian format for accessing buffer data. These features are programmed in the receive and transmit function code registers associated with the SCCs, SMCs, and SPI.

If a bus error occurs on an access by the SDMA, the CPM generates a unique interrupt in the SDMA status register. The interrupt service routine then reads the SDMA address register to determine the address on which the bus error occurred. The channel that caused the bus error is determined by reading the Rx internal data pointer and Tx internal data pointers from the specific protocol parameters area in the parameter RAM for the serial channels. If an SDMA bus error occurs, all CP activity ceases, and the entire CP must be reset in the command register.

7.7.1 SDMA Bus Arbitration and Bus Transfers

On the QUICC, the SDMA, IDMA, and DRAM refresh controller can become internal bus masters. To determine the relative priority of these masters, each is given an arbitration ID. The 14 SDMA channels share the same ID, which is programmed by the user. Therefore, any SDMA channel can arbitrate for the bus against the other internal masters and any external masters that are present.

Once an SDMA channel obtains the system bus, it remains the bus master for one long-word transfer before relinquishing the bus. This feature, in combination with the zero clock arbitration overhead provided by the IMB, allows the simultaneous benefits of bus efficiency and low bus latency.

In the case of character-oriented protocols, the SDMA writes characters to memory (it does not wait for multiple characters to be received before writing), but the SDMA always reads long words. This is consistent with the goal of providing low-latency operation on character-oriented protocols that tend to be used at slower rates.

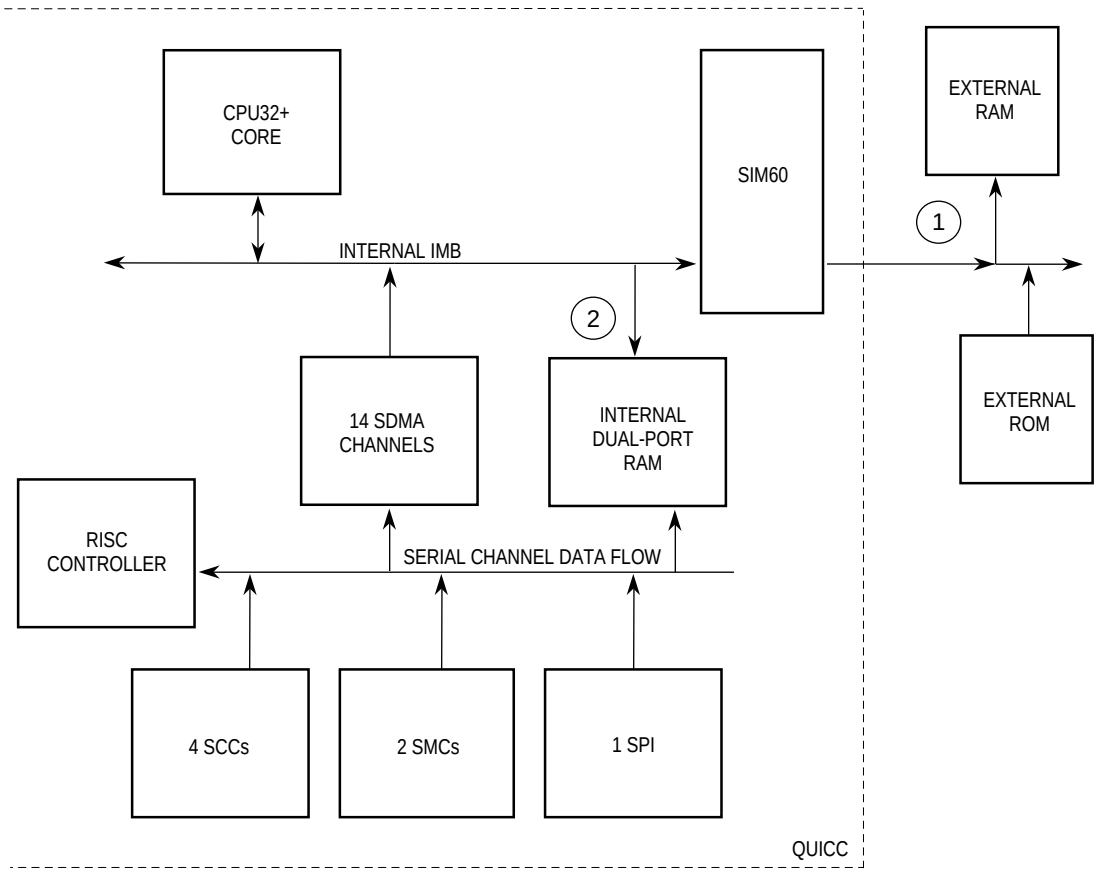
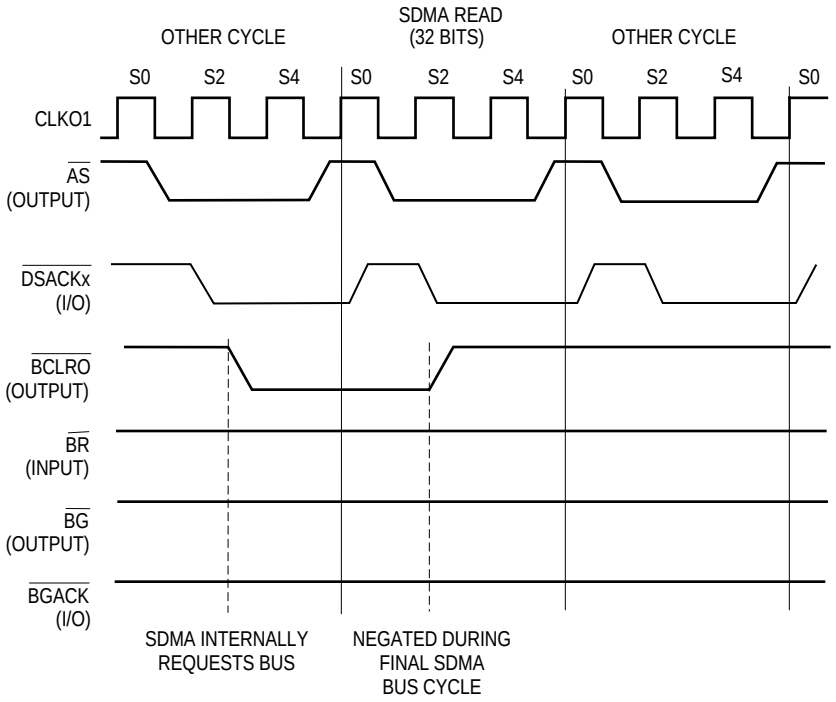


Figure 7-17. SDMA Data Paths

The read or write operation may take multiple bus cycles if the memory provides less than a 32-bit port size. For instance, a 32-bit long-word read from a 16-bit memory will take two SDMA bus cycles. As long as a higher priority bus master does not require the bus during an SDMA transfer, the entire operand (32 bits on reads and 8, 16, or 32 bits on writes) will be transferred in back-to-back bus cycles before the SDMA relinquishes the bus. If a higher priority bus master requests the bus during an operand transfer, it will be granted the bus at the end of that SDMA bus cycle. Once the higher priority bus master relinquishes the bus, the SDMA will reacquire the bus and continue any outstanding bus cycles.

The SDMA can steal cycles with no arbitration overhead when the QUICC is in master mode (i.e., the CPU32+ is enabled) and the external bus is not currently being held by an external master (see Figure 7-18). Note that in normal operation, the $\overline{BR}$, $\overline{BG}$, and $\overline{BGACK}$ signals are not affected by the SDMA; however, an indication of the SDMA internal bus request can be obtained from the $\overline{BCLRO}$ signal.

The SDMA will assert the $\overline{BCLRO}$ signal when it requests the bus if this capability is programmed in the SIM60 module configuration register and port E pin assignment register. $\overline{BCLRO}$ can be used to clear an external bus master from the external bus, if desired. For instance, $\overline{BCLRO}$ can be connected through logic to the external master's $\overline{HALT}$ signal, and then be negated externally when the external master's $\overline{AS}$ signal is negated. $\overline{BCLRO}$, as seen from the QUICC, is negated by the SDMA during its access to memory.



NOTES:
1. The BCLRO signal is only asserted if the SDMA bus arbitration ID is greater than the BCLROID2–BCLROID0 bits in the SIM60 module configuration register.
2. The BR, BG, and BGACK signals are not affected by the SDMA bus arbitration process if the CPU32+ is enabled.

Figure 7-18. SDMA Bus Arbitration (Normal Operation)

The relative priority between the two IDMAs and the SDMA channels is user programmable. Regardless of system configuration, if the IDMA is a bus master when a higher priority SDMA channel needs to transfer over the bus, the SDMA will steal cycles from the IDMA with no arbitration overhead.

When the QUICC is in slave mode (CPU32+ is disabled) the SDMA can steal cycles from the IDMA with no arbitration overhead. See Section 4 Bus Operation for diagrams of bus arbitration by an internal master in slave mode.

7.7.2 SDMA Registers

The SDMA channels have one configuration register; otherwise, they are controlled transparently to the user, through the configuration of the SCCs, SMCs, and SPI. The only user-accessible registers associated with the SDMA are the SDMA configuration register (SDCR), SDMA address register (SDAR), a read-only register used for diagnostics in case of an SDMA bus error, and the SDMA status register (SDSR).

7.7.2.1 SDMA CONFIGURATION REGISTER (SDCR). The 16-bit SDCR is used to configure all 14 SDMA channels. It is always readable and writable in the supervisor mode, although writing the SDCR is not recommended unless the CP is disabled. SDCR is cleared at reset.

15	14 13	12 11	10 9 8	7	6 5 4	3 2	1	0
—	FRZ	—	SISM	—	SAID	—	INTE	INTB

Bits 15, 12, 11, 7, 2—Reserved

FRZ1–FRZ0—Freeze

These bits determine the action to be taken when the FREEZE signal is asserted. The SDMA negates $\overline{BR}$ and keeps it negated until FREEZE is negated or a reset occurs.

00 = The SDMA channels ignore the FREEZE signal.
01 = Reserved.
10 = The SDMA channels freeze on the next bus cycle.
11 = Reserved.

SISM—SDMA Interrupt Service Mask

These bits contain the interrupt service mask. When the interrupt service level on the IMB is greater than the interrupt service mask, the SDMA relinquishes the bus and negates the internal bus request to the IMB until the interrupt level service is less than or equal to the interrupt service mask.

NOTE

This value should be programmed to 7 for typical user applications. This level gives the SDMA channels priority over all interrupt handlers.

SAID—SDMA Arbitration ID

These bits establish bus arbitration priority level among modules that have the capability of becoming bus master. In the QUICC, the DRAM refresh controller, IDMAs, SDMAs, and external bus masters can obtain bus mastership. The SDMA channel arbitration ID is determined by these bits. Zero is the lowest priority, and seven is the highest priority.

NOTE

This value should be programmed to 4 for typical user applications. This value should always be programmed to a value larger than the arbitration IDs for the two IDMA channels. The user must program this field to 7 when the QUICC is configured in slave mode.

INTE—Interrupt Error

This bit enables the SBER status bit in the SDSR.

0 = A zero masks the interrupt generated by the corresponding bit in the SDSR. If a bus error occurs while the SDMA is bus master, the channel does not generate an interrupt to the QUICC interrupt controller. The SBER bit is still set in the SDSR.
1 = If a bus error occurs while the SDMA is bus master, the channel generates an interrupt to the QUICC interrupt controller and sets the SBER bit in the SDSR.

NOTE

An interrupt will only be generated if the SDMA bit is set in the CP interrupt mask register.

INTB—Interrupt Breakpoint

This bit is the enable bit for the SBKP status bit in the SDSR.

0 = A zero masks the interrupt generated by the corresponding bit in the SDSR. When a breakpoint is recognized while the SDMA is bus master, the channel does not generate an interrupt to the QUICC interrupt controller. The SBKP bit is still set in the SDSR.

1 = When a breakpoint is recognized while the SDMA is bus master, the channel generates an interrupt to the QUICC interrupt controller and sets the SBKP bit in the SDSR.

NOTE

An interrupt will only be generated if the SDMA bit is set in the CP interrupt mask register. The interrupt can suspend SDMA activity immediately if it is programmed to be at a higher level than the SDMA channels. Alternatively, the interrupt can be processed after the SDMA transfer is complete.

7.7.2.2 SDMA STATUS REGISTER (SDSR). Shared by all 14 SDMA channels, the SDSR is an 8-bit register used to report events recognized by the SDMA controller. On recognition of an event, the SDMA sets its corresponding bit in the SDSR (regardless of the INTE, INTB, and INTR bits in the SDCR). The SDSR is a memory-mapped register that may be read at any time. A bit is reset by writing a one and is left unchanged by writing a zero. More than one bit may be reset at a time, and the register is cleared by reset.

7	6	5	4	3	2	1	0
—					RINT	SBER	SBKP

Bits 7–3—Reserved

RINT—Reserved Interrupt

This status bit is reserved for factory testing. RINT is cleared by writing a one; writing a zero has no effect.

SBER—SDMA Channel Bus Error

This bit indicates that the SDMA channel terminated with an error during a read or write cycle. The SDMA bus error address can be read from the SDAR. SBER is cleared by writing a one; writing a zero has no effect.

SBKP—SDMA Breakpoint

This bit indicates that the breakpoint signal was asserted during an SDMA transfer. SBKP is cleared by writing a one; writing a zero has no effect.

7.7.2.3 SDMA ADDRESS REGISTER (SDAR). The 32-bit read-only SDAR shows the system address that was accessed during an SDMA bus error. It is undefined at reset.

7.8 SERIAL INTERFACE WITH TIME SLOT ASSIGNER

The SI connects the physical layer serial lines to the four SCCs and two SMCs (see Figure 7-19). In its simplest configuration, the SI allows the four SCCs and two SMCs to be connected their own set of individual pins. Each SCC or SMC that connects to the external world in this way is said to connect to an NMSI. In the NMSI configuration, the SI provides a flexible clocking assignment for each SCC and SMC from a bank of external clock pins and/or internal baud rate generators.

However, the main feature of the SI is its TSA. The TSA allows any combination of SCCs and SMCs to multiplex their data together on either one or two TDM channels. TDM is used in this manual as the generic term that describes any serial channel that is divided into channels separated by time. Common examples of TDMs are the T1 lines in Japan and the United States and the CEPT lines in Europe.

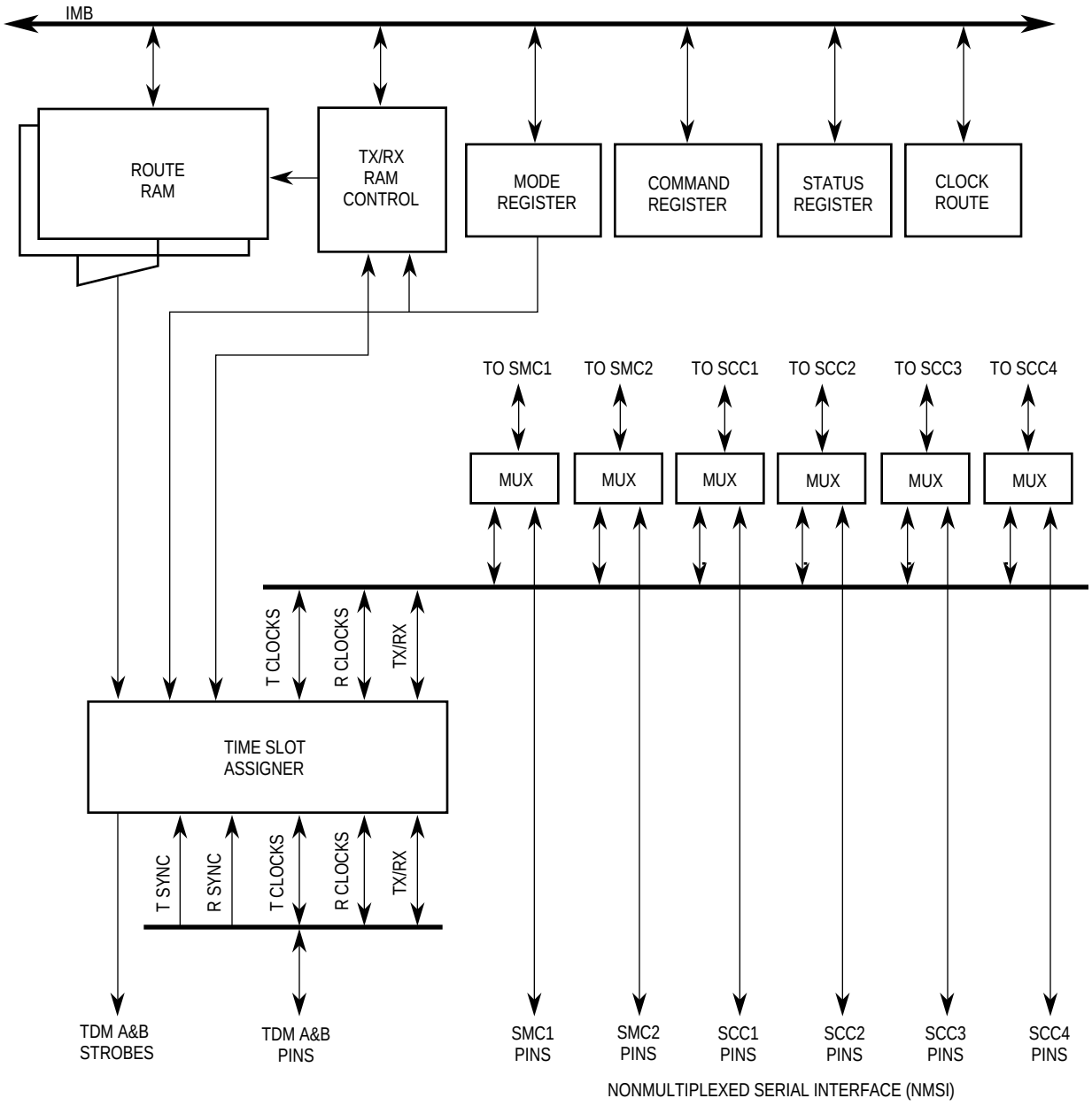
Even if the TSA is not used in its intended capacity, it may still be used to generate complex waveforms on four output pins. For instance, these pins can be programmed by the TSA to implement stepper motor control or variable duty cycle and period control on these pins. Any programmed configuration may be changed on the fly.

7.8.1 SI Key Features

The two major features of the SI are the TSA and the NMSI. The TSA contains the following features:

- Can Connect to Two Independent TDM channels. Each TDM May Be One of the Following:

  —T1 or CEPT Line
  —PCM Highway
  —ISDN Primary Rate
  —ISDN Basic Rate—IDL
  —ISDN Basic Rate—GCI
  —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 (i.e., 1 or 2 Clocks per Data Bit)
- Selectable Delay (0–3 Bits) Between Frame Sync and Frame Start
- Four Programmable Strobe Outputs and Two (2×) Clock Output Pins
- 1- or 8-Bit Resolution in Routing, Masking, and Strobe Selection
- Supports Frames Up to 8192 Bits Long
- Internal Routing and Strobe Selection Can Be Dynamically Programmed
- Supports Automatic Echo and Loopback Mode for Each TDM



NOTE: NMSI clocking paths are not shown.

Figure 7-19. SI Block Diagram

The NMSI contains the following features:

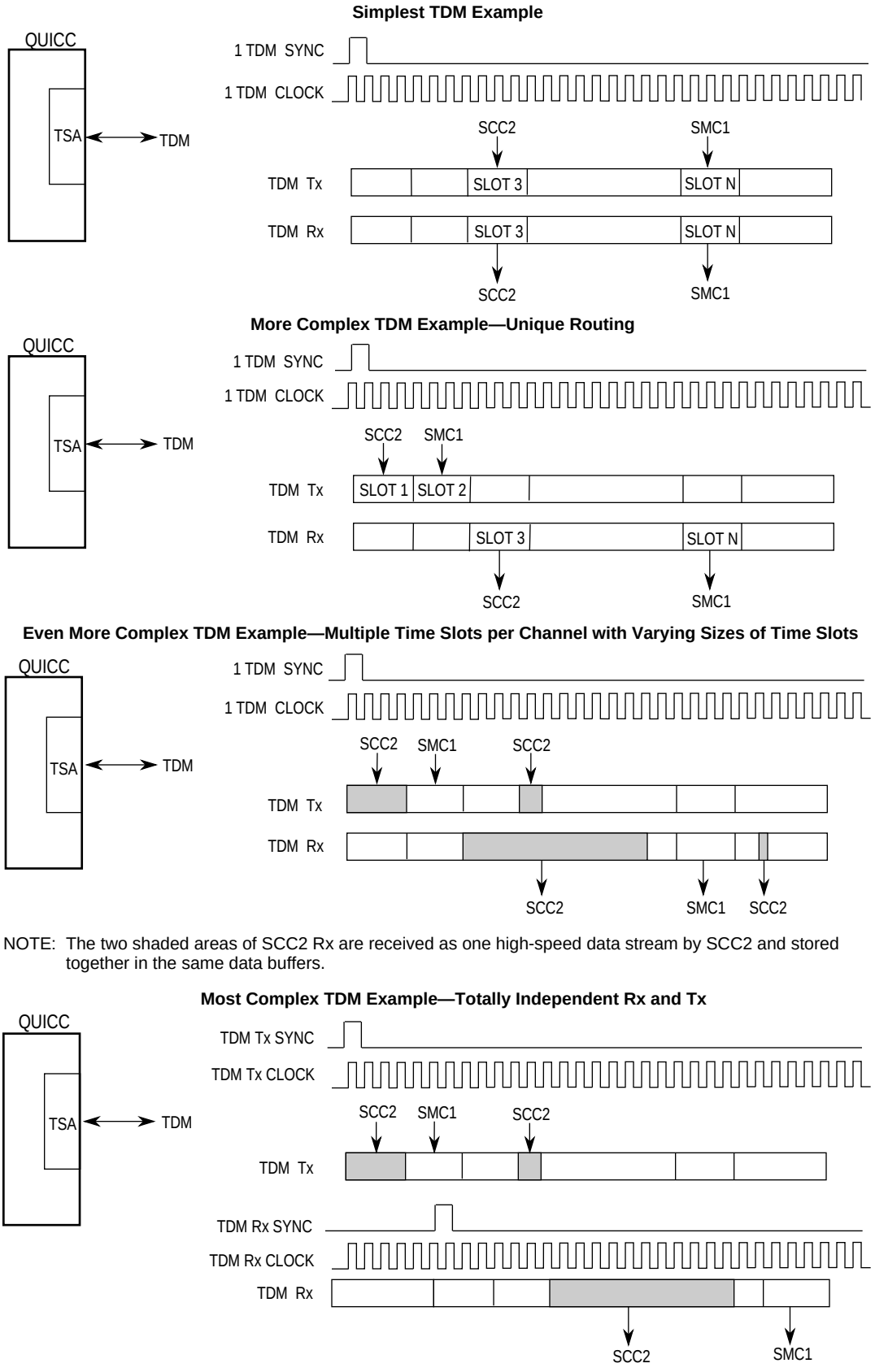
- Each SCC and SMC Can Be Independently Programmed To Work with Its Own Set of Pins in a Nonmultiplexed Manner.
- Each SCC Can Have Its Own Set of Modem Control Pins (TXD, RXD, TCLK, RCLK, $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$).
- Each SMC Can Have Its Own Set of Four Pins (SMTXD, SMRXD, CLK, and SMSYN).
- Each SCC and SMC Can Derive Clocks Externally from a Bank of Eight Clock Pins (CLK1–CLK8) or a Bank of Four Baud Rate Generators (BRG1–BRG4).

7.8.2 TSA Overview

The TSA implements both the internal route selection and time-division multiplexing for multiplexed serial channels. The TSA supports the serial bus rate and format for most standard TDM buses, including the T1 and CEPT highways, the PCM highway, and the ISDN buses in both basic and primary rates. The two popular ISDN basic rate buses, IDL and GCI (also known as IOM-2), are supported. An additional level of flexibility is provided by the TSA in that it supports two TDMs. It is therefore possible to simultaneously support one T1 line and one CEPT line, one basic rate and one primary rate ISDN channel, etc.

TSA programming is completely independent of the protocol used by the SCC or SMC. For instance, the fact that SCC2 may programmed for the HDLC protocol has no impact on the programming of the TSA. The purpose of the TSA is to route the data from the specified pins to the desired SCC or SMC at the correct time. It is the job of the SCC or SMC to handle the data it receives.

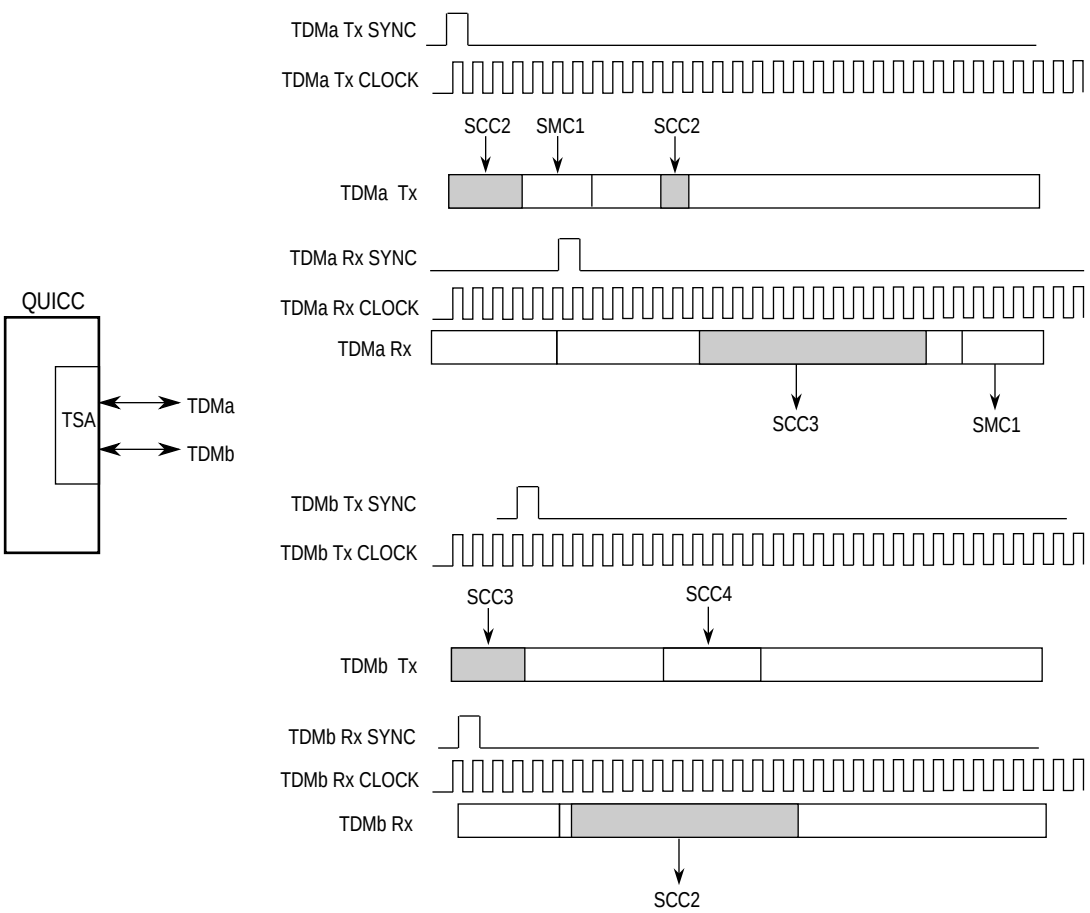
In its simplest mode, the TSA identifies the frame using one sync pulse and one clock signal provided externally by the user. This can be enhanced to allow independent routing of the receive and transmit data on the TDM. Additionally, the definition of a time slot need not be limited to 8 bits or even limited to a single contiguous position within the frame. Finally, the user may provide separate receive and transmit syncs as well as receive and transmit clocks. These various configurations are illustrated in Figure 7-20.



NOTE: The two shaded areas of SCC2 Rx are received as one high-speed data stream by SCC2 and stored together in the same data buffers.

Figure 7-20. Various Configurations of a Single TDM Channel

The TSA also allows two TDM channels to be supported simultaneously. Thus, in its most flexible mode, the TSA can provide two separate TDM channels, each with an independent receive and transmit routing assignment and independent sync pulse and clock inputs (see Figure 7-21). 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.



NOTE: SCCs may receive on one TDM and transmit on another (e.g., SCC2 and SCC3).

Figure 7-21. Dual TDM Channel Example

In addition to channel programming, the TSA supports up to four strobe outputs that may be asserted on a bit basis or a byte basis. These strobes are completely independent from the channel routing used by the SCCs and SMCs. They 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 multi-transmitter architecture. (Note that open-drain programming on the TXDx pins to support a multi-transmitter architecture is programmed in the parallel I/O block.) These strobes can also be used for generating output waveforms to support such applications as stepper motor control.

Most TSA programming is accomplished in two SI RAMs, each of size 64×16 bits. These SI RAMs are directly accessible by the host processor in the internal register section of the QUICC and are not associated with the dual-port RAM. One SI RAM is always used to pro-

gram the transmit routing, and the other SI RAM is always used to program the receive routing. With the SI RAMs, the user can define the number of bits/bytes that are to be routed to which SCC or SMC and the times the external strobes are to be asserted and negated. The size of the SI RAM that is available for time-slot programming depends on the configuration. If two TDM channels are selected, the SI RAM entries available per channel are reduced by one-half. If on-the-fly changes are also allowed, the SI RAM entries are further reduced by one-half. Even in a configuration with two TDM channels and on-the-fly changes allowed, the SI RAM size is still sufficient to allow extensive time-slot programming flexibility. The maximum frame length that can be supported in any configuration is 8192 bits.

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 SCC echo mode in that it can operate on the entire TDM signal rather than just on a particular SCC channel. Loopback mode causes the physical interface to receive the same signal it is transmitting. The SI loopback mode checks more than the individual SCC loopback; it checks the SI and the internal channel routes.

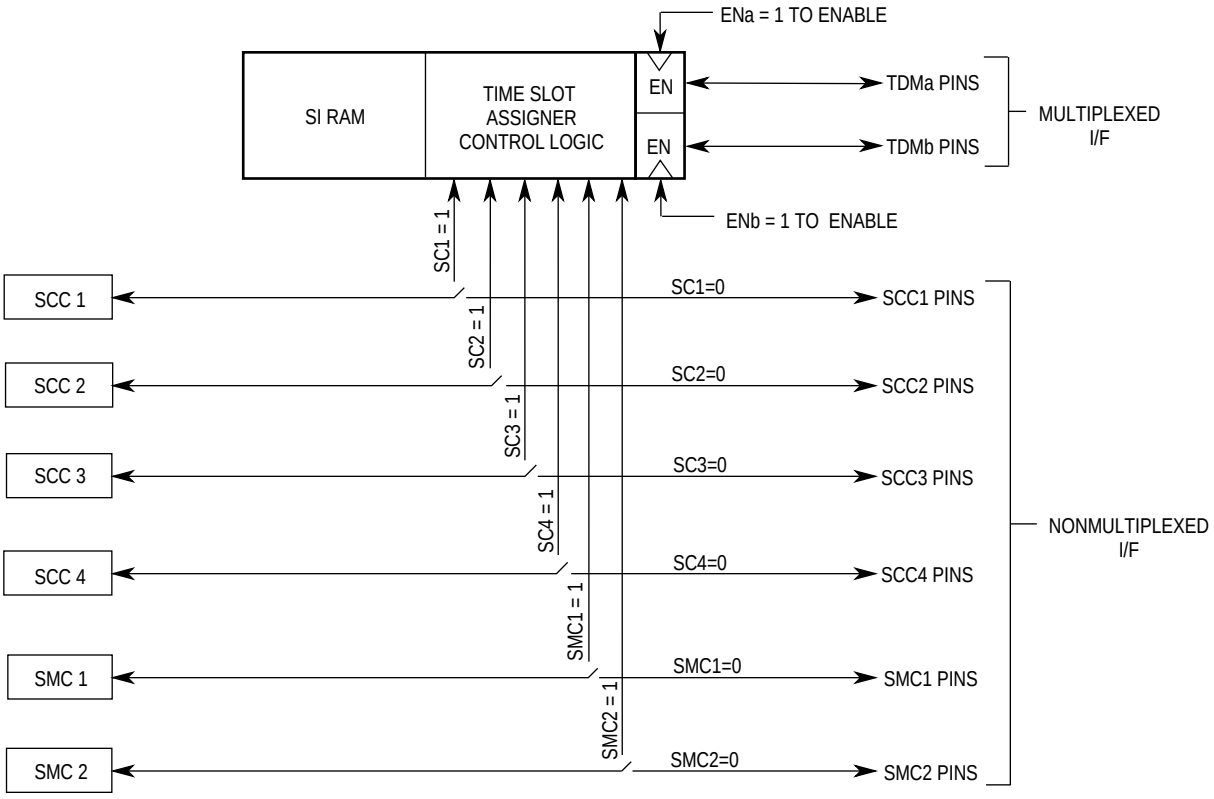
The maximum clock that can be input to the TSA depends on the internal SyncCLK rate. SyncCLK, which is generated in the QUICC clock synthesizer specifically for the SCCs, SMCs, and TSA, defaults to the system frequency (for instance, 25 MHz). However, the clock synthesizer in the SIM60 has an option to divide SyncCLK by 1, 4, 16, or 64 before it leaves the clock synthesizer. Whatever the resulting frequency of SyncCLK, the maximum external serial clock that may be an input to the TSA is SyncCLK/2.5.

The ability to reduce the frequency of SyncCLK before it ever leaves the clock synthesizer is useful for two reasons. First, in a low-power mode, the TSA clocking could potentially be a significant factor in overall QUICC power consumption. Thus, if the TSA does not need to operate at high frequencies, the user may choose a lower frequency SyncCLK as the input to the TSA. (In making this decision, the user must also consider the needs of the other SCCs and SMCs not connected to the TSA and select a sufficiently high SyncCLK value for their use.) Second, the user may wish to dynamically change the general system clock frequency in the clock synthesizer (slow-go mode) while still having the TSA run at the original frequency. The SyncCLK also allows this configuration.

If an SCC or SMC is operating with the NMSI, then the serial clock rate may be slightly faster, at a value not to exceed SyncCLK/2.

7.8.3 Enabling Connections to the TSA

Each SCC and SMC may be independently enabled to be connected to the TSA (see Figure 7-22). Note that separate bits enable whether each SCC or SMC is connected to the TSA or to its own set of external pins. Additionally, the two TDM interfaces must be enabled to be connected to the TSA.



NOTES:
1. The ENx bits are located in SIGMR.
2. The SCx bits are located in SICR.
3. The SMCx bits are located in SIMODE.
4. The clocking paths are not shown for the nonmultiplexed I/F (see Figure 7-35 for more details).

Figure 7-22. Enabling Connections Through the SI

Once the connections are made, the exact routing decisions are made in the SI RAM, as described in the following paragraphs.

7.8.4 SI RAM

The SI has two 64 × 16 static RAMs used to control the routing of the TDM channels to the SCCs and SMCs. The RAMs are uninitialized after power-on. For proper operation, the host should program the RAMs before enabling the multiplexed channels, or undesired results may occur.

The RAM consists of 16-bit entries that are used to define the routing control. Each entry can control from 1 to 16 bits or from 1 to 16 bytes at a time as determined in the entry. In addition to the routing, up to four strobe pins may be asserted according to the programming of the RAM. The strobes are active high.

The two SI RAMs can be configured in four different ways to support various TDM channels. The four possible cases are discussed in the following paragraphs.

7.8.4.1 ONE MULTIPLEXED CHANNEL WITH STATIC FRAMES. With this configuration (see Figure 7-23), there are 64 entries in the SI RAM for transmit data and strobe routing and 64 entries for receive data and strobe routing. This configuration should be chosen when only one TDM is required and the routing on that TDM does not need to be changed dynamically.

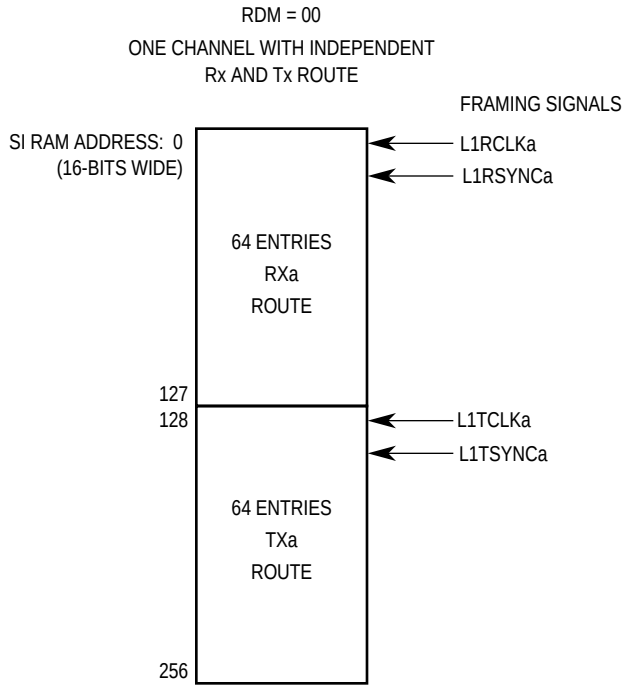


Figure 7-23. SI RAM: One TDM with Static Frames

7.8.4.2 ONE MULTIPLEXED CHANNEL WITH DYNAMIC FRAMES. With this configuration (see Figure 7-24), there is one multiplexed channel. The channel has 32 entries for transmit data and strobe routing and 32 entries for receive data and strobe routing. In each RAM, one of the partitions is the current-route RAM, and the other is a shadow RAM used to allow the user to change the serial routing. After programming the shadow RAM, the user sets the CSRx bit of the associated channel in the SI CR. When the next frame sync arrives, the SI will automatically exchange the current-route RAM for the shadow RAM. Refer to 7.8.4.7 SI RAM Dynamic Changes for more details on how to dynamically change the channel's route. This configuration should be chosen when only one TDM is required but the routing on that TDM may need to be changed dynamically.

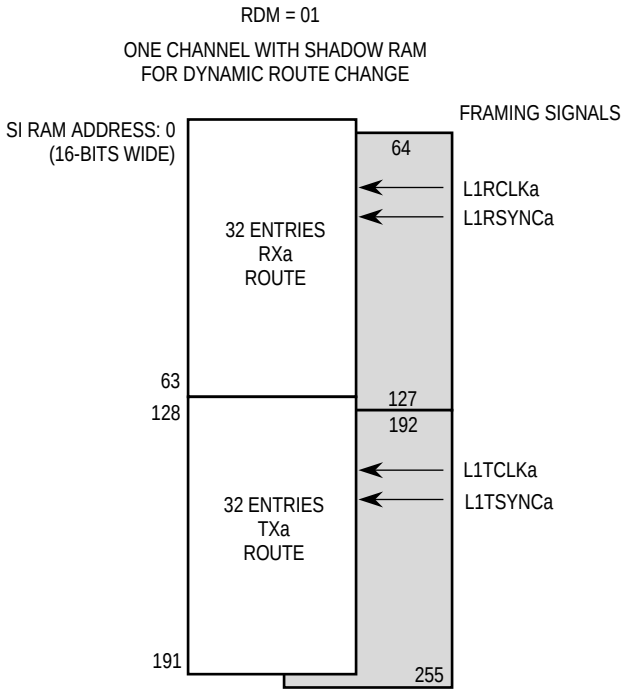


Figure 7-24. SI RAM: One TDM with Dynamic Frames

7.8.4.3 TWO MULTIPLEXED CHANNELS WITH STATIC FRAMES. With this configuration (see Figure 7-25), there are 32 entries for transmit data and strobe routing and 32 entries for receive data and strobe routing. This configuration should be chosen when two TDMs are required and the routing on that TDM does not need to be changed dynamically.

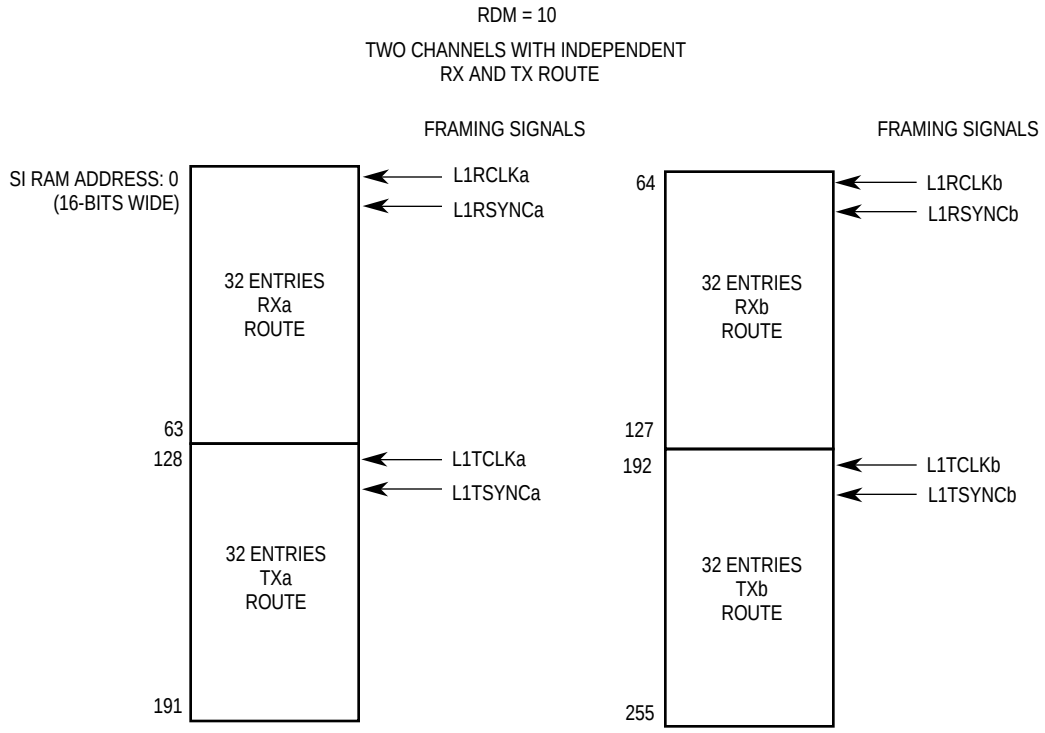


Figure 7-25. SI RAM: Two TDMs with Static Frames

7.8.4.4 TWO MULTIPLEXED CHANNELS WITH DYNAMIC FRAMES. With this configuration (see Figure 7-26), there are two multiplexed channels. Each channel has 16 entries for transmit data and strobe routing and 16 entries for receive data and strobe routing. In each RAM, one of the partitions is the current-route RAM, and the other is a shadow RAM used to allow the user to change the serial routing. After programming the shadow RAM, the user sets the CSRx bit of the associated channel in the SI CR. When the next frame sync arrives, the SI will automatically exchange the current-route RAM for the shadow RAM. Refer to 7.8.4.7 SI RAM Dynamic Changes for more details on how to dynamically change the channel's route. This configuration should be chosen when two TDMs are required and the routing on each TDM may need to be changed dynamically.

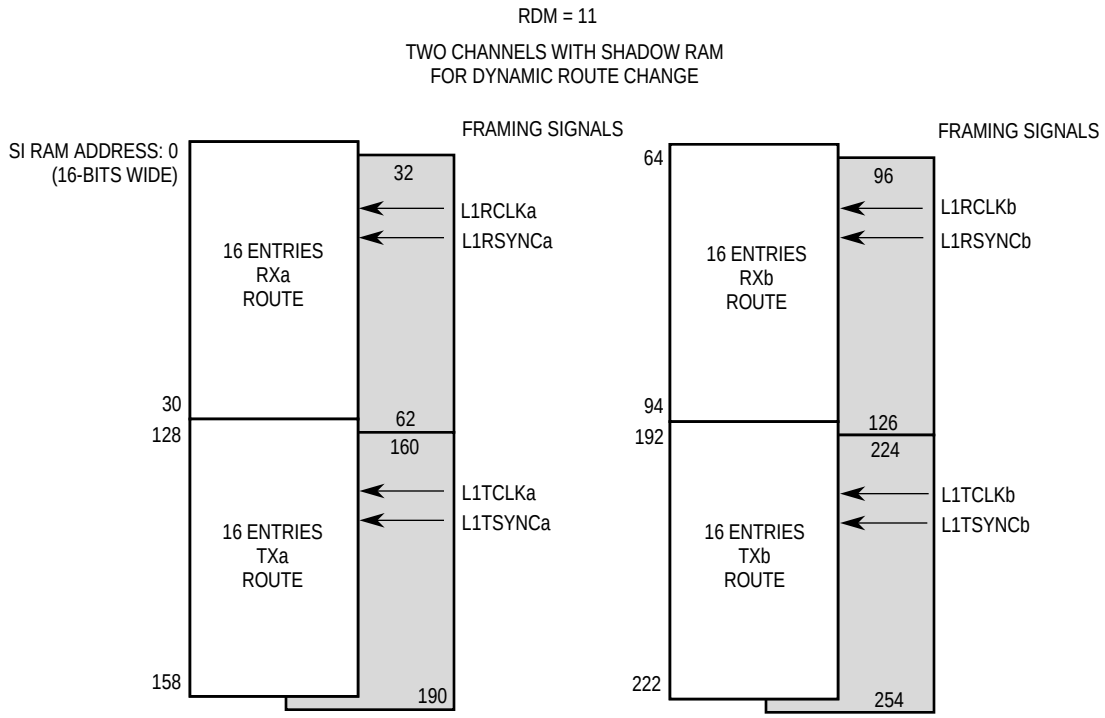


Figure 7-26. Two TDMs with Dynamic Frames

7.8.4.5 PROGRAMMING SI RAM ENTRIES. The programming of each word within the RAM determines the routing of the serial bits (or bit groups) and the assertion of strobe outputs. The RAM programming codes are as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOOP[1]	SWTR	SSEL1–SSEL4				—	CSEL			CNT				BYT	LST

NOTES:
1: Only available on REV C mask or later. NOT Available on REV A or B.
Rev A mask is C63T
Rev B mask are C69T, and F35G
Current Rev C mask are E63C, E68C and F15W

Bit 15 LOOP (Loop back this time slot)

0 = normal mode
1 = loop back mode for this time slot

SWTR—Switch Tx and Rx

The SWTR bit is only valid in the receive route RAM and is ignored in the transmit route RAM. This bit affects the operation of both the L1RXD and L1TXD pins

The SWTR bit would only be set in a special situation where the user desires to receive data from a transmit pin and transmit data on a receive pin. For instance, consider the situation 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, since they both receive the same TDM receive data and transmit on the same TDM transmit signal (see Figure 7-27).

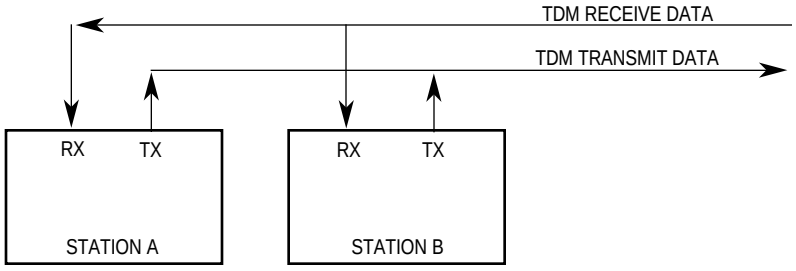


Figure 7-27. Using the SWTR Feature

The SWTR option gives station B the opportunity to listen to transmissions from station A and to transmit data to Station A. To do this, station B would set the SWTR bit in its receive route RAM. For this entry, receive data is taken from the L1TXD pin and data is transmitted on the L1RXD pin. If the user only wants to listen to Station A's transmissions and not transmit data on L1RXD, then the CSEL bits in the corresponding transmit route RAM entry should be cleared to prevent transmission on the L1RXD pin.

It is also possible for station B to transmit data to station A by setting the SWTR bit of the entry in its receive route RAM. Data is transmitted on the L1RXD pin rather than the L1TXD pin, according to the transmit route RAM. Note that this configuration could cause collisions with other data on the L1RXD pin unless care is taken to choose an available (quiet) time slot. If the user only wants to transmit on L1RXD and not receive data on L1TXD, then the CSEL bits in the receive route RAM should be cleared to prevent reception of data on L1TXD.

NOTE

If the transmit and receive sections of the TDM do not use a single clock source, this feature will give erratic results.

0 = Normal operation of the L1TXD and L1RXD pins.
1 = Data is transmitted on the L1RXD pin and is received from the L1TXD pin for the duration of this entry.

SSEL1–SSEL4—Strobe Select

The four strobes (L1STA1, L1STA2, L1STB1, and L1STB2) may be assigned to the receive RAM and asserted/negated with L1RCLKa or L1RCLKb or assigned to the transmit RAM and asserted/negated with L1TCLKa or L1TCLKb. Each bit corresponds to the value the strobe should have during this bit/byte group. Multiple strobes can be asserted simultaneously, if desired.

If a strobe is configured to be asserted in two consecutive SI RAM entries, then it will remain continuously asserted during the processing of both SI RAM entries. If a strobe is asserted on the last entry in the table, the strobe will be negated after the processing of that last entry is complete.

Bit 9—Reserved

NOTES

Each strobe is changed with the corresponding RAM clock and will be 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 (e.g., the SI Rx route for the TDMa entries and the SI Tx route for TDMb entries both select the same strobe), then the assertion of the strobe corresponds to the logical OR of all possible sources. This use of the strobes is not useful for most applications. It is recommended that a given strobe be selected in only one set of SI RAM entries.

CSEL—Channel Select

000 = The bit/byte group is not supported within the QUICC. The transmit data pin is three-stated, and the receive data pin is ignored.
001 = The bit/byte group is routed to SCC1.
010 = The bit/byte group is routed to SCC2.
011 = The bit/byte group is routed to SCC3.
100 = The bit/byte group is routed to SCC4.
101 = The bit/byte group is routed to SMC1.
110 = The bit/byte group is routed to SMC2.
111 = The bit/byte group is not supported within the QUICC. This code is also used in SCIT mode as the D channel grant (refer to 7.8.7.2.2 SCIT Programming.)

CNT—Count

This value indicates the number of bits/bytes (according to the BYT bit) that the routing and strobe select of this entry controls. If CNT = 0000, then 1 bit/byte is chosen; if CNT = 1111, then 16 bits/bytes are selected.

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.

LST—Last Entry in the RAM

Whenever the SI RAM is used, this bit must be set in one of the Tx or Rx entries of each group that is used. Even if all entries of a group are used, this bit must still be set in the last entry.

0 = This is not the last entry in this section of the route RAM.
1 = This is the last entry in this RAM. After this entry, the SI will wait for the sync signal to start the next frame.

NOTE

If a second sync signal is received before the end of a frame (as defined by the last SI RAM entry), an error occurs. The SI will terminate SI RAM processing, and cease transmitting or receiving data until a third sync signal is received.

7.8.4.6 SI RAM PROGRAMMING EXAMPLE. This example shows how to program the RAM to support the 10-bit IDL bus (see Figure 7-33 for the 10-bit IDL bus format).

In this example, the TSA supports the B1 channel with SCC2, 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 SCC4. Additionally, the TSA will mark the D channel with another strobe signal.

First, divide the frame from the start (i.e., the sync) to the end of the frame according to the support that is required:

1. 8 bits (B1)—SCC2
2. 1 bit (D)—SCC1 + strobe1
3. 1 bit—no support
4. 4 bits (B2)—strobe2
5. 4 bits (B2)—SCC4
6. 1 bit (D)—SCC1 + strobe1

Each of these six divisions can be supported by just one SI RAM entry. Thus, a total of only six entries is needed in the SI RAM:

Entry No.	RAM WORD						
	SWTR	SSEL	CSEL	CNT	BYT	LST	description
1	0	0000	010	0000	1	0	8 Bits SCC2
2	0	0001	001	0000	0	0	1 Bit SCC1 Strobe1
3	0	0000	000	0000	0	0	1 Bit No Support
4	0	0010	000	0011	0	0	4 Bits Strobe2
5	0	0000	100	0011	0	0	4 Bits SCC4
6	0	0001	001	0000	0	1	1 Bit SCC1 Strobe1

NOTE

Since IDL requires the same routing for both receive and transmit, an exact duplicate of the above entries should be written to both the receive and transmit sections of the SI RAM. Then the CRTx bit in the SIMODE register can be used to instruct the SI RAM to use the same clock and sync to simultaneously control both sets of SI RAM entries.

7.8.4.7 SI RAM DYNAMIC CHANGES. The SI RAM, described in 7.8.4.5 Programming SI RAM Entries, has four operating modes:

1. One TDM with a static routing definition. SI RAM divided into two parts (Rx and Tx).
2. One TDM allowing dynamic changes. SI RAM divided into four parts.
3. Two TDMs with static routing definition. SI RAM divided into four parts.
4. Two TDMs allowing dynamic changes. SI RAM divided into eight parts.

Dynamic changes mean that the routing definition of a TDM can be modified while the SCCs/SMCs are connected to the TDM. With fixed routing, a change to the routing requires that all SCCs/SMCs connected to the TSA be disabled, the SI routing be modified, and then all SCCs/SMCs connected to the TSA be reenabled before the new routing takes effect.

Dynamic changes divide portions of the SI RAM into current-route RAM and shadow RAM. Once the current-route RAM is programmed, the TSA and SI channels can be enabled, and TSA operation can begin. When the user decides that a change in routing is required, the user programs the shadow RAM with the new route and sets the CSRx bit in the SI CR. As a result, the SI will exchange the shadow RAM and the current-route RAM as soon as the corresponding sync arrives and will reset the CSRx bit to signify that the operation is complete. At this time, the user may change the routing again. Note that the original current-route RAM is now the shadow RAM and vice versa. Figure 7-28 illustrates an example of the shadow RAM exchange process.

If one TDM with dynamic changes is programmed, the initial current-route RAM addresses in the SI RAM are as follows:

- 0–63 RXa Route
- 128–191TXa Route

and the shadow RAMs are at addresses:

- 64–127 RXa Route
- 192–255TXa Route

If two TDMs with dynamic changes are programmed, the initial current-route RAM addresses in the SI RAM are as follows:

- 0–31 RXa Route
- 64–93 RXb Route
- 128–159TXa Route
- 192–223TXb Route

and the shadow RAMs are at addresses:

- 32–63 RXa Route
- 96–93 RXb Route
- 160–191TXa Route
- 224–255TXb Route

The user can read any RAM at any time, but for proper operation of the SI, the user must not attempt to write the current-route RAM. The user can read the SI status register (SISTR) to find which part of the RAM is the current-route RAM.

Beyond knowing which RAM is the current-route RAM, the user may wish to know which entry that the TSA is currently using within the current-route RAM. This information is provided in the SI RAM pointer register (SIRP). The user may also externally connect one of the four strobes to an interrupt pin to generate an interrupt on a particular SI RAM entry starting or ending execution by the TSA.

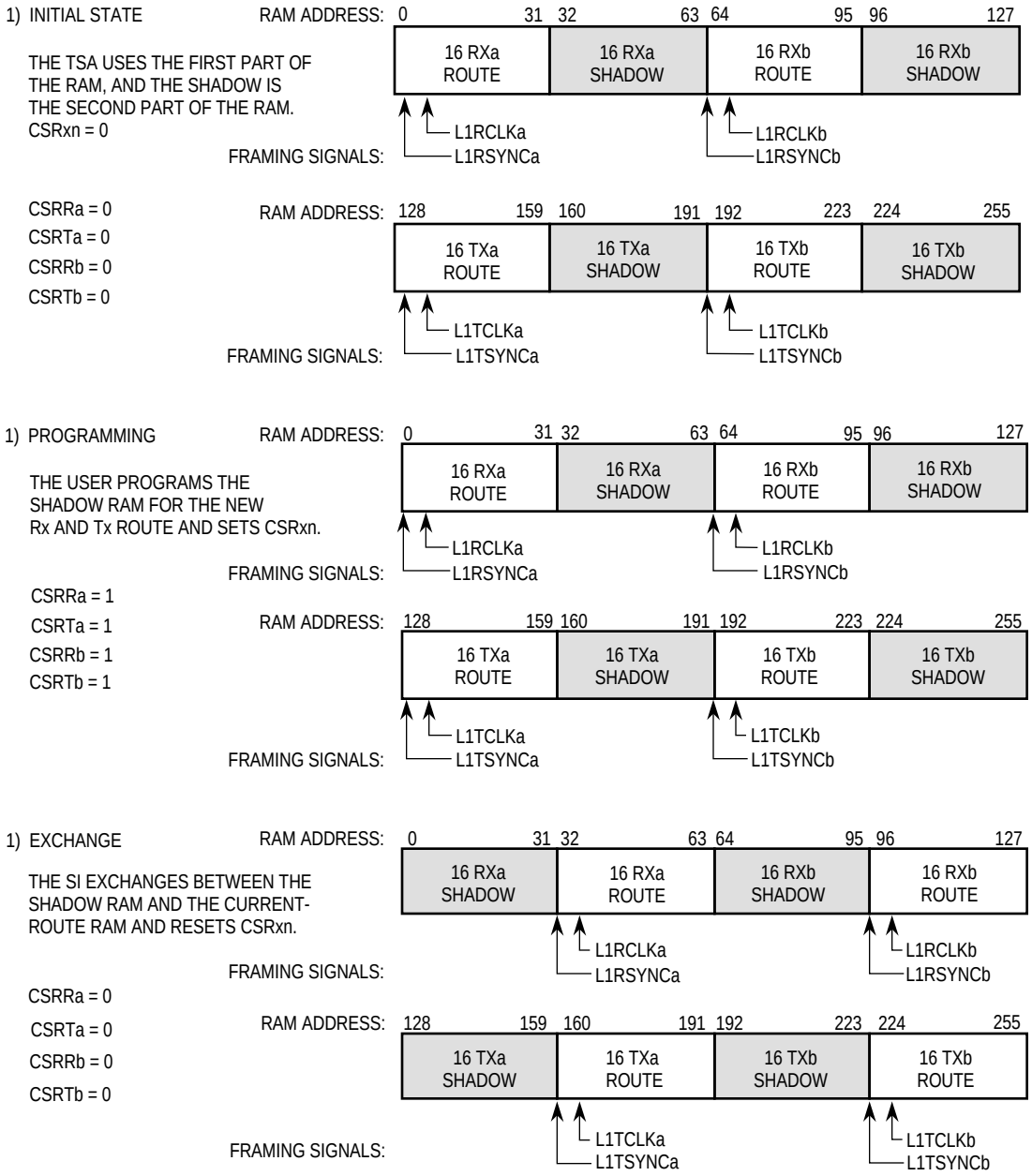


Figure 7-28. SI RAM Dynamic Changes

7.8.5 SI Registers

The following paragraphs describe the SI registers.

7.8.5.1 SI GLOBAL MODE REGISTER (SIGMR). The 8-bit SIGMR defines the RAM division modes. The SIGMR appears to the user as a memory-mapped, read-write register and is cleared at reset.

7	6	5	4	3	2	1	0
—				ENb	ENa	RDM1–RDM0	

Bits 7–4—Reserved

ENb—Enable Channel b

0 = Channel b is disabled. The SI RAMs and TDM routing are in a state of reset, but all other SI functions still operate.
1 = The SI is enabled.

ENa—Enable Channel a

0 = Channel a is disabled. The SI RAMs and TDM routing are in a state of reset, but all other SI functions still operate.
1 = The SI is enabled.

RDM1–RDM0—RAM Division Mode

These bits define the RAM division mode and the number of multiplexed channels supported in the SI.

00 = The SI supports one TDM channel with 64 entries for receive routing and 64 entries for transmit routing.
01 = The SI supports one TDM channel with 32 entries for receive routing and 32 entries for transmit routing. There are an additional 32 shadow entries for the receive routing and 32 shadow entries for transmit routing that may be used to dynamically change the routing.
10 = The SI supports two TDM channels with 32 entries for the receive routing and 32 entries for transmit routing for each of the two TDMs.
11 = The SI supports two TDM channels with 16 entries for receive routing and 16 entries for transmit routing for each channel. There are an additional 16 shadow entries for receive routing and 16 shadow entries for transmit routing that may be used to dynamically change the channel routing.

NOTE

TSAa must be used in RDM1—0 if 00 or 01 setting is desired.

7.8.5.2 SI MODE REGISTER (SIMODE). The 32-bit SIMODE defines the SI operation modes. This register allows the user (in conjunction with the SI RAM) to support any or all of the ISDN channels independently when in IDL or GCI (IOM-2) mode. Any extra SCC channel can then be used for other purposes in NMSI mode. SIMODE appears to the user as a memory-mapped, read-write register and is cleared at reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SMC2	SMC2CS			SDMb		RFSDb		DSCb	CRTb	STZb	CEb	FEb	GMb	TFSDb	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMC1	SMC1CS			SDMa		RFSDa		DSCa	CRTa	STZa	CEa	FEa	GMa	TFSDa	

SMCx—SMCx Connection

0 = NMSI mode. The clock source is determined by the SMCxCS bit, and the data comes from a dedicated pin (SMTXD1 and SMRXD1 for SMC1 or SMTXD2 and SMRXD2 for SMC2) in the NMSI.
1 = SMCx is connected to the multiplexed SI (TDM channel).

SMC2CS—SMC2 Clock Source (NMSI mode)

SMC2 can take its clocks from one of the baud rate generators or one of four pins from the bank of clocks. The SMC2 transmit and receive clocks must be the same when it is connected to the NMSI.

000 = SMC2 transmit and receive clocks are BRG1.
001 = SMC2 transmit and receive clocks are BRG2.
010 = SMC2 transmit and receive clocks are BRG3.
011 = SMC2 transmit and receive clocks are BRG4.
100 = SMC2 transmit and receive clocks are CLK5.
101 = SMC2 transmit and receive clocks are CLK6.
110 = SMC2 transmit and receive clocks are CLK7.
111 = SMC2 transmit and receive clocks are CLK8.

SMC1CS—SMC1 Clock Source (NMSI mode)

SMC1 can take its clocks from one of the baud rate generators or one of four pins from the bank of clocks. The SMC1 transmit and receive clocks must be the same when it is connected to the NMSI.

000 = SMC1 transmit and receive clocks are BRG1.
001 = SMC1 transmit and receive clocks are BRG2.
010 = SMC1 transmit and receive clocks are BRG3.
011 = SMC1 transmit and receive clocks are BRG4.
100 = SMC1 transmit and receive clocks are CLK1.
101 = SMC1 transmit and receive clocks are CLK2.
110 = SMC1 transmit and receive clocks are CLK3.
111 = SMC1 transmit and receive clocks are CLK4.

SDMx—SI Diagnostic Mode for TDM A or B

00 = Normal operation.
01 = Automatic Echo. In this mode, the channel_x transmitter automatically retransmits the TDM received data on a bit-by-bit basis. The receive section operates normally, but the transmit section can only retransmit received data. In this mode, the L1GRx line is ignored.
10 = Internal Loopback. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXDx is connected to L1RXDx). The receiver and transmitter operate normally. The data appears on the L1TXDx pin. In this mode, the L1RQx line is asserted normally. The L1GRx line is ignored.
11 = Loopback Control. In this mode, the TDM transmitter output is internally connected to the TDM receiver input (L1TXDx is connected to L1RXDx). The transmitter output (L1TXDx) and the L1RQx pin will be 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 the transmit clocks should be identical.

RFSDx—Receive Frame Sync Delay for TDM A or B

These two bits determine the number of clock delays between the receive sync and the first bit of the receive frame. Even if the CRTx bit 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

Refer to Figure 7-29 and Figure 7-30 for an example of the use of these bits.

DSCx—Double-Speed Clock for TDM A or B

Some TDMs such as GCI define the input clock to be 2× faster than the data rate. 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.)

CRTx—Common Receive and Transmit Pins for TDM A or B

This bit is useful when the transmit and receive sections of a given TDM use the same clock and sync signals. In this mode, L1TCLKx and L1TSYNCx pins can be used as general-purpose I/O pins.

0 = Separate pins. The receive section of this TDM uses L1RCLKx and L1RSYNCx pins for framing, and the transmit section uses L1TCLKx and L1TSYNCx for framing.
1 = Common pins. The receive and transmit sections of this TDM use L1RCLKx as clock pin of channel x and L1RSYNCx as the receive and transmit sync pin. (Use for IDL and GCI.)

STZx—Set L1TXDx to Zero for TDM A or B

0 = Normal operation.
1 = L1TXDx is set to zero until serial clocks are available, which is useful for GCI activation. Refer to 7.8.7.1 SI GCI Activation/Deactivation Procedure.

CEx—Clock Edge for TDM A or B

When DSCx =0

0 = The data is transmitted on the rising edge of the clock and received on the falling edge. (Use for IDL and GCI.)
1 = The data is transmitted on the falling edge of the clock and received on the rising edge.

When DSCx = 1

0 = The data is transmitted on the rising edge of the clock and received on the rising edge. (Use for IDL and GCI.)
1 = The data is transmitted on the falling edge of the clock and received on the falling edge.

FEx—Frame Sync Edge for TDM A or B

The L1RSYNCx and L1TSYNCx pulses are sampled with the falling/rising edge of the channel clock according to this bit.

0 = Falling edge (Use for IDL and GCI.)
1 = Rising edge

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 SI RAM with 111 to assert an internal strobe on it. Refer to 7.8.7.2.2 SCIT Programming.
1 = IDL mode. A GRANT mechanism is supported if the corresponding GR1–GR4 bits in the SIMODE register are set. The grant is a sample of the L1GRx pin while L1TSYNCx is asserted. This GRANT mechanism implies the IDL access controls for transmission on the D channel. Refer to 7.8.6.2 IDL Interface Programming.

TFSDx—Transmit Frame Sync Delay for TDM A or B

These two bits determine the number of clock delays between the transmit sync and the first bit of the transmit frame. If the CRTx bit is set (recommended with IDL or GCI), then the transmit sync is not used, and these bits are ignored.

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

Refer to Figure 7-29 and Figure 7-30 for an example of the use of these bits.

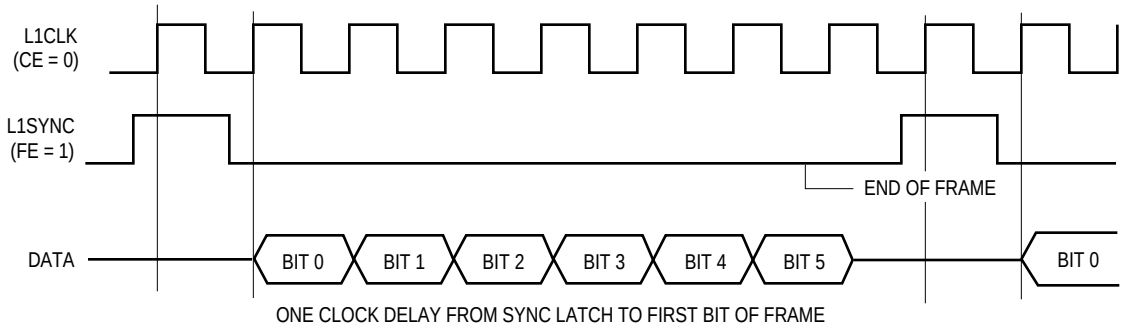


Figure 7-29. One Clock Delay from Sync to Data (RFSD = 01)

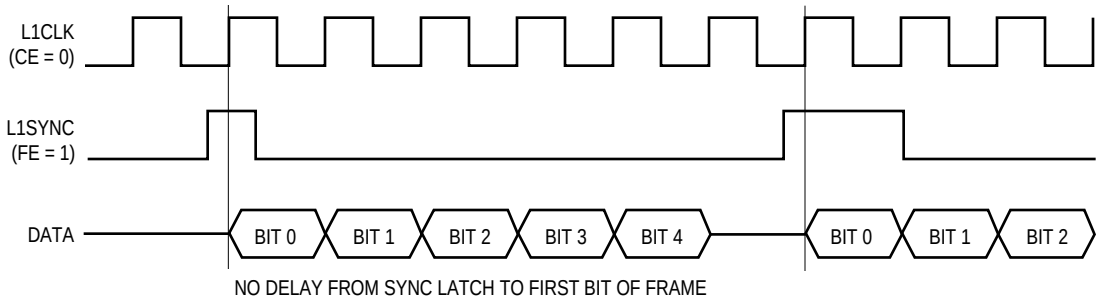
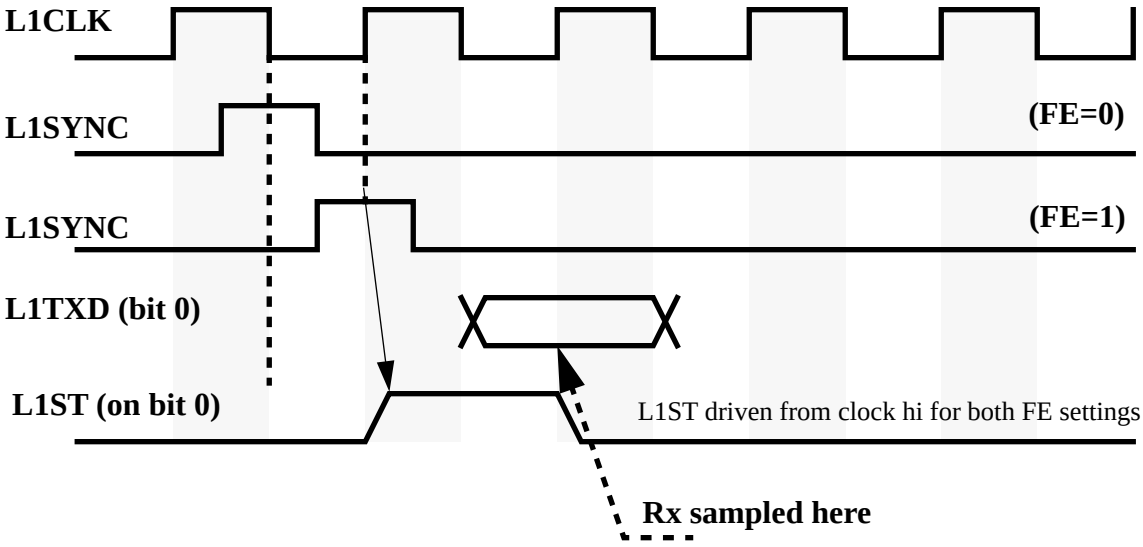


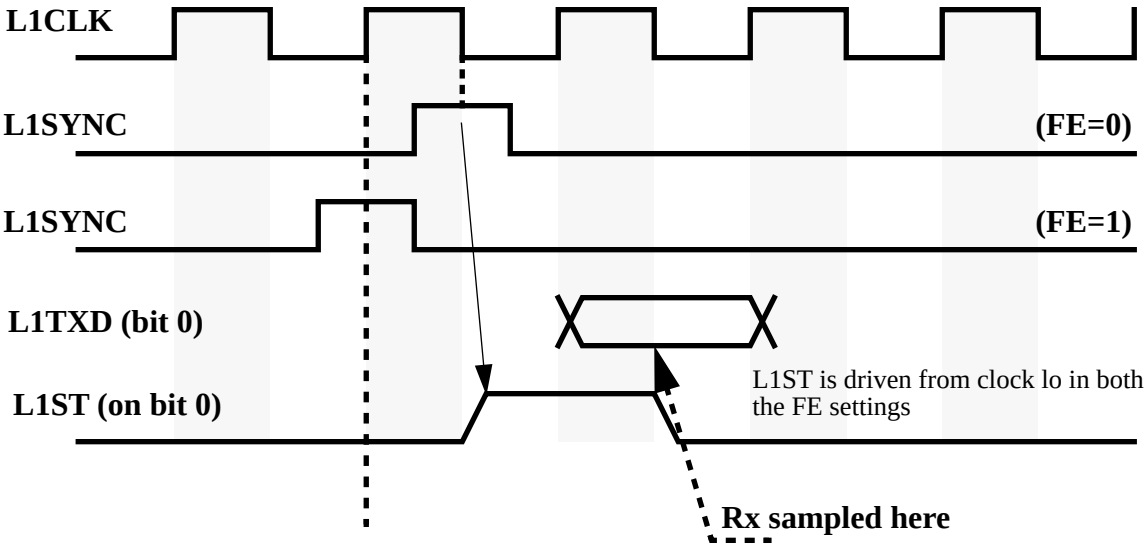
Figure 7-30. No Delay from Sync to Data (RFSD = 00)

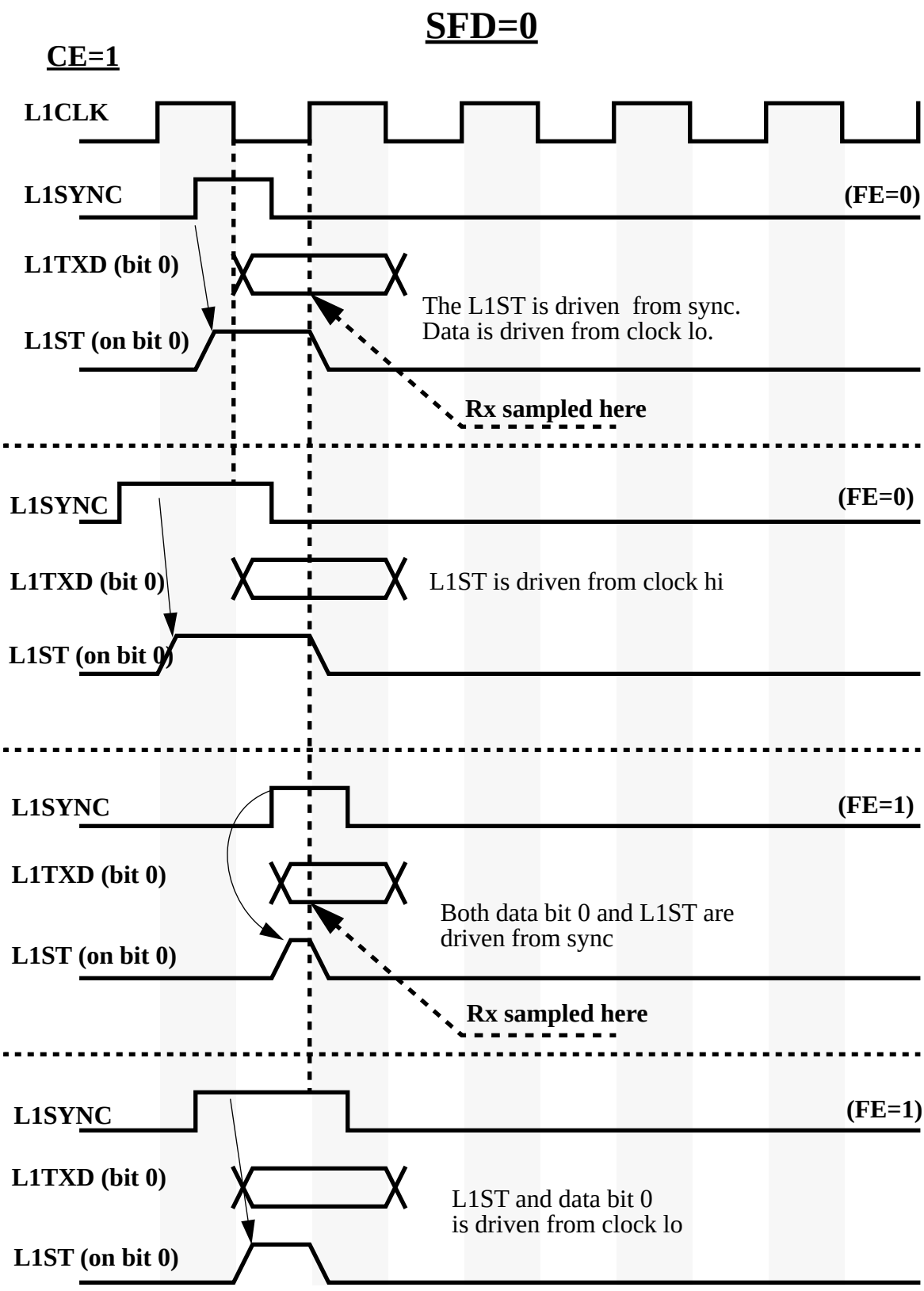
SFD=1

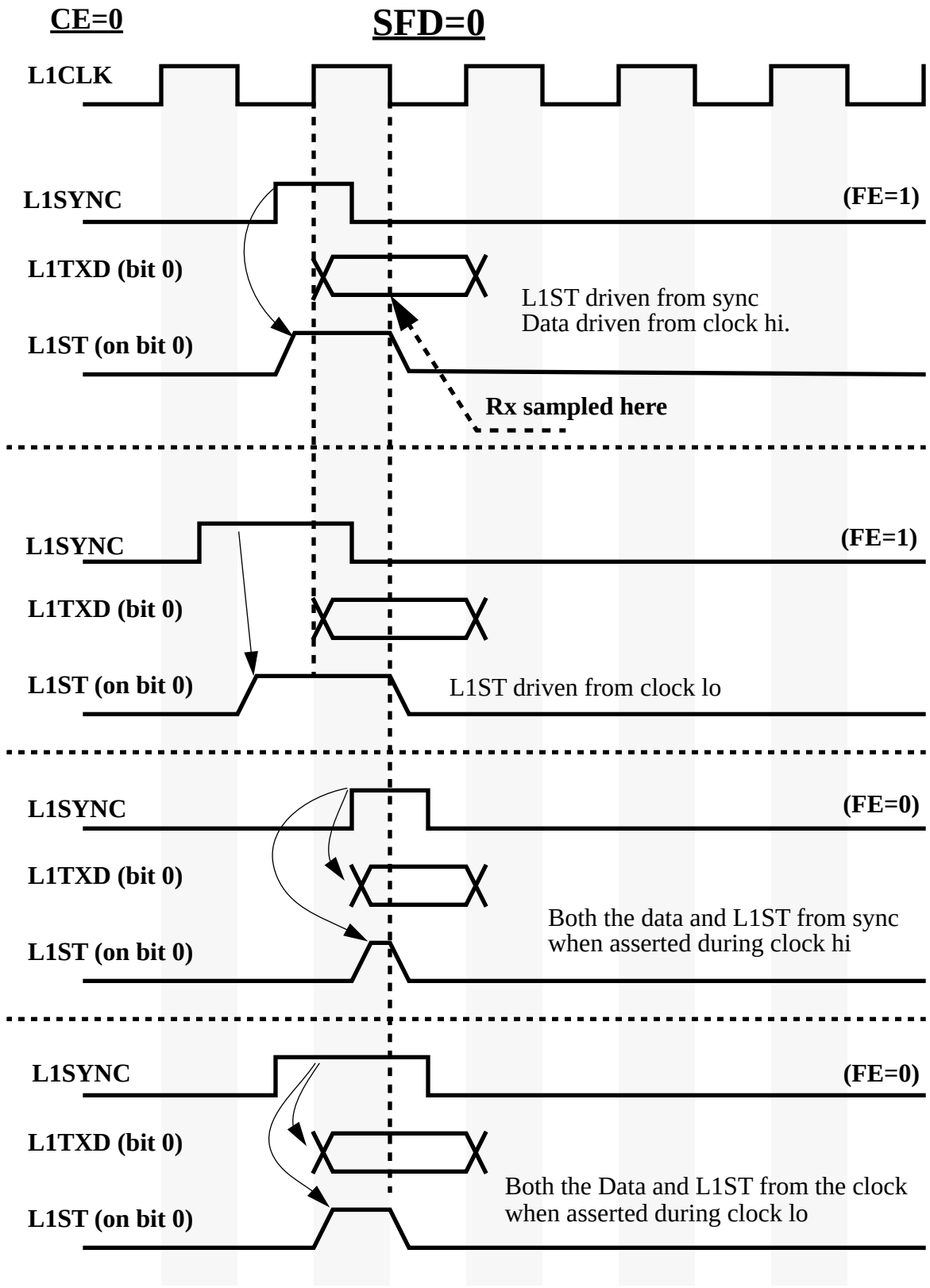
CE=1


L1CLK
L1SYNC
(FE=0)
L1SYNC
(FE=1)
L1TXD (bit 0)
L1ST (on bit 0)
L1ST driven from clock hi for both FE settings
Rx sampled here

CE=0


L1CLK
L1SYNC
(FE=0)
L1SYNC
(FE=1)
L1TXD (bit 0)
L1ST (on bit 0)
L1ST is driven from clock lo in both the FE settings
Rx sampled here


SFD=0
CE=1
L1CLK
L1SYNC
(FE=0)
L1TXD (bit 0)
L1ST (on bit 0)
The L1ST is driven from sync.
Data is driven from clock lo.
Rx sampled here
L1SYNC
(FE=0)
L1TXD (bit 0)
L1ST (on bit 0)
L1ST is driven from clock hi
L1SYNC
(FE=1)
L1TXD (bit 0)
L1ST (on bit 0)
Both data bit 0 and L1ST are
driven from sync
Rx sampled here
L1SYNC
(FE=1)
L1TXD (bit 0)
L1ST (on bit 0)
L1ST and data bit 0
is driven from clock lo


CE=0
SFD=0
L1CLK
L1SYNC
(FE=1)
L1TXD (bit 0)
L1ST (on bit 0)
L1ST driven from sync
Data driven from clock hi.
Rx sampled here
L1SYNC
(FE=1)
L1TXD (bit 0)
L1ST (on bit 0)
L1ST driven from clock lo
L1SYNC
(FE=0)
L1TXD (bit 0)
L1ST (on bit 0)
Both the data and L1ST from sync
when asserted during clock hi
L1SYNC
(FE=0)
L1TXD (bit 0)
L1ST (on bit 0)
Both the Data and L1ST from the clock
when asserted during clock lo

7.8.5.3 SI CLOCK ROUTE REGISTER (SICR). The 32-bit SICR is used to define the SCC clock sources. The clock source can be one of the four baud rate generators or an input from a bank of clock pins. The SICR appears to the user as a memory-mapped, read-write register and is cleared at reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GR4	SC4	R4CS			T4CS			GR3	SC3	R3CS			T3CS		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GR2	SC2	R2CS			T2CS			GR1	SC1	R1CS			T1CS		

GRx—Grant Support of SCCx

0 = SCCx transmitter does not support the grant mechanism. The grant is always asserted internally.
1 = SCCx transmitter supports the grant mechanism as determined by the GMx bit of its channel.

SCx—SCCx Connection

0 = SCCx is not connected to the multiplexed SI but is either connected directly to the NMSIx pins or is not used. The choice of general-purpose I/O port pins versus SCCn pins is made in the parallel I/O control register.
1 = SCCx is connected to the multiplexed SI. The NMSIx receive pins are available for other purposes.

RxCS—Receive Clock Source for SCCx

These bits are ignored when the SCCx is connected to the TSA (SCx = 1).

000 = SCCx receive clock is BRG1.
001 = SCCx receive clock is BRG2.
010 = SCCx receive clock is BRG3.
011 = SCCx receive clock is BRG4.
100 = SCCx receive clock for x = 1,2 is CLK1 and for x = 3,4 is CLK5.
101 = SCCx receive clock for x = 1,2 is CLK2 and for x = 3,4 is CLK6.
110 = SCCx receive clock for x = 1,2 is CLK3 and for x = 3,4 is CLK7.
111 = SCCx receive clock for x = 1,2 is CLK4 and for x = 3,4 is CLK8.

TxCS—Transmit Clock Source for SCCx

These bits are ignored when SCCx is connected to the TSA (SCx = 1).

000 = SCCx transmit clock is BRG1.
001 = SCCx transmit clock is BRG2.
010 = SCCx transmit clock is BRG3.
011 = SCCx transmit clock is BRG4.
100 = SCCx transmit clock for x = 1,2 is CLK1 and for x = 3,4 is CLK5.
101 = SCCx transmit clock for x = 1,2 is CLK2 and for x = 3,4 is CLK6.
110 = SCCx transmit clock for x = 1,2 is CLK3 and for x = 3,4 is CLK7.
111 = SCCx transmit clock for x = 1,2 is CLK4 and for x = 3,4 is CLK8.

7.8.5.4 SI COMMAND REGISTER (SICMR). The 8-bit SICMR allows the user to dynamically program the SI RAM. For more information about dynamic programming, refer to 7.8.4.7 SI RAM Dynamic Changes

The contents of this register are valid only in the RAM division mode (RDM1–RDM0 bits in SIGMR equal 01 or 11). This register is cleared at reset.

7	6	5	4	3	2	1	0
CSRRa	CSRTa	CSRRb	CSRTb	—			

CSRRx—Change Shadow RAM for TDM A or B Receiver

When set, this bit will cause the SI receiver to replace the current route with the shadow RAM. The bit is 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 will exchange between the RAMs and take the new receive routing from the receiver shadow RAM. This bit is cleared as soon as the switch has completed.

CSRTx—Change Shadow RAM for TDM A or B Transmitter

When set, this bit will cause the SI transmitter to replace the current route with the shadow RAM. The bit is 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 will exchange between the RAMs and take the new transmitter routing from the receiver shadow RAM. This bit is cleared as soon as the switch has completed.

Bits 3–0—Reserved

These bits should be set to zero by the user.

7.8.5.5 SI STATUS REGISTER (SISTR). The 8-bit SISTR indicates to the user which part of the SI RAM is the current-route RAM. The value of this register is valid only when the corresponding bit in the SIGMR is clear. This register is cleared at reset.

CRORa—Current Route of TDMa Receiver

7	6	5	4	3	2	1	0
CRORa	CROTa	CRORb	CROTb	—			

0 = The current-route receiver RAM is in address:
0–63 when the SI supports one TDM (RDM = 01)
0–31 when the SI supports two TDMs (RDM = 11)
1 = The current route receiver RAM is in address:
64–127 when the SI supports one TDM (RDM = 01)
32–63 when the SI supports two TDMs (RDM = 11)

CROTa—Current Route of TDMa Transmitter

0 = The current-route transmitter RAM is in address:
128–191 when the SI supports one TDM (RDM = 01).
128–159 when the SI supports two TDMs (RDM = 11).
1 = The current-route transmitter RAM is in address:
192–255 when the SI supports one TDM (RDM = 01).
160–191 when the SI supports two TDMs (RDM = 11).

CRORb—Current Route of TDMb Receiver

This bit is valid only in the RAM division mode (RDM bits in the SIGMR equal 11).

0 = The current-route receiver RAM is in address 64–95.
1 = The current-route receiver RAM is in address 96–127.

CROTb—Current Route of TDMb Transmitter

This bit is valid only in the RAM division mode (RDM bits in the SIGMR equal 11).

0 = The current-route transmitter RAM is in address 192–223.
1 = The current-route transmitter RAM is in address 224–255.

Bits 3–0—Reserved

7.8.5.6 SI RAM POINTERS (SIRP). This 32-bit, read-only register indicates to the user which RAM entry is currently being serviced. This gives a real-time status of where the SI current is inside the TDM frame.

Although SIRP does not need to be accessed by most users, it does provide information that may be helpful for debugging and synchronization of some system activity to the activity on the TDMs. Reading SISTR should be sufficient for most applications.

The user can determine which RAM entry in the SI RAM is currently in progress, but cannot determine the status within that entry. For instance, if the RAM entry is programmed to select four contiguous time slots from the TDM and the SIRP indicates the entry is currently active, the user does not know which of the four time slots is currently in progress. The SIRP will, however, change its status immediately when the next SI RAM entry begins to be processed.

NOTE

The user may also connect one of the four strobes externally to an interrupt pin to generate an interrupt on a particular SI RAM entry starting or ending execution by the TSA.

The value of this register is changed upon transitions of the serial clocks. Before acting on the information in this register, the user should perform two reads and verify that the two reads returned the same value.

The pointers provided by this register indicate the SI RAM entry word offset that is currently in progress. The register is cleared at reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
—	—	V	RbPTR					—	—	V	RaPTR				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	V	TbPTR					—	—	V	TaPTR				

In all cases, the value in the TxPTR or RxPTR increments by one for each entry (i.e., 16-bit SI RAM word) that is processed by the SI. Since each TxPTR and RxPTR is 5 bits each, the values in each TxPTR and RxPTR can range from 0 to 31, corresponding to 32 different SI RAM entries.

The full pointer range may not necessarily be used. For instance, if the last bit is set in the fifth SI RAM entry, then the pointer will only reflect values from 0 to 4. Once the fifth entry is processed by the SI, the pointer is reset to 0.

The V-bit in each entry shows that the entry is valid. This information is particularly useful if the PTR value happens to be zero. Additionally, the V-bits save the user from having to read both the SIRP and the SISTR to obtain the needed information.

The pointer values are described based on the four possible ways the SI RAM can be configured.

7.8.5.6.1 SIRP When RDM = 00 (One Static TDM). •In this case, since 64 entries cannot be signified with a single 5-bit pointer, two 5-bit pointers are used—one for the first 32 entries and one for the second 32 entries.

RaPTR and RbPTR contain the address of the RAM entry currently active. When the SI services entries 1–32, RaPTR will be incremented, and RbPTR will be continuously cleared. When the SI services entries 33–64, RaPTR will be continuously cleared, and RbPTR will be incremented.

TaPTR and TbPTR contain the address of the Tx entry currently active. When the SI services entries 1–32, TaPTR will be incremented, and TbPTR will be continuously cleared. When the SI services entries 33–64, TaPTR will be continuously cleared, and TbPTR will be incremented.

7.8.5.6.2 SIRP When RDM = 01 (One Dynamic TDM). •For the receiver, either RaPTR or RbPTR is used, depending on which portion of the SI Rx RAM is currently active. For the transmitter, either TaPTR or TbPTR is used, depending on which portion of the SI Tx RAM is currently active.

If its V-bit is set, RaPTR contains the address of the Rx entry currently active. The SI RAM receive address block in use is 0–63, and CRORa = 0 in SISTR.

If its V-bit is set, RbPTR contains the address of the Rx entry currently active. The SI RAM receive address block in use is 64–127, and CRORa = 1 in SISTR.

If its V-bit is set, TaPTR contains the address of the Tx entry currently active. The SI RAM transmit address block in use is 128–191, and CROTa = 0 in SISTR.

If its V-bit is set, TbPTR contains the address of the Tx entry currently active. The SI RAM transmit address block in use is 192–255, and CROTa = 1 in SISTR.

7.8.5.6.3 SIRP When RDM = 10 (Two Static TDMs). •This is the simplest case, since each pointer is continuously used and has only one function.

RaPTR contains the address of the RXa entry currently active.

RbPTR contains the address of the RXb entry currently active.

TaPTR contains the address of the TXa entry currently active.

TbPTR contains the address of the TXb entry currently active.

7.8.5.6.4 SIRP When RDM = 11 (Two Dynamic TDMs). •In this case, each pointer is continuously used, but points to different sections of the SI RAM, depending on whether the pointer's value is in the first half (0–15) or the second half (16–31).

RaPTR contains the address of the RXa entry currently active. If the pointer has a value from 0–15, the current-route RAM is SI RAM address block 0–31, and CRORa = 0 in SISTR. If the pointer has a value from 16–31, the current-route RAM is SI RAM address block 32–63, and CRORa = 1 in SISTR.

RbPTR contains the address of the RXb entry currently active. If the pointer has a value from 0–15, the current route RAM is SI RAM address block 64–95, and CRORb = 0 in SISTR. If the pointer has a value from 16–31, the current-route RAM is SI RAM address block 96–127, and CRORb = 1 in SISTR.

TaPTR contains the address of the TXa entry currently active. If the pointer has a value from 0–15, the current route RAM is SI RAM address block 128–159, and CROTa = 0 in SISTR. If the pointer has a value from 16–31, the current-route RAM is SI RAM address block 160–191, and CROTa = 1 in SISTR.

TbPTR contains the address of the TXb entry currently active. If the pointer has a value from 0–15, the current-route RAM is SI RAM address block 192–223, and CROTb = 0 in SISTR. If the pointer has a value from 224–255, the current-route RAM is SI RAM address block 160–191, and CROTb = 1 in SISTR.

7.8.6 SI IDL Interface Support

The IDL interface is a full-duplex ISDN interface used to connect a physical layer device to the QUICC. The QUICC supports both the basic rate and the primary rate of the IDL bus. In the basic rate of IDL, data on three channels, B1, B2, and D, is transferred in a 20-bit frame, providing 160-kbps full-duplex bandwidth. The QUICC is an IDL slave device that is clocked by the IDL bus master (physical layer device) and has separate receive and transmit sections. Because the QUICC can support two TDMs, it can actually support two independent IDL buses using separate clocks and sync pulses as shown in Figure 7-31.

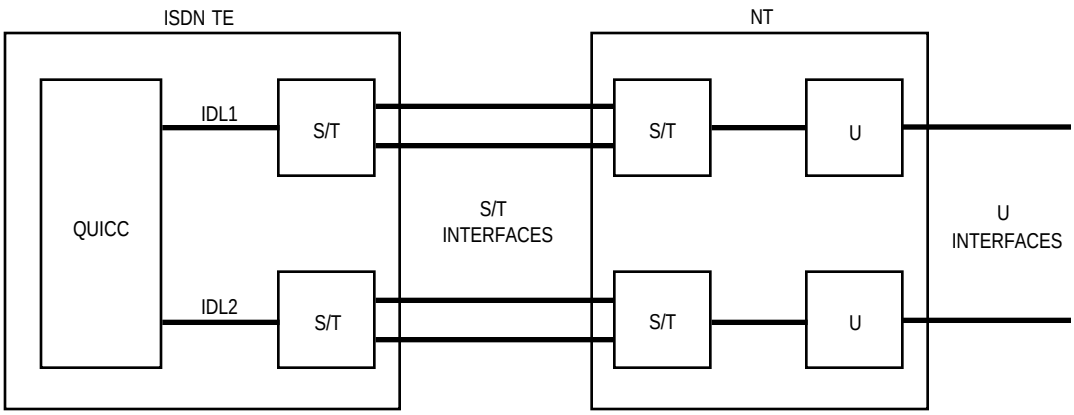


Figure 7-31. Dual IDL Bus Application Example

7.8.6.1 IDL INTERFACE EXAMPLE. An example of the IDL application is the ISDN terminal adaptor shown in Figure 7-32. In such an application, the IDL interface is used to connect the 2B+D channels between the QUICC, CODEC, and S/T transceiver. One of the QUICC SCCs would be configured to HDLC mode to handle the D channel; another QUICC SCC would be used to rate adapt the terminal data stream over the first B channel. That SCC would be configured for HDLC mode if V.120 rate adaption is required. The second B channel could be routed to the CODEC as a digital voice channel, if desired. The SPI is used to send initialization commands and periodically check status from the S/T transceiver. The SCC connected to the terminal would be configured for UART or other protocol depending on the terminal protocol used. Alternatively, instead of a terminal, a connection to a LAN could be made via Ethernet.

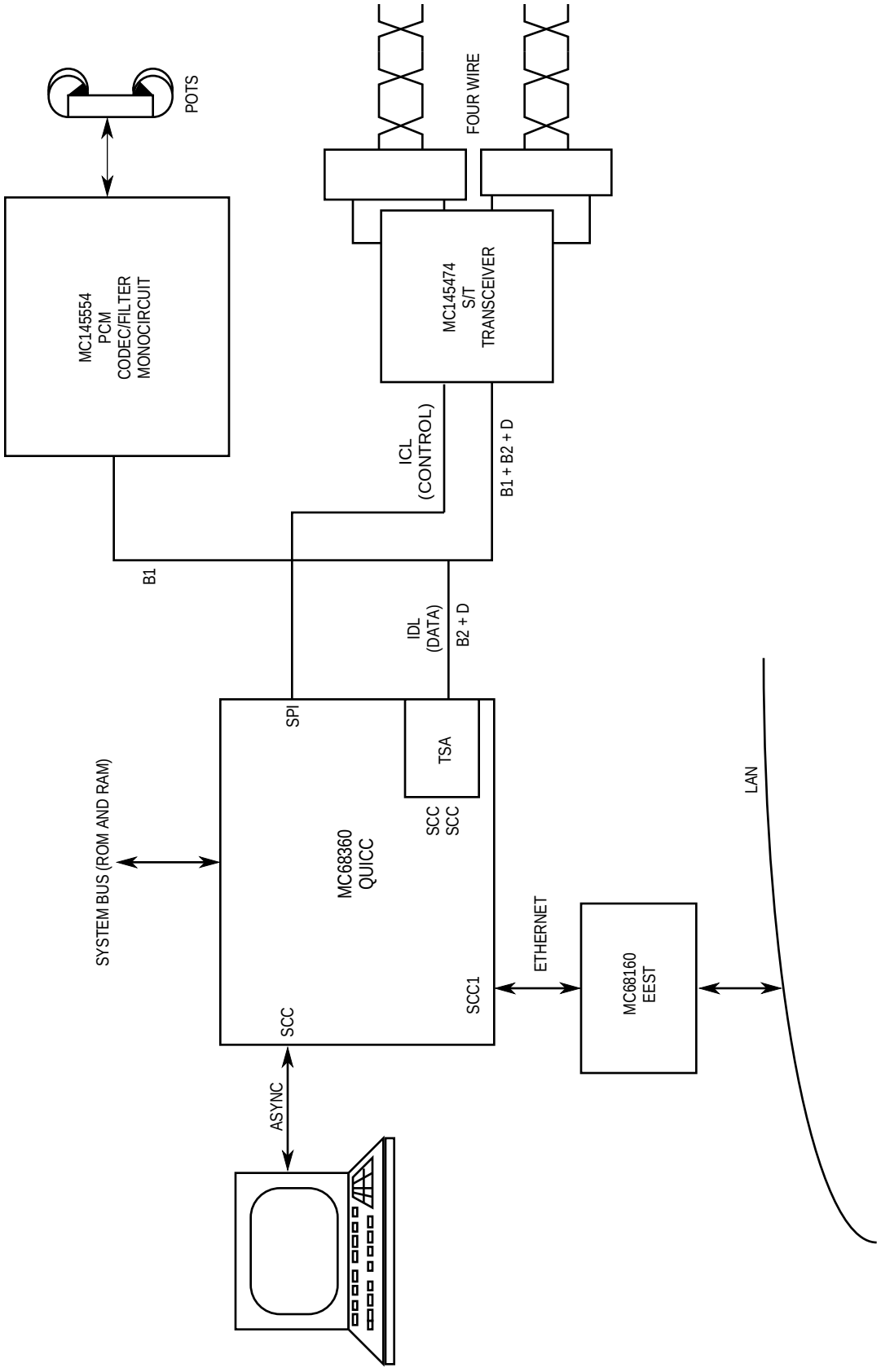


Figure 7-32. IDL Terminal Adapter

The QUICC 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 as follows:

1. L1RCLKx—IDL clock; input to the QUICC.
2. L1RSYNCx—IDL sync signal; input to the QUICC. This signal indicates that the clock periods following the pulse designate the IDL frame.
3. L1RXDx—IDL receive data; input to the QUICC. Valid only for the bits that are supported by the IDL; ignored for other signals that may be present.
4. L1TXDx—IDL transmit data; output from the QUICC. Valid only for the bits that are supported by the IDL; three-stated otherwise.
5. L1RQx—IDL request permission to transmit on the D channel; output from the QUICC on L1RQx pin.
6. L1GRx—IDL grant permission to transmit on the D Channel; input to the QUICC on L1TSYNCx pin.

NOTE

x = a and b for TDMa and TDMb.

The basic rate IDL bus has three channels:

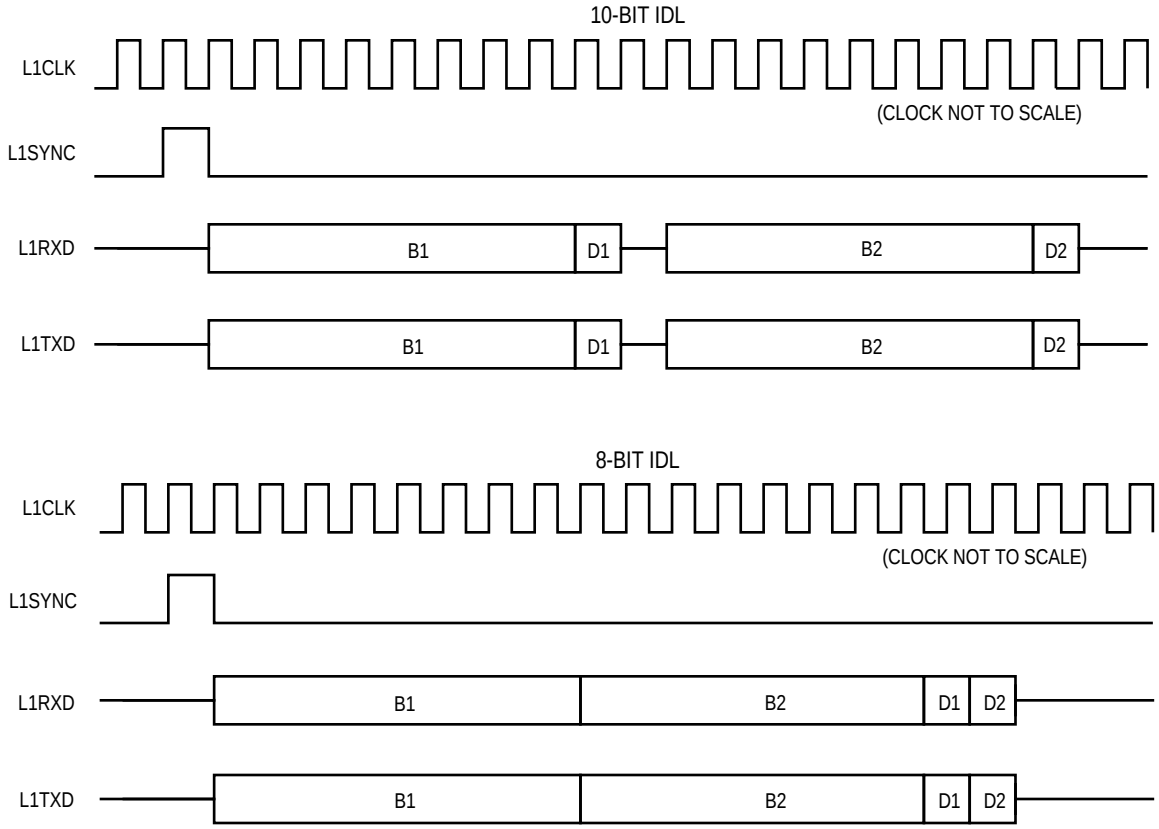
- B1—64 kbps bearer channel
- B2—64 kbps bearer channel
- D—16 kbps signaling channel

There are two definitions of the IDL bus frame structure: 8 bits and 10 bits (see Figure 7-33). The difference between them is only the channel order within the frame.

NOTE

Previous versions of Motorola's IDL-defined bit functions, called auxiliary (A) and maintenance (M), were eliminated from the IDL definition when it was decided that the IDL control channel would be out-of-band. They were defined as a subset of the Motorola SPI format called serial control port (SCP). If a user wishes to implement the A and M bit functions as originally defined, the TSA may be programmed to access these bits and to route them transparently to an SCC or SMC. To perform the out-of-band signaling required, the QUICC's SPI may be used.

The QUICC supports all channels of the IDL bus in the basic rate. Each bit in the IDL frame can be routed to every SCC and SMC or can assert a strobe output for supporting an external device.



NOTE: L1RQx and L1GRx are not shown.

Figure 7-33. IDL Bus Signals

The QUICC supports the request-grant method for contention detection on the D channel of the IDL basic rate. When the QUICC has data to transmit on the D channel, it asserts L1RQx. The physical layer device monitors the physical layer bus for activity on the D channel and indicates that the channel is free by asserting L1GRx. The QUICC samples the L1GRx signal when the IDL sync signal (L1RSYNCx) is asserted. If L1GRx is high (active), the QUICC transmits the first zero of the opening flag in the first bit of the D channel. If a collision is detected on the D channel, the physical layer device negates L1GRx. The QUICC then stops its transmission and retransmits the frame when L1GRx is reasserted. This procedure is handled automatically for the first two buffers of a frame.

For the primary rate IDL, the QUICC can support up to four 8-bit channels in the frame, determined by the programming of the SI RAM. To support more channels, the user can route more than one channel to every SCC, which the SCC will treat as one high-speed stream and store in the same data buffers (this approach is appropriate only for transparent data). Additionally, the QUICC can be used to assert strobes for support of additional IDL channels externally.

The IDL interface supports the CCITT I.460 recommendation for data rate adaptation, since it can separately access 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 will receive only the bits that are enabled by the receiver route RAM. The transmitter will transmit only

the bits that are enabled by the transmitter route RAM and will three-state L1TXDx otherwise.

7.8.6.2 IDL INTERFACE PROGRAMMING. The user can program the channels used for the IDL bus interface to the appropriate configuration. First, the user should program the SIMODE to the IDL grant mode for that channel, using the GMx bits. The user can program more than one channel to interface to the IDL bus. If the receive and transmit section are used for interfacing to the same IDL bus, the user can internally connect the receive clock and sync signals to the SI RAM transmit section, using the CRTx bits. The user has to program the RAM section used for the IDL channels to the desired routing. (An example is shown in 7.8.4.6 SI RAM Programming Example.) The user should then define the IDL frame structure to be a delay of 1 bit from frame sync to data, to falling edge sample sync, and the clock edge to transmit on the rising edge of the clock. The L1TXDx pin should be programmed to be three-stated when inactive (through the parallel I/O open-drain register). To support the D channel, the user must program the appropriate GRx bit in SIMODE and program the RAM entry to route data to that serial controller. The two definitions of IDL, 8 bits and 10 bits, are supported by only modifying the SI RAM programming. In both cases, the L1GRx pin will be sampled with the L1TSYNCx signal and transferred to the D channel SCC as a grant indication. The same procedure is valid for supporting an IDL bus in the second channel.

For example, assuming the 7.8.4.6 SI RAM Programming Example, which uses SCC1, SCC2, and SCC4, connected to the TDMx pins, with no other SCCs connected, the initialization sequence is as follows:

1. Program the SI RAM. Write all entries that are not used with $0001, setting the LST bit and disabling the routing function.

Entry	RAM Word						
No.	SWTR	SSEL	CSEL	CNT	BYT	LST	description
1	0	0000	010	0000	1	0	8 Bits SCC2
2	0	0000	001	0000	0	0	1 Bit SCC1
3	0	0000	000	0000	0	0	1 Bit No Support
4	0	0000	100	0000	1	0	8 Bits SCC4
5	0	0001	001	0000	0	1	1 Bit SCC1 Strobe1

NOTE

Since IDL requires the same routing for both receive and transmit, an exact duplicate of the above entries should be written to both the receive and transmit sections of the SI RAM beginning at SI RAM addresses 0 and 128, respectively.

2. SIMODE = $00000145. Only TDMa is used; the SMCs are not connected.
3. SICR = $400040C0. Only SCC4, SCC2, and SCC1 are connected to the TSA. SCC1 supports the grant mechanism since it is on the D channel.
4. PAODR bit 6 = 1. Configures L1TXDa to an open-drain output.

5. PAPAR bits 6, 7, and 8 = 1. Configures L1TXDa, L1RXDa, and L1RCLKa.
6. PADIR bits 6 and 7 = 1. PADIR bit 8 = 0. Configures L1TXDa, L1RXDa, and L1RCLKa.
7. PCPAR bits 3, 10, and 11 = 1. Configures L1RQa, L1TSYNCa, and L1RSYNCa.
8. PCDIR bit 3 = 0. L1RQa is an input. L1TSYNCa will perform the L1GRa function and is therefore an output, but it does not need to be configured with a PCDIR bit. L1RSYNCa is an input, but it does not need to be configured with a PCDIR bit.
9. SIGMR = $04. Enable TDMa (one static TDM).
10. 1SICMR is not used.
11. 1SISTR and SIRP do not need to be read, but can be used for debugging information once the channels are enabled.
12. 1Enable the SCC1 for HDLC operation (to handle the LAPD protocol of the D channel), and set SCC2 and SCC4 as desired.

7.8.7 SI GCI Support

The normal mode of the GCI, also known as the ISDN-oriented modular rev 2.2 (IOM-2), and the SCIT are fully supported by the QUICC. The QUICC also supports the D channel access control in S/T interface terminals by using the command/indication (C/I) channel for that function.

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 QUICC can support two independent GCI buses and has independent receive and transmit sections for each one. 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 2048 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 QUICC 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.

The GCI signals for each transmit and receive channel are as follows:

L1RSYNCx—Used as GCI sync signal; input to the QUICC. This signal indicates that the clock periods following the pulse designate the GCI frame.

L1RCLKx—Used as GCI clock; input to the QUICC. The L1RCLKx signal is twice the data clock.

L1RXDx—Used as GCI receive data; input to the QUICC.

L1TXDx—Used as GCI transmit data; open-drain output. Valid only for the bits that are supported by the IDL; three-stated otherwise.

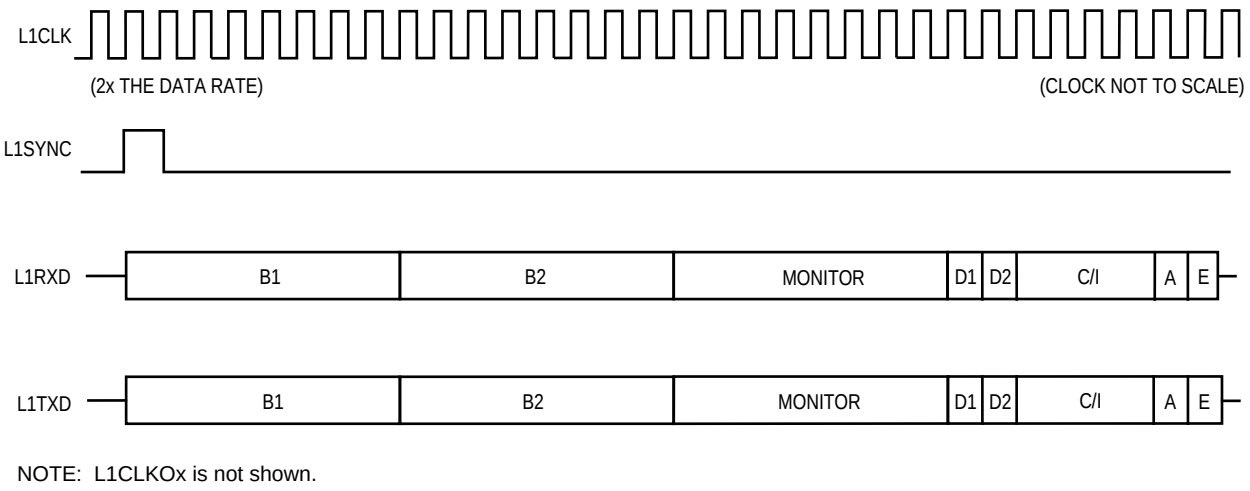
L1CLKOx—Optional signal; output from QUICC. This 1× clock output can be used to clock devices that do not interface directly to GCI. If the double-speed clock

is used, (DSCx bit is set in the SIMODE), this output is the L1RCLKx divided by 2; otherwise, it is simply a 1× output of the L1RCLKx signal. Note that on the MC68302 this signal was known as GCIDCL.

NOTE

x = a and b for TDMa and TDMb.

Figure 7-34 shows the GCI bus signals.



NOTE: L1CLKOx is not shown.

Figure 7-34. GCI Bus Signals

In addition to the 144-kbps ISDN 2B+D channels, the GCI provides five channels for maintenance and control functions:

- B1—64 kbps bearer channel
- B2—64 kbps bearer channel
- M—64 kbps monitor (M) channel
- D—16 kbps signaling channel
- C/I—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 (i.e., the CPU32+ core). 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 M and C/I channel protocols.

The QUICC can support any channel of the GCI bus in the primary rate by modifying the SI 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 by which serial controller. The receiver will receive only the bits that are enabled by the SI RAM. The transmitter will transmit only the bits that

are enabled by the SI RAM and will not drive L1TXDx; otherwise, L1TXDx is an open-drain output and should be pulled high externally.

The QUICC supports contention detection on the D channel of the SCIT bus. When the QUICC has data to transmit on the D channel, it checks a SCIT bus bit that is marked with a special route code (generally, 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 QUICC then aborts its transmission and retransmits the frame when this bit is set again. This procedure is handled automatically for the first two buffers of a frame.

7.8.7.1 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 QUICC by enabling the clock pulses and by an indication in the channel 0 C/I channel. The QUICC will report to the CPU32+ core by a maskable interrupt that a valid indication is in the SMC receive BD.

When the CPU32+ core activates the line, the data output of L1TXDn is programmed to zero by setting the STZx bit in the SIMODE register. Code 0 (command timing TIM) will be transmitted on channel 0 C/I channel to the layer 1 device until the STZx bit is reset. The physical layer device will resume the clock pulses and will give an indication in the channel 0 C/I channel. The CPU32+ core should reset the STZx bit to enable data output.

7.8.7.2 SI GCI PROGRAMMING. The following paragraphs describe programming for both the normal mode GCI and SCIT.

7.8.7.2.1 Normal Mode GCI Programming. The user can program the channels used for the GCI bus interface to the appropriate configuration. First, the user should program the SIMODE to the GCI/SCIT mode for that channel, 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 can internally connect the receive clock and sync signals to the SI RAM transmit section, using the CRTx bits. The user should then define the GCI frame routing and strobe select using the SI RAM. When the receive and transmit section use the same clock and sync signals, the user should program the receive section as well as the transmit section to the same configuration. 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, the user should route those channels to one of the SMCs. To support the D channel when there is no possibility of collision, the user should clear the GRx bit corresponding to the SCC that supports the D channel in the SIMODE.

7.8.7.2.2 SCIT Programming. For interfacing the GCI/SCIT bus, the user should program the SIMODE to the GCI/SCIT mode. The SI 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 receive and transmit sections to use the same clock and sync signals, using the

CRTx bits, and program the GRx bits to transfer the D channel grant to the SCC that supports this channel. The user should mark the received bit, which is the grant bit, by programming the channel select bits of the SI RAM to 111 for an internal assertion of a strobe on this bit. This bit will be 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 GCI, but any other bit may be selected using the SI RAM.

For example, assuming SCC1 is connected to the D channel, SCC2 is connected to the B1 channel, and SCC4 is connected 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 the SI RAM. Write all entries that are not used with $0001, setting the LST bit and disabling the routing function.

Entry No.	RAM Word						
	SWTR	SSEL	CSEL	CNT	BYT	LST	Description
1	0	0000	010	0000	1	0	8 Bits SCC2
2	0	0000	100	0000	1	0	8 Bits SCC4
3	0	0000	101	0000	1	0	8 Bits SMC1
4	0	0000	001	0001	0	0	2 Bits SCC1
5	0	0000	101	0101	0	0	6 Bits SMC1
6	0	0000	000	0110	1	0	Skip 7 Bytes
7	0	0000	000	0001	0	0	Skip 2 Bits
8	0	0000	111	0000	0	1	D Grant Bit

NOTE

Since GCI requires the same routing for both receive and transmit, an exact duplicate of the above entries should be written to both the receive and transmit sections of the SI RAM beginning at addresses 0 and 128, respectively.

2. SIMODE = $000080E0. Only TDMa is used; SMC1 is connected. SCIT mode is used in this example.

NOTE

If SCIT mode is not used, delete the last three entries of the SI RAM and set the LST bit in the new last entry.

3. SICR = $400040C0. SCC4, SCC2, and SCC1 are connected to the TSA. SCC1 supports the grant mechanism since it is on the D channel.
4. PAODR bit 6 = 1. Configures L1TXDa to an open-drain output.
5. PAPAR bits 6, 7, and 8 = 1. Configures L1TXDa, L1RXDa, and L1RCLKa.
6. PADIR bits 6 and 7 = 1. PADIR bit 8 = 0. Configures L1TXDa, L1RXDa, and L1RCLKa.

7. If the 1× GCI data clock is required, set PBPAR bit 11 and PBDIR bit 11, which configures L1CLKOa as an output.
8. PCPAR bit 11 = 1. Configures L1RSYNCa.
9. SIGMR = $04. Enable TDMa (one static TDM).
10. 1SICMR is not used.
11. 1SISTR and SIRP do not need to be read, but can be used for debugging information once the channels are enabled.
12. 1Enable the SCC1 for HDLC operation (to handle the LAPD protocol of the D channel), set SCC2 and SCC4 as desired, and enable SMC1 for SCIT operation.

7.8.8 Serial Interface Synchronization

On rev A and B of the QUICC, the SI would reset itself if an unexpected sync pulse was seen during the middle of a time frame. This would cause the SI to sync again on the following sync pulse but it would also lead to an unresolved loss of synchronization of an SCC or SMC operating in transparent or GCI modes (assuming that SCC or SMC was receiving data from the SI).

In revision C.1 and later of the QUICC, the SI will ignore this unexpected sync pulse and synchronize on the next sync pulse (it will not reset itself). This may lead to a reception of one or two “bad” slots but the SCC or SMC will remain synchronized.

NOTE

Rev A mask is C63T

Rev B mask are C69T, and F35G

Current Rev C mask are E63C, E68C and F15W

7.8.9 NMSI Configuration

The SI supports an NMSI mode for each of the SCCs and SMCs. The decision of whether to connect a given SCC to the NMSI is made in the SICR. The decision of whether to connect a given SMC to the NMSI is made in the SIMODE register.

An SCC or SMC may be connected to the NMSI, regardless of which other channels are connected to a TDM channel. The user should note, however, that NMSI pins may be multiplexed with other functions at the parallel I/O lines. Therefore, if a combination of TDM and NMSI channels is used, the decision of which SCCs and SMCs to connect and where to connect them should be made consulting the QUICC pinout. Generally speaking, the TDMa channel is multiplexed with many of the SCC4 pins; whereas, the TDMb channel is multiplexed with many of the SCC3 pins.

The clocks that are provided to the SCCs and SMCs are derived from twelve sources: four internal baud rate generators and eight external CLK pins (see Figure 7-35). There are two main advantages to the bank-of-clocks approach. First, an SCC or SMC is not forced to

choose its clock from a pre-defined pin or baud rate generator, which allows flexibility in the pinout mapping strategy. Second, if a group of SCC receivers and transmitters need the same clock rate, they can share the same pin. This configuration leaves additional pins for other functions and minimizes potential skew between multiple clock sources.

The four baud rate generators also make their clocks available to external logic, regardless of whether the baud rate generators are being used by an SCC or SMC. Note that the BRGOx pins are multiplexed with other functions; therefore, all BRGOx pins may not always be available. Note that BRGO3 has the flexibility to be output on both port A 12 and port B 16. See the pinout description in Section 11 Ordering Information and Mechanical Data for more details.

There are a few restrictions in the bank-of-clocks mapping. First, only eight of the twelve sources can be connected to any given SCC receiver or transmitter. Second the SMC transmitter must have the same clock source as the receiver when connected to the NMSI pins.

Once the clock source is selected, the clock is given an internal name. For the SCCs, the name is RCLKx and TCLKx. For the SMCs, the name is simply SMCLKx. These internal names are used only in NMSI mode to specify the clock that is sent to the SCC or SMC. These names do not correspond to any pins on the QUICC.

NOTE

The internal RCLKx and TCLKx may be used as inputs to the DPLL unit, which is inside the SCC. Thus, the RCLKx and TCLKx signals are not required to always reflect the actual bit rate on the line.

The exact pins available to each SCC and SMC in the NMSI mode are summarized in Figure 7-35.

The SCC1 in NMSI mode has its own set of modem control pins:

TXD1

RXD1

TCLK1 <- BRG1–BRG4, CLK1–CLK4

RCLK1 <- BRG1–BRG4, CLK1–CLK4

$\overline{\text{RTS1}}$

$\overline{\text{CTS1}}$

$\overline{\text{CD1}}$

The SCC2 in NMSI mode has its own set of modem control pins:

TXD2

RXD2

TCLK2 <- BRG1–BRG4, CLK1–CLK4

RCLK2 <- BRG1–BRG4, CLK1–CLK4

RTS2

CTS2

CD2

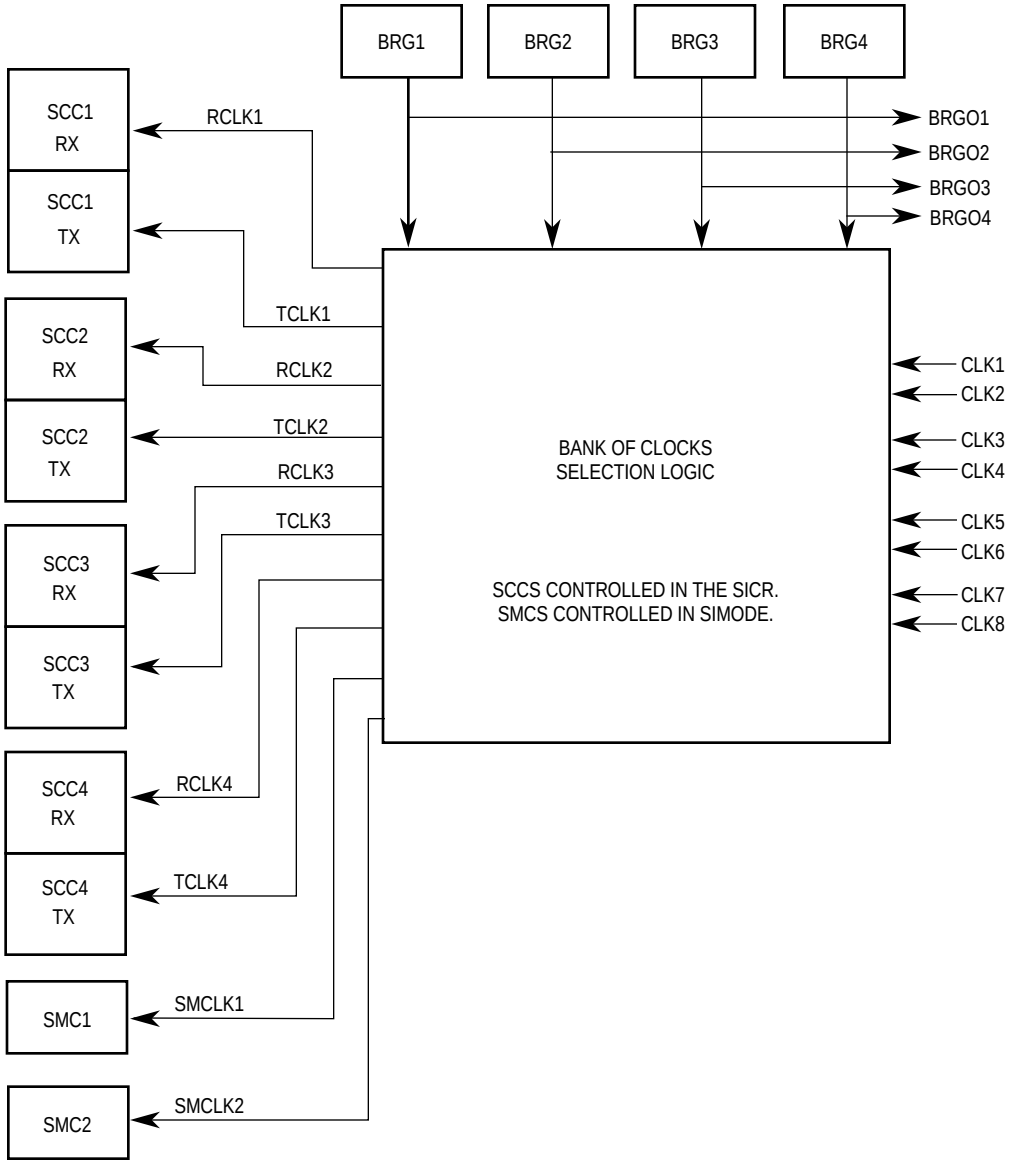


Figure 7-35. Bank of Clocks

The SCC3 in NMSI mode has its own set of modem control pins:

TXD3

RXD3

TCLK3 <- BRG1–BRG4, CLK5–CLK8

RCLK3 <- BRG1–BRG4, CLK5–CLK8

RTS3

CTS3

CD3

The SCC4 in NMSI mode has its own set of modem control pins:

TXD4

RXD4

TCLK4 <- BRG1–BRG4, CLK5–CLK8

RCLK4 <- BRG1–BRG4, CLK5–CLK8

RTS4

CTS4

CD4

The SMC1 in NMSI mode has its own set of modem control pins:

SMTXD1

SMRXD1

SMCLK1 <- BRG1–BRG4, CLK1–CLK4

SMSYN1 (used only in the totally transparent protocol)

The SMC2 in NMSI mode has its own set of modem control pins:

SMTXD2

SMRXD2

SMCLK2 <- BRG1–BRG4, CLK5–CLK8

SMSYN2 (used only in the totally transparent protocol)

Any SCC or SMC that requires fewer pins that those listed may use that pin for another function or configure that pin as a parallel I/O pin.

Since some SCCs use external clock pins CLK1–CLK4 and other SCCs use external clock pins CLK5–CLK8, it might seem that there is no way to provide one external clock source on one CLK pin to be used by all four SCCs. However, the QUICC provides a simple clock bridge function from external CLK8 to the internal CLK4 connection, even if the CLK4 pin is used for another of its programmable functions. This configuration allows SCC1/SCC2 to share clocks with SCC3/SCC4 without wasting an external pin. This is shown in the port A registers in 7.14.4 Port A Registers.

7.9 BAUD RATE GENERATORS (BRGS)

The CPM contains four, independent, identical BRGs that can be used with the SCCs and SMCs. The clocks produced in the BRG are sent to the bank-of-clocks selection logic, where they can be routed to the SCCs and/or SMCs. The bank-of-clocks logic is described in 7.8.9 NMSI Configuration. In addition, the output of the BRG may be routed to a pin to be used externally. The main features of the BRGs are as follows:

- Four, Independent, Identical BRGs
- On-the-Fly Changes Allowed
- Each BRG May Be Routed to One or More SCCs or SMCs
- A 16× Divider Option Allows Slow Baud Rates at High System Frequencies
- Each BRG Contains an Autobaud Support Option

- Each BRG Output May Be Routed to a Pin (e.g., BRGO1)

Refer to Figure 7-36 for the BRG block diagram.

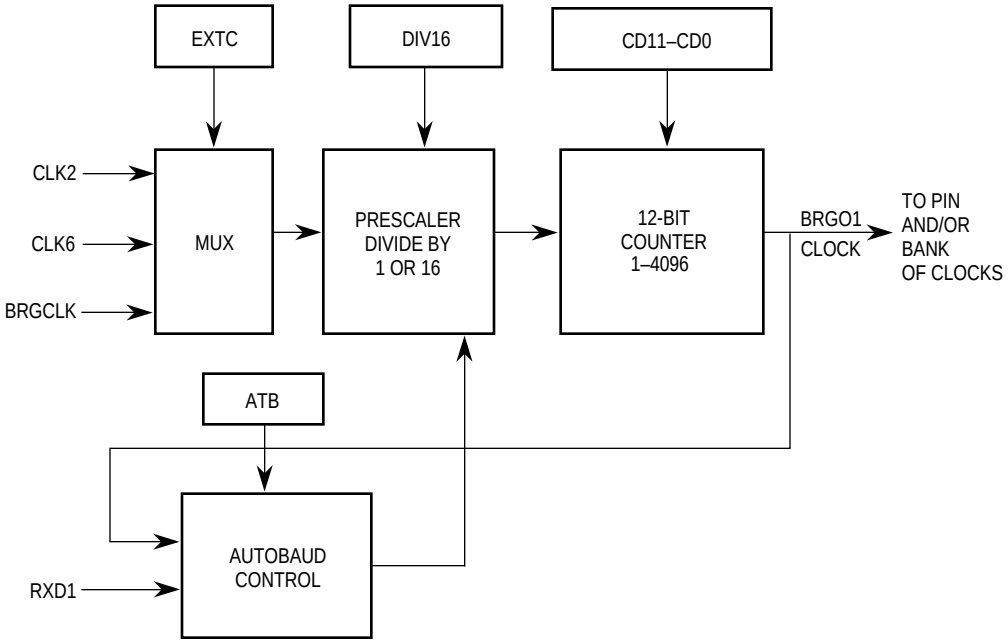


Figure 7-36. Baud Rate Generator Block Diagram

The clock input to the prescaler may be selected by the EXTC bits to come from one of three sources: BRGCLK, CLK2, or CLK6. Each source is discussed in the following paragraphs.

The BRGCLK is generated in the QUICC clock synthesizer specifically for the four BRGs (as well as a fifth BRG that is part of the SPI) and defaults to the system frequency (e.g., 25 MHz). However, the clock synthesizer in the SIM60 has an option to divide the BRGCLK by 1, 4, 16, or 64 before it leaves the clock synthesizer. Whatever the resulting frequency of BRGCLK, the user may use that frequency as the input to the QUICC BRGs.

The ability to reduce the frequency of BRGCLK before it leaves the clock synthesizer is useful in low-power applications. In a low-power mode, the BRG clocking could be a significant factor in overall QUICC power consumption. Thus, if the BRGs do not need to generate high frequencies or do not require a high resolution in the user application, a lower frequency BRGCLK may be input to the BRGs. The user may wish to dynamically change the general system clock frequency in the clock synthesizer (slow go mode) while still having the BRG run at the original frequency. The BRGCLK allows this option also.

NOTE

The BRG configuration register may be written at any time, regardless of the BRGCLK input frequency.

Alternatively, the user may choose the CLK2 or CLK6 pins to be the clock source. An external pin allows flexible baud rate frequency generation, regardless of the system frequency. Additionally, the CLK2 or CLK6 pins allow a single external frequency to become the input

clock for multiple BRGs. The clock signals on the CLK2 and CLK6 pins are not synchronized internally prior to being used by the BRG.

Next, the BRG provides a divide-by-16 option before the clock reaches the prescaler. This option is chosen by the DIV16 bit.

The clock is then divided in the prescaler by up to 4096. This input clock divide ratio can be programmed on the fly. Two on-the-fly BRG changes should not occur within a time shorter than the period of at least two BRG input clocks.

The output of the prescaler is sent internally to the bank of clocks and may also be output externally on the BRGOx pins of either the port A or port B parallel I/O. One BRGOx pin (BRGO4–BRGO1) is an output from the corresponding BRG. If the BRG divides the clock by an even value, the transitions of the BRGO pin will always occur on the falling edge of the input clock to the BRG. If the BRG is programmed to an odd value, the transitions will alternate between the falling and rising edges of the input clock.

Additionally, the output of the BRG may be sent to the autobaud control block described in the following paragraphs.

7.9.1 Autobaud Support

In the autobaud process, a UART deduces the baud rate of its received character stream by looking at the pattern received as well as the timing information of that pattern. The QUICC BRGs have a built-in autobaud control function that automatically measures the length of a start bit and modifies the baud rate accordingly. (This capability was only available on the MC68302 with a special microcode option.)

If the ATB bit in the BRG is set, the autobaud control block starts to search for a low level on the corresponding RXDx input line (RXD4–RXD1). When it finds a low level on the RXDx line, it assumes that this is 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 input clock rate, and then stops with the BRGO output clock in the low state.

After the RXDx line changes back to the high level, the autobaud control block rewrites the CD and DIV16 bits in the BRG configuration register to the divide ratio it found. Due to measurement error that can occur at high baud rates, this divide rate written by the autobaud controller may not be the precise, final baud rate desired by the user (e.g., 56600 could be the resulting baud rate, rather than 57600). Thus, an interrupt is provided to the user in the UART SCC event register to signify that the BRG configuration register was rewritten by the autobaud controller. On recognition of this interrupt, the user should rewrite the BRG configuration register with the desired value. The user is encouraged to do this as quickly as possible, even prior to the first character being fully received, to ensure that all characters are recognized correctly by the UART. The first data must have a transition from 1 to 0, then look for a one again. Thus the first cahracter must be an odd character.

Once a full character is received, the user may check in software to see if the received character matches a predefined value (such as "a" or "A"; it must be an odd character). Software should then check for other characters (such at "t" or "T") and program the SCC to the

desired parity mode. Changes in the parity mode may be accomplished in the UART protocol specific mode register (PSMR).

NOTES

The SCC associated with this BRG must be programmed to UART mode. The SCC must have the TDCR and RDCR bits in the general SCC mode register set to the 16× option for the autobaud function to operate correctly.

The input clock that is supplied to the BRG in autobaud mode should be as fast as possible to improve the accuracy of the start bit measurement. Input frequencies such as 1.8432MHz, 3.68MHz, 7.36MHz and 14.72MHz should be used.

For autobaud to operate sucessfully, the SCC performing the autobaud function must be connected to the baud rate generator for that SCC. In other words, for SCC2 to correctly perform the autobaud function, it must be clocked by BRG2. Also, for the SCC to correctly detect an autobaud lock and an interrupt to be generated, the SCC must receive three full Rx clocks from the BRG before the autobaud process begins. To do this, first set the GSMR with the ATB=0 and enable the BRG Rx clock to the highest frequency. Immediately prior to the start of the autobaud process (after device initialization) set the ATB bit equal to a one.

7.9.2 BRG Configuration Register (BRGC)

Each BRGC is a 24-bit, memory-mapped, read/write register that is cleared at reset. A reset disables the BRG and puts the BRGO output clock to the high level. The BRGC can be written at any time with no need to disable the SCCs or the external devices that are connected to the BRGO output clock. The BRG changes will occur at the end of the next BRG clock cycle (no spikes will occur on the BRGO output clock). The BRGC allows on-the-fly changes. Two on-the-fly changes to the BRG should not occur within a time shorter than the period of at least two BRG input clocks.

23	22	21	20	19	18	17	16	15	14	13	12
—						RST	EN	EXTC1–EXTC0		ATB	CD11
11	10	9	8	7	6	5	4	3	2	1	0
CD10	CD9	CD8	CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0	DIV16

Bits 23–18—Reserved

RST—Reset BRG

This bit performs a software reset of the BRG identical to that of an external reset. A reset disables the BRG and sets the BRGO output clock. (This can only be seen externally if the BRGO function is enabled to reach the corresponding port B parallel I/O pin.)

0 = Enable the BRG.
1 = Reset the BRG (software reset).

EN—Enable BRG Count

This bit is used to dynamically stop the BRG from counting, which may be useful for low-power modes.

0 = Stop all clocks to the BRG.
1 = Enable clocks to the BRG.

EXTC1–EXTC0—External Clock Source

The EXTC bits select the BRG input clock from the internal BRGCLK or one of three external pins.

00 = The BRG input clock comes from the BRGCLK (internal clock generated by the clock synthesizer in the SIM60).
01 = The BRG input clock comes from the CLK2 pin.
10 = The BRG input clock comes from the CLK6 pin.
11 = Reserved.

ATB—Autobaud

When set, this bit selects autobaud operation of the BRG on the corresponding RXDx pin.

0 = Normal operation of the BRG.
1 = When RXDx goes low, the BRG will determine the length of the start bit and synchronize the BRG to the actual baud rate.

NOTE

This bit must remain clear (0) until the SCC receives 3 Rx clocks, then the user must set this bit to one in order to obtain the correct baud rate. When the baud rate is obtained and locked it will be indicated by setting the AB bit in the UART event register.

CD11–CD0—Clock Divider

The clock divider bits, CD11–CD0, and the prescaler determine the BRG output clock rate. CD11–CD0 are used to preset a 12-bit counter that is decremented at the prescaler output rate. The counter is not accessible to the user. When the counter reaches zero, it is reloaded from the clock divider bits. Thus, a value of $FFF in CD11–CD0 produces the minimum clock rate (divide by 4096), and a value of $0000 produces the maximum clock rate (divide by 1).

Even 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.

DIV16—BRG Clock Prescaler Divide by 16

The BRG clock prescaler bit selects a divide-by-1 or divide-by-16 prescaler for the clock divider input.

7.9.3 UART Baud Rate Examples

For synchronous communication using internal baud rate generator, the BRGO output clock must never be faster than the SyncCLK frequency divided by 2. Produced in the clock synthesizer in the SIM60, SyncCLK is the frequency used internally by the synchronization circuitry in the SCCs, SMCs, and SPI. It defaults to the main system frequency (e.g., 25 MHz). Thus, with a 25-MHz system where the SyncCLK is the same as the main system frequency, the maximum BRGO output clock rate is 12.5 MHz.

The user should program the UART to 16× oversampling (RDCR and TDCR bits in the general SCC mode register) when using the SCC as a UART. (On the QUICC, 8× and 32× options are also available.) Assuming 16× oversampling is chosen in the UART, a data rate of 25MHz ÷ 16 = 1.5625 Mbits/sec is the maximum possible UART speed.

Putting this together, the following formula for calculating the bit rate based on a particular BRG configuration for a UART: async baud rate = (BRGCLK or CLK2 or CLK6) ÷ (clock divider + 1) ÷ (1 or 16 depending on the DIV16 bit) ÷ (8 or 16 or 32 according to the RDCR and TDCR bits in the general SCC mode register).

Table 7-3 lists examples of typical bit rates of asynchronous communication. Note that for this mode, the internal clock rate is assumed to be 16× the baud rate.

Table 7-4. Typical Baud Rates of Asynchronous Communication

Baud Rates	QUICC System Frequency (MHz)								
	20			25			24.5760		
	Div16	Div	Actual Frequency	Div16	Div	Actual Frequency	Div16	Div	Actual Frequency
50	1	1561	50.02	1	1952	50	1	1919	50
75	1	1040	75.05	1	1301	75	1	1279	75
150	1	520	149.954	1	650	150	1	639	150
300	1	259	300.48	1	324	300.5	1	319	300
600	0	2082	600.09	0	2603	600	0	2559	600
1200	0	1040	1200.7	0	1301	1200	0	1279	1200
2400	0	520	2399.2	0	650	2400.1	0	639	2400
4800	0	259	4807.7	0	324	4807.69	0	319	4800
9600	0	129	9615.4	0	162	9585.9	0	159	9600
19200	0	64	19231	0	80	19290	0	79	19200
38400	0	32	37879	0	40	38109	0	39	38400
57600	0	21	56818	0	26	57870	0	26	56889
115200	0	10	113636	0	13	111607	0	12	118154

NOTE: All values are decimal.

For synchronous communication, the internal clock is identical to the baud rate output. To get the desired rate, the user can select the appropriate system clock according to the following equation:

sync baud rate = (BRGCLK or CLK2 or CLK6) ÷ (clock divider + 1) ÷ (1 or 16 according to the DIV16 bit)

For example, to get the rate of 64 kbps, the system clock can be 24.96 MHz, DIV16 = 0, and the clock divider = 389.

7.10 SERIAL COMMUNICATION CONTROLLERS (SCCS)

The SCC key features are as follows:

- Implements HDLC/SDLC, HDLC Bus, BISYNC, Synchronous Start/Stop, Asynchronous Start/Stop (UART), AppleTalk (LocalTalk), and Totally Transparent Protocols
- Ethernet Version of QUICC Supports Full 10 Mbps Ethernet/IEEE 802.3 on SCC1
- Additional Protocols Supported Through Motorola-Supplied RAM Microcodes: Profibus, Signaling System#7 (SS7), Async HDLC, DDCMP, V.14, and X.21 (see Appendix C RISC Microcode from RAM**)**.
- 2 Mbps HDLC, HDLC Bus, and/or Transparent Data Rates Supported on All Four SCCs Simultaneously (Full Duplex).
- 10 Mbps Ethernet (Half Duplex) on SCC1 and 2 Mbps on the Other SCCs Supported Simultaneously (Full Duplex)
- A Single HDLC or Transparent Channel Can Be Supported at 8 Mbps (Full Duplex)
- SCC Clocking Rates up to 12.5 MHz at 25 MHz.
- DPLL Circuitry for Clock Recovery with NRZ, NRZI, FM0, FM1, Manchester, and Differential Manchester (Also Known as Differential Biphase-L)
- SCC Clocks May Be Derived from a Baud Rate Generator, an External Pin, or DPLL. Data Clock May Be as High as 3.125 MHz with a 25-MHz Clock
- Supports Automatic Control of the $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ Modem Signals
- Multibuffer Data Structure for Receive and Transmit (up to 224 BDs May Be Partitioned in Any Way Desired)
- Deep FIFOs (SCC1 Has 32-Byte Rx and Tx FIFOs; SCC2, SCC3, and SCC4 Have 16-Byte Rx and Tx FIFOs)
- Transmit-On-Demand Feature Decreases Time to Frame Transmission
- Low FIFO Latency Option for Transmit and Receive in Character-Oriented and Totally Transparent Protocols
- Frame Preamble Options
- Full-Duplex Operation
- Fully Transparent Option for Receiver/Transmitter While Another Protocol Executes on the Transmitter/Receiver
- Echo and Local Loopback Modes for Testing

NOTE

The performance figures listed in the key features assume a 25-MHz system clock.

7.10.1 SCC Overview

The QUICC has four SCCs that can be configured independently to implement different protocols. Together, they can be used to implement bridging functions, routers, gateways, and interface with a wide variety of standard wide area networks, local area networks, and proprietary networks. The SCCs have many physical interface options such as interfacing to TDM buses, ISDN buses, and standard modem interfaces (see 7.8 Serial Interface with Time Slot Assigner).

On the QUICC, the SCC does not include the physical interface, but it is the logic which formats and manipulates the data obtained from the physical interface. That is why the SI section is described separately. The choice of protocol is independent of the choice of physical interface.

The SCC is described in terms of the protocol that it is chosen to run. When an SCC 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 ISO model). Many functions of the SCC are common to all of the protocols. These functions are described first in the SCC description. Following that, the specific implementation details that make one protocol different from the others are discussed, beginning with the UART protocol. Thus, the reader should read from this point up to the UART protocol, and then skip to the particular protocol desired. Since the SCCs use similar data structures across all protocols, the reader's learning time will decrease dramatically after understanding the first protocol.

Each SCC supports a number of protocols: HDLC/SDLC, HDLC bus, BISYNC, asynchronous or synchronous start/stop (UART), totally transparent operation, and AppleTalk (i.e., LocalTalk). In addition, Ethernet is available on SCC1 of the Ethernet version of the QUICC. Although the protocol that is chosen usually applies to both the SCC transmitter and receiver, the SCCs have an option of running one-half of the SCC with transparent operation while the other half runs the standard protocol.

Each of the internal clocks (RCLK, TCLK) for each SCC can be programmed with either an external or internal source. The internal clocks can originate from one of four baud rate generators or one of four external clock pins. These clocks may be as fast as a 1:2 ratio of the system clock (i.e., 12.5 MHz); however, the SCC’s ability to support a sustained bit stream depends on the protocol as well as other factors. See Appendix A Serial Performance for more details.

Associated with each SCC is a digital phase-locked loop (DPLL) for external clock recovery. The clock recovery options include NRZ, NRZI, FM0, FM1, Manchester, and Differential Manchester. The DPLL may be configured to NRZ operation in order to pass the clocks and data to/from the SCCs without modifying them.

Each SCC may be connected to its own set of pins on the QUICC. This configuration is called the NMSI and is described in 7.8 Serial Interface with Time Slot Assigner. In this configuration, each SCC can support the standard modem interface signals ($\overline{RTS}$, $\overline{CTS}$, and $\overline{CD}$) through the port C pins and the CPM interrupt controller (CPIC). Additional handshake signals may be supported with additional parallel I/O lines.

Refer to the SCC block diagram in Figure 7-37.

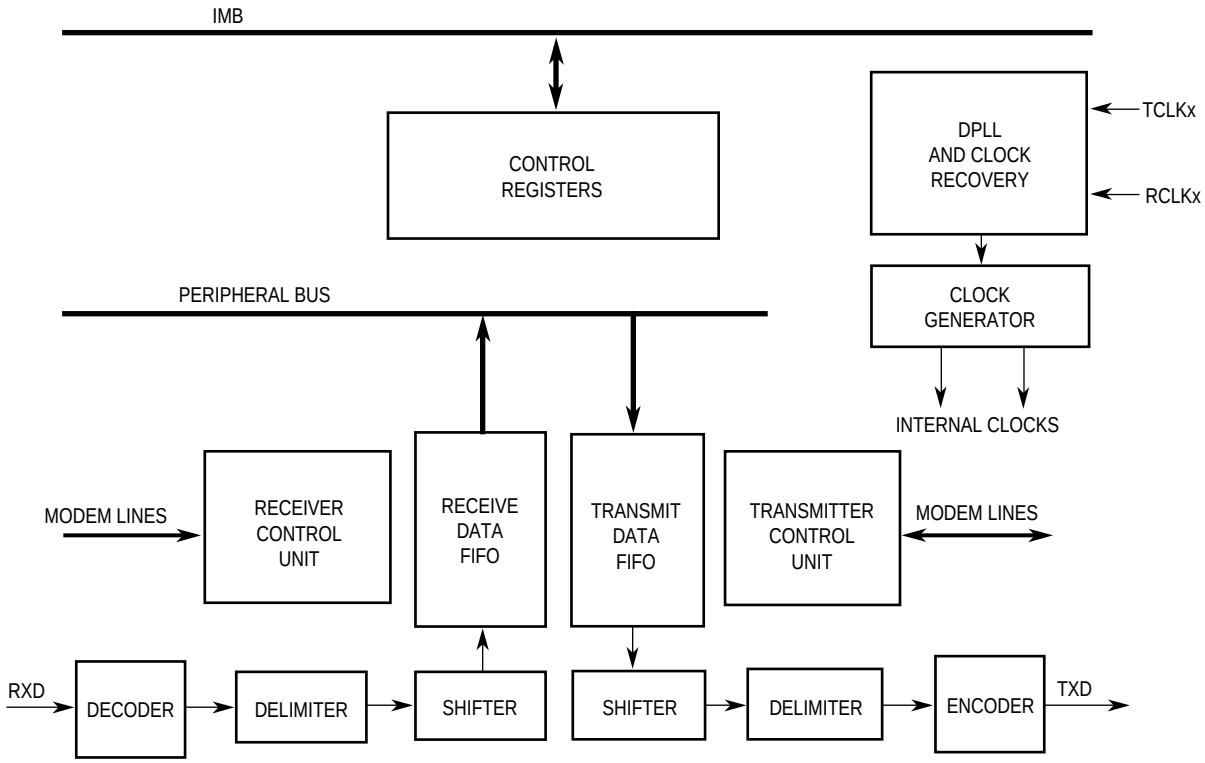


Figure 7-37. SCC Block Diagram

7.10.2 General SCC Mode Register (GSMR)

Each SCC contains a GSMR that defines all the options that are common to each SCC, regardless of the protocol. Detailed descriptions of some of the operations of the GSMR are given in later sections. GSMR is a read-write register that is cleared at reset. Since GSMR is 64 bits in length, it is accessed as GSMR_L and GSMR_H. GSMR_L contains the first (low-order) 32 bits of GSMR; GSMR_H contains the last 32 bits of GSMR.

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
—															GDE
47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
TCRC		REVD	TRX	TTX	CDP	CTSP	CDS	CTSS	TFL	RFW	TXSY	SYNL		RTSM	RSYN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
—	EDGE		TCI	TSNC		RINV	TINV	TPL			TPP		Tend	TDCR	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDCR		RENC		TENC				DIAG		ENR	ENT	MODE			

Bits 63–49, 31—Reserved

GDE—Glitch Detect Enable

This bit determines whether the SCC will look for glitches on the external receive and transmit serial clock lines provided to this SCC. If this feature is enabled, the presence of a glitch will be reported in the SCC event register. Whether or not GDE is set, the SCC always attempts to clean up the clocks that it uses internally via a Schmitt trigger on the input lines.

0 = No glitch detection is performed. This option should be chosen if the external serial clock exceeds the limits of the glitch detection logic (6.25 MHz assuming a 25-MHz system clock). This option should also be chosen if the SCC clock is provided from one of the internal baud rate generators. Lastly, this option should be chosen if external clocks are used and it is more important to minimize power consumption than to watch for glitches.

1 = Glitch detection is performed with a maskable interrupt generated in the SCC event register.

TCRC—Transparent CRC (Valid for a Totally Transparent Channel Only)

These bits select the type of frame checking that is provided on the transparent channels of this SCC (either the receiver, transmitter, or both as defined by TTX and TRX). Although this configuration selects a frame check type, the actual decision to send the frame check is made in the Tx BD. Thus, it is not required to send a frame check in transparent mode. If a frame check is not used, the user may simply ignore the frame check errors that are generated on the receiver.

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

REVD—Reverse Data (Valid for a Totally Transparent Channel Only)

0 = Normal operation.

1 = When set, this bit will cause the totally transparent channels on this SCC (either the receiver, transmitter, or both as defined by TTX and TRX) to reverse the bit order, transmitting the MSB of each octet first. See 7.10.20.11 BISYNC Mode Register (PSMR) for the method of reversing the bit order in the BISYNC protocol.

TRX—Transparent Receiver

The QUICC SCCs offer totally transparent operation. However, to increase flexibility, totally transparent operation is not configured with the MODE bits, but with the TTX and TRX bits. This gives the user the opportunity to implement unique applications, such as an SCC transmitter configured to UART and the receiver configured to totally transparent operation. To do this, set MODE = UART, 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 SCC is obtained by setting both TTX and TRX.

An SCC cannot operate with Ethernet on its transmitter simultaneously with transparent operation on its receiver, or erratic behavior will result. In other words, if the GSMR MODE = Ethernet, TTX must equal TRX, or erratic operation will result.

TTX—Transparent Transmitter

The QUICC SCCs offer totally transparent operation. However, to increase flexibility, totally transparent operation is not configured with the MODE bits, but with the TTX and TRX bits. This gives the user the opportunity to implement unique applications, such as an SCC receiver configured to HDLC and the transmitter configured to totally transparent operation. To do this, set MODE = HDLC, TTX = 1, and TRX = 0.

0 = Normal operation.
1 = The transmitter operates in totally transparent mode, regardless of the protocol selected for the receiver in the MODE bits.

NOTE

Full-duplex totally transparent operation for an SCC is obtained by setting both TTX and TRX.

An SCC cannot operate with Ethernet on its receiver simultaneously with transparent operation on its transmitter, or erratic behavior will result. In other words, if the GSMR MODE = Ethernet, TTX must equal TRX, or erratic operation will result.

CDP—$\overline{\text{CD}}$ Pulse

0 = Normal operation—envelope mode. The $\overline{\text{CD}}$ pin should envelope the frame, and negating $\overline{\text{CD}}$ while receiving will cause a CD lost error.
1 = Pulse mode. Once the $\overline{\text{CD}}$ pin is asserted, synchronization has been achieved, and further transitions of $\overline{\text{CD}}$ will have no effect on reception.

This mode is similar to the way the $\overline{\text{CD}}$ (sync) pin is used on the MC68302 in totally transparent mode. To mimic this behavior on the QUICC, the external sync signal should be connected to the $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ pins on the QUICC, and the CDP and CTSP bits should be set.

NOTE

This bit must be set if this SCC is used in TSA.

CTSP—$\overline{\text{CTS}}$ Pulse

0 = Normal operation—envelope mode. The $\overline{\text{CTS}}$ pin should envelope the frame, and a negation of $\overline{\text{CTS}}$ while transmitting will cause a CTS lost error.
1 = Pulse mode. Once the $\overline{\text{CTS}}$ pin is asserted, synchronization has been achieved, and further transitions of $\overline{\text{CTS}}$ will have no effect on transmission.

CDS—$\overline{CD}$ Sampling

0 = The $\overline{CD}$ input is assumed to be asynchronous with the data. It is internally synchronized by the SCC, and then data is received.

1 = The $\overline{CD}$ input is assumed to be synchronous with the data, giving faster operation. In this mode, $\overline{CD}$ must transition while the receive clock is in the low state. As soon as $\overline{CD}$ is low, data begins being received. This mode is especially useful when connecting QUICCs in transparent mode since it allows the $\overline{RTS}$ pin of one QUICC to be directly connected to the $\overline{CD}$ pin of the other QUICC.

CTSS—$\overline{CTS}$ Sampling

0 = The $\overline{CTS}$ input is assumed to be asynchronous with the data. It is internally synchronized by the SCC, and data is then transmitted after several serial clock delays.

1 = The $\overline{CTS}$ input is assumed to be synchronous with the data, giving faster operation. In this mode, $\overline{CTS}$ must transition while the transmit clock is in the low state. As soon as $\overline{CTS}$ is low, data immediately begins transmission. This mode is especially useful when connecting QUICCs in transparent mode since it allows the $\overline{RTS}$ pin of one QUICC to be directly connected to the $\overline{CTS}$ pin of the other QUICC.

TFL—Transmit FIFO Length

0 = Normal operation. The transmit FIFO is 32 bytes for SCC1 and 16 bytes for the other SCCs.

1 = The transmit FIFO is 1 byte. This may be used with character-oriented protocols such as UART to ensure a minimum FIFO latency at the expense of performance.

RFW—Rx FIFO Width

0 = Rx FIFO is 32-bits wide for maximum performance. Data will not normally be written to receive buffers until at least 32 bits have been received. This configuration is required for HDLC-type protocols and Ethernet; it is the recommended configuration for high-performance transparent modes. In this mode, the receive FIFO is 32 bytes for SCC1 and 16 bytes for the other SCCs.

1 = Low-latency operation. The Rx FIFO is 8-bits wide, and the receive FIFO is one-fourth its normal size (8 bytes for SCC1 and 4 bytes for the other SCCs). This allows data to be written to the data buffer each time a character is received, without waiting for 32 bits to be received. This configuration must be chosen for character-oriented protocols such as UART and BISYNC. It may also be used for low-performance, low-latency, totally transparent operation if desired. It must not be used with HDLC, HDLC Bus, AppleTalk, or Ethernet, or erratic behavior may result.

TXSY—Transmitter Synchronized to the Receiver

The TXSY bit is particularly intended for X.21 applications where the transmitted data must begin an exact multiple of 8-bit periods after the receive data arrives.

0 = No synchronization between receiver and transmitter (default).

1 = The transmit bit stream is synchronized to the receiver. Additionally, if RSYN = 1, then transmission in the totally transparent mode will not occur until the receiver has synchronized with the bit stream and the $\overline{CTS}$ signal is asserted to the SCC. Assuming $\overline{CTS}$ is already asserted, transmission will begin eight clocks after the

receiver begins receiving data. This behavior is similar to the MC68302 totally transparent mode behavior when the EXSYN bit in its SCC mode register is set.

SYNL—Sync Length (BISYNC and Transparent Mode Only)

These bits determine the operation of an SCC receiver that is configured for BISYNC or totally transparent operation only. See the data synchronization register definition in the BISYNC and totally transparent descriptions for more information.

00 = The sync pattern in the DSR is not used. An external sync signal is used instead ($\overline{CD}$ pin asserted).
01 = 4-bit sync. The receiver will synchronize on a 4-bit sync pattern stored in DSR. This character and additional syncs can be programmed to be stripped using the SYNC character in the parameter RAM. The transmitter will transmit the entire contents of the DSR prior to each frame.
10 = 8-bit sync. This option should be chosen along with the BISYNC protocol to implement mono-sync. The receiver will synchronize on an 8-bit sync pattern stored in DSR. The transmitter will transmit the entire contents of the DSR prior to each frame.
11 = 16-bit sync. Also called BISYNC. The receiver will synchronize on a 16-bit sync pattern stored in DSR. The transmitter will transmit the DSR prior to each frame.

RTSM—$\overline{RTS}$ Mode

This bit may be changed on the fly.

0 = Send idles between frames as defined by the protocol and the Tend bit. $\overline{RTS}$ is negated between frames (default).
1 = Send flags/syncs between frames according to the protocol. $\overline{RTS}$ is always asserted whenever the SCC is enabled.

RSYN—Receive Synchronization Timing (Valid for a Totally Transparent Channel Only)

0 = Normal operation.
1 = If CDS = 1, then the $\overline{CD}$ pin should be asserted on the second bit of the receive frame, rather than the first. This configuration matches the behavior of the MC68302 totally transparent receiver when its EXSYN bit is set; it is included on the QUICC for compatibility.

EDGE—Clock Edge

The EDGE bits determine the clock edge used by the DPLL for adjusting the receive sample point due to jitter in the received signal. The selection of the EDGE bits is ignored in the UART protocol or the x1 mode of the RDCR bits.

00 = Both the positive and negative edges are use for changing the sample point (default).
01 = Positive edge. Only the positive edge of the received signal is used for changing the sample point.
10 = Negative edge. Only the negative edge of the received signal is used for changing the sample point.
11 = No adjustment is made to the sample points.

TCI—Transmit Clock Invert

0 = Normal operation.

1 = The internal transmit clock (TCLK) is inverted by the SCC before it is used. This option allows the SCC to clock data out one-half clock earlier, on the rising edge of TCLK rather than the falling edge. In this mode, 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 transmit clock can be latched by the external receiver one clock cycle later on the next rising edge of the same transmit clock. This option is recommended for Ethernet, HDLC, or transparent operation when the clock rates are high (e.g., above 8 MHz) to improve data setup time for the external receiver.

TSNC—Transmit Sense

This bit indicates the amount of time the internal sense signal will stay active after the last transition on the RXD pin, indicating that the line is free. For instance, these bits can be used in the AppleTalk protocol to avoid the spurious $\overline{CS}$-changed interrupt that would otherwise occur during the frame sync sequence that precedes the opening flags.

If RDCR is configured to 1× 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 lesser of the two numbers listed.

00 = Infinite—carrier sense is always active (default)

01 = 14 or 6.5 bit times as determined by the RDCR bits

10 = 4 or 1.5 bit times as determined by the RDCR bits (normally chosen for AppleTalk)

11 = 3 or 1 bit times as determined by the RDCR bits

RINV—DPLL Receive Input Invert Data

0 = No invert

1 = Invert the data before it is sent to the on-chip DPLL for reception. This setting is used to produce FM1 from FM0, NRZI space from NRZI mark, etc. It may also be used in regular NRZ mode to invert the data stream.

NOTE

This bit must be 0 in HDLC BUS mode.

TINV—DPLL Transmit Input Invert Data

0 = No invert

1 = Invert the data before it is sent to the on-chip DPLL for transmission. This setting is used to produce FM1 from FM0, NRZI space from NRZI mark, etc. It may also be used in regular NRZ mode to invert the data stream.

NOTE

This bit must be 0 in HDLC BUS mode.

In T1 applications, setting TINV and TEND creates a continuously inverted HDLC data stream.

TPL—Tx Preamble Length

The TPL bits determine 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

TPP—Tx Preamble Pattern

The TPP bits determine what, if any, bit pattern should precede the start of each transmit frame. The preamble pattern will be sent prior to the first flag/sync of the frame. TPP is ignored if the SCC is programmed to UART mode. The length of the preamble is programmed in TPL. The preamble pattern is typically transmitted to a receiving station that uses a DPLL for clock recovery. The receiving DPLL uses the regular pattern of the preamble to help it lock onto the received signal in a short, predictable time period.

00 = All zeros
01 = Repeating 10's (Select this setting for Ethernet operation.)
10 = Repeating 01's
11 = All ones (Select this setting for LocalTalk operation.)

Tend—Transmitter Frame Ending

This bit is intended particularly for the NMSI transmitter encoding of the DPLL. Tend determines whether the TXD line should idle in a high state or in an encoded ones state (which may be either high or low). It may, however, be used with other encodings besides NMSI.

0 = Default operation. The TXD line is encoded only when data is transmitted (including the preamble and opening and closing flags/syncs). When no data is available to transmit, the line is driven high.
1 = The TXD line is always encoded (even when idles are transmitted).

TDCR—Transmit Divide Clock Rate

The TDCR bits determine the divider rate of the transmitter. If the DPLL is not used, the 1× value should be chosen, except in asynchronous UART mode where 8×, 16×, or 32× must be chosen. The user should program TDCR to equal RDCR in most applications.

If the DPLL is used in the application, the selection of TDCR depends on the encoding. NRZI usualy requires 1×; whereas, FM0/FM1, Manchester, and Differential Manchester allow 8×, 16×, or 32×. The 8× option allows highest speed; whereas, the 32× option provides the greatest resolution. TDCR is usually equal to RDCR to allow the same clock frequency source to control both the transmitter and receiver.

00 = 1× clock mode (Only NRZ or NRZI encodings are allowed.)
01 = 8× clock mode
10 = 16× clock mode (normally chosen for UART and AppleTalk)
11 = 32× clock mode

RDCR—Receive DPLL Clock Rate

The RDCR bits determine the divider rate of the receive DPLL. If the DPLL is not used, the 1× value should be chosen, except in asynchronous UART mode where 8×, 16×, or 32× must be chosen. The user should program RDCR to equal TDCR in most applications.

If the DPLL is used in the application, the selection of RDCR depends on the encoding. NRZI usualy requires 1×; whereas, FM0/FM1, Manchester, and Differential Manchester allow 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 decodings are allowed.)
01 = 8× clock mode
10 = 16× clock mode (normally chosen for UART and AppleTalk)
11 = 32× clock mode

RENC—Receiver Decoding Method

Select NRZ if the DPLL is not used. The user should program RENC to equal TENC in most applications. Do not use this internal DPLL for Ethernet mode.

000 = NRZ (default setting if DPLL is not used)
001 = NRZI Mark (set RINV also for NRZI Space)
010 = FM0 (set RINV also for FM1)
011 = Reserved
100 = Manchester
101 = Reserved
110 = Differential Manchester (Differential Biphase-L)
111 = Reserved

TENC—Transmitter Encoding Method

Select NRZ if the DPLL is not used. The user should program TENC to equal RENC in most applications. Do not use this internal DPLL for Ethernet mode.

000 = NRZ (default setting if DPLL is not used)
001 = NRZI Mark (set TINV also for NRZI Space)
010 = FM0 (set TINV also for FM1)
011 = Reserved
100 = Manchester
101 = Reserved
110 = Differential Manchester (Differential Biphase-L)
111 = Reserved

DIAG—Diagnostic Mode

In normal operation mode, the SCC operates normally. The receive data enters the RXD pin and the transmit data is shifted out through the TXD pin. The SCC uses the modem signals ($\overline{CD}$ and $\overline{CTS}$) to automatically enable and disable transmission and reception. These timings are shown in 7.10.11 SCC Timing Control.

00 =Normal operation ($\overline{CTS}$ and $\overline{CD}$ signals under automatic control)

In local loopback mode, the transmitter output is internally connected to the receiver input, while the receiver and the transmitter operate normally. The value on the RXD pin is ig-

nored. Data can be programmed to appear on the TXD pin, or the TXD pin can remain high by programming the port A register. The $\overline{RTS}$ line can also be programmed to be disabled in the appropriate parallel I/O register. In TDM modes, the L1TXDx and L1RQx lines can be programmed to be either asserted normally or to remain inactive by programming the serial interface mode register (SIMODE).

When using local loopback mode, the clock source for the transmitter and the receiver must be the same. Thus, the same baud rate generator may be used for both transmitter and receiver, or the same external CLKx pin may be used for both transmitter or receiver. (Separate CLKx pins may be used with the transmitter and receiver as long as the CLKx pins are connected to the same external clock signal source.)

01 =Local loopback mode

NOTE

If external loopback is desired, the DIAG bits should be selected for normal operation, and an external connection should be made between the TXD and RXD pins. Clocks may be generated internally by a baud rate generator or generated externally. The user may physically connect the appropriate control signals ($\overline{RTS}$ connected to $\overline{CD}$, and $\overline{CTS}$ grounded) or the port C register may be used to cause the $\overline{CD}$ and $\overline{CTS}$ pins to be permanently asserted to the SCC.

In automatic echo mode, the channel automatically retransmits the received data on a bit-by-bit basis using whatever receive clock is provided. The receiver operates normally and can receive data if $\overline{CD}$ is asserted. The transmitter simply transmits received data. In this mode, the $\overline{CTS}$ line is ignored.

The echo function may also be accomplished in software by receiving buffers from an SCC, linking them to Tx BDs and then transmitting them back out of that SCC.

10 = Automatic echo mode

In loopback/echo mode, loopback operation and echo operation occur simultaneously. The $\overline{CD}$ pin and $\overline{CTS}$ pins are ignored. See the loopback bit description for clocking requirements.

11 = Loopback and echo mode

NOTE

Users familiar with the MC68302 may notice that the QUICC does not contain "software operation" mode. The software operation mode as implemented on the MC68302 can be implemented on the QUICC using parallel I/O port C.

ENR—Enable Receive

This bit enables the receiver hardware state machine for this SCC. When ENR is cleared, the receiver is disabled, and any data in the receive FIFO is lost. If ENR is cleared during reception, the receiver aborts the current character. ENR may be set or cleared regard-

less of whether serial clocks are present. See 7.10.14 Disabling the SCCs on the Fly for a description of the proper methods to disable and reenable an SCC.

NOTE

The SCC provides other tools to control reception besides the ENR bit. They are the ENTER HUNT MODE command, the CLOSE Rx BD command, and the E-bit in the Rx BD.

ENT—Enable Transmit

This bit enables the transmitter hardware state machine for this SCC. When ENT is cleared, the transmitter is disabled. If ENT is cleared during transmission, the transmitter aborts the current character, and the TXD pin returns to the idle state. Data already in the transmit shift register will not be transmitted. ENT may be set or cleared regardless of whether serial clocks are present. See 7.10.14 Disabling the SCCs on the Fly for a description of the proper methods to disable and reenable an SCC.

NOTE

The SCC provides other tools to control transmission besides the ENT bit. They are the STOP TRANSMIT command, the GRACEFUL STOP TRANSMIT command, the RESTART TRANSMIT command, the freeze option in UART mode, the $\overline{\text{CTS}}$ flow control option in UART mode, and the ready (R) bit in the Tx BD.

MODE—Channel Protocol mode

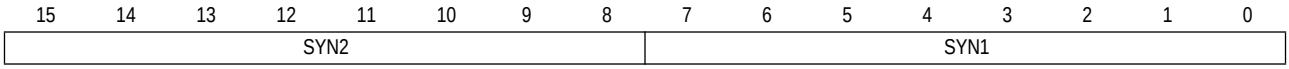
0000 = HDLC
0001 = Reserved
0010 = AppleTalk (LocalTalk)
0011 = SS7 (reserved for RAM microcode)
0100 = UART
0101 = Profibus (reserved for RAM microcode)
0110 = Async HDLC (reserved for RAM microcode)
0111 = V.14 (reserved for RAM microcode)
1000 = BISYNC
1001 = DDCMP (reserved for RAM microcode)
1010 = Reserved
1011 = Reserved
1100 = Ethernet
11xx = Reserved

7.10.3 SCC Protocol-Specific Mode Register (PSMR)

The functionality of the SCC varies according to the protocol selected by the MODE bits in the GSMR. Each of the four SCC has an additional 16-bit, memory-mapped, read-write PSMR that configures the SCC to the special configurations within a chosen mode. A detailed description of the PSMR bits is contained within each specific protocol. The PSMRs are cleared at reset.

7.10.4 SCC Data Synchronization Register (DSR)

Each of the four SCC has a 16-bit, memory-mapped, read-write DSR. The DSR specifies the pattern used in the frame synchronization procedure in the synchronous protocols. In the UART protocol, it is used to configure fractional stop bit transmission. In the BISYNC and totally transparent protocol, it should be programmed with the desired SYNC pattern. In the Ethernet protocol, it should be programmed with $D555. At reset, it defaults to $7E7E (two HDLC flags), so it does not need to be written for HDLC mode. When DSR is used to send out SYNCs (such as in BISYNC or transparent mode), the contents of the DSR are always transmitted LSB first.



7.10.5 SCC Transmit on Demand Register (TODR)

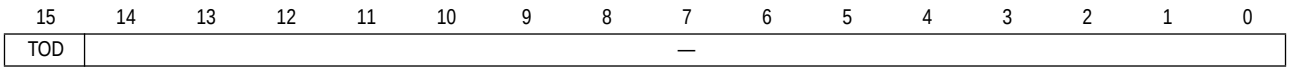
If no frame is currently being transmitted by an SCC, the RISC controller periodically polls the R-bit of the next Tx BD to see if the user has requested a new frame/buffer to be transmitted. This polling algorithm depends on the SCC configuration, but occurs every 8 to 32 serial transmit clocks. The user, however, has an option to request that the RISC begin the processing of the new frame/buffer immediately, without waiting until the normal polling time. To obtain immediate processing, the TOD bit in the transmit-on-demand register is set by the user once the user has set the R-bit in the Tx BD.

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 Tx BD, it can conceivably affect the servicing of the other SCC FIFOs. Therefore, it is recommended that the transmit-on-demand feature only be used when a high-priority Tx BD has been prepared and transmission on this SCC has not occurred for a period of time.

The TOD bit does not need to be set if a new Tx BD is added to the circular queue but other Tx BDs in that queue have not fully completed transmission. In that case, the new Tx BD will be processed immediately following the completion of the older Tx BD s.

The first bit of the frame will typically be clocked out 5-6 bit times after TOD has been written to a 1.

TOD—Transmit on Demand



0 = Normal operation
1 = The RISC will give a high priority to the current Tx BD and will not wait for the normal polling time to check that the Tx BD’s R-bit has been set. It will begin transmitting the frame. This bit will be cleared automaticaly after one serial clock.

Bits 14–0—Reserved

These bits should be written with zeros.

7.10.6 SCC Buffer Descriptors

Data associated with each SCC channel is stored in buffers. Each buffer is referenced by a BD, which may be located anywhere in internal memory. The QUICC internal memory has space for 224 BDs to be shared between the four SCCs and any SMCs and SPIs that are used. However, the allocation of BDs to the transmitter or receiver of a serial channel is user-defined. Thus, the user may select 100 BDs for the SCC1 receiver, 20 BDs for the SCC1 transmitter, etc.

The BD table forms a circular queue with a programmable length. The user can program the start address of each channel BD table in the internal memory (see Figure 7-38). The user is allowed to allocate the parameter area of an unused channel to the other used channels as BD tables or as actual buffers.

The format of the BDs is the same for each SCC mode of operation and for both transmit and receive. The first word in each BD contains a status and control word, which also determines the BD table length. Only this first field (containing the status and control bits) differs for each protocol. The second word determines the data length referenced to this BD, and the two last words in the BD contain the 32-bit address pointer that points to the actual buffer in memory.

	15 … 0
OFFSET + 0	STATUS AND CONTROL
OFFSET + 2	DATA LENGTH
OFFSET + 4	HIGH-ORDER DATA BUFFER POINTER
OFFSET + 6	LOW-ORDER DATA BUFFER POINTER

For frame-oriented protocols, a message may 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 the unlinked buffers will be provided by the CPU32+ core in time to be either transmitted or received. Failure to do so will result in an error condition being reported by the CP. An underrun error is reported in the case of transmit, and a busy error is reported in the case of receive.

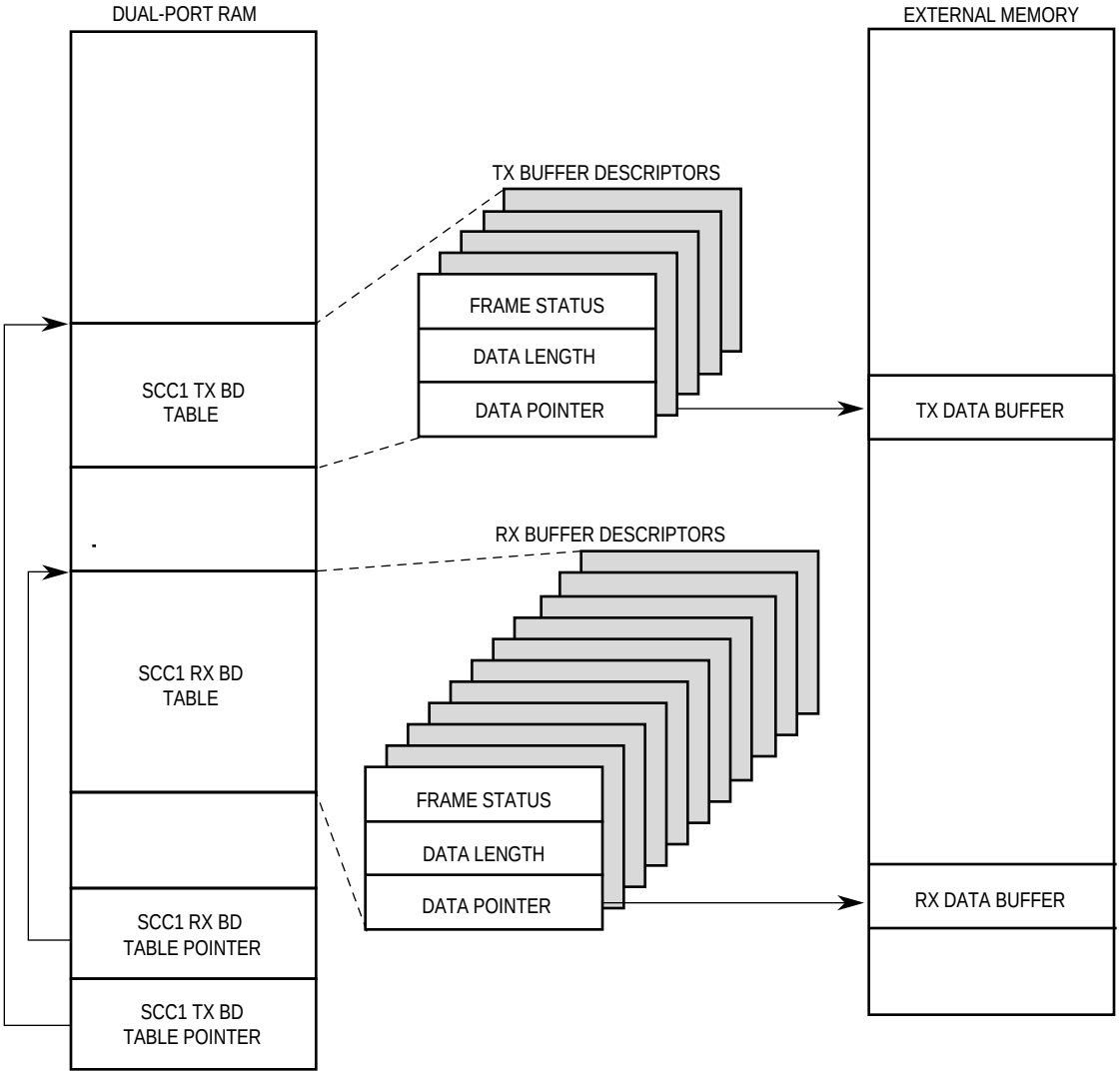


Figure 7-38. SCC Memory Structure

All protocols can have their buffer descriptors point to data buffers that are located in internal dual-port RAM. Typically, however, due to the internal RAM being used for buffer descriptors, it is customary for the data buffers to be located in external RAM, especially if the data buffers are large in size. In all cases, the IMB is used to transfer the data to the data buffer.

The CP processes the Tx BDs in a straightforward fashion. Once the transmit side of an SCC is enabled, it starts with the first BD in that SCC's transmit table. Once the CP detects that the Tx BD R-bit was set, it will begin processing the buffer. (The CP will detect that the BD is ready either by polling the R-bit periodically or by the user writing to the transmit-on-demand register (TODR).) Once the data from the BD has been placed in the transmit FIFO, the CP moves on to the next BD, again waiting for that BD's 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 sets the R-bit to not-

ready; thus, the CP will not use a BD twice until the BD has been confirmed by the CPU32+ core. (The one exception to this rule is that the QUICC supports an option for repeated transmission, called the continuous mode, whereby the R-bit is left in the ready position. This is available in some protocols.)

The CP uses the Rx BDs in a similar fashion. Once the receive side of an SCC is enabled, it starts with the first BD in that SCC's Rx BD table. Once data arrives from the serial line into the SCC, the CP performs certain required protocol processing on the data and moves the resultant data to the data buffer pointed to by the first BD. Use of a BD is complete when there is no more room left in the buffer or when certain events occur, such as detection of an error or an end-of-frame. Whatever the reason, the buffer is then said to be closed, and additional data will be stored using the next BD. Whenever the CP needs to begin using a BD because new data is arriving, it will check the E-bit of that BD. If the current BD is not empty, it will report a busy error. However, it will not move from the current BD until it becomes empty. When the CP sees the 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 sets the E-bit to not-empty; thus, the CP will never use a BD twice until the BD has been processed by the CPU32+ core. (The one exception to this rule is that the QUICC supports an option for repeated reception, called the continuous mode, whereby the E-bit is left in the empty position. This is available in some protocols.)

7.10.7 SCC Parameter RAM

Each SCC parameter RAM area begins at the same offset from each SCC base area. The protocol-specific portions of the SCC parameter RAM are discussed in the specific protocol descriptions. The part of the SCC parameter RAM that is the same for all SCC protocols is shown in Table 7-5.

Certain parameter RAM values (marked in boldface) need to be initialized by the user before the SCC is enabled; other values are initialized/written by the CP. Once initialized, most parameter RAM values will not need to be accessed in user software since most of the activity is centered around the transmit and Rx BDs, not the parameter RAM. However, if the parameter RAM is accessed by the user, the following regulations should be noted. The parameter RAM can be read at any time. The parameter time values related to the SCC transmitter can only be written whenever the transmitter is disabled (see 7.10.14 Disabling the SCCs on the Fly), after a STOP TRANSMIT and before a RESTART TRANSMIT command, or after the buffer/frame completes transmission as a result of a GRACEFUL STOP TRANSMIT command and before a RESTART TRANSMIT command. The parameter RAM values related to the SCC receiver can only be written when the receiver is disabled (see 7.10.14 Disabling the SCCs on the Fly).

Table 7-5. SCC Parameter RAM Common to All Protocols

Address	Name	Width	Description
SCC Base + 00	RBASE	Word	Rx BD Base Address
SCC Base + 02	TBASE	Word	Tx BD Base Address
SCC Base + 04	RFCR	Byte	Rx Function Code
SCC Base + 05	TFCR	Byte	Tx Function Code
SCC Base + 06	MRBLR	Word	Maximum Receive Buffer Length
SCC Base + 08	RSTATE	Long	Rx Internal State
SCC Base + 0C		Long	Rx Internal Data Pointer
SCC Base + 10	RBPTR	Word	Rx BD Pointer
SCC Base + 12		Word	Rx Internal Byte Count
SCC Base + 14		Long	Rx Temp
SCC Base + 18	TSTATE	Long	Tx Internal State
SCC Base + 1C		Long	Tx Internal Data Pointer
SCC Base + 20	TBPTR	Word	Tx BD Pointer
SCC Base + 22		Word	Tx Internal Byte Count
SCC Base + 24		Long	Tx Temp
SCC Base + 28	RCRC	Long	Temp Receive CRC
SCC Base + 2C	TCRC	Long	Temp Transmit CRC
SCC Base + 30			First Word of Protocol-Specific Area
SCC Base + xx			Last Word of Protocol-Specific Area

NOTE: The items in boldface should be initialized by the user.

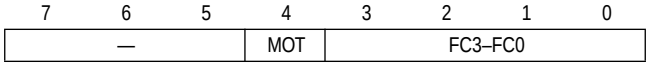
7.10.7.1 BD TABLE POINTER (RBASE, TBASE). The RBASE and TBASE entries define the starting location in the dual-port RAM for the set of BDs for receive and transmit functions of the SCC. This provides a great deal of flexibility in how BDs for an SCC are partitioned. By selecting RBASE and TBASE entries for all SCCs, and by setting the W-bit in the last BD in each BD list, the user may select how many BDs to allocate for the transmit and receive side of every SCC. The user must initialize these entries before enabling the corresponding channel. Furthermore, the user should not configure BD tables of two enabled SCCs to overlap, or erratic operation will occur.

NOTE

RBASE and TBASE should contain a value that is divisible by 8.

7.10.7.2 SCC FUNCTION CODE REGISTERS (RFCR, TFCR). There are eight separate function code registers for the four SCC channels: four for receive data buffers (RFCRx) and four for transmit data buffers (TFCRx). The FC entry contains the value that the user would like to appear on the function code pins FC3–FC0 when the associated SDMA channel accesses memory. It also controls the byte-ordering convention to be used in the transfers.

Receive Function Code Register



Bits 7–5—Reserved

MOT—Motorola

This bit should be set by the user to achieve normal operation. If this bit is modified on the fly, it will take effect at the beginning of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD otherwise. MOT must be set if the data buffer is located in external memory and has a 16-bit wide port size.

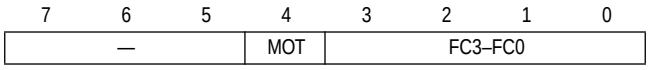
0 = Intel convention is used for byte ordering—swapped operation. It is also called little-endian byte ordering. The bytes stored in each buffer word are reversed as compared to the Motorola mode.

1 = Motorola byte ordering—normal operation. It is also called big-endian byte ordering. As data is received from the serial line and put into the buffer, the most significant byte of the buffer word contains data received earlier than the least significant byte of the same buffer word.

FC3–FC0—Function Code 3–0

These bits contain the function code value used during this SDMA channel's memory accesses. It is suggested that the user write bit FC3 with a one, to identify this SDMA channel access as a DMA-type access. Example: FC3–FC0 = 1000 (binary). Do not write the value 0111 (binary) to these bits.

Transmit Function Code Register



Bits 7–5—Reserved

MOT—Motorola

This bit should be set by the user to achieve normal operation. If this bit is modified on the fly, it will take effect at the beginning of the next frame (Ethernet, HDLC, and transparent) or at the beginning of the next BD otherwise. The MOT must be set if the data buffer is located in external memory and has a 16-bit wide port size.

0 = 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 Motorola mode.

1 = Motorola 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.

NOTE

The MOT must be set if the data buffer is located in external memory and has a 16-bit wide port size.

FC3–FC0—Function Code 3–0

These bits contain the function code value used during this SDMA channel's memory accesses. It is suggested that the user write bit FC3 with a one to identify this SDMA channel access as a DMA-type access. Example: FC3–FC0 = 1000 (binary). Do not write the value 0111 (binary) to these bits.

7.10.7.3 MAXIMUM RECEIVE BUFFER LENGTH REGISTER (MRBLR). Each SCC has one MRBLR to define the receive buffer length for that SCC. MRBLR defines the maximum number of bytes that the QUICC will write to a receive buffer on that SCC before moving to the next buffer. The QUICC may write fewer bytes to the buffer than MRBLR if a condition such as an error or end-of-frame occurs, but it will never write more bytes than the MRBLR value. It follows, then, that buffers supplied by the user for use by the QUICC should always be of size MRBLR (or greater) in length.

The transmit buffers for an SCC are not affected in any way by the value programmed into MRBLR. Transmit buffers may be individually chosen to have varying lengths, as needed. The number of bytes to be transmitted is chosen by programming the data length field in the Tx BD.

NOTE

MRBLR is not intended to be changed dynamically while an SCC is operating. However, if it is modified in a single bus cycle with one 16-bit move (NOT two 8-bit bus cycles back-to-back), then a dynamic change in receive buffer length can be successfully achieved. This takes place when the CP moves control to the next Rx BD in the table. Thus, a change to MRBLR will not have an immediate effect. To guarantee the exact Rx BD on which the change will occur, the user should change MRBLR only while the SCC receiver is disabled.

The MRBLR value should be greater than zero for all modes.

For Ethernet and HDLC the MRBLR should be evenly divisible by 4. In totally transparent mode, MRBLR should also be divisible by 4, unless the receive FIFO width (RFW) bit in GSMR is set to 8 bits.

7.10.7.4 RECEIVER BD POINTER (RBPTR). The RBPTR for each SCC channel points to the next BD that the receiver will transfer 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 this pointer to the value programmed in the RBASE entry. Although RBPTR need never be written by the user in most applications, it may be modified by the user when the receiver is disabled or when the user is sure that no receive buffer is currently in use.

7.10.7.5 TRANSMITTER BD POINTER (TBPTR). The TBPTR for each SCC channel points to the next BD that the transmitter will transfer 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 initializes this pointer to the value programmed in the TBASE entry.

Although TBPTR need never be written by the user in most applications, it may be modified by the user when the transmitter is disabled or when the user is sure that no transmit buffer is currently in use (e.g., after STOP TRANSMIT command is issued, or after a GRACEFUL STOP TRANSMIT command is issued, and the frame completes its transmission.)

7.10.7.6 OTHER GENERAL PARAMETERS. Additional parameters are listed in Table 7-5. These parameters do not need to be accessed by the user in normal operation, and are listed only because they may provide helpful information for experienced users and for debugging.

The Rx and Tx internal data pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.

The Tx internal byte count is a down-count value that is initialized with the Tx BD data length and decremented with every byte read by the SDMA channels. The Rx internal byte count is a down-count value that is initialized with the MRBLR value and decremented with every byte written by the SDMA channels.

NOTE

To extract data from a partially full receive buffer, the CLOSE Rx BD command may be used.

The Rx internal state, Tx internal state, Rx temp, Tx temp, and reserved areas are for RISC use only.

7.10.8 Interrupts from the SCCs

Interrupt handling for each of the SCC channels is configured on a global (per channel) basis in the CPM interrupt pending register, CPM interrupt mask register, and CPM in service register. Within each of these registers, one bit is used to either mask, enable, or report the presence of an interrupt in an SCC channel. The interrupt priority between the four SCCs is programmable in the CP interrupt configuration register. An SCC interrupt may be caused by a number of events. To allow interrupt handling for these (SCC-specific) events, further event registers are provided within the SCCs.

A number of events can cause the SCC to interrupt the processor. The events differ slightly according to the protocol selected. For a detailed description of the events see the specific protocol paragraphs. These events are handled independently for each channel by the SCC event register and the SCC mask register.

Events that can cause interrupts due to the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ modem lines are described in 7.14.9 Port C Pin Functions.

7.10.8.1 SCC EVENT REGISTER (SCCE). The 16-bit SCC event register is used to report events recognized by any of the SCCs. On recognition of an event, the SCC will set its corresponding bit in the SCC event register (regardless of the corresponding mask bit). The SCC event register appears to the user as a memory-mapped register and 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. This register is cleared at reset.

7.10.8.2 SCC MASK REGISTER (SCCM). The 16-bit, read-write SCC mask register allows the user to either enable or disable interrupt generation by the CP for specific events in each SCC channel. Note that an interrupt will only be generated if the SCC interrupts in this channel are enabled in the CPM interrupt mask register.

If a bit in the SCC mask register is zero, the CP will not proceed with its usual interrupt handling whenever that event occurs. Anytime a bit in the SCC mask register is set, a one in the corresponding bit in the SCC event register will set the SCC event bit in the CPM interrupt pending register.

The bit position of the SCC mask register is identical to that of the SCC event register.

7.10.8.3 SCC STATUS REGISTER (SCCS). The 8-bit, read-write SCC status register allows the user to monitor real-time status conditions on the RXD line, such as flags, idle, and data carrier sense. It does not show the real-time status of the $\overline{CTS}$ and $\overline{CD}$ pins. Their real-time status is available in the port C parallel I/O.

7.10.9 SCC Initialization

The SCCs require a number of registers and parameters to be configured after a power-on reset. The following outline is the proper sequence for initializing the SCCs, regardless of the protocol used. More detailed examples are given in the protocol sections.

1. Write the parallel I/O ports to configure the I/O pins to connect to the SCCs.
2. The SDCR (SDMA Configuration Register) should be initialized to $0740, rather than being left at its default value of $0000.
3. Write the port C registers to configure the $\overline{CTS}$ and $\overline{CD}$ pins to be parallel I/O with interrupt capability or to be direct connections to the SCC (if modem support is needed).
4. If the TSA is used, the SI must be configured. See 7.8 Serial Interface with Time Slot Assigner for a description of the steps required. If the SCC is used in the NMSI mode, then the SICR must still be initialized.
5. Write GSMR, but do not write the ENT or ENR bits yet.
6. Write the PSMR.
7. Write DSR.
8. Initialize the required values for this SCC in its parameter RAM.
9. Clear out any current events in the SCCE, if desired.
10. Write SCCM to enable the interrupts in the SCCE.
11. 1Write CICR to configure the SCC's interrupt priority.
12. 1Clear out any current interrupts in CIPR, if desired.
13. 1Write CIMR to enable interrupts to the CP interrupt controller.
14. 1Set the ENT and ENR bits in the GSMR.

The BDs may have their ready/empty bits set at any time. Notice that the CR does not need to be accessed following a power-on reset. An SCC 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 the RST bit in the CR is a comprehensive reset that may also be used.

7.10.10 SCC Interrupt Handling

The following describes what would normally take place within an interrupt handler for the SCC.

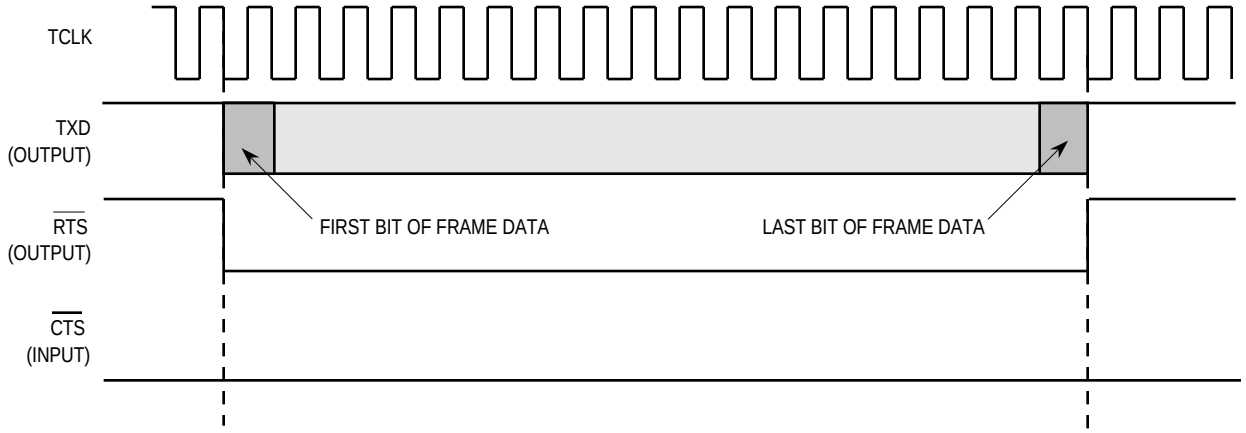
1. Once an interrupt occurs, the SCCE should be read by the user to see which sources have caused interrupts. The SCCE bits that are going to be "handled" in this interrupt handler would normally be cleared at this time.
2. Process the Tx BDs to reuse them if the TX bit or TXE bit was set in SCCE. If the transmit speed is fast or the interrupt delay is long, more than one transmit buffer may have been sent by the SCC. Thus, it is important to check more than just one Tx BD during the interrupt handler. One common practice is to process all Tx BDs in the interrupt handler until one is found with its R-bit set.
3. Extract data from the Rx BD if the RX, RXB, or RXF bit was set in SCCE. If the receive speed is fast or the interrupt delay is long, more than one receive buffer may have been received by the SCC. Thus, it is important to check more than just one Rx BD during the interrupt handler. One common practice is to process all Rx BDs in the interrupt handler until one is found with its E-bit set.
4. Clear the SCCx bit in the CISR.
5. Execute the RTE instruction.

7.10.11 SCC Timing Control

When the DIAG bits of the GSMR are programmed to normal operation, the $\overline{CD}$ and $\overline{CTS}$ lines are controlled automatically by the SCC. The following paragraphs describe the behavior in this mode. In the following description, the TCI bit in the GSMR is assumed to be cleared, implying normal transmit clock operation.

7.10.11.1 SYNCHRONOUS PROTOCOLS. The $\overline{RTS}$ pin is asserted when the SCC data is loaded into the transmit FIFO and a falling transmit clock occurs. At this point, the SCC begins transmitting the data, once the appropriate conditions occur on the $\overline{CTS}$ pin. In all cases, the first bit of data is the first bit of the opening flag, sync pattern, or the preamble (if a preamble was programmed to be sent prior to the frame).

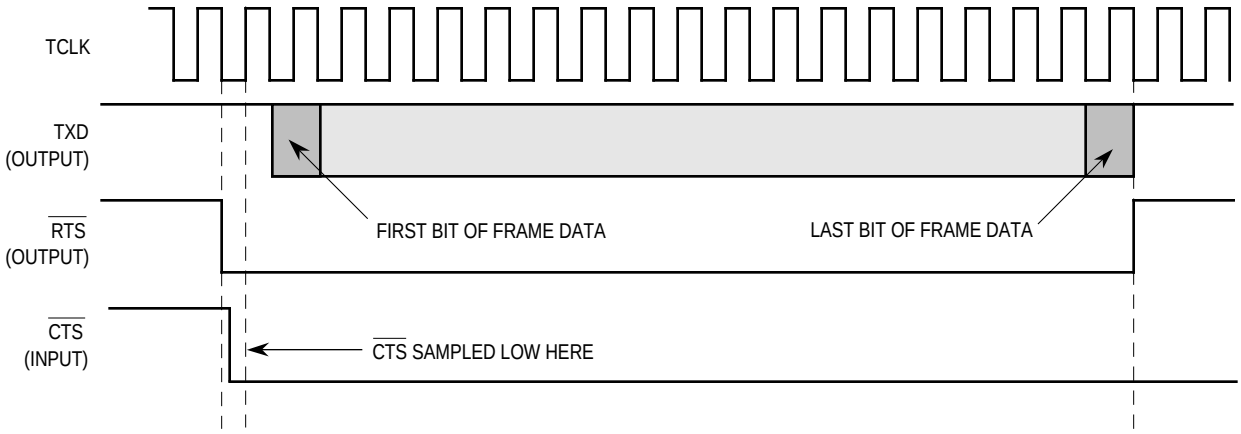
Figure 7-39 shows that the delay between $\overline{RTS}$ and data is 0 bit times, regardless of the CTSS bit in the GSMR. This operation assumes that the $\overline{CTS}$ pin is already asserted to the SCC or that the $\overline{CTS}$ pin is reprogrammed to be a parallel I/O line, in which case the $\overline{CTS}$ signal to the SCC is always asserted. $\overline{RTS}$ is negated one clock after the last bit in the frame.



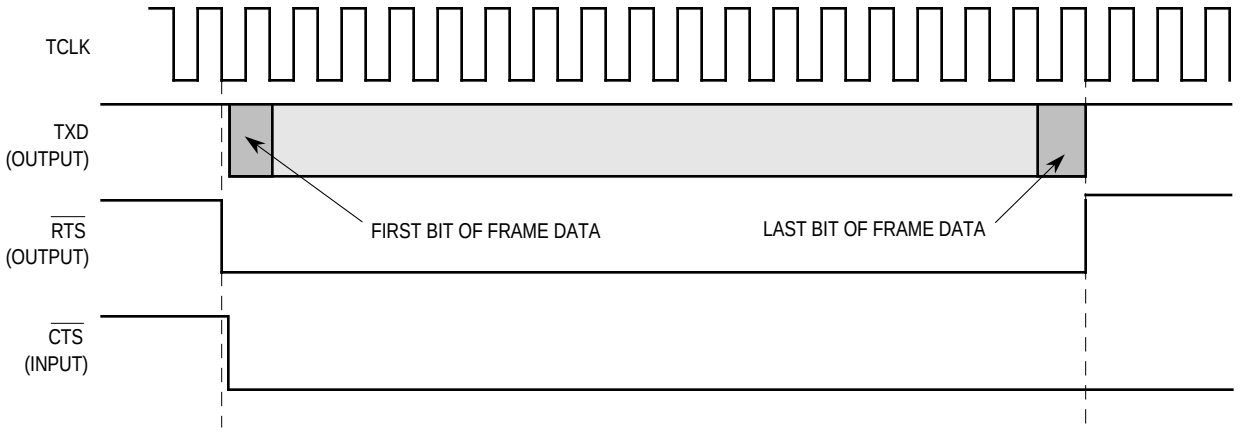
NOTES:
1. A frame includes opening and closing flags and syncs, if present in the protocol.

Figure 7-39. Output Delays from $\overline{\text{RTS}}$ Asserted for Synchronous Protocols

If the $\overline{\text{CTS}}$ pin is not already asserted when the $\overline{\text{RTS}}$ pin is asserted, then the delays to the first bit of data depend on when $\overline{\text{CTS}}$ is asserted. Figure 7-40 shows the delay between $\overline{\text{CTS}}$ and the data can be approximately 0.5 to 1 bit time or 0 bit times, depending on the CTSS bit in the GSMR.



NOTE: CTSS = 0 in GSMR. CTSP is a don't care.



NOTE: CTSS = 1 in GSMR. CTSP is a don't care.

Figure 7-40. Output Delays from $\overline{\text{CTS}}$ Asserted for Synchronous Protocols

If the $\overline{\text{CTS}}$ pin is programmed to envelope the data, the $\overline{\text{CTS}}$ pin must remain asserted during frame transmission, or a CTS lost error occurs (see Figure 7-41). The negation of the $\overline{\text{CTS}}$ pin forces the $\overline{\text{RTS}}$ pin high, forcing the transmit data to the idle state. If the CTSS bit in the GSMR is zero, the $\overline{\text{CTS}}$ pin must be sampled by the SCC before a CTS lost is recognized. Otherwise, the negation of $\overline{\text{CTS}}$ immediately causes the CTS lost condition.

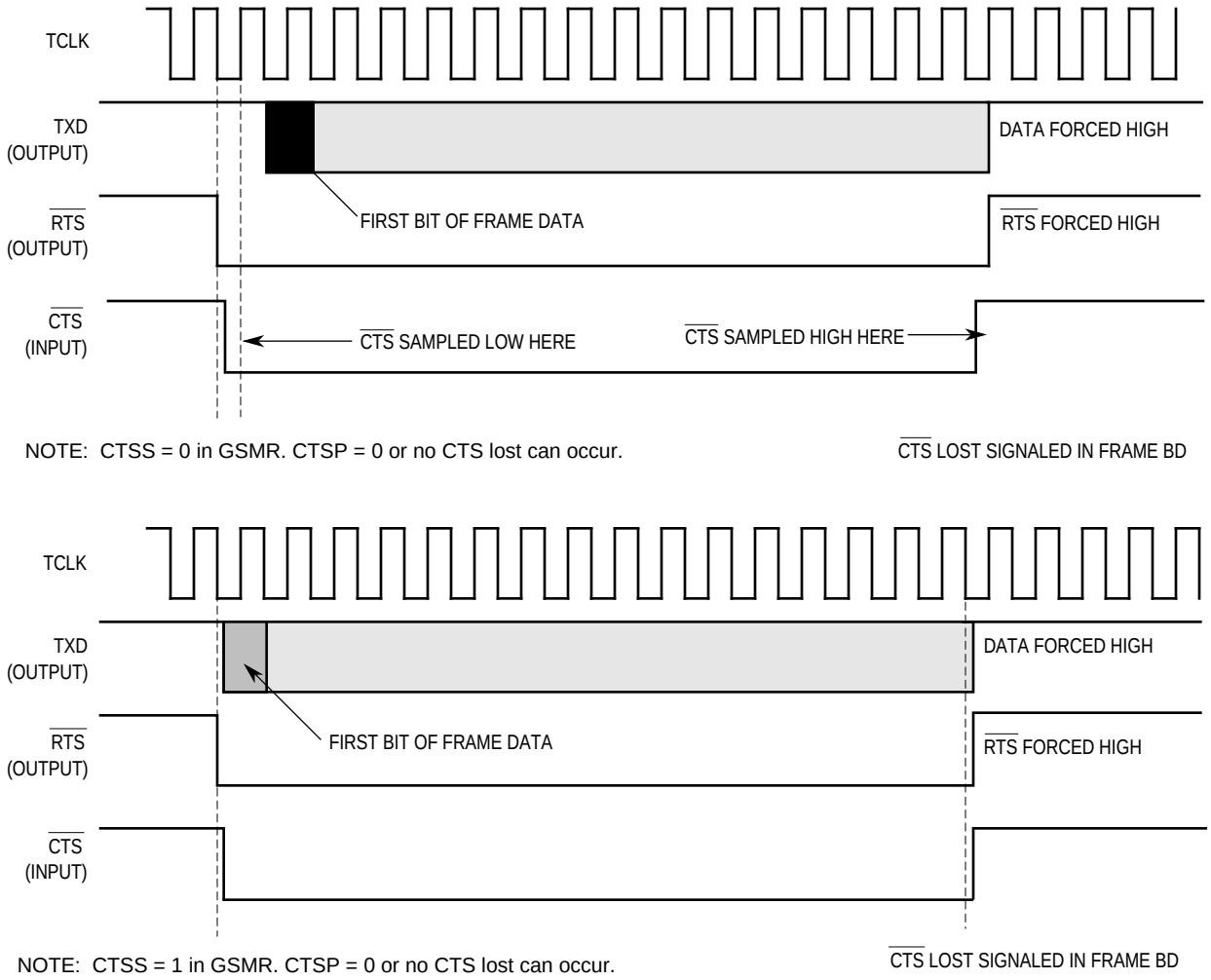
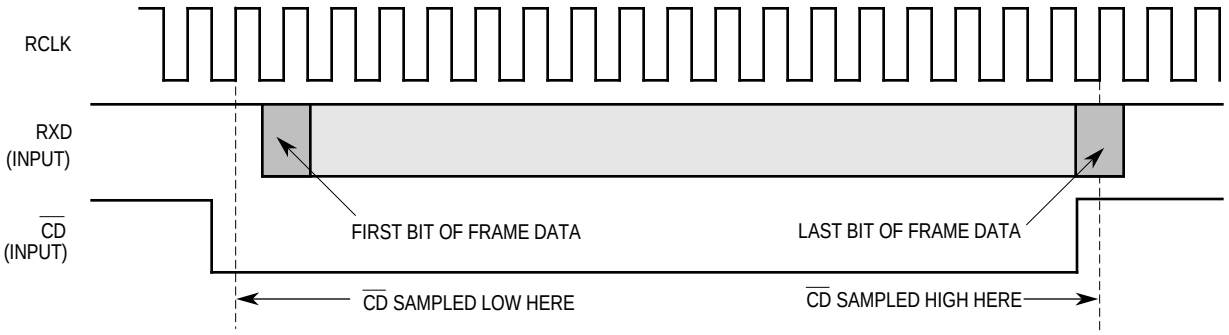


Figure 7-41. $\overline{\text{CTS}}$ Lost in Synchronous Protocols

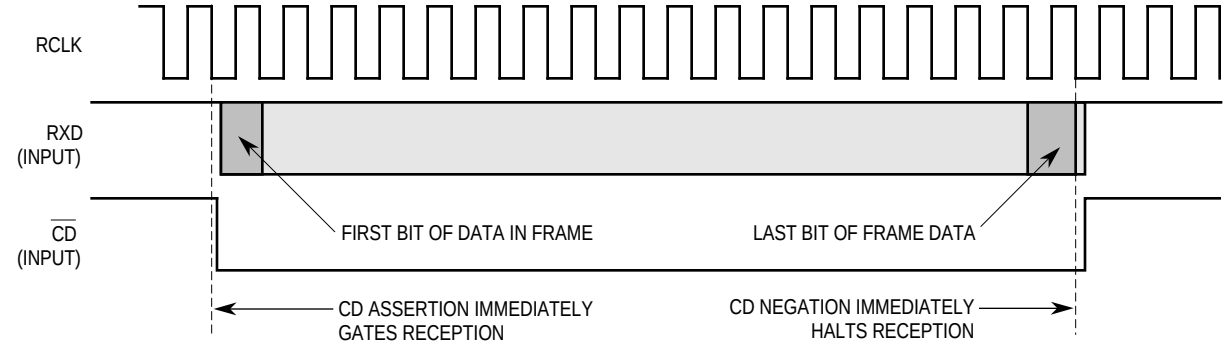
NOTE

If the CTSS bit in GSMR is set, all $\overline{\text{CTS}}$ transitions must occur while the transmit clock is low.

Reception delays are determined by the $\overline{\text{CD}}$ pin as shown in Figure 7-42. If the CDS bit in GSMR is zero, then the $\overline{\text{CD}}$ pin is sampled on the rising receive clock edge prior to data being received. If the CDS bit in GSMR is one, then the $\overline{\text{CD}}$ pin transitions cause data to be immediately gated into the receiver.



NOTES:
1. CDS = 0 in GSMR; CDP = 0.
2. If CD is negated prior to the last bit of the receive frame, CD LOST is signaled in the frame BD.
3. If CDP = 1, CD LOST cannot occur, and CD negation has no effect on reception.



NOTES:
1. CDS = 1 in GSMR; CDP = 0.
2. If CD is negated prior to the last bit of the receive frame, 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 7-42. Using CD to Control Reception of Synchronous Protocols

If the CD pin is programmed to envelope the data, the CD pin must remain asserted during frame transmission, or a CD lost error occurs. The negation of the CD pin terminates reception. If the CDS bit in the GSMR is zero, the CD pin must be sampled by the SCC before a CD lost is recognized. Otherwise, the negation of CD immediately causes the CD lost condition.

NOTE

If the CDS bit in GSMR is set, all CD transitions must occur while the receive clock is low.

7.10.11.2 ASYNCHRONOUS PROTOCOLS. The RTS pin is asserted when the SCC data is loaded into the transmit FIFO and a falling transmit clock occurs. The CD and CTS pins may be used to control reception and transmission in the same manner as the synchronous protocols. The first bit of data transmission in an asynchronous protocol is the start bit of the first character. In addition, the UART protocol has an option for CTS flow control as described in 7.10.16 UART Controller.

If $\overline{CTS}$ is already asserted when $\overline{RTS}$ is asserted, transmission begins in two additional bit times. If $\overline{CTS}$ is not already asserted when $\overline{RTS}$ is asserted and CTSS = 0, then transmission begins in three additional bit times. If $\overline{CTS}$ is not already asserted when $\overline{RTS}$ is asserted and CTSS = 1, then transmission begins in two additional bit times.

7.10.12 Digital Phase-Locked Loop (DPLL)

DPLL data encoding and DPLL operations are discussed in the following paragraphs.

7.10.12.1 DATA ENCODING. Each SCC contains a DPLL unit that may be programmed to encode and decode the SCC data as NRZ, NRZI Mark, NRZI Space, FM0, FM1, Manchester, and Differential Manchester. Examples of the different encoding methods are shown in Figure 7-43.

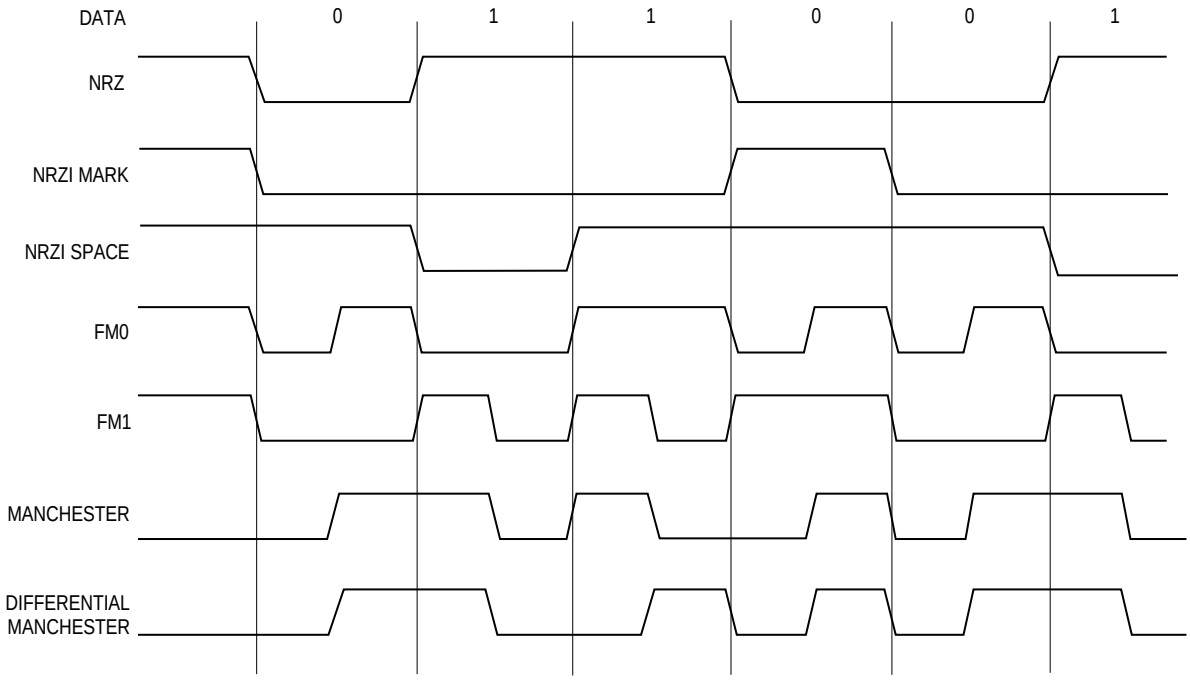


Figure 7-43. DPLL Encoding Examples

If it is not desired to use the DPLL, the NRZ coding may be chosen by the user in the GSMR. The definition of the encodings are as follows:

```
    NRZ             A 1 is represented by a high level for the
duration of the bit.
                    A 0 is represented by a low level for the duration
of the bit.
    NRZI Mark       A 1 is represented by no transition at all.
                    A 0 is represented by a transition at the
beginning of the bit
                    (i.e., the level present in the preceding bit is
reversed).
```

NRZI SpaceA 1 is represented by a transition at the beginning of the bit
(i.e., the level present in the preceding bit is reversed).

A 0 is represented by no transition at all.

FM0 A 1 is represented by a transition at the beginning of the bit only.

A 0 is represented by a transition at the beginning of the bit and another transition at the center of the bit.

FM1 A 1 is represented by a transition at the beginning of the bit and another transition at the center of the bit.

A 0 is represented by a transition at the beginning of the bit only.

Manchester A 1 is represented by a high to low transition at the center of the bit.

A 0 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 1 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 0 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.

7.10.12.2 DPLL OPERATION. Each SCC channel includes a DPLL used to recover clock information from a received data stream. For applications that provide a direct clock source to the SCC, the DPLL may be bypassed as programmed in the GSMR.

The DPLL must not be used when an SCC is programmed to Ethernet. It is optional for other protocols.

The DPLL receive block diagram is shown in Figure 7-44; the transmit block diagram is shown in Figure 7-45.

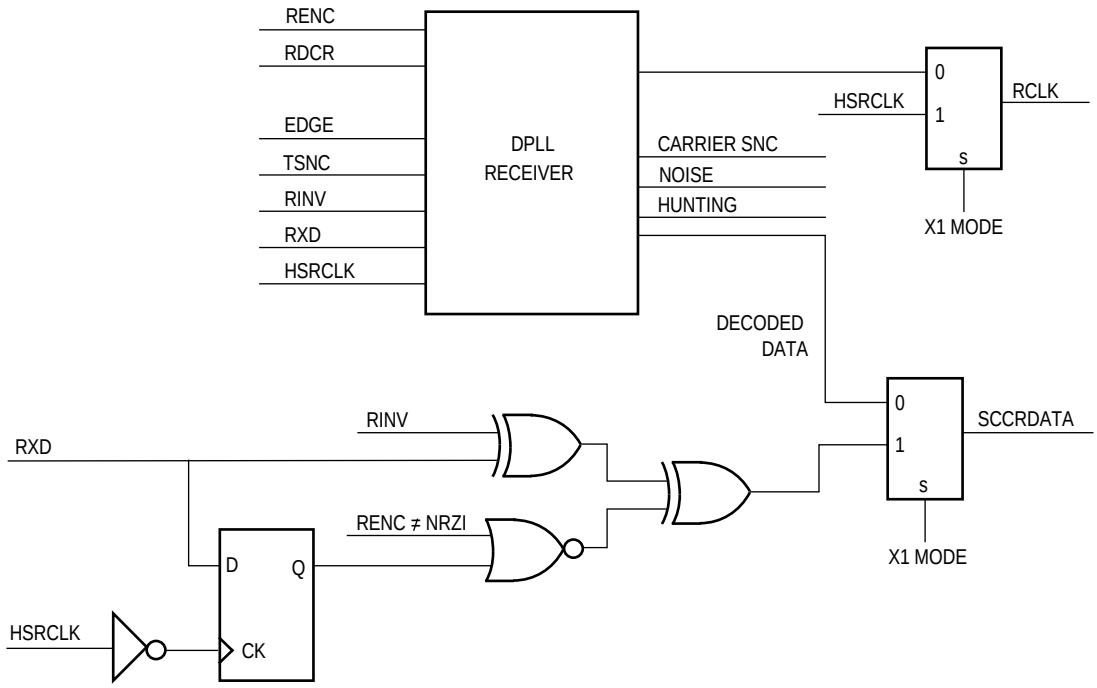


Figure 7-44. DPLL Receive Block Diagram

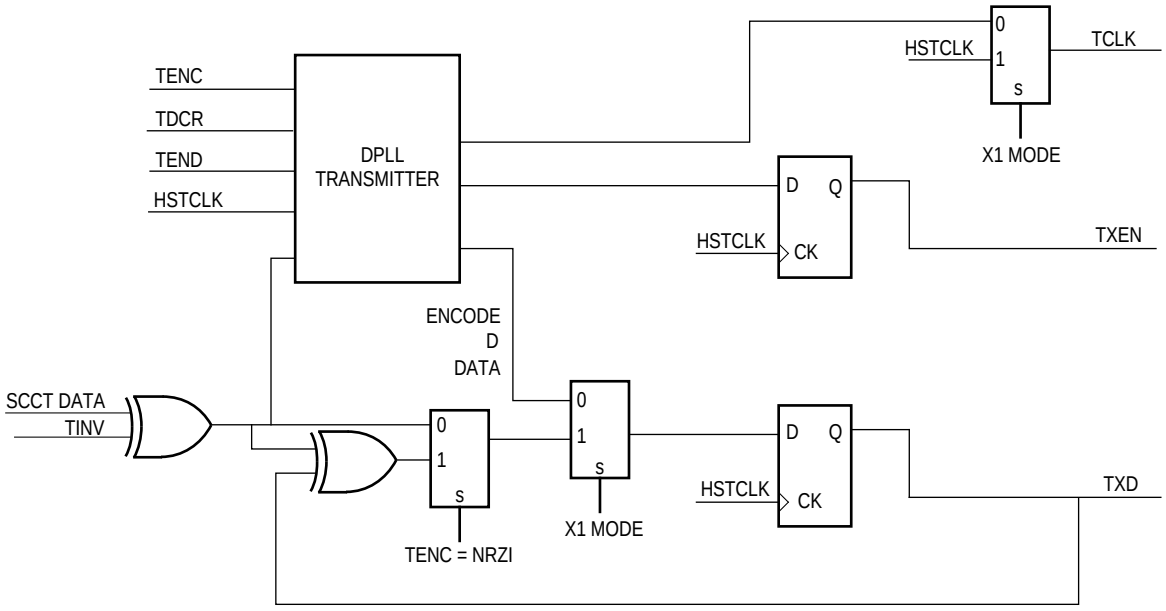


Figure 7-45. DPLL Transmit Block Diagram

The DPLL is driven by either an external clock or one of the baud rate generator outputs. This clock should be approximately 8×, 16×, or 32× times the data rate, depending on the encoding/decoding desired. The DPLL uses this clock, along with the data stream, to con-

struct a data clock that may be used as the SCC receive and/or transmit clock. In all modes, the DPLL uses the input clock to determine the nominal bit time.

At the beginning of operation, the DPLL is in search mode. In this mode, 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. Whenever a transition is detected, the DPLL makes a count adjustment to produce an output clock that tracks the incoming bits.

The DPLL provides a carrier-sense signal. The carrier-sense signal indicates that there are data transfers on the RXD line. It is asserted as soon as a transition is detected on the RXD line, and it is negated after a programmable number of clocks have been detected with no transitions, using the TSNC bits in the GSMR.

To prevent the DPLL from locking on the wrong edges and to provide a fast synchronization, the DPLL should generally receive a preamble pattern prior to the data. In some protocols, the preceding flags or syncs are used. However, some protocols require a special pattern, such as alternating ones and zeros. For the case of transmission, the SCC has an option to generate preamble patterns as programmed in the TPP and TPL bits of the GSMR.

Table 7-6. Preamble Requirement

Decoding Method	Preamble Pattern	Max Preamble Length Required
NRZI Mark	All zeros	8-bits
NRZI Space	All ones	8-bits
FM0	All ones	8-bits
FM1	All zeros	8-bits
Manchester	Repeating 10's	8-bits
Differential Manchester.	All ones	8-bits

NOTES:
The QUICC receiver require the above preambles.

The DPLL can also be used to invert the data stream on receive or transmit. This feature is available in all encodings, including the standard NRZ data format.

The DPLL offers a choice on the transmitter during idle of whether to force the TXD line to a high voltage or to continue encoding the data supplied to it.

The DPLL is used for the UART encoding/decoding. This gives the user the option of selecting the divide ratio used in the UART decoding process (8, 16, or 32). Typically, the 16× option is chosen by users.

The maximum data rate that can be supported with the DPLL is 3.125 MHz when working with a 25-MHz system clock, which assumes the 8× option is chosen: 25 MHz/8 = 3.125 MHz. Thus, the frequency applied to the CLKx pin or generated by an internal baud rate generator may be up to 25 MHz on a 25-MHz QUICC, if the DPLL 8×, 16×, or 32× options are used.

NOTE

The 1:2 ratio of the SyncCLK to the serial clock does not apply when the DPLL is used to recover the clock in the 8×, 16×, or 32× modes. The synchronization actually occurs internally after the receive clock is generated by the DPLL; therefore, even the fastest DPLL clock generation (the 8× option) easily meets the required 1:2 ratio clocking limit.

7.10.13 Clock Glitch Detection

A clock glitch occurs when an input clock signal transitions between a one and zero state twice, within a small enough time period to violate the minimum high/low time specification of the input clock. Spikes are one type of glitch. Additionally, glitches can occur when excessive noise is present on a slowly rising/falling signal.

Glitched clocks are a worry to many communications systems. Not only can they cause systems to experience errors, they can potentially cause errors to occur without even being detected by the system. Systems that supply an external clock to a serial channel are often susceptible to glitches from situations such as noise, connecting/disconnecting the physical cable from the application board, or excessive ringing on the clock lines.

The SCCs on the QUICC have a special circuit designed to detect glitches that may occur in the system. The glitch circuit is designed to detect glitches that could cause the SCC to transition to the wrong state. This status information can be used to alert the system of a problem at the physical layer.

The glitch detect circuit is not a specification test. Thus, if the user develops a circuit that does not meet the input clocking specifications for the SCCs, erroneous data may be received/transmitted that is not indicated by the glitch detection logic. Conversely, if a glitch indication is signaled, it does not guarantee that erroneous data was received/transmitted.

Regardless of whether the DPLL is used, the received clock is passed through a noise filter that eliminates any noise spikes that affect a single sample. This sampling is enabled with the GDE bit of the GSMR.

If a spike is detected, a maskable receive or transmit glitched clock interrupt is generated in the event register of the SCC channel. Although the user may choose to reset the SCC receiver or transmitter or to continue operation, he should keep statistics on clock glitches for later evaluation. In addition, the glitched status indication may be used as a debugging aid during the early phases of prototype testing.

7.10.14 Disabling the SCCs on the Fly

If an SCC is not needed for a period of time, it may be disabled and reenabled later. In this case, a sequence of operations is followed.

These sequences ensure that any buffers in use will be properly closed and that new data will be transferred to/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. For instance, the internal baud rate generators allow on-the-fly changes, but the DPLL-related bits in the GSMR do not.

NOTE

The modification of parameter RAM does not require a full disabling of the SCC. See the parameter RAM description for details on when parameter RAM values may be modified.

If the user desires to disable all SCCs, SMCs, and the SPI, then the CR may be used to reset the entire CP with a single command.

7.10.14.1 SCC TRANSMITTER FULL SEQUENCE. •For the SCC transmitter, the full disable and enable sequence is as follows:

1. STOP TRANSMIT command. This command is recommended if the SCC is currently in the process of transmitting data since it stops transmission in an orderly way. If the SCC is not transmitting (e.g., no Tx BDs are ready or the GRACEFUL STOP TRANSMIT command has been issued and has completed), then the STOP TRANSMIT command is not required. Furthermore, if the TBPTR will be overwritten by the user or the INIT TX PARAMETERS command will be executed, this command is not required.
2. Clear the ENT bit in the GSMR, which disables the SCC transmitter and puts it in a reset state.
3. Make modifications. The user may make modifications to the SCC transmit parameters including the parameter RAM. If the user desires to switch protocols or restore the SCC transmit parameters to their initial state, the INIT TX PARAMETERS command may now be issued.
4. RESTART TRANSMIT command. This command is required if the INIT TX PARAMETERS command was not issued in step 3.
5. Set the ENT bit in the GSMR. Transmission will now begin using the Tx BD pointed to by the TBPTR as soon as the Tx BD R-bit is set.

7.10.14.2 SCC TRANSMITTER SHORTCUT SEQUENCE. •A shorter sequence is possible if the user desires to reinitialize the transmit parameters to the state they had after reset. This sequence is as follows:

1. Clear the ENT bit in the GSMR.
2. INIT TX PARAMETERS command. Any additional modifications may now be made.
3. Set the ENT bit in the GSMR.

7.10.14.3 SCC RECEIVER FULL SEQUENCE. •The full disable and enable sequence for the receiver is as follows:

1. Clear the ENR bit in the GSMR. Reception will be aborted immediately. This disables the receiver of the SCC and puts it in a reset state.
2. Make modifications. The user may make modifications to the SCC receive parameters

including the parameter RAM. If the user desires to switch protocols or restore the SCC receive parameters to their initial state, the INIT RX PARAMETERS command may now be issued.

3. ENTER HUNT MODE command. This command is required if the INIT RX PARAMETERS command was not issued in step 2.
4. Set the ENR bit in the GSMR. Reception will now begin immediately using the Rx BD pointed to by the RBPTR if the Rx BD E-bit is set.

7.10.14.4 SCC RECEIVER SHORTCUT SEQUENCE. •A shorter sequence is possible if the user desires to reinitialize the receive parameters to the state they had after reset. This sequence is as follows:

1. Clear the ENR bit in the GSMR.
2. INIT RX PARAMETERS command. Any additional modifications may now be made.
3. Set the ENR bit in the GSMR.

7.10.14.5 SWITCHING PROTOCOLS. •Sometimes the user desires to switch the protocol that the SCC is executing (e.g., UART to HDLC) without resetting the board or affecting any other SCC. This can be accomplished using only one command and a short number of steps:

1. Clear the ENT and ENR bits in the GSMR.
2. INIT TX AND RX PARAMETERS command. This one command initializes both transmit and receive parameters. Any additional modifications may now be made in the GSMR to change the protocol, etc.
3. Set the ENT and ENR bits in the GSMR. The SCC is enabled with the new protocol.

7.10.15 Saving Power

When the ENT and ENR bits of an SCC are cleared, that SCC consumes a minimal amount of power.

7.10.16 UART Controller

Many applications need a simple method of communicating low-speed data between equipment. The universal asynchronous receiver transmitter (UART) protocol is the de-facto standard for such communications. The term asynchronous is used because it is not necessary to send clocking information with the data that is sent. UART links are typically 38400 baud or less in speed and are character oriented (i.e., the smallest unit of data that can be correctly received or transmitted is a character). Typical applications of asynchronous links are connections between terminals and computer equipment. Even in applications where synchronous communications are required, the UART is often used for a local debugging port to run board debugger software. The character format of the UART protocol is shown in Figure 7-46.

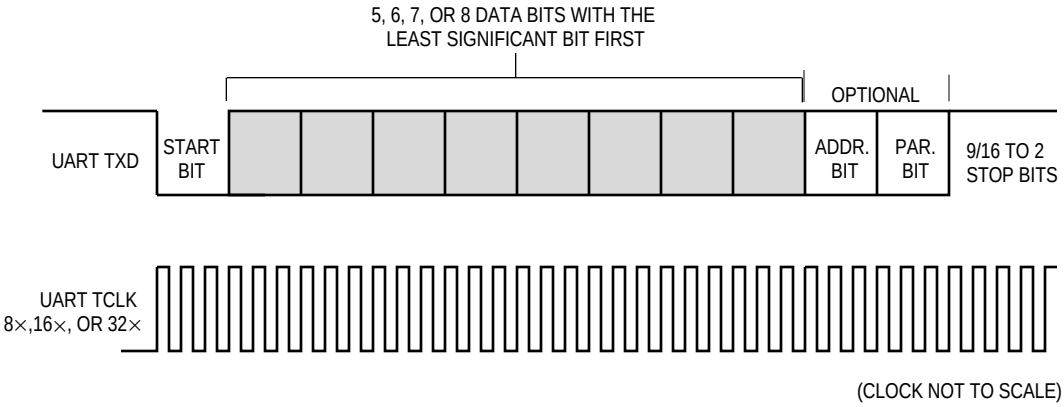


Figure 7-46. UART Character Format

Since the transmitter and receiver work asynchronously, there is no need to connect 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 three samples of the sixteen samples are used. Two UARTs can communicate using a system like this if parameters, such as the parity scheme and character length, are the same for both transmitter and receiver.

When data is not transmitted in the UART protocol, a continuous stream of ones is transmitted, called the idle condition. Since the start bit is always a zero, the receiver can detect when real data is once again present on the line. UART also specifies an all-zeros character called a break, which is used to abort a character transfer sequence.

Many different protocols have been defined using asynchronous characters, but the most popular of these is the RS-232 standard. RS-232 specifies standard baud rates, handshaking protocols, and mechanical/electrical details. Another popular standard using the same character format is RS-485, which defines a balanced line system allowing longer cables than RS-232 links. Synchronous protocols like HDLC are sometimes defined to run over asynchronous links. Other protocols like Profibus extend the UART protocol to include LAN-oriented features such as token passing.

All the standards provide handshaking signals, but some systems require just three physical lines: transmit data, receive data, and ground.

Many proprietary standards have been built around the asynchronous character frame, and some even implement a multidrop configuration. In multidrop systems, more than two stations may be present on a network, with each having a specific address. Frames made up of many characters may be broadcast, with the first character acting as a destination address. To allow this, the UART frame is extended by one bit to distinguish between an address character and the normal data characters.

Additionally, a synchronous form of the UART protocol exists where start and stop bits are still present, but a clock is provided with each bit, so the oversampling technique is not required. This mode is called "isochronous" operation or, more often, synchronous UART.

By appropriately setting the GSMR, any of the SCC channels may be configured to function as a UART.

The UART provides standard serial I/O using asynchronous character-oriented (start-stop) protocols with RS-232C-type lines. The UART may be used to communicate with any existing RS-232-type device.

The UART provides a port for serial communication to other microprocessors, terminals, etc., either locally or via modems. It includes facilities for communication using standard asynchronous bit rates and protocols. The UART supports a multidrop mode for master/slave operation with wake-up capability on both idle line and address bit. The UART also supports a synchronous mode of operation where a clock must be provided with each bit received.

The UART transmits data from memory (either internal or external) to the TXD line and receives data from the RXD line into memory. In a synchronous UART mode, the clock must also be supplied. It may be generated internally or externally. Modem lines are supported via the port C pins.

The UART consists of separate transmit and receive sections whose operations are asynchronous with the CPU32+ core.

7.10.16.1 UART KEY FEATURES. •The UART contains the following key features:

- Flexible Message-Oriented Data Structure
- Implements Synchronous and Asynchronous UART
- Multidrop Operation
- Receiver Wake-Up on Idle Line or Address Mode
- Eight Control Character Comparison
- Two Address Comparison
- Maintenance of Four 16-Bit Error Counters
- Received Break Character Length Indication
- Programmable Data Length (5–8 Bits)
- Programmable 1 to 2 Stop Bits in Transmission
- Capable of Reception without a Stop Bit
- Programmable Fractional Stop Bit Length
- Even/Odd/Force/No Parity Generation
- Even/Odd/Force/No Parity Check
- Frame Error, Noise Error, Break, and idle Detection
- Transmit Preamble and Break Sequences
- Freeze Transmission Option with Low-Latency Stop

7.10.16.2 NORMAL ASYNCHRONOUS MODE. •In a normal asynchronous mode, the receive shift register receives the incoming data on the RXDx pin. The length and format

of the UART character is defined by the control bits in the UART mode register. The order of reception is:

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 times faster then 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 the samples do not all agree, a noise indication counter is incremented. When a complete byte has been clocked in, the contents of the shift register are transferred to the UART receive data register. If there is an error in this character, then the appropriate error bits will be set by the CP.

The UART may receive fractional stop bits. The next character's start bit may begin anytime after the three middle samples have been taken.

The UART transmit shift register transmits the outgoing data on the TXDx pin. Data is clocked synchronously with the transmit clock, which may have either an internal or external source. The order of bit transmission is LSB first.

Only the data portion of the UART frame is actually stored in the data buffers. The start and stop bits are always generated and stripped by the UART controller. The parity bit may also be generated in transmission and checked during reception. Although parity is not stored in the data buffer, its value may be inferred from the reporting mechanism in the data buffer (i.e., character with parity errors are identified). Similarly, the optional address bit is not stored in the transmit or receive data buffer, but is implied from the buffer descriptor itself. Parity is generated and checked for the address bit, when present.

The RFW bit in the GSMR must be set for an 8-bit receive FIFO for the UART receiver.

7.10.16.3 SYNCHRONOUS MODE. In synchronous mode, the UART controller uses the 1× data clock for timing. The receive shift register receives the incoming data on the RXD pin synchronously to the clock. The length and format of the serial word in bits are defined by the control bits in the UART mode register in the same manner as for asynchronous mode. When a complete byte has been clocked in, the contents of the shift register are transferred to the UART receive data register. If there is an error in this character, then the appropriate error bits will be set by the CP.

The UART transmit shift register transmits the outgoing data on the TXD pin. Data is clocked synchronously with the transmit clock, which may have either an internal or external source.

The RFW bit in the GSMR must be set for an 8-bit receive FIFO for the UART receiver.

7.10.16.4 UART MEMORY MAP. When configured to operate in UART mode, the QUICC overlays the structure listed in Table 7-5 with the UART-specific parameters described in Table 7-7.

Table 7-7. UART-Specific Parameters

Address	Name	Width	Description
SCC Base + 30	**RES**	Long	Reserved
SCC Base + 34	**RES**	Long	Reserved
SCC Base + 38	**MAX_IDL**	Word	Maximum idle Characters
SCC Base + 3A	IDLC	Word	Temporary idle Counter
SCC Base + 3C	**BRKCR**	Word	Break Count Register (Transmit)
SCC Base + 3E	**PAREC**	Word	Receive Parity Error Counter
SCC Base + 40	**FRMEC**	Word	Receive Framing Error Counter
SCC Base + 42	**NOSEC**	Word	Receive Noise Counter
SCC Base + 44	**BRKEC**	Word	Receive Break Condition Counter
SCC Base + 46	BRKLN	Word	Last Received Break Length
SCC Base + 48	**UADDR1**	Word	UART Address Character 1
SCC Base + 4A	**UADDR2**	Word	UART Address Character 2
SCC Base + 4C	RTEMP	Word	Temp Storage
SCC Base + 4E	**TOSEQ**	Word	Transmit Out-of-Sequence Character
SCC Base + 50	**CHARACTER1**	Word	CONTROL Character 1
SCC Base + 52	**CHARACTER2**	Word	CONTROL Character 2
SCC Base + 54	**CHARACTER3**	Word	CONTROL Character 3
SCC Base + 56	**CHARACTER4**	Word	CONTROL Character 4
SCC Base + 58	**CHARACTER5**	Word	CONTROL Character 5
SCC Base + 5A	**CHARACTER6**	Word	CONTROL Character 6
SCC Base + 5C	**CHARACTER7**	Word	CONTROL Character 7
SCC Base + 5E	**CHARACTER8**	Word	CONTROL Character 8
SCC Base + 60	**RCCM**	Word	Receive Control Character Mask
SCC Base + 62	RCCR	Word	Receive Control Character Register
SCC Base + 64	RLBC	Word	Receive Last Break Character

NOTE: The boldface items should be initialized by the user.

MAX_IDL. Once a character is received on the line, the UART controller begins counting any idle characters received. If a MAX_IDL number of idle characters is received before the next data character is received, an idle timeout occurs, and the buffer is closed. This in turn can produce an interrupt request to the CPU32+ core to receive the data from the buffer. Thus, MAX_IDL provides a convenient way to demarcate frames in the UART mode. To disable the MAX_IDL feature, simply program it to $0000. A character of idle is calculated as the following number of bit times: 1 + data length (5, 6, 7, 8, or 9) + 1 (if parity bit is used) + number of stop bits (1 or 2). Example: for 8 data bits, no parity, and 1 stop bit, the character length is 10 bits.

IDLC. This value is used by the RISC to store the current idle counter value in the MAX_IDL timeout process. IDLC is a down-counter; it does not need to be initialized or accessed by the user.

BRKCR. The UART controller will send an a break character sequence whenever a STOP TRANSMIT command is given. The number of break characters sent by the UART controller is determined by the value in BRKCR. In the case of 8 data bits, no parity, 1 stop bit, and 1 start bit, each break character is 10 bits in length and consists of all zeros.

PAREC, FRMEC, NOSEC, and BRKEC. These 16-bit (modulo–2^{16}) counters are initialized by the user. When the associated condition occurs, they will be incremented by the RISC controller. PAREC counts received parity errors. FRMEC counts received characters with framing errors. NOSEC counts received characters with noise errors (one of the three samples was different). BRKEC counts the number of break conditions that occurred on the line. Note that one break condition may last for hundreds of bit times, yet this counter is incremented only once during that period.

BRKLN. This value is used to store the length of the last break character received. This value is the length in characters of the break. Example: If the receive pin is low for 20 bit times, BRKLN will show the value $0010. BRKLN is accurate to within one character unit of bits. For example, for 8 data bits, no parity, 1 stop bit, and 1 start bit, BRKLN is accurate to within 10 bits.

UADDR1, UADDR2. In the multidrop mode, the UART controller can provide automatic address recognition of two addresses. In this case, the lower order bytes of UADDR1 and UADDR2 are programmed by the user with the two desired addresses.

TOSEQ. This value is used to transmit out-of-sequence characters in the transmit stream such as the XOFF and XON characters. Using this field, the desired characters can be inserted into the transmit FIFO without affecting any transmit buffer that might currently be in progress.

CHARACTER1–8. These characters define the receive control characters on which interrupts may be generated.

RCCM. This value is used to mask the comparison of CHARACTER1–8 so that classes of control characters may be defined. A one enables the bit comparison, a zero masks it.

RCCR. This value is used to hold the value of any control character that is NOT to be written to the data buffer.

RLBC. This entry is used in synchronous UART, when the RZS bit is set in the PSMR. This entry contains the actual pattern of the last break character. By counting the zeros in this entry, the CPU32+ core can measure the break length to a bit resolution. The user reads RLBC by counting the number of zeros written, starting at bit 15 down to the point where the first one is written. Therefore, RLBC = 001xxxxxxxxxxxx (binary) indicates two zeros, and RLBC = 1xxxxxxxxxxxxxx (binary) indicates no zeros.

7.10.16.5 UART PROGRAMMING MODEL. An SCC configured as a UART uses the same data structure as in the other modes. The UART data structure supports multibuffer operation. The UART may be programmed to perform address comparison whereby messages not destined for a given programmable address are discarded. Also, the user can program the UART to accept or reject control characters. If a control character is rejected, an interrupt may be generated. The receive character may be accepted using a receive character mask value. The UART enables the user to transmit break and preamble sequences. Overrun, parity, noise, and framing errors are reported via the BD table and/or error counters. An indication of the status of the line (idle) is reported through the status register, and a maskable interrupt is generated upon a status change. In its simplest form, the UART can function in a character-oriented environment. Each character is transmitted with accompanied stop bits and parity (as configured by the user) and received into separate 1-byte buffers. Reception of each buffer may generate a maskable interrupt.

Many applications may want to take advantage of the message-oriented capabilities supported by the UART by using linked buffers (in either receive or transmit). In this case, data is handled in a message-oriented environment; users can work on entire messages rather than operating on a character-by-character basis. A message may span several linked buffers. For example, before handling the input data, a terminal driver may wish to wait until an end-of-line character has been typed by the user rather than being interrupted upon the reception of each character.

As another example, when transmitting ASCII files, the data may be transferred as messages ending on the end-of-line character. Each message could be both transmitted and received as a linked list of buffers without any intervention from the CPU32+, which achieves ease in programming and significant savings in processor overhead.

On the receive side, the user may define up to eight control characters. Each control character may be configured to designate the end of a message or generate a maskable interrupt without being stored in the data buffer. The latter option is useful when flow control characters such as XON or XOFF need to alert the CPU32+, yet do not belong to the message being received.

7.10.16.6 UART COMMAND SET. The following transmit and receive commands are issued to the CR.

7.10.16.6.1 Transmit Commands. The following paragraphs describe the UART transmit commands.

STOP TRANSMIT Command*.* After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the transmit enable mode and starts polling the first BD in the table every 8 transmit clocks (immediately if the TOD bit in the TODR is set).

The STOP TRANSMIT command disables the transmission of characters on the transmit channel. If this command is received by the UART controller during message transmission, transmission of that message is aborted. The UART completes transmission of all data

already transferred to its FIFO and then stops transmitting data. The TBPTR is not advanced.

The UART transmitter will transmit a programmable number of break sequences and then start to transmit idles. The number of break sequences (which may be zero) should be written to the break count register before this command is given to the UART controller.

GRACEFUL STOP TRANSMIT Command*.* The GRACEFUL STOP TRANSMIT command is used to stop transmission in an orderly way rather than abruptly, as performed by the regular STOP TRANSMIT command. It stops transmission after the current buffer has completed transmission, or immediately if there is no buffer being transmitted. The GRA bit in the SCCE will be set once transmission has stopped. After transmission ceases, the UART transmit parameters, including BDs, may be modified. The TBPTR will point to the next Tx BD in the table. Transmission will begin once the R-bit of the next BD is set and the RESTART TRANSMIT command is issued.

RESTART TRANSMIT Command*.* The RESTART TRANSMIT command enables the transmission of characters on the transmit channel. This command is expected by the UART controller after disabling the channel in its SCC mode register, after a STOP TRANSMIT command, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error (underrun or CTS lost). The UART controller will resume transmission from the current TBPTR in the channel's Tx BD table.

INIT TX PARAMETERS Command. This command initializes all transmit parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the transmitter is disabled. Note that the INIT TX; AND RX PARAMETERS command may also be used to reset both transmit and receive parameters.

7.10.16.6.2 Receive Commands. The following paragraphs describe the UART receive commands.

ENTER HUNT MODE ENTER HUNT MODE;Command*.* After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in receive enable mode and will use the first BD in the table.

The ENTER HUNT MODE command is used to force the UART controller to close the current Rx BD if it is being used and enter the hunt mode. The UART controller will resume reception to the next BD if a message was in progress.

In the multidrop hunt mode, the UART controller continually scans the input data stream for the address character. When not in multidrop mode, the UART controller will wait for the idle sequence (one character of idle) without losing any data that was in the receive FIFO when this command was executed.

CLOSE Rx BD Command*.* The CLOSE Rx BD command is used to force the SCC to close the Rx BD, if it is currently being used, and to use the next BD for any subsequent data that is received. If the SCC is not in the process of receiving data, no action is taken by this command.

NOTE

The CLOSE Rx BD command in UART mode does the same job as the ENTER HUNT MODE command except for one distinction. The CLOSE Rx BD does not require that a character of idle be present on the line for reception to continue.

INIT RX PARAMETERS Command. This command initializes all receive parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the receiver is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both receive and transmit parameters.

7.10.16.7 UART ADDRESS RECOGNITION (RECEIVER). In multidrop systems, more than two stations may be present on a network, each having a specific address. Figure 7-47 shows two examples of such a configuration. Frames made up of many characters may be broadcast, with the first character acting as a destination address. To achieve this, the UART frame is extended by one bit, called the address bit, to distinguish between an address character, and the normal data characters.

The UART can be configured to operate in a multidrop environment in which the following two modes are supported:

Automatic Multidrop Mode—The UART controller automatically checks the incoming address character and accepts the data following it only if the address matches one of two preset values.

Nonautomatic Multidrop Mode—The UART controller receives all characters. An address character is always written to a new buffer (it may be followed by data characters).

Each UART controller has two 16-bit address registers to support address recognition, UADDR1 and UADDR2. The upper 8 bits of these registers should be written with zero; only the lower 8 bits are used. In the automatic mode, the incoming address is checked against UADDR1 and UADDR2. Upon an address match, the M-bit in the BD is set to indicate which address character was matched, and the data following it is written to the data buffers.

NOTE

For less than 8-bit characters, the MSBs should be zero.

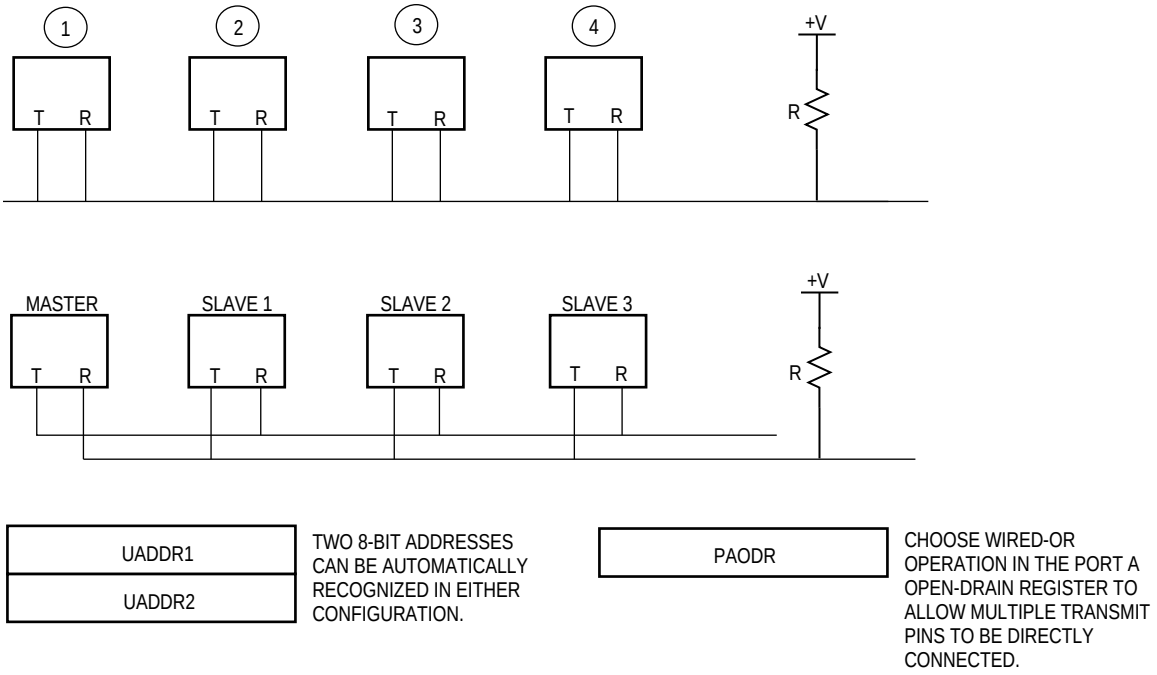


Figure 7-47. Two Configurations of UART Multidrop Operation

7.10.16.8 UART CONTROL CHARACTERS (RECEIVER). The UART has the capability to recognize special control characters. These characters may be used when the UART functions in a message-oriented environment. Up to 8 control characters may be defined by the user in the control characters table. Each character may be either written to the receive buffer (upon which the buffer is closed and a new receive buffer taken) or rejected. If rejected, the character is written to the received control character register (RCCR) in internal RAM, and a maskable interrupt is generated. This method is useful for notifying the user of the arrival of control characters (e.g., XOFF) that are not part of the received messages.

The UART uses a table of 16-bit entries to support control character recognition. Each entry consists of the control character, a valid bit, and a reject character bit.

	15	14	13	12	11	10	9	8	7 … 0
OFFSET + 0	E	R							CHARACTER1
OFFSET + 2	E	R							CHARACTER2
OFFSET + 4	E	R							CHARACTER3
	•								
	•								
	•								
OFFSET + E	E	R							CHARACTER8
OFFSET + 10	1	1							RCCM
OFFSET + 12									RCCR

E—End of Table

0 = This entry is valid. The lower 8 bits will be checked against the incoming character.
1 = The entry is not valid. This must be the last entry in the control characters table.

In tables with 8 control characters, E is always zero.

R—Reject Character

0 = The character is not rejected but is written into the receive buffer. The buffer is then closed, and a new receive buffer is used if there is more data in the message. A maskable (I-bit in the Rx BD) interrupt is generated.
1 = If this character is recognized, it will not be written to the receive buffer. Instead, it is written to the RCCR, and a maskable interrupt is generated. The current buffer is not closed when a control character is received with R set.

CHARACTER1–8—Control Character Values

These fields define control characters that should be compared to the incoming character. For less than 8-bit characters, the MSB should be zero.

RCCM—Received Control Character Mask

The value in this register is used to mask the comparison of CHARACTER1–8. The lower eight bits of RCCM correspond to the lower eight bits of CHARACTER1–8 and are decoded as follows:

0 = Mask this bit in the comparison of the incoming character and CHARACTER1–8.
1 = The address comparison on this bit proceeds normally; no masking occurs.

Bits 15 and 14 of RCCM must be set, or erratic operation may occur during the control character recognition process.

RCCR—Received Control Character Register

Upon a control character match for which the reject bit is set, the UART will write the control character into the RCCR and generate a maskable interrupt. The CPU32+ core must process the interrupt and read the RCCR before a second control character arrives. Failure to do so will result in the UART overwriting the first control character.

7.10.16.9 WAKE-UP TIMER (RECEIVER). By issuing the ENTER HUNT MODE command, the user can temporarily disable the UART receiver. It will remain inactive until an idle or address character is recognized (depending on the setting of the UM bits).

If the UART is still in the process of receiving a message that the user has already decided to discard, the user may abort its reception by issuing the ENTER HUNT MODE command. The UART receiver will be reenabled when the message is finished by detecting the idle line (one character of idle) or by the address bit of the next message, depending on the UM bits.

When the receiver is in sleep mode and a break sequence is received, the receiver will increment the BRKEC counter and generate the BRK interrupt if it is enabled.

7.10.16.10 BREAK SUPPORT (RECEIVER). The UART offers very flexible break support for the receiver.

Transmission of out-of-sequence characters is also supported by the UART and is normally used for the transmission of flow control characters such as XON or XOFF. This procedure is performed using the TOSEQ entry in the UART parameter RAM.

The UART will poll TOSEQ whenever the transmitter is enabled for UART operation. This includes during UART freeze operation, during UART buffer transmission, and when no buffer is ready for transmission. The TOSEQ character is transmitted at a higher priority than the other characters in the transmit buffer (if any), but does not preempt characters already in the transmit FIFO. This means that the XON or XOFF character may not be transmitted for eight character times (SCC1) or four character times (SCC2, SCC3, and SCC4). To reduce this latency, the TFL bit in the GSMR should be set to decrease the FIFO size to one character prior to enabling the SCC transmitter.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	REA	I	CT	0	0	A	CHARSEND							

Bits 15–14—Don't Care. May be written with ones or zeros.

The fact that these bits are don’t cares allows full compatibility between TOSEQ on the QUICC and CHARACTER8 on the MC68302.

REA—Ready

This bit is set by the CPU32+ core when the character is ready for transmission and will remain one while the character is being transmitted. The CP clears this bit after transmission.

I—Interrupt

If set, the CPU32+ core will be interrupted when this character has been transmitted. (The TX bit will be set in the UART event register.)

CT—Clear-to-Send Lost

This status bit indicates that the $\overline{\text{CTS}}$ signal was negated during transmission of this character. If this occurs, the CTS bit in the UART event register will also be set. This bit operates only if the $\overline{\text{CTS}}$ line is monitored by the SCC as determined by the DIAG bits.

NOTE

If the $\overline{\text{CTS}}$ signal was negated during transmission and the CP transmits this character in the middle of buffer transmission, the $\overline{\text{CTS}}$ signal could actually have been negated either during this character’s transmission or during a buffer character’s transmission. In this case, the CP sets the CT bit both here and in the Tx BD status word.

Bits 10–9—Should be written with zeros.

A—Address

When working in a multidrop configuration, the user should include the address bit in this position.

CHARSEND

This value contains the character to be transmitted. Any 5-, 6-, 7-, or 8-bit character value may be transmitted in accordance with the UART's configuration. The character should comprise the LSBs of CHARSEND. This value may be modified only while the REA bit is cleared.

7.10.16.11 SEND BREAK (TRANSMITTER). A break is an all-zeros character without a stop bit(s). A break is sent by issuing the STOP TRANSMIT command. The UART completes transmission of any outstanding data, sends a programmable number of break characters according to the break count register, and then reverts to idle or sends data if the RESTART TRANSMIT command was given before completion. At the completion of the break code, the transmitter sends at least one high bit before transmitting any data to guarantee recognition of a valid start bit.

The break characters do not preempt characters already in the transmit FIFO. This means that the break character may not be transmitted for 8 character times (SCC1) or 4 character times (SCC2, SCC3, and SCC4). To reduce this latency, the TFL bit in the GSMR should be set to decrease the FIFO size to one character prior to enabling the SCC transmitter.

7.10.16.12 SENDING A PREAMBLE (TRANSMITTER). A preamble sequence gives the programmer a convenient way of ensuring that the line goes idle before starting a new message. The preamble sequence length is constructed of consecutive ones of one character length. If the preamble bit in a BD is set, the SCC will send a preamble sequence before transmitting that data buffer. Example: 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.

7.10.16.13 FRACTIONAL STOP BITS (TRANSMITTER). The asynchronous UART transmitter can be programmed to transmit fractional stop bits. Four bits in the SCC data synchronization register (DSR) are used to program the length of the last stop bit transmitted. These DSR bits may be modified at any time. If two stop bits are transmitted, only the second one is affected. Idle characters are always transmitted as full-length characters.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FSB				1	1	0	0	1	1	1	1	1	1	0

In normal UART mode with 16× oversampling, the FSB bits (14–11) in the DSR are decoded as follows:

1111 = Last Transmitted Stop Bit 16/16 (the default value after reset)
1110 = Last Transmitted Stop Bit 15/16
1101 = Last Transmitted Stop Bit 14/16
1100 = Last Transmitted Stop Bit 13/16
1011 = Last Transmitted Stop Bit 12/16
1010 = Last Transmitted Stop Bit 11/16
1001 = Last Transmitted Stop Bit 10/16
1000 = Last Transmitted Stop Bit 9/16
0xxx = Invalid. Do not use.

When the UART is configured for 32× oversampling, the FSB bits (14–11) in the DSR are decoded as follows:

1111 = Last Transmitted Stop Bit 32/32 (the default value after reset)
1110 = Last Transmitted Stop Bit 31/32
1101 = Last Transmitted Stop Bit 30/32
1100 = Last Transmitted Stop Bit 29/32
1011 = Last Transmitted Stop Bit 28/32
1010 = Last Transmitted Stop Bit 27/32
1001 = Last Transmitted Stop Bit 26/32
1000 = Last Transmitted Stop Bit 25/32
0111 = Last Transmitted Stop Bit 24/32
0110 = Last Transmitted Stop Bit 23/32
0101 = Last Transmitted Stop Bit 22/32
0100 = Last Transmitted Stop Bit 21/32
0011 = Last Transmitted Stop Bit 20/32
0010 = Last Transmitted Stop Bit 19/32
0001 = Last Transmitted Stop Bit 18/32
0000 = Last Transmitted Stop Bit 17/32

When the UART is configured for 8× oversampling, the FSB bits (14–11) in the DSR are decoded as follows:

1111 = Last Transmitted Stop Bit 8/8 (the 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.
01xx = Invalid. Do not use.
00xx = Invalid. Do not use.

The UART receiver can always receive fractional stop bits. The next character's start bit may begin at any time after the three middle samples of the stop bit have been taken.

7.10.16.14 UART ERROR-HANDLING PROCEDURE. The UART controller reports character reception and transmission error conditions via the channel BDs, the error counters,

and the UART event register. The modem interface lines can be monitored by the port C pins.

7.10.16.14.1 Transmission Error. The following paragraph describes a UART transmission error.

CTS Lost During Character Transmission*.* When this error occurs, the channel stops transmission after finishing transmission of the current character from the buffer. The channel then sets the CT bit in the Tx BD and generates the TX interrupt if it is not masked. The channel will resume transmission after the RESTART TRANSMIT command is issued and the $\overline{\text{CTS}}$ pin is asserted.

NOTE

The UART also offers an asynchronous flow control option using $\overline{\text{CTS}}$ that does not generate an error. See the FLC bit in the PSMR description.

7.10.16.14.2 Reception Errors. The following paragraphs describe various types of UART reception errors.

Overrun Error*.* Data is moved from the receive FIFO to the data buffer when the first byte is received into the FIFO. If a receiver FIFO overrun occurs, the channel writes the received character into the internal FIFO over the previous received character (previous character is lost.) The channel writes the received character to the buffer, closes the buffer, sets the OV bit in the Rx BD, and generates the RX interrupt if it is enabled. In automatic multidrop mode, the receiver enters hunt mode immediately.

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 the CD bit in the Rx BD, and generates the RX interrupt (if enabled). 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 hunt mode immediately.

Parity Error*.* When a parity error occurs, the channel writes the received character to the buffer, closes the buffer, sets the PR bit in the Rx BD, and generates the RX interrupt (if enabled). The channel also increments the PAREC counter. In automatic multidrop mode, the receiver enters hunt mode immediately.

Noise Error*.* Noise error is detected by the UART controller when the three samples taken on every bit are not identical. When this error occurs, the channel writes the received character to the buffer and proceeds normally, but increments the NOSEC.

NOTE

In the synchronous mode of the UART controller, this error cannot occur.

Idle Sequence Receive*.* An idle is detected when one character consisting of all ones is received. When the UART is receiving data into a receive buffer, and an idle is received, the

channel counts the number of consecutive idle characters received. If the count reaches the value programmed into MAX_IDL, the buffer is closed, and an RX interrupt is generated. If no receive 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.

NOTE

To disable the idle sequence function entirely, set the MAX_IDL value to zero.

Framing Error*.* A framing error is detected by the UART controller when a character is received with no stop bit. All framing errors are report by the UART controller, regardless of the UART mode. When this error occurs, the channel writes the received character to the buffer, closes the buffer, sets the FR bit in the BD, and generates the RX interrupt (if enabled). The channel also increments the FRMEC. When this error occurs, parity is not checked for this character. In automatic multidrop mode, the receiver enters hunt mode immediately.

If the RZS bit is set in the UART mode register when the UART is in the synchronous mode (SYN is set), then the receiver reports all framing errors, but continues reception with the assumption that the unexpected zero is really the start bit of the next character. If RZS is set, user software may not wish to consider a reported UART framing error as a true UART framing error, unless two or more framing errors occur within a short period of time.

Break Sequence*.* The UART offers very flexible break support for the receiver. When the first break sequence is received (one or more all-zero characters), the UART increments the BRKEC and issues the break start (BRKs) event in the UART event register, which can generate an interrupt (if enabled). The UART then measures the break length, and, when the break sequence is complete, writes the length to the BRKLN register. After the first one is received, the UART also issues the break end (BRKe) event in the UART event register, which can generate an interrupt (if enabled). If the UART was in the process of receiving characters when the break was received, it will also close the receive buffer, set the BR bit in the Rx BD, and write the RX bit in the event register, which can generate an interrupt (if enabled).

If the RZS bit is set in the UART mode register when the UART is in the synchronous mode (SYN is set), then a break sequence will be detected only after two successive break characters are received.

7.10.16.15 UART MODE REGISTER (PSMR). Each PSMR is a 16-bit, memory-mapped, read-write register that controls SCC operation. When the SCC is configured as a UART, this register is called the UART mode register. This register is cleared at reset. Many of the PSMR bits may be modified on the fly (i.e., while the receiver and transmitter are enabled).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLC	SL	CL		UM		FRZ	RZS	SYN	DRT	—	PEN	RPM		TPM	

FLC—Flow Control

0 = Normal operation. The GSMR and port C registers determine the mode of the $\overline{CTS}$ pin.

1 = Asynchronous flow control. When the $\overline{CTS}$ pin is negated, the transmitter will stop transmitting at the end of the current character. (If $\overline{CTS}$ is negated past the middle of the current character, the next full character may be sent, and then transmission will be stopped.) When $\overline{CTS}$ is asserted once more, transmission will continue where it left off. No $\overline{CTS}$ lost error will be reported. No characters except idles will be transmitted while $\overline{CTS}$ is negated.

SL—Stop Length

The SL bit selects the number of the stop bits transmitted by the UART. This bit may be modified on the fly. The receiver is always enabled for one stop bit unless the UART is in synchronous mode and the RZS bit is set. Fractional stop bits are configured in the DSR.

0 = One Stop Bit
1 = Two Stop Bits

CL—Character Length

The CL bits determine the number of data bits in the character, not including the optional parity or multidrop address bits. When less than an 8-bit character is used, the MSBs in memory are written as zeros, and on transmission the MSBs in memory are a don’t care. These bits may be modified on the fly.

00 = 5 Data Bits
01 = 6 Data Bits
10 = 7 Data Bits
11 = 8 Data Bits

UM—UART Mode

The UART mode bits select the protocol that is implemented over the ASYNC channel. These bits may be modified on the fly.

00 = Normal UART operation. Multidrop mode is disabled, and an idle-line wake-up is selected. In the idle-line wake-up mode, the UART receiver is reenabled by receiving one character of all ones.

01 = Multidrop non-automatic mode. In the multidrop mode, an additional address/data bit is transmitted with each character. The multidrop asynchronous modes are compatible with the MC68681 DUART, the MC68HC11 SCI, the DSP56000 SCI, and the Intel 8051 serial interface. The UART receiver is reenabled when the last data bit received in the character (i.e., the address bit) is a one. This means that the received character is an address that has to be processed by all inactive processors. The UART receives the address character and writes it to a new buffer. The CPU32+ core then compares the written address with its own address to decide whether to ignore or process the following characters.

10 = Reserved

11 = Multidrop automatic mode. In this mode, the CP automatically checks the address of the incoming address character using the UADDR1 and UADDR2 parameter

RAM values, and automatically accepts or discards the data that follows the address.

FRZ—Freeze Transmission

This bit allows the user to halt the UART transmitter and continue transmission from the same point at a later time.

0 = Normal operation. If the UART was previously frozen, the UART resumes transmission from the next character in the same buffer that was frozen.

1 = The UART completes transmission of any data already transferred to the UART FIFO (the number of characters depends on the TFL bit in the GSMR) and then freezes (stops transmitting data). After this bit is reset, transmission will proceed from the next character.

RZS—Receive Zero Stop Bits

The RZS bit configures the UART receiver to receive data without stop bits. This configuration is useful in V.14 applications where UART data is supplied synchronously and all stop bits of a particular character may be omitted for the purpose of across-network rate adaptation. RZS should only be set if the SYN bit is also set.

0 = The receiver operates normally with at least one stop bit required between characters. A framing error is issued upon a missing stop bit, and a break status is set if a character with all-zero data bits is received with a zero stop bit.

1 = The receiver will continue reception if a missing stop bit is detected. If the stop bit is a zero, then the next bit is considered as the first data bit of the next character. A framing error is issued if a stop bit is missing, but a break status will be reported only after back-to-back reception of two break characters without stop bits.

SYN—Synchronous Mode

0 = Normal asynchronous operation. Note that the user would normally program the TENC and RENC bits in the GSMR to NRZ, and must select either 8×, 16×, or 32× in the RDCR and TDCR bits of the GSMR (16× is the recommended value for most applications).

1 = Synchronous UART using 1× clock. Note that the user would normally program the TENC and RENC bits in the GSMR to NRZ, and must select the RDCR and TDCR bits in the GSMR to be 1× mode.
A one bit is transferred with each clock and is synchronous to the clock. (As with the other modes, the clock may be provided internally or externally.) This mode is sometimes referred to as isochronous operation of a UART channel.

DRT—Disable Receiver While Transmitting

0 = Normal operation

1 = While data is being transmitted by the SCC, the receiver is disabled, being gated by the internal $\overline{\text{RTS}}$ signal. This configuration is useful if the UART is configured onto a multidrop line and the user does not wish to receive his own transmission.

Bit 5—Reserved

PEN—Parity Enable

0 = No Parity
1 = Parity is enabled and determined by the parity mode bits.

RPM—Receiver Parity Mode

The RPM bits select the type of parity check to be performed by the receiver. The RPM bits can be modified on the fly.

00 = Odd Parity
01 = Low Parity (always check for a zero in the parity bit position)
10 = Even Parity
11 = High Parity (always check for a one in the parity bit position)

When odd parity is selected, the transmitter will count the number of ones in the data word. If the total number of ones is not an odd number, the parity bit is set to one and thus produces an odd number. If the receiver counts an even number of ones, an error in transmission has occurred. In the same manner, for even parity, an even number must result from the calculation performed at both ends of the line. In high/low parity (sometimes called mark/space parity), if the parity bit is not high/low, a parity error is reported.

NOTE

The receive parity errors cannot be disabled, but can be ignored if desired.

TPM—Transmitter Parity Mode

The TPM bits select the type of parity to be performed for the transmitter. The TPM bits can be modified on the fly.

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)

7.10.16.16 UART RECEIVE BUFFER DESCRIPTOR (RX BD). The CP reports information concerning the received data on a per-buffer basis via Rx BDs.

The CP closes the current buffer, generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:

1. Receiving a user-defined control character (when the reject bit = 0 in the control character table entry)
2. Detecting an error during message processing
3. Detecting a full receive buffer
4. Receiving a MAX_IDL number of consecutive idle characters
5. Issuing the ENTER HUNT MODE command
6. Issuing the CLOSE Rx BD command

7. Receiving an address character when working in multidrop mode (The address character is written to the next buffer for software comparison.)

An example of the SCC UART Rx BD process is shown in Figure 7-48.

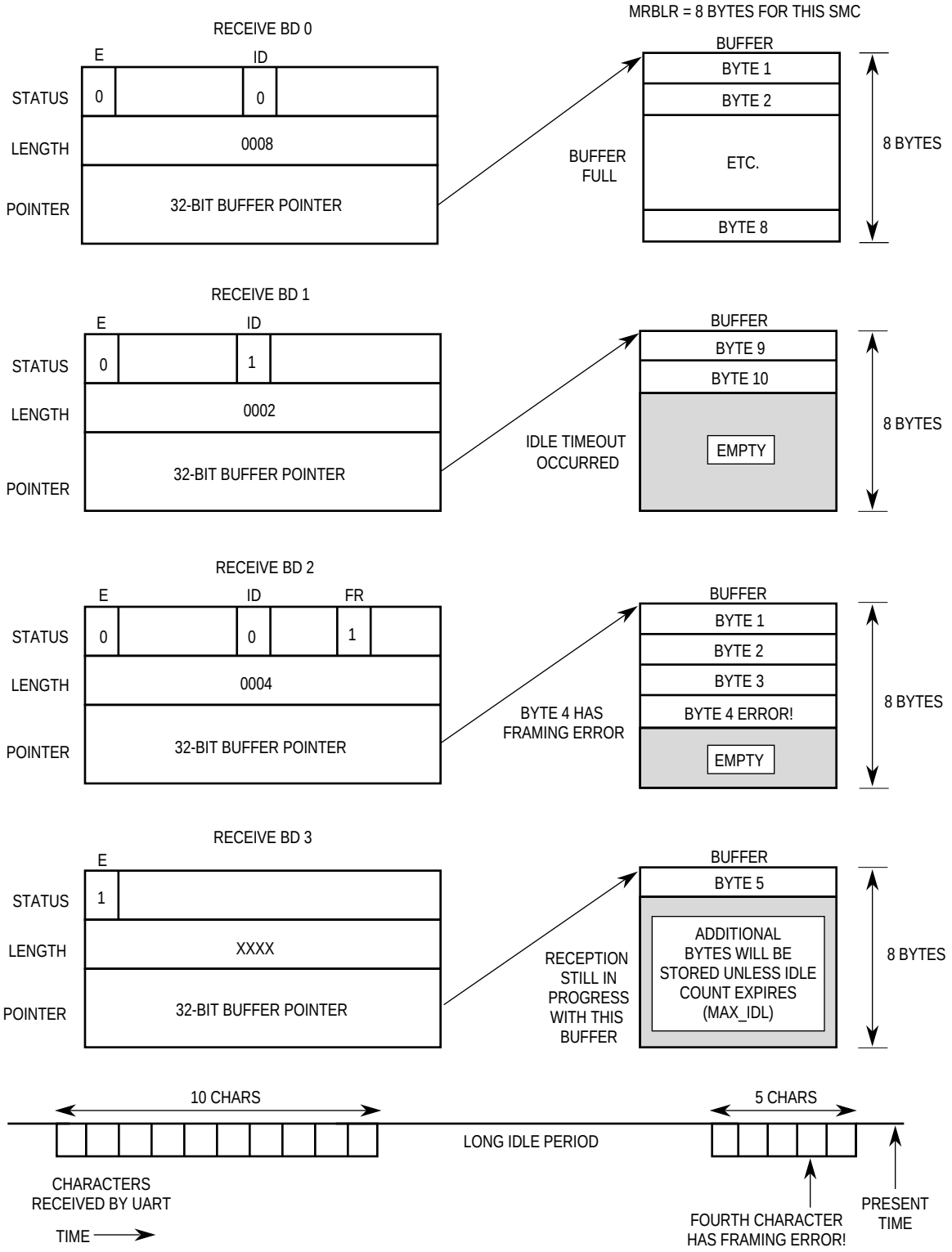


Figure 7-48. SCC UART Rx BD Example

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	—	W	I	C	A	CM	ID	AM	—	BR	FR	PR	—	OV	CD
OFFSET + 2	DATA LENGTH															
OFFSET + 4	**RX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: Entries in boldface must be initialized by the user.

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 CPU32+ 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 CPU32+ core should not write any fields of this Rx BD.

Bits 14, 6, 2—Reserved

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 BD s in this table is programmable and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been filled.

1 = The RX bit in the UART event register will be set when this buffer has been completely filled by the CP, indicating the need for the CPU32+ core to process the buffer. The RX bit can cause an interrupt if it is enabled.

C—Control Character

0 = This buffer does not contain a control character.

1 = This buffer contains a control character. The last byte in the buffer is one of the user-defined control characters.

A—Address

0 = The buffer contains data only.

1 = When working in non-automatic multidrop mode, this bit indicates that the first byte of this buffer contains an address byte. The address comparison should be implemented in software. In automatic multidrop mode, this bit indicates that the BD contains a message received immediately after an address recognized in UADDR1 or UADDR2. This address is not written into the receive buffer.

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 overwritten automatically when the CP next accesses this BD. However, the E-bit will be cleared if an error occurs during reception, regardless of the CM bit.

ID—Buffer Closed on Reception of Idles

The buffer was closed due to the reception of the programmable number of consecutive idle sequences (defined in MAX_IDL).

AM—Address Match

This bit has meaning only if the address bit is set and the automatic multidrop mode was selected in the UM bits. Following an address match, this bit defines which address character matched the user-defined address character, enabling the UART to receive data.

0 = The address matched the value in UADDR2.
1 = The address matched the value in UADDR1.

BR—Break Received

A break sequence was received while receiving data into this buffer.

FR—Framing Error

A character with a framing error was received and is located in the last byte of this buffer. A framing error is a character without a stop bit. A new receive buffer will be used for further data reception.

PR—Parity Error

A character with a parity error was received and is located in the last byte of this buffer. A new receive buffer will be used for further data reception.

OV—Overrun

A receiver overrun occurred during message reception.

CD—Carrier Detect lost

The carrier detect signal was negated during message reception.

Data Length

Data length is the number of octets written by the CP into this BD's data buffer. It is written by the CP once 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.

Rx Data Buffer Pointer

The receive 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.

7.10.16.17 UART TRANSMIT BUFFER DESCRIPTOR (TX BD). Data is presented to the CP for transmission on an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The CP confirms transmission or indicates error conditions via the BDs to inform the processor that the buffers have been serviced.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	—	W	I	CR	A	CM	P	NS	—	—	—	—	—	—	CT
OFFSET + 2	**DATA LENGTH**															
OFFSET + 4	**TX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: Entries in boldface must be initialized by the user.

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.

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 TxBD in the table. After this buffer has been used, the CP will transmit data from the first BD in the table (the BD pointed to by TBASE). The number of Tx BDs in this table is programmable, and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.

1 = The TX bit in the UART event register will be set when this buffer has been serviced by the CP, which can cause an interrupt.

CR—Clear-to-Send Report

This bit allows a choice of either no delay between buffers transmitted in UART mode, or a more accurate CTS lost error reporting and three bits of idle between buffers.

0 = The buffer following this buffer will be transmitted with no delay (assuming it is ready), but the CT bit may not be set in the correct Tx BD or may not be set at all in a CTS lost condition. Asynchronous flow control, however, continues to function normally.

1 = Normal CTS lost (CT bit) error reporting, and three bits of idle occur between back-to-back buffers.

A—Address

This bit is valid only in multidrop mode (either automatic or non-automatic).

0 = This buffer contains data only.
1 = Set by the CPU32+ core, this bit indicates that this buffer contains address character(s). All the buffer's data will be transmitted as address characters.

CM—Continuous Mode

0 = Normal operation.
1 = The R-bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this BD. However, the R-bit will be cleared if an error occurs during transmission, regardless of the CM bit.

P—Preamble

0 = No preamble sequence is sent.
1 = The UART sends one character of all ones before sending the data so that the other end will detect an idle line before the data. If this bit is set and the data length of this BD is zero, only a preamble will be sent.

NS—No Stop Bit Transmitted

0 = Normal operation. Stop bits are sent with all characters in this buffer.
1 = The data in this buffer will be sent without stop bits if the SYNC mode is selected by setting the SYN bit in the PSMR. If ASYNC is selected the stop bit is SHAVED according to the value of the DSR register.

The following bit is written by the CP after it has finished transmitting the associated data buffer.

CT—CTS Lost

0 = The $\overline{\text{CTS}}$ signal remained asserted during transmission.
1 = The $\overline{\text{CTS}}$ signal was negated during transmission.

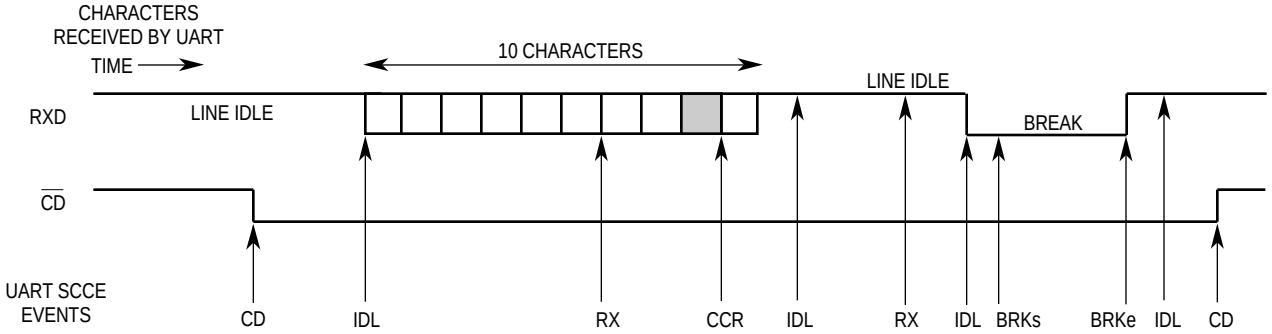
Data Length

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. Normally, this value should be greater than zero. The data length may be equal to zero with the P-bit set, and only a preamble will be sent.

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.

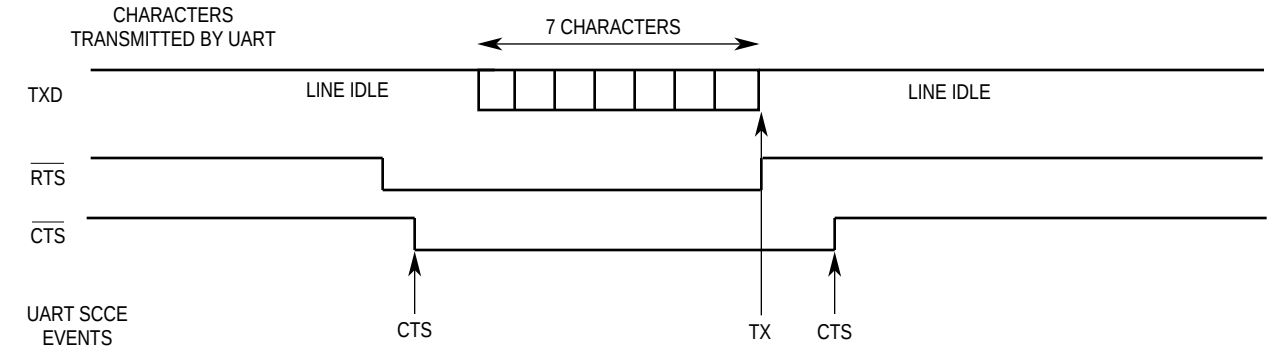
7.10.16.18 UART EVENT REGISTER (SCCE). The SCCE is called the UART event register when the SCC is operating as a UART. It is a 16-bit register used to report events recognized by the UART channel and to generate interrupts. On recognition of an event, the UART controller will set the corresponding bit in the UART event register. Interrupts generated by this register may be masked in the UART mask register. An example of interrupts that may be generated by the UART is shown in Figure 7-49.



NOTES:
1. The first RX event assumes receive 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 receive buffer.



NOTES:
1. TX event assumes all seven characters were put into a single buffer and CR = 1 in the Tx BD.
2. The CTS event must be programmed in the port C parallel I/O, not in the SCC itself.

Figure 7-49. UART Interrupt Events Example

The UART event register is a memory-mapped register that 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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—			GLr	GLt	—	AB	IDL	GRA	BRKe	BRKs	—	CCR	BSY	TX	RX

Bits 15–13, 10, 4—Reserved

These bits should be written with zeros.

GLr—Glitch on Rx

A clock glitch was detected by this SCC on the receive clock.

GLt—Glitch on Tx

A clock glitch was detected by this SCC on the transmit clock.

AB—Auto Baud

An auto baud lock was detected. The CPU32+ core should rewrite the baud rate generator with the precise divider value for the desired baud rate. See 7.9 Baud Rate Generators (BRGs) for more details.

IDL—Idle Sequence Status Changed

A change in the status of the serial line was detected on the UART channel. The real-time status of the line may be read in SCCS. Idle is entered when one character of all ones is received. It is exited when a single zero is received.

GRA—Graceful Stop Complete

A graceful stop, which was initiated by the GRACEFUL STOP TRANSMIT command, is now complete. This bit is set as soon the transmitter has finished transmitting any buffer that was in progress when the command was issued. It will be set immediately if no buffer was in progress when the command was issued.

BRKe—Break End

The end of a break sequence was detected. This indication will be set no sooner than after one idle bit is received following a break sequence.

BRKs—Break Start

A break character was received. This is the first break of a break sequence. The user will not receive multiple BRKs events if a long break sequence is received.

CCR—Control Character Received

A control character was received (with reject (R) character = 1) and stored in the receive control character register (RCCR).

BSY—Busy Condition

A character was received and discarded due to lack of buffers. If the multidrop mode is selected, the receiver automatically enters hunt mode immediately. Otherwise, reception continues as soon as an empty buffer is provided. The latest that an Rx BD can be made empty (have its E-bit set) and still guarantee avoiding the busy condition is the middle of the stop bit of the first character to be stored in that buffer.

TX—Tx Buffer

A buffer has been transmitted over the UART channel. If CR = 1 in the Tx BD, this bit is set no sooner than when the last stop bit of the last character in the buffer begins to be transmitted. If CR = 0, this bit is set after the last character was written to the transmit FIFO.

RX—Rx Buffer

A buffer has been received over the UART channel. This event occurs no sooner than the middle of the first stop bit of the character that caused the buffer to be closed.

7.10.16.19 UART MASK REGISTER (SCCM). The SCCM is referred to as the UART mask register when the SCC is operating as a UART. It is a 16-bit read-write register with the same bit formats as the UART event register. If a bit in the UART mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

7.10.16.20 SCC STATUS REGISTER (SCCS). The SCCs is an 8-bit read-only register that allows the user to monitor real-time status conditions on the RXD line. The real-time status of the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins are part of the port C parallel I/O.

7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	ID

Bits 7–1—Reserved

ID—Idle Status

ID is set when the RXD pin has been a logic one for at least one full character time.

0 = The line is not currently idle.
1 = The line is currently idle.

7.10.16.21 SCC UART EXAMPLE. The following list is an initialization sequence for 9600 baud, 8 data bits, no parity, and stop bit of an SCC UART operation assuming a 25-MHz system frequency. BRG1 and SCC4 are used. The UART is configured with the $\overline{\text{RTS4}}$, $\overline{\text{CTS4}}$, and $\overline{\text{CD4}}$ pins active. In addition, the $\overline{\text{CTS4}}$ pin is used as an automatic flow control signal.

1. The SDCR (SDMA Configuration Register) should be initialized to $0740, rather than being left at its default value of $0000.
2. Configure the port A pins to enable the TXD4 and RXD4 pins. Write PAPAR bits 6 and 7 with ones. Write PADIR bits 6 and 7 with zeros. Write PAODR bits 6 and 7 with zeros.
3. Configure the port C pins to enable $\overline{\text{RTS4}}$, $\overline{\text{CTS4}}$, and $\overline{\text{CD4}}$. Write PCPAR bit 3 with one and bits 10 and 11 with zeros. Write PCDIR bits 3, 10, and 11 with zeros. Write PCSO bits 10 and 11 with ones.
4. Configure BRG1. Write BRGC1 with $010144. The DIV16 bit is not used, and the divider is 162 (decimal). The resulting BRG1 clock is 16× the desired bit rate of the UART.
5. Connect the BRG1 clock to SCC4 using the SI. Write the R4CS bits in SICR to 000. Write the T4CS bits in SICR to 000.
6. Program the CR to execute the INIT RX & TX PARAMS command for this channel. For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
7. Write $0740 to the SDCR to initialize the SDMA Configuration Register.
8. Connect the SCC4 to the NMSI (i.e., its own set of pins). Clear the SC4 bit in the

SICR.

9. Write RBASE and TBASE in the SCC parameter RAM to point to the Rx BD and Tx BD in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.
10. Program the CR to execute the INIT RX & TX PARAMS command for this channel. For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
11. Write RFCR with $15 and TFCR with $15 for normal operation.
12. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 16 bytes, so MRBLR = $0010.
13. Write MAX_IDL with $0000 in the SCC UART-specific parameter RAM to disable the MAX_IDL functionality for this example.
14. Set BRKCR to $0001, so that if a STOP TRANSMIT command is issued, one break character will be sent.
15. Clear PAREC, FRMEC, NOSEC, and BRKEC in the SCC UART-specific parameter RAM for the sake of clarity.
16. Clear UADDR1 and UADDR2. They are not used.
17. Clear TOSEQ. It is not used.
18. Write CHARACTER1–8 with $8000. They are not used.
19. Write RCCM with $C0FF. It is not used.
20. Initialize the Rx BD. Assume the Rx data buffer is at $00404000 in main memory. Write $B000 to Rx_BD_Status. Write $0000 to Rx_BD_Length (not required—done for instructional purposes only). Write $00404000 to Rx_BD_Pointer.
21. Initialize the Tx BD. Assume the Tx data buffer is at $00404100 in main memory and contains five 8-bit characters. Write $B000 to Tx_BD_Status. Write $0010 to Tx_BD_Length. Write $00404100 to Tx_BD_Pointer.
22. Write $FFFF to the SCCE to clear any previous events.
23. Write $0003 to the SCCM to enable the TX and RX interrupts.
24. Write $08000000 to the CIMR to allow SCC4 to generate a system interrupt. (The CICR should also be initialized.)
25. Write $00000020 to GSMR_H4 to configure a small receive FIFO width.
26. Write $00028004 to GSMR_L4 to configure 16× sampling for transmit and receive, the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins to automatically control transmission and reception (DIG bits), and the UART mode. Notice that the transmitter (ENT) and receiver (ENR) have not been enabled yet.
27. Set the PSMR4 to $B000 to configure automatic flow control using the $\overline{\text{CTS}}$ pin, 8-bit characters, no parity, 1 stop bit, and asynchronous UART operation.
28. Write $00028034 to GSMR_L4 to enable the SCC4 transmitter and receiver. This additional write ensures that the ENT and ENR bits will be enabled last.

NOTE

After 16 bytes have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after 16 bytes have been received. Any additional receive data beyond 16 bytes will cause a busy (out-of-buffers) condition since only one Rx BD was prepared.

7.10.16.22 S-RECORDS PROGRAMMING EXAMPLE. In the following paragraphs, an example of a downloading application is given that utilizes an SCC channel as a UART controller. The application performs downloads and uploads of S-records between a host computer and an intelligent peripheral through a serial asynchronous line.;

The S-records are strings of ASCII characters that begin with ‘S’ and end in an end-of-line character. This characteristic will be used to impose a message structure on the communication between the devices. Note that each device may also transmit XON and XOFF characters for flow control, which do not form part of the program being uploaded or downloaded.

The UART mode register should be set as required, with the FRZ bit cleared and the ENT and ENR bits set. Receive buffers should be linked to the receive buffer table with the interrupt (I) bit set. For simplicity, assume that the line is not multidrop (no addresses are transmitted) and that each S-record will fit into a single data buffer.

Three characters should first be entered into the UART control character table:

1. Line Feed—Both the E and R bits should be cleared. When an end-of-line character is received, the current buffer is closed (the next BD taken by the CP) and made available to the CPU32+ core for processing. This buffer contains an entire S record, which the processor can now check and copy to memory or disk as required.
2. XOFF—E should be cleared, and R should be set. Whenever the CPU32+ core receives a control character received interrupt and the receive control character register contains XOFF, the software should immediately stop transmitting to the other station by setting the FRZ bit in the UART mode register. This prevents data from being lost by the other station when it runs out of receive buffers.
3. XON—XON should be received after XOFF. E should be cleared, and R should be set. The FRZ bit on the transmitter should now be cleared. The CP 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 the S-records, the CPU32+ core must only wait for the RX interrupt, indicating the reception of a complete S-record buffer. Transmission requires assembling S-records into data buffers and linking them to the transmit buffer table (transmission may be temporarily halted by reception of an XOFF character). This scheme minimizes the number of interrupts received by the CPU32+ core (one per S-record) and relieves it from the task of continually scanning for control characters.

7.10.17 HDLC Controller

Layer 2 of the seven-layer OSI model is the data link layer. One of the most common layer 2 protocols is HDLC. In fact, many other common layer 2 protocols are heavily based on

HDLC, particularly the framing structure of HDLC: namely, SDLC, SS#7, AppleTalk, LAPB, and LAPD. The framing structure of HDLC is shown in Figure 7-50.

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 to provide a method of clocking and synchronizing the transmitter/receiver.

Since the layer 2 frame can be transmitted over a point-to-point link, a broadcast network, or packet and circuit switched systems, an address field is needed to carry 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. SS#7 has no address field at all because it is always used in point-to-point signaling links. LAPD further divides its 16-bit address into different fields to specify various access points within one piece of equipment. It also defines a broadcast address. Some HDLC-type protocols also allow for 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 CRC (CRC) to the frame, which in most protocols is 16-bits long but may 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 the MODE bits of the GSMR select the HDLC mode, then that SCC functions as an HDLC controller. When an SCC in HDLC mode is used with a nonmultiplexed modem interface, then the SCC outputs are connected directly to the external pins. Modem signals may be supported through the port C pins. The receive and transmit clocks can be supplied either from the bank of baud rate generators, by the DPLL, or externally. The HDLC controller may also be connected to one of the two 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 CPU32+ core and may be either synchronous or asynchronous with respect to the other SCCs. The user can allocate up to 196 BDs for receive and transmit tasks so that many frames may be transmitted or received without host intervention.

7.10.17.1 HDLC CONTROLLER KEY FEATURES. The HDLC contains the following key features:

- Flexible Data Buffers with Multiple Buffers per Frame
- Separate Interrupts for Frames and Buffers (Receive and Transmit)
- Received Frames Threshold To Reduce Interrupt Overhead

- May Be Used with the SCC DPLL
- Four Address Comparison Registers with Mask
- Maintenance of Five 16-Bit Error Counters
- Flag/Abort/Idle Generation/Detection
- Zero Insertion/Deletion
- 16-Bit 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
- Automatic Retransmission in Case of Collision

7.10.17.2 HDLC CHANNEL FRAME TRANSMISSION PROCESSING. The HDLC transmitter is designed to work with almost no intervention from the CPU32+ core. When the CPU32+ core enables one of the transmitters, it will start transmitting flags or idles as programmed in the HDLC mode register. The HDLC controller will poll the first BD in the transmit channel's BD table. When there is a frame to transmit, the HDLC controller will fetch the data from memory and start transmitting the frame (after first transmitting the user-specified minimum number of flags between frames). When the end of the current BD has been reached and the last buffer in the frame bit is set, the CRC, if selected, and the closing flag are appended. In HDLC, the LSB of each octet is transmitted first, and the MSB of the CRC is transmitted first. A typical HDLC frame is shown in Figure 7-50.

OPENING FLAG	ADDRESS	CONTROL	INFORMATION (OPTIONAL)	CRC	CLOSING FLAG
8 BITS	16 BITS	8 BITS	8N BITS	16 BITS	8 BITS

Figure 7-50. HDLC Framing Structure;

Following the transmission of the closing flag, the HDLC controller writes the frame status bits into the BD and clears the R-bit. When the end of the current BD has been reached and the last bit is not set (working in multibuffer mode), only the R-bit is cleared. In either mode, an interrupt may be issued if the I-bit in the Tx BD is set. The HDLC controller will then proceed to the next Tx BD in the table. In this way, the user may be interrupted after each buffer, after a specific buffer has been transmitted, or after each frame.

To rearrange the transmit queue before the CP has completed transmission of all buffers, issue the STOP TRANSMIT command. This technique can be useful for transmitting expedited data before previously linked buffers or for error situations. When receiving the STOP TRANSMIT command, the HDLC controller will abort the current frame being transmitted and start 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 may be issued. A special interrupt (GRA) can be generated in the event register when the current frame is complete.

7.10.17.3 HDLC CHANNEL FRAME RECEPTION PROCESSING. The HDLC receiver is also designed to work with almost no intervention from the CPU32+ core. The HDLC receiver can perform address recognition, CRC checking, and maximum frame length checking. The received frame is available to the user for performing any HDLC-based protocol.

When the CPU32+ core enables one of the receivers, the receiver waits for an opening flag character. When the receiver detects the first byte of the frame, the HDLC controller will compare 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 will compare 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 will fetch the next BD and, if it is empty, will start to transfer the incoming frame to the BD's associated data buffer. When the data buffer has been filled, the HDLC controller clears the E-bit in the BD and generates an interrupt if the I-bit in the BD is set. If the incoming frame exceeds the length of the data buffer, the HDLC controller will fetch the next BD in the table and, if it is empty, will continue to transfer the rest of the frame to this BD's associated data buffer.

During this process, the HDLC controller will check for a frame that is too long. When the frame ends, the CRC field is checked against the recalculated value and is written to the data 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 the 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. The HDLC controller next generates a maskable interrupt, indicating that a frame has been received and is in memory. The HDLC controller then waits for a new frame. Back-to-back frames may be received with only a single shared flag between frames.

The user can configure the HDLC controller not to interrupt the CPU32+ core until a certain number of frames has been received. This is configured in the received frames threshold (RFTHR) location of the parameter RAM. The user can combine this function with a timer to implement a timeout if less than the threshold number of frames is received.

7.10.17.4 HDLC MEMORY MAP. When configured to operate in HDLC mode, the QUICC overlays the structure listed in Table 7-5 with the HDLC-specific parameters described in Table 7-8.

Table 7-8. HDLC-Specific Parameters

Address	Name	Width	Description
SCC Base + 30	RES	Long	Reserved
SCC Base + 34	C_MASK	Long	CRC Constant
SCC Base + 38	C_PRES	Long	CRC Preset
SCC Base + 3C	DISFC	Word	Discard Frame Counter
SCC Base + 3E	CRCEC	Word	CRC Error Counter

Table 7-8. HDLC-Specific Parameters

SCC Base + 40	ABTSC	Word	Abort Sequence Counter
SCC Base + 42	NMARC	Word	Nonmatching Address Rx Counter
SCC Base + 44	RETRC	Word	Frame Retransmission Counter
SCC Base + 46	MFLR	Word	Max Frame Length Register
SCC Base + 48	MAX_cnt	Word	Max_Length Counter
SCC Base + 4A	RFTHR	Word	Received Frames Threshold
SCC Base + 4C	RFCNT	Word	Received Frames Count
SCC Base + 4E	HMASK	Word	User-Defined Frame Address Mask
SCC Base + 50	HADDR1	Word	User-Defined Frame Address
SCC Base + 52	HADDR2	Word	User-Defined Frame Address
SCC Base + 54	HADDR3	Word	User-Defined Frame Address
SCC Base + 56	HADDR4	Word	User-Defined Frame Address
SCC Base + 58	TMP	Word	Temp Storage
SCC Base + 5A	TMP_MB	Word	Temp Storage

NOTE: The boldfaced items should be initialized by the user.

C_MASK. For the 16-bit CRC-CCITT, C_MASK should be initialized with $0000F0B8. For the 32-bit CRC-CCITT, C_MASK should be initialized with $DEBB20E3.

C_PRES. For the 16-bit CRC-CCITT, C_PRES should be initialized with $0000FFFF. For the 32-bit CRC-CCITT, C_PRES should be initialized with $FFFFFFFF.

DISFC, CRCEC, ABTSC, NMARC, and RETRC. These 16-bit (modulo 2^{16}) counters are maintained by the CP. They may be initialized by the user while the channel is disabled. The counters are as follows:

```
DISFC       Discarded Frame Counter (error-free frames but no 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       Non-Matching Address Received Counter (error-free frames only)
RETRC       Frame Retransmission Counter (due to collision)
```

MFLR**.** The HDLC controller checks the length of an incoming HDLC frame against the user-defined value given in this 16-bit register. If this limit is exceeded, the remainder of the incoming HDLC 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 to the end of the frame and reports the frame status and the frame length in the last Rx BD. MFLR is defined as all the in-frame bytes between the opening flag and the closing flag (address, control, data, and CRC). MAX_cnt is a temporary down-counter used to track the frame length.

HMASK, HADDR1, HADDR2, HADDR3, and HADDR4. Each HDLC controller has five 16-bit registers for address recognition—one mask register and four address registers. The HDLC controller reads the frame's address from the HDLC receiver, checks it against the four address register values, and then masks the result with the user-defined mask register. A one in the mask register represents a bit position for which address comparison should

occur; a zero represents a masked bit position. Upon an address match, the address and the data following are written into the data buffers. When the addresses are not matched and the frame is error-free, the nonmatching address received counter (NMARC) is incremented.

NOTE

For 8-bit addresses, mask out (clear) the eight high-order bits in the HMASK register.

The eight low-order bits of HMASK and HADDRx should contain the address byte that immediately follows the opening flag. Example: To recognize a frame that begins $7E (Flag), $68, $AA, using 16-bit address recognition, HADDRx should contain $AA68, and HMASK should contain $FFFF (see Figure 7-51).

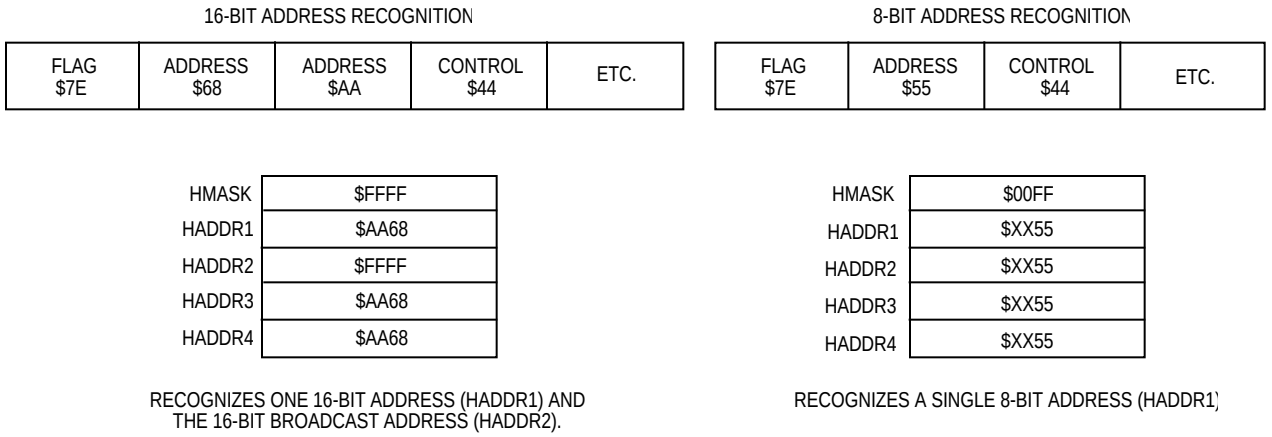


Figure 7-51. HDLC Address Recognition Example

RFTHR. The received frames threshold value is used to reduce the interrupt overhead that might otherwise occur when a series of short HDLC frames arrives, each causing an RXF interrupt. By setting the RFTHR value, the user can limit the frequency of RXF interrupts. The RXF interrupt will only occur when the RFTHR value is reached. RFCNT is a down-counter used to implement this feature.

NOTE

The user should provide enough empty Rx BDs to receive the number of frames specified in RFTHR.

7.10.17.5 HDLC PROGRAMMING MODEL. The CPU32+ core configures each SCC to operate in one of the protocols by the MODE bits in the GSMR. The HDLC controller uses the same data structure as in all other modes. This data structure supports multibuffer operation and address comparisons.

The receive errors (overrun, nonoctet aligned frame, CD lost, aborted frame, and CRC error) are reported through the Rx BD. The transmit errors (underrun and CTS lost) are reported through the Tx BD.

7.10.17.6 HDLC COMMAND SET. The following transmit and receive commands are issued to the CR.

7.10.17.6.1 Transmit Commands. The following paragraphs describe the HDLC transmit commands.

STOP TRANSMIT Command*.* After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the transmit enable mode and starts polling the first BD in the table every 64 transmit clocks (immediately if the TOD bit in the TODR is set).

The channel 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 transmitted, and the transmit FIFO is flushed. The TBPTR is not advanced, no new BD is accessed, and no new frames are transmitted for this channel. The transmitter will transmit an abort sequence consisting of 01111111 (if the command was given during frame transmission) and then begin to transmit flags or idles, as indicated by the HDLC mode register.

NOTE

If the MFF bit in the PSMR is set, then it is possible for one or more small frames to be flushed from the transmit FIFO. To avoid this, the GRACEFUL STOP TRANSMIT command may be used.

GRACEFUL STOP TRANSMIT Command*.* The channel GRACEFUL STOP TRANSMIT command is used to stop transmission in an orderly way rather than abruptly, as performed by the regular STOP TRANSMIT command. It stops transmission after the current frame has completed transmission, or immediately if there is no frame being transmitted. The GRA bit in the SCCE will be set once transmission has stopped. After transmission ceases, the HDLC transmit parameters, including BDs, may be modified. The TBPTR will point to the next Tx BD in the table. Transmission will begin once the R-bit of the next BD is set and the RESTART TRANSMIT command is issued.

RESTART TRANSMIT Command*.* The RESTART TRANSMIT command enables the transmission of characters on the transmit channel. This command is expected by the HDLC controller after a STOP TRANSMIT command, after a STOP TRANSMIT command and disabling the channel in its SCC mode register, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error (underrun or CTS lost when no automatic frame retransmission is performed). The HDLC controller will resume transmission from the current TBPTR in the channel's Tx BD table.

INIT TX PARAMETERS Command*.* This command initializes all transmit parameters in this serial channel's parameter RAM to their reset state. This command should only be issued

when the transmitter is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both transmit and receive parameters.

7.10.17.6.2 Receive Commands. The following paragraphs describe the HDLC receive commands.

ENTER HUNT MODE Command*.* After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the receive enable mode and will use 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 the 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, and the CRC is reset. Further frame reception will use the next BD.

CLOSE Rx BD Command*.* This command should not be used in the HDLC protocol.

INIT RX PARAMETERS Command*.* This command initializes all the receive parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the receiver is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both receive and transmit parameters.

7.10.17.7 HDLC ERROR-HANDLING PROCEDURE. The HDLC controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the HDLC event register.

7.10.17.7.1 Transmission Errors. The following paragraphs describe various types of HDLC transmission errors.

Transmitter Underrun*.* When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the underrun (U) bit in the BD, and generates the TXE interrupt if it is enabled. The channel will resume transmission after reception of the RESTART TRANSMIT command. The transmit FIFO size is 32 bytes on SCC1 and 16 bytes on SCC2, SCC3, and SCC4.

CTS Lost During Frame Transmission*.* When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the CT bit in the BD, and generates the TXE interrupt if it is enabled. The channel will resume transmission after reception of the RESTART TRANSMIT command.

If this error occurs on the first or second buffer of the frame and the RTE bit in the HDLC mode register is set, the channel will retransmit the frame when the CTS line becomes active again. When working in an HDLC mode with collision possibility, to ensure the retransmission method functions properly, the first and second data buffers should contain more than 36 bytes of data (SCC1), and 20 bytes of data (SCC2, SCC3, and SCC4) if multiple buffers per frame are used. (Small frames consisting of a single buffer are not subject to this requirement.) The channel will also increment the retransmission counter.

7.10.17.7.2 Reception Errors. The following paragraphs describe various types of HDLC reception errors.

Overrun Error*.* The HDLC controller maintains an internal FIFO; for receiving data. The CP begins programming the SDMA channel (if the data buffer is in external memory) and updating the CRC when 8 or 32 bits (according to the RFW bit in the GSMR) are received in the FIFO. When a receive FIFO overrun occurs, the channel writes the received data byte to the internal FIFO over the previously received byte. The previous data byte and the frame status are lost. The channel closes the buffer with the overrun (OV) bit in the BD set and generates the RXF interrupt if it is enabled. The receiver then enters the hunt mode.

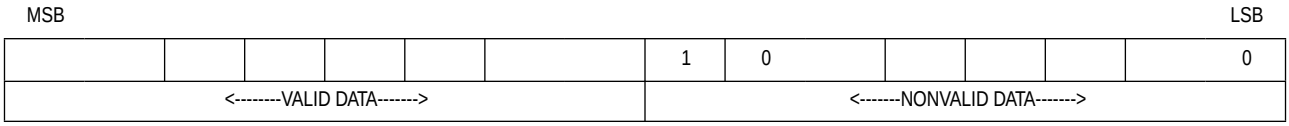
Even if the overrun occurs during a frame whose address is not matched in the address recognition logic, an Rx BD with data length two will be opened to report the overrun, and the RXF interrupt will be generated if it is enabled.

CD Lost During Frame Reception*.* When this error occurs, the channel terminates frame reception, closes the buffer, sets the CD bit in the Rx BD, and generates the RXF interrupt if it is enabled. This error has the highest priority. The rest of the frame is lost, and other errors are not checked in that frame. The receiver then enters the hunt mode.

Abort Sequence*.* An abort sequence is detected by the HDLC controller when seven or more consecutive ones are received. When this error occurs and the HDLC controller is currently receiving a frame, the channel closes the buffer by setting the AB bit in the Rx BD 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.

If the HDLC controller is not currently receiving a frame when an abort is received, no indication is given to the user.

Nonoctet Aligned Frame*.* When this error occurs, the channel writes the received data to the data buffer, closes the buffer, sets the Rx nonoctet aligned frame (NO) bit in the Rx BD, and generates the RXF interrupt (if enabled). The CRC error status should be disregarded on nonoctet frames. After a nonoctet aligned frame is received, the receiver enters hunt mode. (An immediately following back-to-back frame will still be received.) The nonoctet data may be derived from the last word in the data buffer as follows:



NOTE

If the data buffer swapping option is used (MOT bit cleared in the RFCR), then the above diagram refers to the last byte of the data buffer, not the last word. In HDLC, the LSB of each octet is transmitted first, and the MSB of the CRC is transmitted first.

CRC Error**.** When this error occurs, the channel writes the received CRC to the data buffer, closes the buffer, sets the CR bit in the Rx BD, and generates the RXF interrupt (if enabled). The channel also increments the CRC error counter. After receiving a frame with a CRC error, the receiver enters hunt mode. (An immediately following back-to-back frame will still be received.) CRC checking cannot be disabled, but the CRC error may be ignored if checking is not required.

7.10.17.8 HDLC MODE REGISTER (PSMR). Each HDLC mode register is a 16-bit, memory-mapped, read-write register that controls SCC operation. The term HDLC mode register refers to the PSMR of the SCC when that SCC is configured for HDLC. The HDLC mode register is cleared at reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOF				CRC		RTE	—	FSE	DRT	BUS	BRM	MFF	—		

NOF—Number of Flags

Minimum number of flags between frames or before frames (0 to 15 flags). If NOF = 0000, then no flags will be inserted between frames. Thus, the closing flag of one frame will be immediately followed by the opening flag of the next frame in the case of back-to-back frames. These bits may be modified on the fly.

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.

RTE—Retransmit Enable

0 = No retransmission
1 = Automatic frame retransmission is enabled. This is particularly useful in the HDLC Bus protocol and ISDN applications where multiple HDLC controllers may be colliding on a single channel. Note that retransmission only occurs if the CTS lost happens on the first or second buffer of the frame.

Bits 8, 2–0—Reserved

FSE—Flag Sharing Enable

This bit is only valid if the RTSM bit is set in GSMR This bit may be modified on the fly.

0 = Normal operation
1 = If NOF3–NOF0 = 0000, then a single shared flag is transmitted between back-to-back frames. Other values of NOF3–NOF0 are decremented by one when FSE is set. This is useful in Signaling System #7 applications.

DRT—Disable Receiver While Transmitting

0 = Normal operation
1 = While data is being transmitted by the SCC, the receiver is disabled, being gated by the internal $\overline{\text{RTS}}$ signal. This configuration is useful if the HDLC channel is configured onto a multidrop line and the user does not wish to receive his own transmission.

BUS—HDLC Bus Mode

0 = Normal HDLC operation
1 = HDLC Bus operation selected. See 7.10.18 HDLC Bus Controller for more details.

BRM—HDLC Bus $\overline{\text{RTS}}$ Mode

This bit is only valid 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 transmit 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. This is useful when the HDLC Bus protocol is run locally, and at the same time, transmitted over a long-distance transmission line. Data may be delayed by one bit before it is sent over the transmission line; thus, $\overline{\text{RTS}}$ may be used to enable the transmission line buffers. The result is a clean signal level sent over the transmission line.

MFF—Multiple Frames in FIFO

0 = Normal operation. The transmit FIFO can never contain more than one HDLC frame. The CTS lost status will be reported accurately on a per-frame basis. The receiver is not affected by this bit.
1 = The transmit FIFO can contain multiple frames, but CTS lost is not guaranteed to be reported on the exact buffer/frame on which it truly occurred. This option, however, can improve the performance of HDLC transmissions in cases of small back-to-back frames or in cases where the user desires to strongly limit the number of flags transmitted between frames. The receiver is not affected by this bit.

7.10.17.9 HDLC RECEIVE BUFFER DESCRIPTOR (RX BD). The HDLC controller uses the Rx BD to report information about the received data for each buffer. An example of the Rx BD process is shown in Figure 7-52.

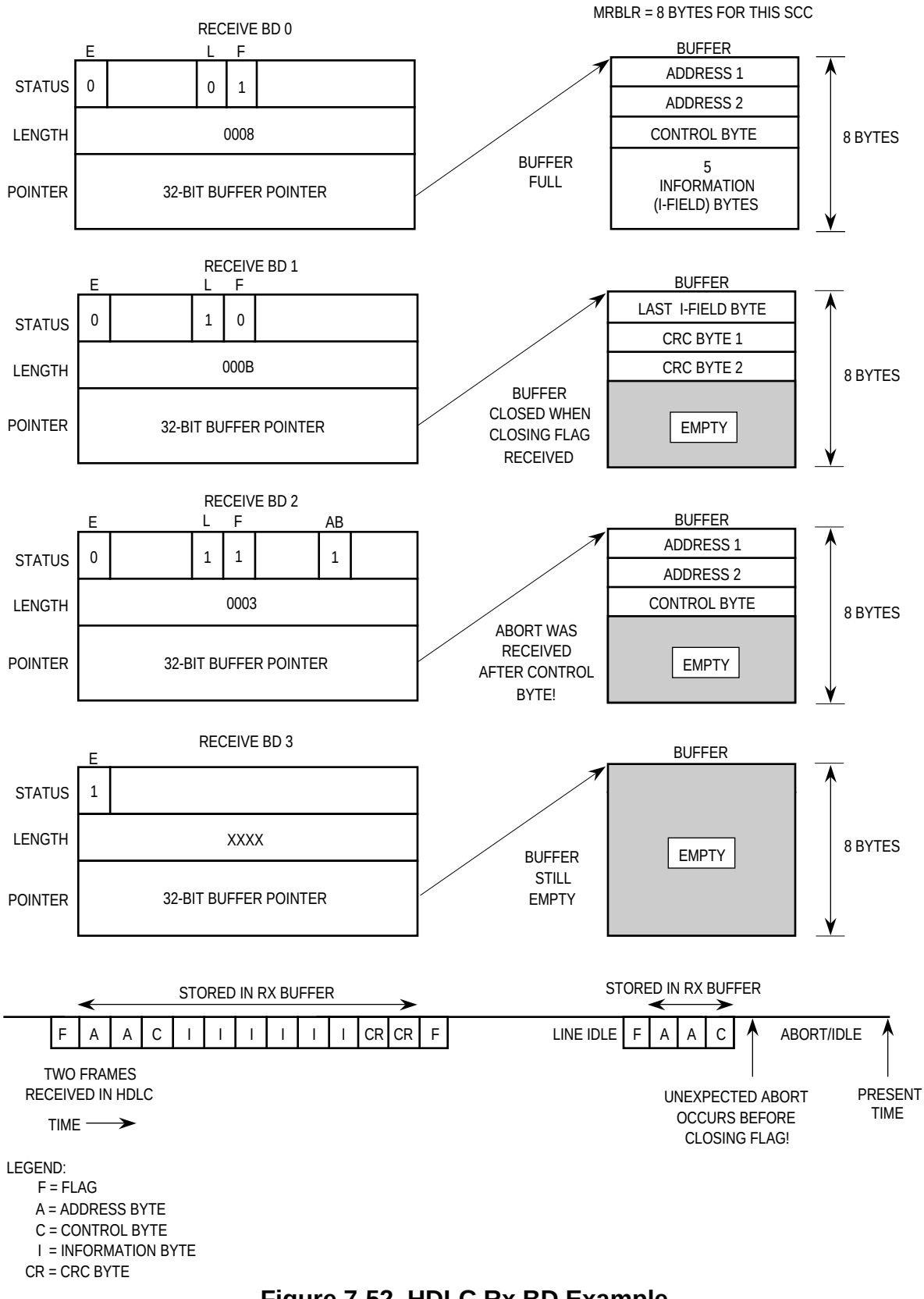


Figure 7-52. HDLC Rx BD Example

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	—	W	I	L	F	CM	—	DE	—	LG	NO	AB	CR	OV	CD
OFFSET + 2	DATA LENGTH															
OFFSET + 4	**RX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: Entries in boldface must be initialized by the user.

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 CPU32+ 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 CPU32+ core should not write any fields of this Rx BD.

Bits 14, 8, 6—Reserved

W—Wrap (Final BD in Table)

0 = This is not the last buffer descriptor in the Rx BD table.

1 = This is the last buffer descriptor 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 BD s in this table is programmable, and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

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 event register will be set when this buffer has been used by the HDLC controller. These two bits may cause interrupts (if enabled).

L—Last in Frame

This bit is set by the HDLC controller when this buffer is the last 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 will write 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.

F—First in Frame

This bit is set by the HDLC controller when this buffer is the first in a frame.

0 = The buffer is not the first in a frame.

1 = The buffer is the first in a frame.

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 overwritten automatically when the CP next accesses this BD. However, the E-bit will be cleared if an error occurs during reception, regardless of the CM bit.

DE—DPLL Error

This bit is set by the HDLC controller when a DPLL error has occurred during the reception of this buffer. In decoding modes where a transition is promised every bit, the DE bit will be set when a missing transition has occurred.

LG—Rx Frame Length Violation

A frame length greater than the maximum defined for this channel was recognized (only the maximum-allowed number of bytes (MFLR) is written to the data buffer). This event will not be reported until the Rx BD is closed and the RXF bit is set, after receipt of the closing flag. The actual number of bytes received between flags is written to the data length field of this BD.

NO—Rx Nonoctet Aligned Frame

A frame that contained a number of bits not exactly divisible by eight was received.

AB—Rx Abort Sequence

A minimum of seven consecutive ones was received during frame reception.

CR—Rx CRC Error

This frame contains a CRC error. The received CRC bytes are always written to the receive buffer.

OV—Overrun

A receiver overrun occurred during frame reception.

CD—Carrier Detect Lost

The carrier detect signal was negated during frame reception. This bit is only valid when working in the NMSI mode.

Data Length

Data length is the number of octets written by the CP into this BD's data buffer. It is written by the CP once as the BD is closed.

When this BD is the last BD in the frame (L = 1), the data length contains the total number of frame octets (including 2 or 4 bytes for CRC).

The actual amount of memory allocated for this buffer should be greater than or equal to the contents of the MRBLR.

Rx Data Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, may reside in either internal or external memory. The Rx buffer pointer must be divisible by 4.

7.10.17.10 HDLC TRANSMIT BUFFER DESCRIPTOR (TX BD). Data is presented to the HDLC controller for transmission on an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The HDLC controller confirms transmission (or indicates error conditions) using the BDs to inform the CPU32+ core that the buffers have been serviced.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**R**	—	**W**	**I**	**L**	**TC**	**CM**	—	—	—	—	—	—	—	UN	CT
OFFSET + 2	**DATA LENGTH**															
OFFSET + 4	**TX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: Entries in boldface must be initialized by the user.

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.

Bits 14, 8–2—Reserved

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 TxBD in the table. After this buffer has been used, the CP will transmit data from the first BD in the table (the BD pointed to by TBASE). The number of Tx BD s in this table is programmable, and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.

1 = Either TXB or TXE in the HDLC event register will be set when this buffer has been serviced by the HDLC controller. These bits can cause interrupts (if enabled).

L—Last

0 = This is not the last buffer in the frame.

1 = This is the last buffer in the current frame.

TC—Tx CRC

This bit is 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 for testing purposes to send a bad CRC after the data.
1 = Transmit the CRC sequence after the last data byte.

CM—Continuous Mode

0 = Normal operation.
1 = The R-bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this BD. However, the R-bit will be cleared if an error occurs during transmission, regardless of the CM bit.

The following status bits are written by the HDLC controller after it has finished transmitting the associated data buffer.

UN—Underrun

The HDLC controller encountered a transmitter underrun condition while transmitting the associated data buffer.

CT—CTS Lost

CTS in NMSI mode or layer 1 grant was lost in GCI mode during frame transmission. If data from more than one buffer is currently in the FIFO when this error occurs, this bit will be set in the Tx BD that is currently open.

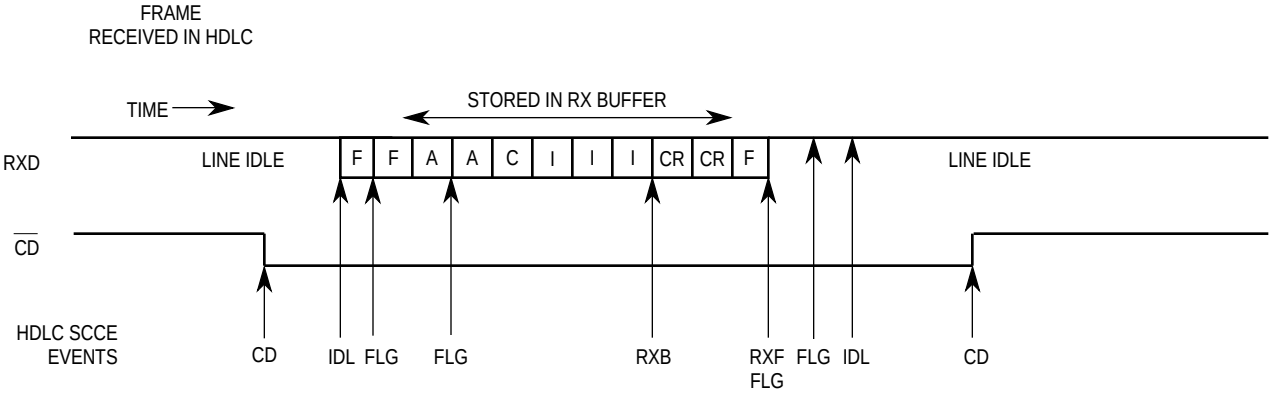
Data Length

The data length is the number of bytes the HDLC controller should transmit from this BD's data buffer. It is never modified by the CP. The value of this field should be greater than zero.

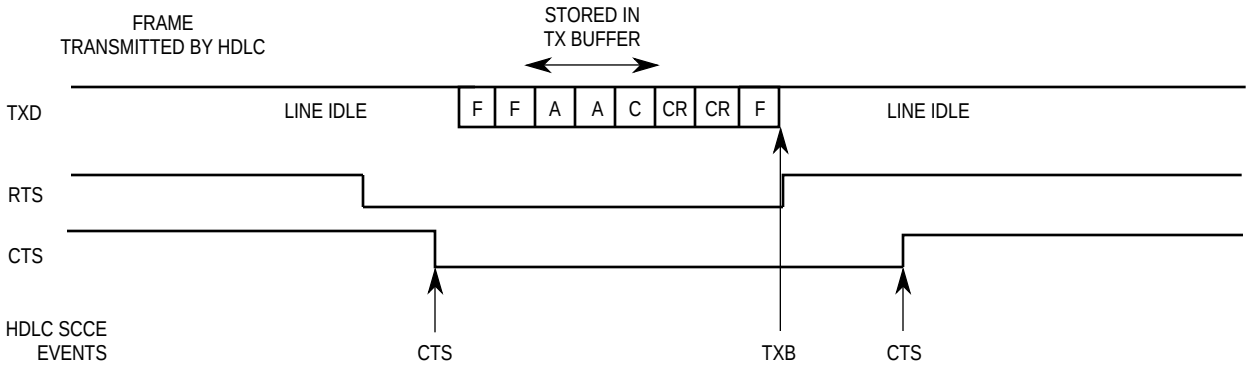
Tx Data Buffer Pointer

The transmit buffer pointer, which contains the address of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory. This value is never modified by the CP.

7.10.17.11 HDLC EVENT REGISTER (SCCE). The SCCE is called the HDLC event register when the SCC is operating as an HDLC controller. It is a 16-bit register used to report events recognized by the HDLC channel and to generate interrupts. On recognition of an event, the HDLC controller will set the corresponding bit in the HDLC event register. Interrupts generated by this register may be masked in the HDLC mask register. An example of interrupts that may be generated in the HDLC protocol is shown in Figure 7-53.



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 7-53. HDLC Interrupt Event Example

The HDLC event register is a memory-mapped register that 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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—			GLr	GLt	DCC	FLG	IDL	GRA	—		TXE	RXF	BSY	TXB	RXB

Bits 15–13, 6, 5—Reserved

These bits should be written with zeros.

GLr—Glitch on Rx

A clock glitch was detected by this SCC on the receive clock.

GLt—Glitch on Tx

A clock glitch was detected by this SCC on the transmit clock.

DCC—DPLL CS Changed

The carrier sense status as generated by the DPLL has changed state. The real-time status may be found in SCCS. This is not the CD pin status (which is reported in port C), and is only valid when the DPLL is used.

FLG—Flag Status

The HDLC controller has stopped or started receiving HDLC flags. The real-time status may be obtained in SCCS.

IDL—Idle Sequence Status Changed

A change in the status of the serial line was detected on the HDLC line. The real-time status of the line may be read in SCCS.

GRA—Graceful Stop Complete

A graceful stop, which was initiated by the GRACEFUL STOP TRANSMIT command, is now complete. This bit is set as soon the transmitter has finished transmitting any frame that was in progress when the command was issued. It will be set immediately if no frame was in progress when the command was issued.

TXE—Tx Error

An error (CTS lost or underrun) occurred on the transmitter channel.

RXF—Rx Frame

A complete frame has been received on the HDLC channel. This bit is set no sooner than two clocks after receipt of the last bit of the closing flag.

BSY—Busy Condition

A frame was received and discarded due to lack of buffers.

TXB—Transmit Buffer

A buffer has been transmitted on the HDLC channel. This bit 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, this bit is set after the last byte of the buffer has been written to the transmit FIFO.

RXB—Receive Buffer

A buffer has been received on the HDLC channel that was not a complete frame.

7.10.17.12 HDLC MASK REGISTER (SCCM). The SCCM is referred to as the HDLC mask register when the SCC is operating as an HDLC controller. It is a 16-bit read-write register with the same bit formats as the HDLC event register. If a bit in the HDLC mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

7.10.17.13 SCC STATUS REGISTER (SCCS). The SCCS is an 8-bit read-only register that allows the user to monitor real-time status conditions on the RXD line. The real-time status of the $\overline{CTS}$ and $\overline{CD}$ pins are part of the port C parallel I/O.

7	6	5	4	3	2	1	0
—	—	—	—	—	FG	CS	ID

Bits 7–3—Reserved

FG—Flags

While FG is cleared, the most recently received 8 bits are examined every bit time to see if a flag is present. FG is set as soon as an HDLC flag ($7E) is received on the line. Once FG is set, it will remain set at least 8 bit times, at which time the next 8 received bits are examined. If another flag occurs, then FG remains set for at least another eight bits; otherwise, FG is cleared and the search begins again.

The examination of the line is made after the data has been decoded by the DPLL.

0 = HDLC flags are not currently being received.
1 = HDLC flags are currently being received.

CS—Carrier Sense (DPLL)

This bit shows the real-time carrier sense of the line as determined by the DPLL, if it is used.

0 = The DPLL does not sense a carrier.
1 = The DPLL does sense a carrier.

ID—Idle Status

ID is set when the RXD pin is a logic one for 15 or more consecutive bit times; it is cleared after a single logic zero is received.

0 = The line is not currently idle.
1 = The line is currently idle.

7.10.17.14 SCC HDLC EXAMPLE #1. The following list is an initialization sequence for an SCC HDLC channel assuming an external clock is provided. SCC4 is used. The HDLC controller is configured with the $\overline{RTS4}$, $\overline{CTS4}$, and $\overline{CD4}$ pins active. The CLK7 pin is used for both the HDLC receiver and transmitter.

1. The SDCR (SDMA Configuration Register) should be initialized to $0740, rather than being left at its default value of $0000.
2. Configure the port A pins to enable the TXD4 and RXD4 pins. Write PAPAR bits 6 and 7 with ones. Write PADIR bits 6 and 7 with zeros. Write PAODR bits 6 and 7 with zeros.
3. Configure the port C pins to enable $\overline{RTS4}$, $\overline{CTS4}$, and $\overline{CD4}$. Write PCPAR bit 3 with one, and bits 10 and 11 with zeros. Write PCDIR bits 3, 10 and 11 with zeros. Write PCSO bits 10 and 11 with ones.
4. Configure port A to enable the CLK7 pin. Write PAPAR bit 14 with a one. Write PADIR bit 14 with a zero.

5. Connect the CLK7 pin to SCC4 using the SI. Write the R4CS bits in SICR to 110. Write the T4CS bits in SICR to 110.
6. Connect the SCC4 to the NMSI (i.e., its own set of pins). Clear the SC4 bit in the SICR.
7. Write $0740 to the SDCR to initialize the SDMA Configuration Register.
8. Write RBASE and TBASE in the SCC parameter RAM to point to the Rx BD and Tx BDs in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.
9. Program the CR to execute the INIT RX & TX PARAMS command for this channel." For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
10. Write RFCR with $18 and TFCR with $18 for normal operation.
11. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 256 bytes, so MRBLR = $0100. The value 256 was chosen to allow an entire receive frame to fit into one receive buffer (see MFLR below).
12. Write C_MASK with $0000F0B8 to comply with 16-bit CCITT-CRC.
13. Write C_PRES with $0000FFFF to comply with 16-bit CCITT-CRC.
14. Clear DISFC, CRCEC, ABTSC, NMARC, and RETRC for the sake of clarity.
15. Write MFLR with $0100 to make the maximum frame size 256 bytes.
16. Write RFTHR with $0001 to allow interrupts after each frame.
17. Write HMASK with $0000 to allow all addresses to be recognized.
18. Clear HADDR1, HADDR2, HADDR3, and HADDR4 for clarity.
19. Initialize the Rx BD. Assume the Rx data buffer is at $00001000 in main memory. Write $B000 to Rx_BD_Status. Write $0000 to Rx_BD_Length (not required—done for instructional purposes only). Write $00001000 to Rx_BD_Pointer.
20. Initialize the Tx BD. Assume the Tx data frame is at $00002000 in main memory and contains five 8-bit characters. Write $BC00 to Tx_BD_Status. Write $0005 to Tx_BD_Length. Write $00002000 to Tx_BD_Pointer.
21. Write $FFFF to the SCCE to clear any previous events.
22. Write $001A to the SCCM to enable the TXE, RXF, and TXB interrupts.
23. Write $08000000 to the CIMR to allow SCC4 to generate a system interrupt. (The CICR should also be initialized.)
24. Write $00000000 to GSMR_H4 to enable normal behavior of the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins and idles between frames (as opposed to flags).
25. Write $00000000 to GSMR_L4 to configure the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins to automatically control transmission and reception (DIAG bits) and the HDLC mode. Normal operation of the transmit clock is used (TCI is cleared). Notice that the transmitter (ENT) and receiver (ENR) have not been enabled. If inverted HDLC operation

is desired, set the RINV and TINV bits.

26. Set the PSMR4 to $0000 to configure one opening and one closing flag, 16-bit CCITT-CRC, and prevention of multiple frames in the FIFO.
27. Write $00000030 to GSMR_L4 to enable the SCC4 transmitter and receiver. This additional write ensures that the ENT and ENR bits will be enabled last.

NOTE

After 5 bytes and CRC have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after a frame is received. Any additional receive data beyond 256 bytes or a single frame will cause a busy (out-of-buffers) condition since only one Rx BD was prepared.

7.10.17.15 SCC HDLC EXAMPLE #2. •The following is an initialization sequence for an SCC HDLC channel that uses the DPLL in a Manchester encoding. The user provides a clock that is 16x the desired bit rate, on the CLK7 pin. CLK7 is then connected to the HDLC transmitter and receiver. (A baud rate generator could have been used instead, if desired). SCC4 is used. The HDLC controller is configured with the $\overline{\text{RTS4}}$, $\overline{\text{CTS4}}$, and $\overline{\text{CD4}}$ pins active.

1. Follow all the steps in the HDLC Example #1, until the step where the GSMR is initialized.
2. Write $00000000 to GSMR_H4 to enable normal behavior of the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins, and idles between frames (as opposed to flags).
3. Write $004AA400 to GSMR_L4 to configure carrier sense always active, a 16-bit preamble of "01" pattern, 16x operation of the DPLL for the receiver and transmitter, Manchester encoding for the receiver and transmitter, the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins to automatically control transmission and reception (DIAG bits), and the HDLC mode. Notice that the transmitter (ENT) and receiver (ENR) have not been enabled yet.
4. Set the PSMR4 to $0000 to configure one opening and one closing flag, 16-bit CCITT-CRC, and not allowing multiple frames in the FIFO.
5. Write $004AA430 to GSMR_L4 to enable the SCC4 transmitter and receiver. This additional write ensures that the ENT and ENR bits will be enabled last.

NOTE

After the preamble and 5 bytes have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after 16 bytes have been received. Any additional receive data beyond 16 bytes will cause a busy (out-of-buffers) condition since only one Rx BD was prepared.

7.10.18 HDLC Bus Controller

HDLC bus is an enhancement of HDLC that allows an HDLC-based LAN and other HDLC point-to-multipoint configurations to be easily implemented. Most versions of HDLC-based controllers only provide point-to-point communications.

HDLC bus is based on the 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, HDLC bus is not fully compliant with I.430 or T1.605, and cannot be used to directly

replace devices that implement these protocols. Instead, HDLC bus is more suited to the needs of non-ISDN LAN and point-to-multipoint configurations.

It may be helpful for the reader to review the basic features of I.430 and T1.605 before learning HDLC bus.

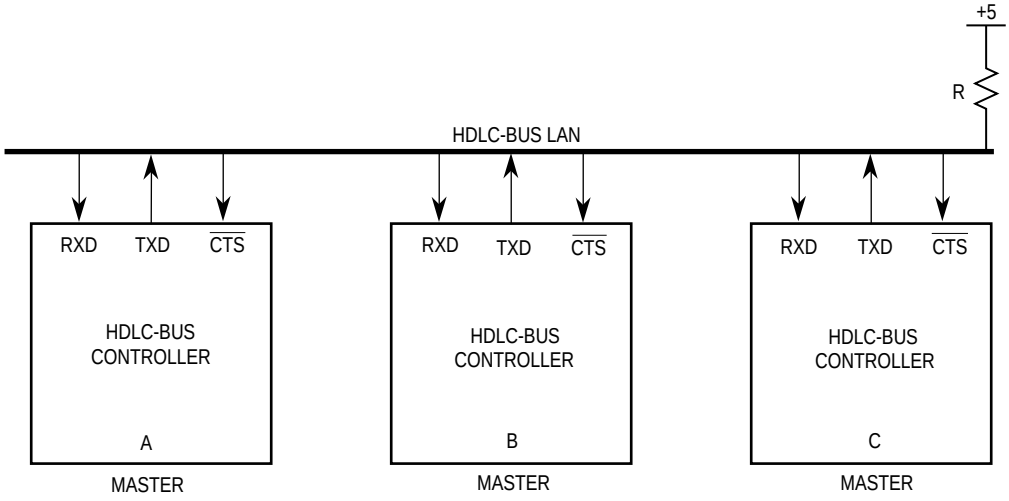
I.430/T1.605 define a method whereby 8 terminals may be connected over the D-channel of the S/T bus of ISDN. The protocol used at layer 2 is a variant of HDLC, called LAPD. However, at layer 1, a method is provided to allow any of the 8 terminals to acquire access to the physical S/T bus to send frames to the switch.

The S/T interface device detects whether the channel is clear by looking at an "echo" bit on the line. The echo bit is designed to echo whatever bit was most recently transmitted on the D channel. Depending on the "class" of the terminal, and the particular situation, the S/T interface device may wait for 7, 8, 9, or 10 ones on the echo bit before allowing the LAPD frame to begin transmission. Once transmission begins, the S/T chip monitors the data that was sent. As long as the echo bit matches the transmit data, the transmission continues. If the echo bit is ever a zero when the transmit bit is a one, then a collision has occurred between terminals, and the station(s) that transmitted a zero immediately stops further transmission. The station that transmitted a one continues normally.

In summary, I.430/T1.605 provides a physical layer protocol that allows multiple terminals to share the same physical connection. These protocols make very efficient use of the bus by dealing with collisions in such a way that one station is always able to complete its transmission. Once a station completes a transmission, it lowers its own priority to give other devices fair access to the physical connection.

HDLC bus works much the same way; however, a few differences exist. First, HDLC bus does not use the echo bit, but rather a separate pin ($\overline{\text{CTS}}$) to monitor the data that was transmitted. The transmit data is simply connected to the $\overline{\text{CTS}}$ input. Second, HDLC bus is a synchronous digital open-drain connection for short-distance configurations, rather than the more complex definition of the S/T interface. Third, HDLC bus allows any HDLC-based frame protocol to be implemented at layer 2, not just LAPD. Fourth, HDLC bus devices wait either 8 or 10 bit times before transmitting, rather than 7, 8, 9, or 10 bits (HDLC bus has one "class" rather than two).

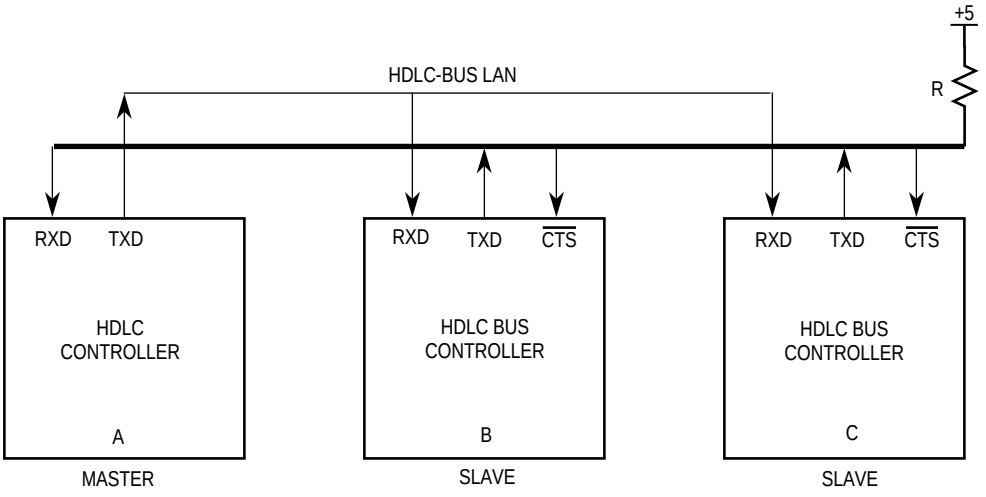
Figure 7-54 shows HDLC-bus in its most common LAN configuration. All stations may transmit and receive data to/from every other station on the LAN. All transmissions are half-duplex, as is typical in LANs.



NOTES:
1. Transceivers may be used to extend the LAN size, if necessary.
2. The TXD pins should be configured to open-drain in the port C parallel I/O port.

Figure 7-54. HDLC Bus Multi-Master Configuration

Figure 7-55 shows the other LAN-type configuration of HDLC bus. In this configuration, a master station transmits to any slave station, with no collisions possible. The slaves communicate only with the master, but may experience collisions in their access over the bus. In this configuration, a slave that must communicate with another slave must first transmit its data to the master, where the data is buffered in RAM and then retransmitted to the other slave. The benefit of this configuration, however, is that full-duplex operation may be obtained. This configuration is preferred in a point-to-multipoint environment.



NOTES:
1. Transceivers may be used to extend the LAN size, if necessary.
2. The TXD pins of slave devices should be configured to open-drain in the port C parallel I/O port.

Figure 7-55. HDLC Bus Single-Master Configuration

7.10.18.1 HDLC BUS KEY FEATURES. The HDLC bus controller contains the following key features:

- Superset of the HDLC Controller Features
- Automatic HDLC Bus Access
- Automatic Retransmission in Case of a Collision
- May Be Used with the NMSI Mode or a TDM Bus
- Delayed $\overline{\text{RTS}}$ Mode

7.10.18.2 HDLC BUS OPERATION. The following paragraphs detail the operation of the HDLC bus Controller.

7.10.18.2.1 Accessing the HDLC Bus. HDLC bus ensures an orderly access to the bus when two or more transmitters attempt to access the bus simultaneously. In such a case, one transmitter will always be successful in completing its transmission. This procedure relies upon the use of HDLC flags consisting of the binary pattern 01111110 ($7E) and the use of the zero bit insertion to prevent flag imitation.

While in the active condition (desiring to transmit), the HDLC bus controller will monitor the bus through the $\overline{\text{CTS}}$ pin. It counts the number of one bits using the $\overline{\text{CTS}}$ pin, and if a zero is detected, the internal counter is cleared.

Once 8 consecutive ones have been received, the HDLC bus controller will begin transmission on the line. While it is transmitting information on the bus, the transmitted data is continuously compared with the data actually on the bus. The $\overline{\text{CTS}}$ pin is used to sample the external bus.

Figure 7-56 shows how the $\overline{\text{CTS}}$ pin is used. The $\overline{\text{CTS}}$ sample is taken halfway through the bit time, using the rising edge of the transmit clock. If the transmitted bit is the same as the received $\overline{\text{CTS}}$ sample, the HDLC bus controller continues its transmission. If, however, the received CTS bit is zero, but the transmitted bit was 1, the HDLC controller ceases transmission following that bit and returns to the active condition. Since the HDLC bus uses a wired-OR scheme, a transmitted zero has priority over a transmitted one.

If the source address is included in the HDLC frame in addition to the destination address, a predefined priority of nodes will result. In addition, the inclusion of a source address will allow collisions to be detected no later than the end of the source address.

NOTE

HDLC bus can be used with many different HDLC-based frame formats. HDLC bus does not specify the type of HDLC protocol used.

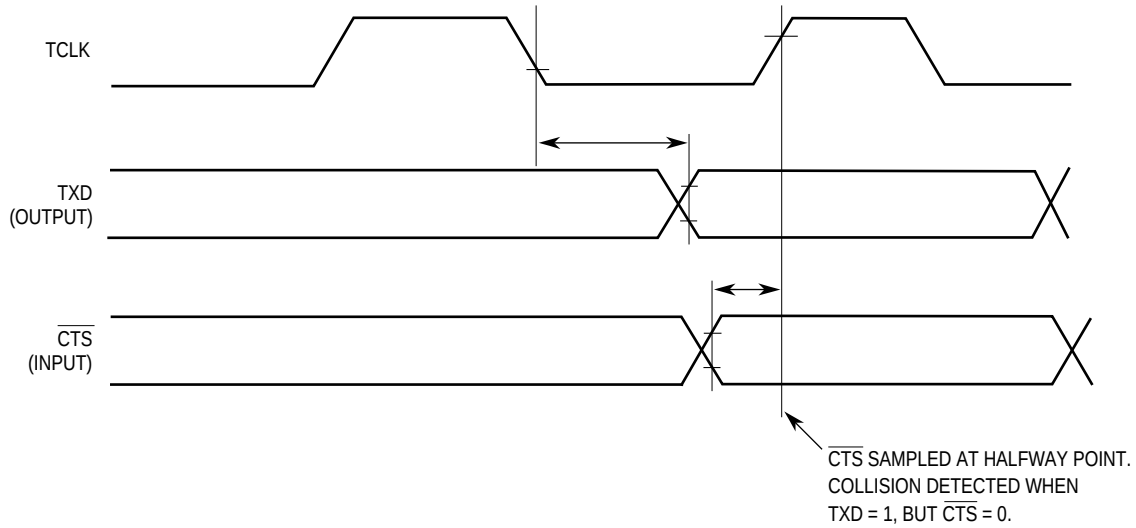


Figure 7-56. HDLC Bus Collision Detection

To ensure that all stations gain an equal share of the bus, a priority mechanism is also implemented in HDLC bus. Once an HDLC bus node has completed the transmission of a frame, it waits for 10 consecutive one bits, rather than just 8, before beginning the next transmission. In this way, all nodes desiring to transmit will obtain the bus, before a node transmits twice. Once a node detects that 10 consecutive ones have occurred on the bus, it may attempt transmission and can reinstate its original priority of waiting for 8 ones.

7.10.18.2.2 More Performance. Since HDLC bus is used in a wired-OR configuration, the limit of HDLC bus operation is determined by the rise time of the one bit.

Figure 7-57 shows a method to increase performance. The user supplies a clock that is high for a shorter duration than it is low, which allows more rise time in the case of a one bit.

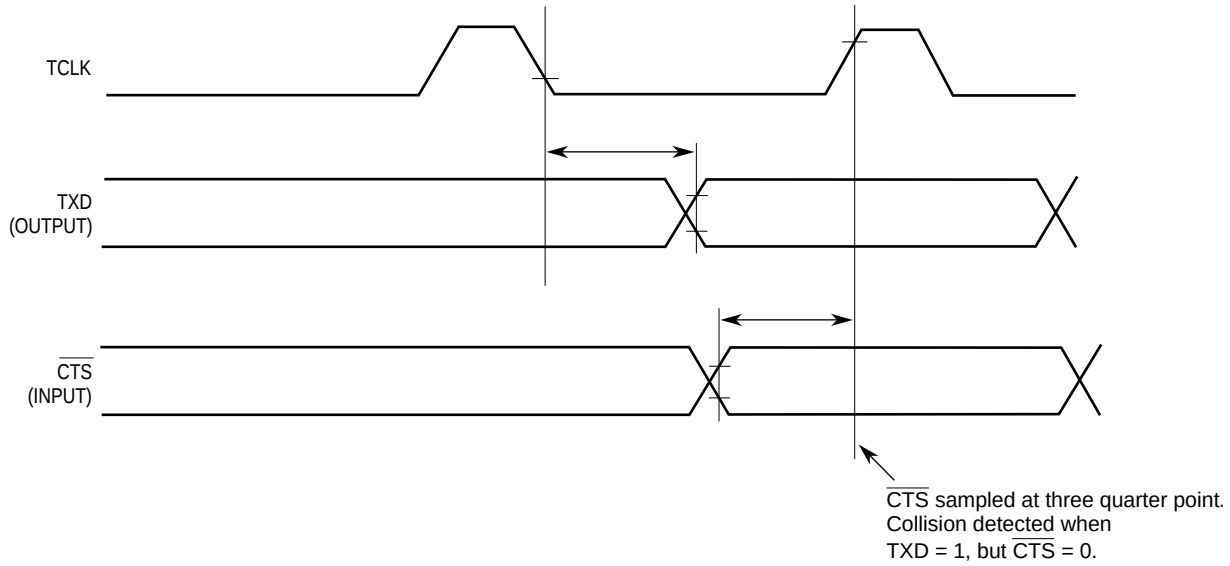
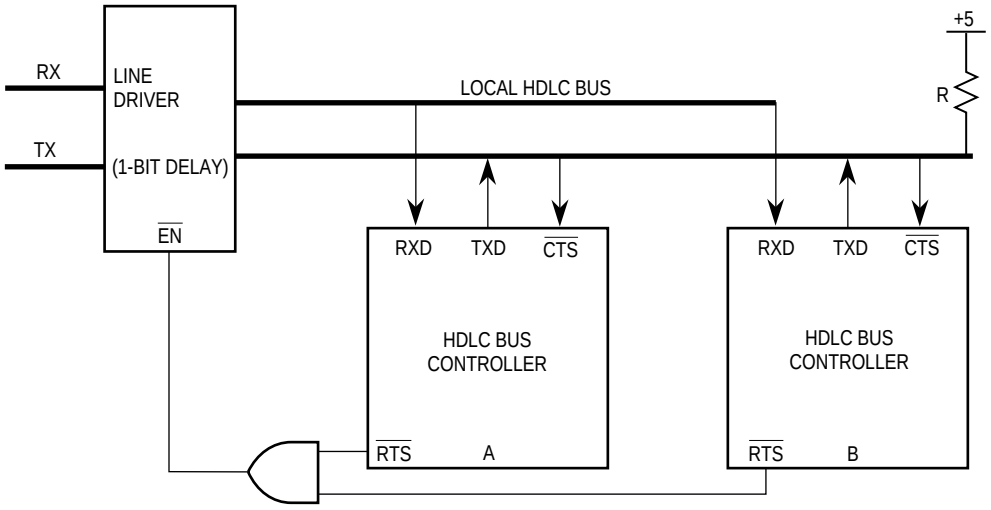


Figure 7-57. Non-Symmetrical Duty Cycle

7.10.18.2.3 Delayed RTS Mode. Sometimes HDLC bus may be used in a configuration having a local HDLC bus and a standard transmission line that is not an HDLC bus. Figure 7-58 illustrates such a case. The local HDLC bus controllers do not communicate with each other, but with a station on the transmission line; yet the HDLC bus protocol is used to control the access to the transmission line. In such a case, the $\overline{\text{RTS}}$ pin may be used as follows.



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 7-58. HDLC Bus Transmission Line Configuration

Normally, the $\overline{\text{RTS}}$ pin goes active at the beginning of the first bit of the opening flag. Use of $\overline{\text{RTS}}$ is not normally required in HDLC bus; however, a mode exists on the QUICC's HDLC bus that delays the $\overline{\text{RTS}}$ signal by one bit with respect to the data. This mode is selected with the BRM bit in the PSMR.

The delayed $\overline{\text{RTS}}$ mode is useful when the HDLC bus is used to connect multiple local nodes to a transmission line. If the transmission line driver has a one-bit delay, then the delayed $\overline{\text{RTS}}$ line can be used to enable the output of the transmission line driver. The result is that the transmission line bits always drive "clean" without any collisions occurring on them. The $\overline{\text{RTS}}$ timing is shown in Figure 7-59.

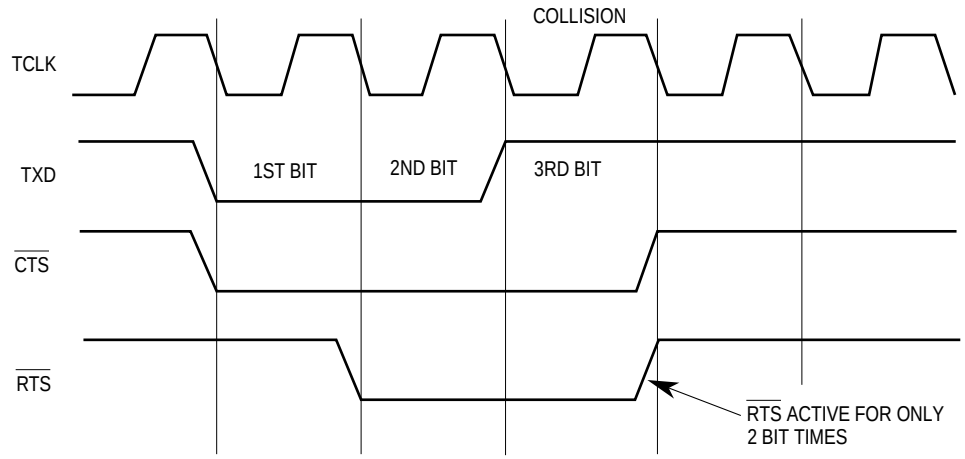
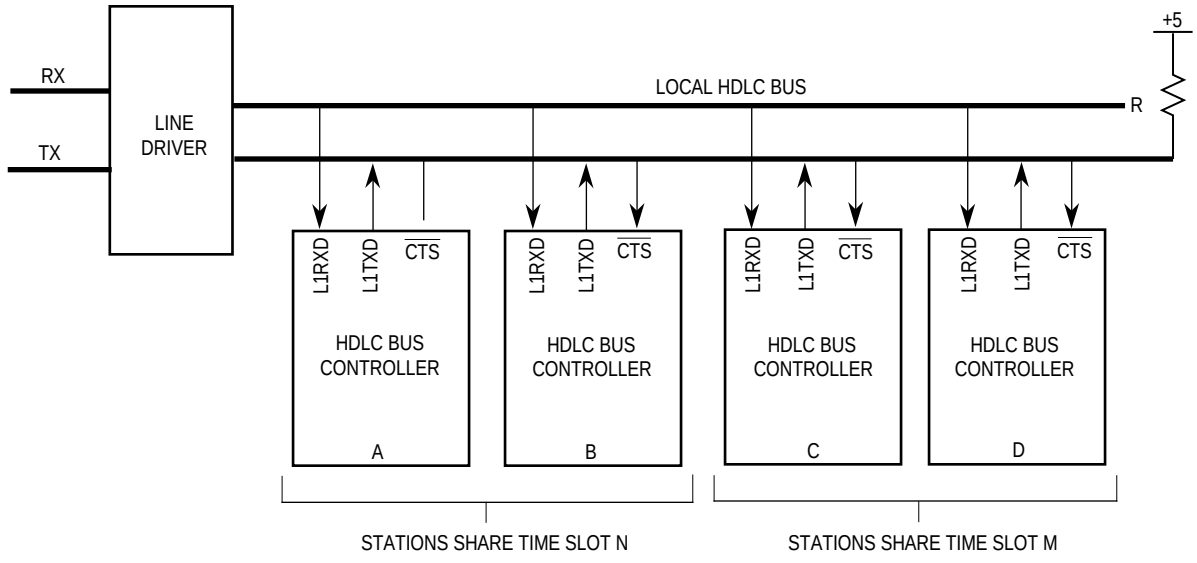


Figure 7-59. Delayed RTS Mode

7.10.18.2.4 Using the TSA. Sometimes HDLC bus may be used in a configuration that has a local HDLC bus, and a TDM transmission line that is not an HDLC bus. Figure 7-60 shows such a case. The local HDLC bus controllers all communicate over time slots; however, more than one HDLC bus controller is assigned to a given time slot, and the HDLC bus protocol is used to control access during that time slot.



NOTES:
1. All Tx 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 desired 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 7-60. HDLC Bus TSA Transmission Line Configuration

Once again, the local HDLC controllers do not communicate with each other, only with the transmission line. If the SCC is configured to operate using the TSA of the SI, then the data will be received and transmitted using the L1TXDx and L1RXDx pins. The collision sensing

is still obtained from the SCC's individual $\overline{\text{CTSx}}$ pin; thus, the $\overline{\text{CTS}}$ pin must be configured in port C to connect to the desired SCC. Since the SCC only receives clocks during its time slot, the $\overline{\text{CTS}}$ pin is only sampled during the transmit clock edges of the SCC's particular time slot.

7.10.18.3 HDLC BUS MEMORY MAP AND PROGRAMMING. HDLC bus on the QUICC is implemented using the HDLC controller with certain bits set. Otherwise, the user should consult the HDLC controller section for detailed information on the programming of HDLC.

7.10.18.3.1 GSMR Programming. The GSMR programming sequence is as follows:

1. Set the MODE bits to HDLC.
2. Set the ENT and ENR bits as desired.
3. Set the DIAG bits for normal operation.
4. Set the RDCR and TDCR bits for 1x clock.
5. Set the TENC and RENC bits for NRZ.
6. Clear RTSM.
7. Set CTSS to one and all other bits to zero or to their default condition.

7.10.18.3.2 PSMR Programming. The PSMR programming sequence is as follows:

1. Set the NOF bits as desired.
2. Set the CRC to 16-bit CRC CCITT.
3. Set the RTE bit.
4. Set the BUS bit.
5. Set the BRM bit to one or zero as desired.
6. Set all other bits to zero or to their default condition.

7.10.18.3.3 HDLC Bus Controller Example. Except for the previously discussed register programming, the HDLC Example #1 may be followed.

7.10.19 AppleTalk Controller

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, the AppleTalk protocols have traditionally been most closely associated with one particular physical and link layer protocol called LocalTalk.

The term LocalTalk refers to an HDLC-based link layer and physical layer protocol that runs at the rate of 230.4 kbps. In this document, the term AppleTalk controller refers to a support that the QUICC provides for the LocalTalk protocol.

The AppleTalk controller provides the required frame synchronization, bit sequence, preamble, and postamble onto standard HDLC frames. These capabilities, as well as the use of the HDLC controller in conjunction with the DPLL operating in FM0 mode, provide the proper connection formats to the LocalTalk bus.

NOTE

The MC68302 also provides the same general level of LocalTalk functionality when it is combined with its companion chip, the MC68195 LocalTalk Adaptor (LA). The LA device, however, is not required with the QUICC.

7.10.19.1 LOCALTALK BUS OPERATION. The following paragraphs detail the operation of the LocalTalk. A LocalTalk frame is a modified HDLC frame as shown in Figure 7-61.

SYNC SEQ	HDLC FLAGS	DEST. ADDR.	SOURCE ADDR.	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 7-61. LocalTalk Frame Format

First, a synchronization sequence of greater than three bits is sent. This sequence consists of at least one logical one bit, FM0 encoded, followed by greater than two bit times of line idle. No particular maximum time is specified for this line idle time. The idle time allows some LocalTalk equipment to sense 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/detection. Two bytes of address, destination and source, are transmitted next. This is followed by a byte of control and 0 to 600 data bytes. Next, two bytes of CRC are sent. The CRC is the common 16-bit CRC-CCITT polynomial referenced in the HDLC standard protocol. The LocalTalk frame is then terminated by a flag and a restricted HDLC abort sequence (a sequence of 12 to 18 logical ones). The transmitter's driver is then disabled.

The control byte within the LocalTalk frame indicates the type of frame. Control byte values from 0x01 to 0x7f are data frames, and control byte values from 0x80 to 0xff are control frames. Four different control frames are currently defined: ENQ (Enquiry), ACK (ENQ acknowledgement), RTS (request to send a data frame), and CTS (clear to send a data frame).

Frames are sent in groups known as dialogs. For instance, to transfer a data frame, three frames are actually sent over the network: an RTS frame (not to be confused with the RS-232 pin RTS) is sent requesting 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. Once a dialog has begun, other nodes cannot begin transmission until the dialog is complete. Dialogs are typically handled in software.

Frames within a dialog are transmitted 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 via software.

Due to the protocol definition, collisions should only be encountered during RTS and ENQ frames. Once a frame's transmission is started, it is fully transmitted, regardless of whether it collides with another frame. ENQ frames are infrequent, being sent only when a node is powered 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. FM0 requires one level transition on every bit boundary. If the value to be encoded is a logic zero, FM0 also requires a second transition in the middle of the bit time. The purpose of the FM0 encoding is to eliminate the need to transmit clocking information on a separate wire. With FM0, the clocking information is present whenever valid data is present.

7.10.19.2 APPLETALK CONTROLLER KEY FEATURES. The AppleTalk controller contains the following key features:

- Superset of the HDLC Controller Features
- Provides FM0 Encoding/Decoding
- Programmable Transmission of Sync Sequence
- Automatic Postamble Transmission
- Reception of Sync Sequence Does Not Cause Extra $\overline{CD}$ Interrupts
- Reception Automatically Disabled While Transmitting a Frame
- Transmit-on-Demand Feature Expedites Frames
- Connects Directly to RS-422 Transceiver

7.10.19.3 QUICC APPLETALK HARDWARE CONNECTION. The QUICC connects to LocalTalk as shown in Figure 7-62. The QUICC interfaces to the RS-422 transceiver through the TXD, $\overline{RTS}$, and RXD pins. The RS-422, in turn, interfaces to the LocalTalk connector. Although it is not shown, a passive RC circuit is recommended between the transceiver and the connector.

The 16x overspeed clock of 3.686 MHz may be generated from an external frequency source or from one of the baud rate generators if the resulting BRG output frequency is close to a multiple of the 3.686-MHz frequency (within the tolerance specified by LocalTalk).

The QUICC asserts the $\overline{RTS}$ signal for the complete duration of the frame; thus, $\overline{RTS}$ may be used to enable the RS-422 transmit driver.

7.10.19.4 APPLETALK MEMORY MAP AND PROGRAMMING MODEL. The AppleTalk controller on the QUICC is implemented using the HDLC controller with certain bits set. Otherwise, the user should consult 7.10.18 HDLC Bus Controller for detailed information on the programming of HDLC.

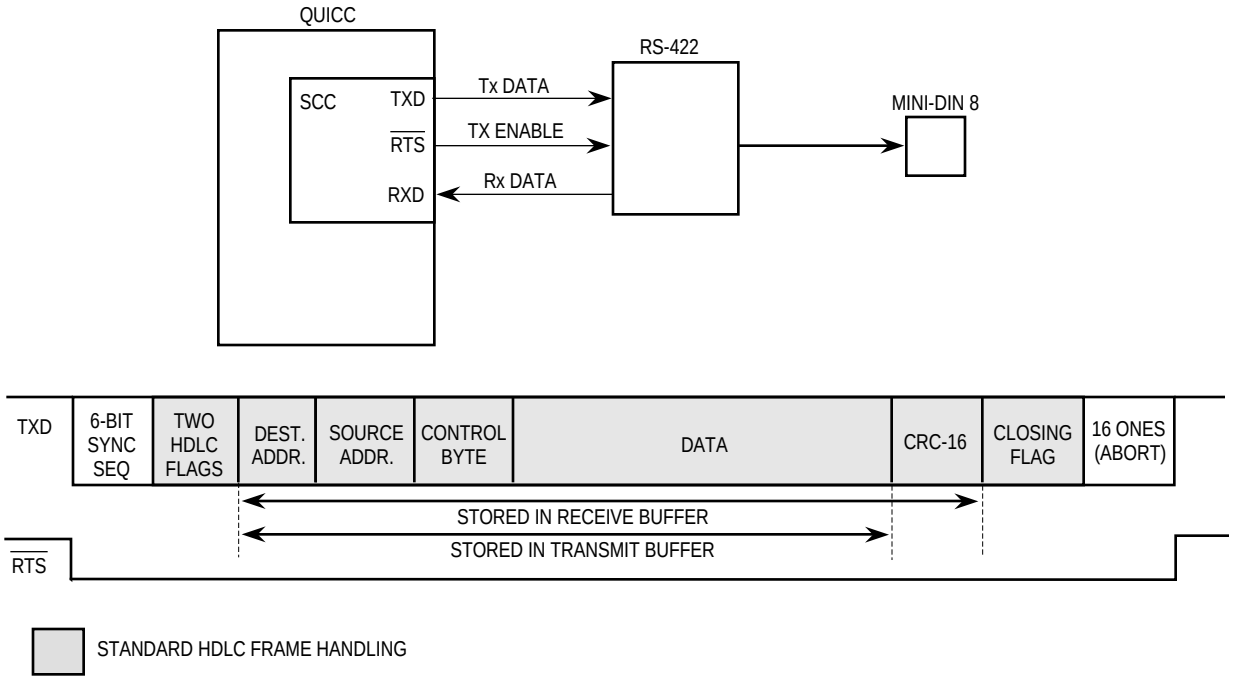


Figure 7-62. Connecting the QUICC to LocalTalk

7.10.19.4.1 GSMR Programming. The GSMR programming sequence is as follows:

1. The MODE bits should be set to AppleTalk.
2. The ENT and ENR bits should be set.
3. The DIG bits should be set for normal operation, with the $\overline{CD}$ and $\overline{CTS}$ pins grounded or with the $\overline{CD}$ and $\overline{CTS}$ pins configured for parallel I/O, which causes $\overline{CD}$ and $\overline{CTS}$ to be internally asserted to the SCC.
4. The RDCR and TDCR bits should usually be set to 16x clock.
5. The TENC and RENC bits should be set for FM0.
6. The Tend bit should be zero.
7. The TPP bits should be 11.
8. The TPL bits should be set to 000 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.
9. The TINV and RINV bits should be zero.
10. The TSNC bits should be set to 1.5 bit times 10.
11. The EDGE bits should be zero.
12. RTSM should be zero.
13. All other bits should be set to zero or to their default condition.

7.10.19.4.2 PSMR Programming. The PSMR programming sequence is as follows:

1. The NOF bits should be set to 0001 (binary) giving two flags before frames (one opening flag, plus one additional flag).
2. The CRC should be set to 16-bit CRC-CCITT.
3. The DRT bit should be set.
4. All other bits should be set to zero or to their default condition.

7.10.19.4.3 TODR Programming. To expedite a transmit frame, the transmit on demand register (TODR) may be used.

7.10.19.4.4 AppleTalk Controller Example. Except for the previously discussed register programming, the HDLC Example #1 may be followed.

7.10.20 BISYNC Controller

The byte-oriented binary synchronous communication (BISYNC) protocol was originated by IBM for use in networking products. The three classes of BISYNC frames are transparent, non-transparent with header, and non-transparent without header (see Figure 7-63). The transparent mode in BISYNC allows full binary data to be transmitted 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.

NOTE

The transparent frame type in BISYNC is not related to the totally transparent protocol supported by the QUICC. See 7.10.21 Transparent Controller for details.

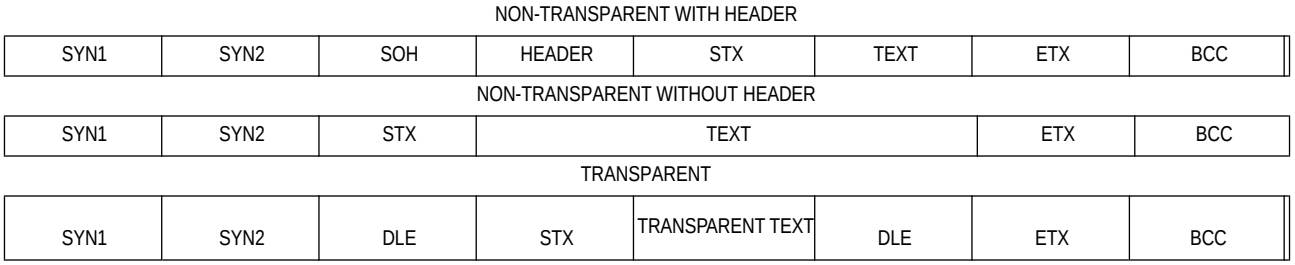


Figure 7-63. Typical BISYNC Frames

The bulk of the frame is divided into fields whose meaning depends on the frame type. The BCC is a 16-bit CRC (CRC16) format if 8-bit characters are used; it is a longitudinal check (a sum check) in combination with vertical redundancy check (parity) if 7-bit characters are used. In transparent operation, to allow the BISYNC control characters to be present in the frame as valid text data, a special character (DLE) is defined, which informs the receiver that the character following the DLE is a text character, not a control character. If a DLE is transmitted as valid data, it must be preceded by a DLE character. This technique is sometimes called byte-stuffing.

The physical layer of the BISYNC communications link must provide a means of synchronizing the receiver and transmitter. This is usually accomplished by sending at least one pair of synchronization characters prior to every frame

BISYNC is unusual in that a transmit underrun need not be an error. If an underrun occurs, the synchronization pattern is transmitted until data is once again ready to transmit. The receiver discards the additional synchronization characters as they are received. In non-transparent operation, all synchronization characters (SYNCs) are discarded. In transparent operation, all DLE-SYNC pairs are discarded. (Correct operation in this case assumes that, on the transmit side, the underrun does not occur between the DLE and its following character, a failure mode that is prevented in the QUICC.)

By appropriately setting the SCC mode register, any of the SCC channels may be configured to function as a BISYNC controller. The BISYNC controller handles the basic functions of the BISYNC protocol in normal mode and in transparent mode.

The SCC in BISYNC mode can work with the TSA or NMSI. The SCC can support modem lines by a connection to the port C pins or by using the general-purpose I/O pins.

The BISYNC controller consists of separate transmit and receive sections whose operations are asynchronous with the CPU32+ core and may be either synchronous or asynchronous with respect to the other SCCs.

7.10.20.1 BISYNC CONTROLLER FEATURES. The BISYNC controller contains the following key features:

- 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 Generation/Checking
- Parity (VRC) Generation/Checking
- Supports BISYNC Transparent Operation (Use of DLE Characters)
- Maintains Parity Error Counter
- Reverse Data Mode

7.10.20.2 BISYNC CHANNEL FRAME TRANSMISSION. The BISYNC transmitter is designed to work with almost no intervention from the CPU32+ core. When this CPU32+ core enables the BISYNC transmitter, it will start transmitting SYN1–SYN2 pairs (located in the data synchronization register) or idle as programmed in the BISYNC mode register. The BISYNC controller polls the first BD in the transmit channel's BD table. If there is a message

to transmit, the BISYNC controller will fetch the data from memory and start transmitting the message (after first transmitting the SYN1–SYN2 pair). The entire SYN1–SYN2 pair is always transmitted, regardless of the programming of the SYNL bits in the GSMR.

When a BD's data has been completely transmitted, L-bit is checked. If both the L-bit, and transmit BCS bit are set in that BD, the BISYNC controller will append the CRC16/LRC. Subsequently, the BISYNC controller writes the message status bits into the BD and clears the R-bit. It will then start transmitting SYN1–SYN2 pairs or idles as programmed in the RTSM bit in the GSMR. When the end of the current BD has been reached and the last bit is not set (working in multibuffer mode), only the R-bit 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 the transmission of each buffer, a specific buffer, or each block. The BISYNC controller will then proceed to the next BD in the table.

If no additional buffers have been presented to the BISYNC controller for transmission, an in-frame underrun is detected, and the BISYNC controller begins transmitting either SYNCs or idles. If the BISYNC controller was in transparent mode, the BISYNC controller transmits DLE-SYNC pairs.

Characters are included in the block check sequence (BCS) calculation on a per-buffer basis. Each buffer can be independently programmed to be included or excluded from the BCS calculation, and any characters to be excluded from the BCS calculation must reside in a separate buffer. The BISYNC controller can reset the BCS generator before transmitting a specific buffer. When functioning in transparent mode, the BISYNC controller automatically inserts a DLE before transmitting a DLE character. In this case, only one DLE is used in the calculation of the BCS.

7.10.20.3 BISYNC CHANNEL FRAME RECEPTION. Although the BISYNC receiver is designed to work with almost no intervention from the CPU32+ core, it allows user intervention on a per-byte basis if necessary. The BISYNC receiver can perform CRC16, longitudinal redundancy check (LRC), or vertical redundancy check (VRC) checking, SYNC stripping in normal mode, DLE-SYNC stripping and stripping of the first DLE in DLE-DLE pairs in transparent mode, and control character recognition. A control character is discussed in 7.10.20.6 BISYNC Control Character Recognition.

When the CPU32+ core enables the BISYNC receiver, it will enter hunt mode. In this mode, as data is shifted into the receiver shift register one bit at a time, the contents of the register are compared to the contents of the SYN1–SYN2 fields in the data synchronization register. If the two are not equal, the next bit is shifted in, and the comparison is repeated. When the registers match, the hunt mode is terminated, and character assembly begins. The BISYNC controller is now character synchronized and will perform SYNC stripping and message reception. The BISYNC controller will revert to the hunt mode when it is issued the ENTER HUNT MODE command, upon recognition of some error condition, or upon reception of an appropriately defined control character.

When receiving data, the BISYNC controller updates the BCS bit (CR) in the BD for every byte transferred. When the data buffer has been filled, the BISYNC controller clears the E-bit in the BD and generates an interrupt if the I-bit in the BD is set. If the incoming data

exceeds the length of the data buffer, the BISYNC controller will fetch the next BD in the table and, if it is empty, will continue to transfer data to this BD's associated data buffer.

When a BCS is received, it is checked and written to the data buffer. The BISYNC controller sets the last bit, writes the message status bits into the BD, and clears the E-bit. Then it generates a maskable interrupt, indicating that a block of data has been received and is in memory. Note that the SYNCs in the non-transparent mode or DLE-SYNC pairs in the transparent mode (i.e., an underrun condition) are not included in the BCS calculations.

NOTE

The receive FIFO width (RFW) bit in the GSMR must be set for an 8-bit receive FIFO for the BISYNC receiver.

7.10.20.4 BISYNC MEMORY MAP. When configured to operate in BISYNC mode, the QUICC overlays the structure listed in Table 7-5 with the BISYNC-specific parameters described in Table 7-9.

Table 7-9. BISYNC-Specific Parameters

Address	Name	Width	Description
SCC Base + 30	**RES**	Long	Reserved
SCC Base + 34	**CRCC**	Long	CRC Constant Temp Value
SCC Base + 38	**PRCRC**	Word	Preset Receiver CRC16/LRC
SCC Base + 3A	**PTCRC**	Word	Preset Transmitter CRC16/LRC
SCC Base + 3C	**PAREC**	Word	Receive Parity Error Counter
SCC Base + 3E	**BSYNC**	Word	BISYNC SYNC Character
SCC Base + 40	**BDLE**	Word	BISYNC DLE Character
SCC Base + 42	**CHARACTER1**	Word	CONTROL Character 1
SCC Base + 44	**CHARACTER2**	Word	CONTROL Character 2
SCC Base + 46	**CHARACTER3**	Word	CONTROL Character 3
SCC Base + 48	**CHARACTER4**	Word	CONTROL Character 4
SCC Base + 4A	**CHARACTER5**	Word	CONTROL Character 5
SCC Base + 4C	**CHARACTER6**	Word	CONTROL Character 6
SCC Base + 4E	**CHARACTER7**	Word	CONTROL Character 7
SCC Base + 50	**CHARACTER8**	Word	CONTROL Character 8
SCC Base + 52	**RCCM**	Word	Receive Control Character Mask

NOTE: Boldfaced items should be initialized by the user.

PRCRC and PTCRC. These value should be preset to all ones or all zeros, depending on the BCS used.

PAREC. This 16-bit (modulo 2^{16}) counter is maintained by the CP. It may be initialized by the user while the channel is disabled. The counter counts parity errors on receive if the parity feature of BISYNC is enabled.

BSYNC. This register contains the value of the SYNC to be transmitted in an underrun condition, transmitted as the second byte of a DLE-SYNC pair, and stripped from incoming data on receive once the receiver has synchronized to the data using the DSR and SYN1–SYN2 pair.

BDLE. This register contains the value to be transmitted as the first byte of a DLE-SYNC pair and stripped on receive.

CHARACTER1–8. These values represent control characters that may be recognized by the BISYNC controller.

RCCM. This value is used to mask the comparison of CHARACTER1–8 so that classes of control characters may be defined. A one enables the bit comparison and a zero masks it.

The CPU32+ core configures each SCC to operate in one of the protocols by the MODE bits in the GSMR. The SYN1–SYN2 synchronization characters are programmed in the data synchronization register.

The BISYNC controller uses the same basic data structure as that used in the other modes. Receive and transmit errors are reported through their respective BDs. The status of the line is reflected via the port C pins, and a maskable interrupt can be generated upon each status change.

There are two basic ways of handling the BISYNC channels. First, data may be inspected on a per-byte basis, with the BISYNC controller interrupting the CPU32+ core upon receipt of every byte of data. Second, the BISYNC controller may be operated so that software is only necessary for handling the first two to three bytes of data; subsequent data (until the end of the block) can be handled by the BISYNC controller without interrupting the CPU32+ core.

7.10.20.5 BISYNC COMMAND SET. The following transmit and receive commands are issued to the CR.

7.10.20.5.1 Transmit Commands. The following paragraphs describe the BISYNC transmit commands.

STOP TRANSMIT Command*.* After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the transmit enable mode and starts polling the first BD in the table every 64 transmit clocks (immediately if the TOD bit in the TODR is set).

The STOP TRANSMIT command aborts transmission after a maximum of 64 additional bits are transmitted, without waiting until the end of the buffer is reached, and the transmit FIFO is flushed. The TBPTR is not advanced. No new BD is accessed, and no new buffers are transmitted for this channel. SYNC characters consisting of SYNC-SYNC or DLE-SYNC pairs (according to the transmitter mode) will be continually transmitted until transmission is reenabled by issuing the RESTART TRANSMIT command. The STOP TRANSMIT command may be used when it is necessary to abort transmission and transmit an EOT control

sequence. The EOT sequence should be the first buffer presented to the BISYNC controller for transmission after re-enabling transmission.

NOTE

The BISYNC controller will remain in the transparent or normal mode after receiving the STOP TRANSMIT or RESTART TRANSMIT commands.

GRACEFUL STOP TRANSMIT Command*.* The channel GRACEFUL STOP TRANSMIT command is used to stop transmission in an orderly way rather than abruptly, as performed by the regular STOP TRANSMIT command. It stops transmission after the current frame has completed transmission, or immediately if there is no frame being transmitted. The GRA bit in the SCCE will be set once transmission has stopped. After transmission ceases, the BISYNC transmit parameters, including BDs, may be modified. The TBPTR will point to the next Tx BD in the table. Transmission will begin once the R-bit of the next BD is set and the RESTART TRANSMIT command is issued.

RESTART TRANSMIT Command*.* The RESTART TRANSMIT command enables the transmission of characters on the transmit channel. This command is expected by the BISYNC controller after a STOP TRANSMIT command, after a STOP TRANSMIT command and disabling the channel in its SCC mode register, after a GRACEFUL STOP TRANSMIT command, or after a transmitter error (underrun or CTS lost). The BISYNC controller will resume transmission from the current TBPTR in the channel's Tx BD table.

INIT TX PARAMETERS Command*.* Initializes all the transmit parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the transmitter is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both transmit and receive parameters.

7.10.20.5.2 Receive Commands. The following paragraphs describe the BISYNC receive commands.

RESET BCS CALCULATION Command*.* The RESET BCS CALCULATION command resets the receive BCS accumulator immediately. For example, it may be used to reset the BCS after recognizing a control character (such as SOH), signifying that a new block is commencing.

ENTER HUNT MODE Command**.** After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the receive enable mode and will use the first BD in the table.

The ENTER HUNT MODE command is used to force the BISYNC controller to abort reception of the current block and enter the hunt mode. In the hunt mode, the BISYNC controller continually scans the input data stream for the SYN1–SYN2 sequence as programmed in the data synchronization register. After receiving the command, the current receive buffer is closed, and the BCS is reset. Message reception continues using the next BD.

CLOSE Rx BD Command**.** The CLOSE Rx BD command is used to force the SCC to close the current Rx BD, if it is currently being used, and to use the next BD for any subsequent data that is received. If the SCC is not in the process of receiving data, no action is taken by this command.

INIT RX PARAMETERS Command**.** This command initializes all the receive parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the receiver is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both receive and transmit parameters.

7.10.20.6 BISYNC CONTROL CHARACTER RECOGNITION. The BISYNC controller can recognize special control characters. These characters are used to customize the BISYNC protocol implemented by the BISYNC controller and may be used to aid its operation in a DMA-oriented environment. Their main use is for receive buffers longer than one byte. In single-byte buffers, each byte can easily be inspected, and control character recognition should be disabled.

The purpose of the control characters table is to enable automatic recognition (by the BISYNC controller) of the end of the current block. Since the BISYNC controller imposes no restrictions on the format of the BISYNC blocks, user software must respond to the received characters and inform the BISYNC controller of mode changes and certain protocol events (e.g., resetting the BCS). However, correct use of the control characters table allows the remainder of the block to be received without interrupting the user software.

Up to 8 control characters may be defined. These characters inform the BISYNC controller that the end of the current block has been reached and whether a BCS is expected following this character. For example, the end of text (ETX) character 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, a BCS expected bit, and a hunt mode bit. The RCCM entry is used to define classes of control characters with a masking option.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	B	H						CHARACTER1							
OFFSET + 2	E	B	H						CHARACTER2							
OFFSET + 4	E	B	H						CHARACTER3							
OFFSET + 6	E	B	H													
•																
•																
OFFSET + E	E	B	H						CHARACTER8							
OFFSET + 10	1	1	1						MASK VALUE(RCCM)							

E—End of Table

0 = This entry is valid. The lower eight bits will be checked against the incoming character.
1 = The entry is not valid. No valid entries exist beyond this entry.

NOTE

In tables with 8 control characters, the E-bit should be zero in all eight positions.

B—BCS Expected

0 = The character is written into the receive buffer. The buffer is then 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.

NOTE

A maskable interrupt is generated after the buffer is closed.

H—HUNT MODE

0 = The BISYNC controller will maintain character synchronization after closing this buffer.
1 = The BISYNC controller will enter hunt mode after closing the buffer. When the B bit is set, the controller will enter hunt mode after the reception of the BCS.

CHARACTER1–8—Control Character Value

These fields define control characters.

NOTE

When using 7-bit characters with parity, the parity bit should be included in the control character value.

RCCM—Received Control Character Mask

The value in this register is used to mask the comparison of CHARACTER1–8. The lower eight bits of RCCM correspond to the lower eight bits of CHARACTER1–8, and are decoded as follows.

0 = Mask this bit in the comparison of the incoming character and CHARACTER1–8.
1 = The address comparison on this bit proceeds normally. No masking occurs.

NOTE

Bits 15 through 13 of RCCM must be set, or erratic operation may occur during the control character recognition process.

7.10.20.7 BSYNC-BISYNC SYNC REGISTER. The 16-bit, memory-mapped, read-write BSYNC register is used to define the BISYNC stripping and insertion of the SYNC character. When an underrun occurs during message transmission, the BISYNC controller will insert SYNC characters until the next data buffer is available for transmission. When the BISYNC

receiver is not in hunt mode and a SYNC character has been received, the receiver will discard this character if the valid bit is set.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
V	0	0	0	0	0	0	0	SYNC							

NOTE

When using 7-bit characters with parity, the parity bit should be included in the SYNC register value.

7.10.20.8 BDLE-BISYNC DLE REGISTER. The 16-bit, memory-mapped, read-write BDLE register is used to define the BISYNC stripping and insertion of the DLE character. When the BISYNC controller is in transparent mode and an underrun occurs during message transmission, the BISYNC controller inserts DLE-SYNC pairs until the next data buffer is available for transmission.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
V	0	0	0	0	0	0	0	DLE							

When the BISYNC receiver is in transparent mode and a DLE character is received, the receiver discards this character and excludes it from the BCS if the valid bit is set. If the second (next) character is a SYNC character, the BISYNC controller discards it and excludes it from the BCS. If the second character is a DLE, the BISYNC controller will write it to the buffer and include it in the BCS. If the character is not a DLE or SYNC, the BISYNC controller will examine the control characters table and act accordingly. If the character is not in the table, the buffer will be closed with the DLE follow character error (DLE) bit set. If the valid bit is not set, the receiver will treat the character as a normal character.

NOTE

When using 7-bit characters with parity, the parity bit should be included in the DLE register value.

7.10.20.9 TRANSMITTING AND RECEIVING THE SYNCHRONIZATION SEQUENCE. The BISYNC channel can be programmed to transmit and receive a synchronization pattern. The pattern is defined in the DSR. The length of the SYNC pattern is defined in the SYNL bits in the GSMR. The receiver synchronizes on the synchronization pattern that is located in the DSR. If the SYNL bits specify a non-zero synchronization pattern, then the transmitter sends the entire contents of the DSR prior to each frame, starting with the LSB first. Thus, the user may wish to repeat the desired SYNC pattern in the other DSR bits as well.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4-BIT SYNC															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8-BIT SYNC															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
16-BIT SYNC															

7.10.20.10 BISYNC ERROR-HANDLING PROCEDURE. The BISYNC controller reports message reception and transmission error conditions using the channel BDs, the error counters, and the BISYNC event register. The modem interface lines can also be directly monitored via the port C pins.

7.10.20.10.1 Transmission Errors. The following paragraphs describe various types of BISYNC transmission errors.

Transmitter Underrun**.** When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the UN bit in the BD, and generates the TXE interrupt (if enabled). The channel resumes transmission after the reception of the RESTART TRANSMIT command. Underrun cannot occur between frames or during a DLE-XXX pair in transparent mode.

CTS Lost During Message Transmission**.** When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the CTS lost bit in the BD, and generates the TXE interrupt (if enabled). The channel will resume transmission after reception of the RESTART TRANSMIT command.

7.10.20.10.2 Reception Errors. The following paragraphs describe various types of BISYNC reception errors.

Overrun Error**.** The BISYNC controller maintains an internal FIFO for receiving data. The CP begins programming the SDMA channel (if the data buffer is in external memory) and updating the CRC when the first byte is received into the FIFO. If a FIFO overrun occurs, the BISYNC controller writes the received data byte to the internal FIFO over the previously received byte. The previous character and its status bits are lost. Following this, the channel closes the buffer, sets the OV-bit in the BD, and generates the RX interrupt (if enabled). The receiver then enters hunt mode immediately.

CD Lost During Message Reception**.** When this error occurs, the channel terminates message reception, closes the buffer, sets the carrier detect lost bit in the BD, and generates the RX interrupt (if enabled). This error has the highest priority; the rest of the message is lost, and no other errors are checked in the message. The receiver then enters hunt mode immediately.

Parity Error**.** When this error occurs, the channel writes the received character to the buffer and sets the PR bit in the BD. The channel terminates message reception, closes the buffer, sets the PR bit in the BD, and generates the RX interrupt (if enabled). The channel also increments the PAREC, and the receiver enters hunt mode immediately.

CRC Error**.** The channel updates the CR bit in the BD every time a character is received with a byte delay (eight serial clocks) between the status update and the CRC calculation. When using control character recognition to detect the end of the block and cause the checking of the CRC that follows, the channel closes the buffer, sets the CR bit in the BD, and generates the RX interrupt (if enabled).

7.10.20.11 BISYNC MODE REGISTER (PSMR). Each BISYNC mode register is a 16-bit, memory-mapped, read-write register that controls SCC operation. The term BISYNC mode

register refers to the PSMR of the SCC when that SCC is configured for BISYNC mode. This register is cleared at reset. Some of the PSMR bits can be modified on the fly (i.e., while the receiver and transmitter are enabled).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOS				CRC		RBCS	RTR	RVD	DRT	—		RPM		TPM	

NOS—Minimum Number of SYNCs Between or Before Messages

If NOS3–NOS0 = 0000, then 1 SYN1–SYN2 pair will be transmitted; if NOS3–NOS0 = 1111, then 16 SYN1–SYN2 pairs will be transmitted. The SYN1–SYN2 pair is defined in the DSR. The entire SYN1–SYN2 pair will always be transmitted regardless of the setting of the SYNL bits in the GSMR. The NOS bits may be modified on the fly.

CRC—CRC Selection

00 = Reserved.
01 = CRC16 (BISYNC). (X16 + X15 + X2 + 1). The PRCRC and PTCRC registers should be initialized to a preset value of all zeros or all ones before the channel is enabled. In both cases, the transmitter sends the calculated CRC non-inverted, and the receiver checks the CRC against zero. Eight-bit data characters (without parity) are configured when CRC16 is chosen.
10 = Reserved
11 = LRC (sum check). (BISYNC). For even LRC, the PRCRC and PTCRC registers should be initialized to zero before the channel is enabled. For odd LRC, the PR-CRC and PTCRC registers should be initialized to ones.
The receiver will check character parity when BCS is programmed to LRC and the receiver is not in transparent mode. The transmitter will transmit character parity when BCS is programmed to LRC and the transmitter is not in transparent mode. Use of parity in BISYNC assumes the use of 7-bit data characters.

RBCS—Receive Block Check Sequence

The BISYNC receiver internally stores two BCS calculations with a byte delay (eight serial clocks) between them. This enables the user to examine a received data byte and then decide whether or not it should be part of the BCS calculation. This is useful when control character recognition and stripping are to be performed in software. The bit should be set (or reset) within the time taken to receive the following data byte. When this bit is reset, the BCS calculations exclude the latest fully received data byte. When RBCS is set, the BCS calculations continue normally.

0 = Disable receive BCS
1 = Enable receive BCS

RTR—Receiver Transparent Mode

0 = The receiver is placed in normal mode with SYNC stripping and control character recognition operative.
1 = The receiver is placed in transparent mode. SYNCs, DLEs, and control characters are only recognized after a leading DLE character. The receiver will calculate the CRC16 sequence, even if it is programmed to LRC while in transparent mode. PR-CRC should be initialized to the CRC16 preset value before setting this bit.

RVD—Reverse Data

0 = Normal operation.
1 = Any portion of this SCC that is defined to operate in BISYNC mode (either the receiver or transmitter or both) will operate by reversing the character bit order, transmitting the MSB first.

DRT—Disable Receiver While Transmitting

0 = Normal operation.
1 = While data is being transmitted by the SCC, the receiver is disabled, being gated by the internal $\overline{RTS}$ signal. This is useful if the BISYNC channel is being configured onto a multidrop line, and the user does not wish to receive his own transmission. Note that although BISYNC is usually implemented as a half-duplex protocol, the receiver is not actually disabled during transmission. Thus, for typical BISYNC operation, DRT should not be set.

Bits 5–4—Reserved

RPM—Receiver Parity Mode

The RPM bits select the type of parity check to be performed by the receiver. The RPM bits can be modified on the fly. The RPM bits are ignored unless the CRC bits are selected to be LRC.

00 = Odd Parity
01 = Low Parity (always check for a zero in the parity bit position)
10 = Even Parity
11 = High Parity (always check for a one in the parity bit position)

When odd parity is selected, the transmitter will count the number of ones in the data word. If the total number of ones is not an odd number, the parity bit is set to one and thus produces an odd number. If the receiver counts an even number of ones, an error in transmission has occurred. In the same manner, for even parity, an even number must result from the calculation performed at both ends of the line. In high/low parity, if the parity bit is not high/low, a parity error is reported. The receive parity errors cannot be disabled, but can be ignored if desired.

TPM—Transmitter Parity Mode

The TPM bits select the type of parity to be performed by the transmitter. The TPM bits can be modified on the fly. The TPM bits are ignored unless the CRC bits are selected to be 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)

7.10.20.12 BISYNC RECEIVE BUFFER DESCRIPTOR (RX BD). The CP reports information about the received data for each buffer using BDs. The CP closes the current buffer,

generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:

1. Receiving a user-defined control character
2. Detecting an error
3. Detecting a full receive buffer
4. Issuing the ENTER HUNT MODE command
5. Issuing the CLOSE Rx BD command

212	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**E**	—	**W**	I	L	F	**CM**	—	DE	—	—	NO	—	CR	OV	CD
OFFSET + 2	DATA LENGTH															
OFFSET + 4	RX DATA BUFFER POINTER															
OFFSET + 6																

NOTE: Entries in boldface must be initialized by the user.

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 CPU32+ 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 Rx 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 CPU32+ core should not write any fields of this Rx BD.

Bits 14, 8, 6, 5—Reserved

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.

I—Interrupt

0 = No interrupt is generated after this buffer has been used.

1 = The RX bit in the BISYNC event register will be set when this buffer has been closed by the BISYNC controller. The RX bit can cause an interrupt if it is enabled.

L—Last in Frame

This bit is set by the transparent controller when this buffer is the last in a frame. This implies the negation of CD in envelope mode or the reception of an error, in which

case one or more of the OV, CD, and DE bits are set. the transparent controller will write the number of frame octets to the data length field.

0 = The buffer is not the first in a frame.
1 = The buffer is the first in a frame

F—First in Frame

This bit is set by the transparent controller when this buffer is the first in a frame.

0 = The buffer is not the last in a frame.
1 = The buffer is the last in a frame

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 overwritten automatically when the CP next accesses this BD. However, the E-bit will be cleared if an error occurs during reception, regardless of the CM bit.

DE—DPLL Error

This bit is set by the BISYNC controller when a DPLL error has occurred during the reception of this buffer. In decoding modes where a transition is promised every bit, the DPLL error will be set when a missing transition has occurred.

OV—Overrun

A receiver overrun occurred during message reception.

CD—Carrier Detect Lost

The carrier detect signal was negated during message reception.

Data Length

The data length is the number of octets that the CP has written into this BD's data buffer, including the BCS (if selected). In BISYNC mode, the data length should initially be set to zero by the user; it is incremented each time a received character is written to the data buffer.

NOTE

The actual amount of memory allocated for this buffer should be greater than or equal to the contents of the MRBLR.

Rx Data Buffer Pointer

The receive 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.

7.10.20.13 BISYNC TRANSMIT BUFFER DESCRIPTOR (TX BD). Data is presented to the CP for transmission on an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The CP confirms transmission or indicates error conditions using the BDs to inform the processor that the buffers have been serviced.

The status and control bits are prepared by the user before transmission and are set by the CP after the buffer has been transmitted.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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																

NOTE: Entries in boldface must be initialized by the user.

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.

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 receive incoming data into the first BD in the table (the BD pointed to by TBASE). The number of Tx BDs in this table is programmable, and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.

1 = Either TX or TXE in the BISYNC event register will be set when this buffer has been serviced by the CP, which can cause an interrupt.

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 will enter (remain in) normal mode after sending the last character in the buffer and the BCS (if enabled).

TB—Transmit BCS

This bit is valid only when the L-bit is set.

0 = Transmit the SYN1–SYN2 sequence or idle (according to the RTSM bit in the GSMR) after the last character in the buffer.

1 = Transmit the BCS sequence after the last character. The BISYNC controller will also reset the BCS generator after transmitting the BCS.

CM—Continuous Mode

0 = Normal operation.

1 = The R-bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this BD.

However, the R-bit will be cleared if an error occurs during transmission, regardless of the CM bit.

BR—BCS Reset

0 = The transmitter BCS accumulation is not reset.
1 = The transmitter BCS accumulation is reset (used for STX or SOH) before sending the data buffer.

TD—Transmit DLE

0 = No automatic DLE transmission is to occur before the data buffer.
1 = The transmitter will transmit a DLE character before sending the data buffer, which saves writing the first DLE to a separate data buffer when working in transparent mode. See the TR bit for information on control characters.

TR—Transparent Mode

0 = The transmitter will enter (remain in) the normal mode after sending the data buffer. In this mode, the transmitter will automatically insert SYNCs in an underrun condition.
1 = The transmitter enters or remains in transparent mode after sending the data buffer. In this mode, the transmitter automatically inserts DLE-SYNC pairs in the underrun condition. Underrun occurs when the BISYNC controller finishes a buffer with the L-bit set to zero and the next BD is not available. The transmitter also checks all characters before sending them; if a DLE is detected, another DLE is automatically sent. The user must insert a DLE or program the BISYNC controller to insert it (using TD) before each control character required. The transmitter will calculate the CRC16 BCS even if the BCS bit in the BISYNC mode register is programmed to LRC. The PTCRC should be initialized to CRC16 before setting this bit.

B—BCS Enable

0 = Buffer consists of characters to be excluded from the BCS accumulation.
1 = Buffer consists of characters to be included in the BCS accumulation.

The following status bits are written by the CP after it has finished transmitting the associated data buffer.

UN—Underrun

The BISYNC controller encountered a transmitter underrun condition while transmitting the associated data buffer.

CT—CTS Lost

CTS was lost during message transmission.

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. The data length should be greater than zero.

Tx Data Buffer Pointer

The transmit buffer pointer, which always points to the first byte of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory.

7.10.20.14 BISYNC EVENT REGISTER (SCCE). The SCCE is called the BISYNC event register when the SCC is operating as a BISYNC controller. It is a 16-bit register used to report events recognized by the BISYNC channel and to generate interrupts. On recognition of an event, the BISYNC controller will set the corresponding bit in the BISYNC event register. Interrupts generated by this register may be masked in the BISYNC mask register.

The BISYNC event register is a memory-mapped register that may be read at any time. A bit is reset by writing a one (writing a zero does not affect a bit's value). More than one bit may be reset at a time. All unmasked bits must be reset before the CP will negate the internal interrupt request signal. This register is cleared at reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—			GLr	GLt	DCC	—	—	GRA	—	—	TXE	RCH	BSY	TX	RX

Bits 15–13, 9, 8, 6, 5—Reserved

GLr—Glitch on Rx

A clock glitch was detected by this SCC on the receive clock.

GLt—Glitch on Tx

A clock glitch was detected by this SCC on the transmit clock.

DCC—DPLL CS Changed

The carrier sense status as generated by the DPLL has changed state. The real-time status may be found in SCCS. This is not the $\overline{CD}$ pin status that is discussed elsewhere; it is only valid when the DPLL is used.

GRA—Graceful Stop Complete

A graceful stop, which was initiated by the GRACEFUL STOP TRANSMIT command, is now complete. This bit is set as soon the transmitter has finished transmitting any mes-

sage that was in progress when the command was issued. It will be set immediately if no message was in progress when the command was issued.

TXE—Tx Error

An error (CTS lost or underrun) occurred on the transmitter channel.

RCH—Receive Character

A character has been received and written to the buffer.

BSY—Busy Condition

A character was received and discarded due to lack of buffers. The receiver will resume reception after an ENTER HUNT MODE command.

TX—Tx Buffer

A buffer has been transmitted. This bit is set as the last bit of data or the BCS (if sent) begins transmission.

RX—Rx Buffer

A receive buffer has been closed by the CP on the BISYNC channel.

7.10.20.15 BISYNC MASK REGISTER (SCCM). The SCCM is referred to as the BISYNC mask register when the SCC is operating as a BISYNC controller. It is a 16-bit read-write register that has the same bit format as the BISYNC event register. If a bit in the BISYNC mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

7.10.20.16 SCC STATUS REGISTER (SCCS). The SCCS is an 8-bit read-only register that allows the user to monitor real-time status conditions on the RXD line. The real-time status of the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins are part of the port C parallel I/O.

7	6	5	4	3	2	1	0
—	—	—	—	—	—	CS	—

CS—Carrier Sense (DPLL)

This bit shows the real-time carrier sense of the line as determined by the DPLL if it is used.

0 = The DPLL does not sense a carrier.
1 = The DPLL does sense a carrier.

7.10.20.17 PROGRAMMING THE BISYNC CONTROLLER. There are two general techniques that the software may employ to handle data received by the BISYNC controllers. The simplest way is to allocate single-byte receive buffers, request (in the status word in each BD) an interrupt on reception of each buffer (i.e., byte), and implement the BISYNC protocol entirely in software on a byte-by-byte basis. This simple approach is flexible and may be adapted to any BISYNC implementation. The obvious penalty is the overhead caused by interrupts on each received character.

A more efficient method is as follows. Multibyte buffers are prepared and linked to the receive buffer table. Software is used to analyze the first two to three bytes of the buffer to determine what type of block is being received. When this has been determined, reception can continue without further intervention from the user's software until a control character is encountered. The control character signifies the end of the block, causing the software to revert back to a byte-by-byte reception mode.

To accomplish this, the RCH bit in the BISYNC mask register should initially be set, enabling an interrupt on every byte of data received to allow software to analyze the type of block being received on a byte-by-byte basis. After analyzing the initial characters of a block, the user should either set the RTR bit in the BISYNC mode register or issue the RESET BCS CALCULATION command. For example, if DLE-STX is received, transparent mode should be entered. By setting the appropriate bit in the BISYNC mode register, the BISYNC controller automatically strips the leading DLE from <DLE-character> sequences. Thus, control characters are only recognized when they follow a DLE character. The RTR bit should be cleared after a DLE-ETX is received.

Alternatively, after receiving an SOH, the RESET BCS CALCULATION command should be issued. This command causes the SOH to be excluded from BCS accumulation and the BCS to be reset. Note that the RBCS bit in the BISYNC mode register (used to exclude a character from the BCS calculation) is not needed here since SYNCs and leading DLEs (in transparent mode) are automatically excluded by the BISYNC controller.

After recognizing the type of block above, the RCH interrupt should be masked. Data reception then continues without further interruption of the CPU32+ core until the end of the current block is reached. This is defined by the reception of a control character matching that programmed in the receive control characters table.

The control characters table should be set to recognize the end of the block as follows:

Control Characters	E	B	H
ETX	0	1	1
ITB	0	1	0
ETB	0	1	1
ENQ	0	0	0
Next Entry	0	X	X

After the end of text (ETX), a BCS is expected; then the buffer should be closed. Hunt mode should be entered when the line turnaround occurs (BISYNC is normally half-duplex). ENQ characters are used to abort transmission of a block. For the receiver, the ENQ character designates the end of the block, but no CRC is expected.

Following control character reception (i.e., end of the block), the RCH bit in the BISYNC mask register should be set, reenabling interrupts for each byte of received data.

7.10.20.18 SCC BISYNC EXAMPLE. The following list is an initialization sequence for an SCC BISYNC channel assuming an external clock is provided. SCC4 is used. The BISYNC

controller is configured with the $\overline{\text{RTS4}}$, $\overline{\text{CTS4}}$, and $\overline{\text{CD4}}$ pins active. The CLK7 pin is used for both the BISYNC receiver and transmitter.

1. The SDCR (SDMA Configuration Register) should be initialized to $0740, rather than being left at its default value of $0000.
2. Configure the port A pins to enable the TXD4 and RXD4 pins. Write PAPAR bits 6 and 7 with ones. Write PADIR bits 6 and 7 with zeros. Write PAODR bits 6 and 7 with zeros.
3. Configure the port C pins to enable $\overline{\text{RTS4}}$, $\overline{\text{CTS4}}$, and $\overline{\text{CD4}}$. Write PCPAR bit 3 with one and bits 10 and 11 with zeros. Write PCDIR bits 3, 10, and 11 with zeros. Write PCSO bits 10 and 11 with ones.
4. Configure port A to enable the CLK7 pin. Write PAPAR bit 14 with a one. Write PADIR bit 14 with a zero.
5. Connect the CLK7 pin to SCC4 using the SI. Write the R4CS bits in SICR to 110. Write the T4CS bits in SICR to 110.
6. Connect the SCC4 to the NMSI (i.e., its own set of pins). Clear the SC4 bit in the SICR.
7. Write $0740 to the SDCR to initialize the SDMA Configuration Register.
8. Write RBASE and TBASE in the SCC parameter RAM to point to the Rx BD and Tx BD in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM, and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.
9. Program the CR to execute the INIT RX & TX PARAMS command for this channel. For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
10. Write RFCR with $18 and TFCR with $18 for normal operation.
11. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 16 bytes, so MRBLR = $0010.
12. Write PRCRC with $0000 to comply with CRC16.
13. Write PTCRC with $0000 to comply with CRC16.
14. Clear PAREC for the sake of clarity.
15. Write BSYNC with $8033, assuming a SYNC value of $33.
16. Write BDLE with $8055, assuming a DLE value of $55.
17. Write CHARACTER1–8 with $8000. They are not used.
18. Write RCCM with $E0FF. It is not used.
19. Initialize the Rx BD. Assume the Rx data buffer is at $00001000 in main memory. Write $B000 to Rx_BD_Status. Write $0000 to Rx_BD_Length (not required—done for instructional purposes only). Write $00001000 to Rx_BD_Pointer.
20. Initialize the Tx BD. Assume the Tx data buffer is at $00002000 in main memory and contains five 8-bit characters. Write $BD20 to Tx_BD_Status. Write $0005 to

Tx_BD_Length. Write $00002000 to Tx_BD_Pointer.

21. Write $FFFF to the SCCE to clear any previous events.
22. Write $0013 to the SCCM to enable the TXE, TX, and RX interrupts.
23. Write $08000000 to the CIMR to allow SCC4 to generate a system interrupt. (The CICR should also be initialized.)
24. Write $00000020 to GSMR_H4 to configure a small receive FIFO width.
25. Write $00000008 to GSMR_L4 to configure the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins to automatically control transmission and reception (DIAG bits) and the BISYNC mode. Notice that the transmitter (ENT) and receiver (ENR) have not been enabled yet.
26. Set the PSMR4 to $0600 to configure CRC16, CRC checking on receive, and normal operation (not transparent).
27. Write $00000038 to GSMR_L4 to enable the SCC4 transmitter and receiver. This additional write ensures that the ENT and ENR bits will be enabled last.

NOTE

After 5 bytes have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after 16 bytes have been received. Any additional receive data beyond 16 bytes will cause a busy (out-of-buffers) condition since only one Rx BD was prepared.

7.10.21 Transparent Controller

The transparent controller allows the transmission and reception of serial data over an SCC without any modification to that data stream. Transparent mode provides a clear channel on which no bit-level manipulation is performed by the SCC. Any protocol run over transparent mode is performed in software. The job of an SCC in transparent mode is to function simply as a high-speed serial-to-parallel and parallel-to-serial converter. This mode is also referred to as "totally transparent" or "promiscuous" operation.

There are several basic applications for transparent mode. First, some data may need to be moved serially, but requires no protocol superimposed—for example, voice data. Second, some board-level applications require a serial-to-parallel and parallel-to-serial conversion. Often this is done to allow communication between chips on the same board. Third, some applications require the switching of data without interfering with the protocol encoding itself. For instance, in a multiplexer, data from a high-speed time-multiplexed serial stream is multiplexed into multiple low-speed data streams. The concept is to switch the data path, not alter the protocol encoded on that data path.

By appropriately setting the GSMR, any of the SCC channels may be configured to function in transparent mode. The QUICC can both receive and transmit the entire serial bit stream transparently. This mode is configured by selecting the TTx and TRx bits in the in the GSMR for the transmitter and receiver, respectively. Both bits must be set for full-duplex transparent operation.

If just one of the TTx or TRx bits is set, the other half of the SCC can operate with another protocol as programmed in the MODE bits of the GSMR. (This allows loopback modes to DMA data from one memory location to another while converting the data to a specific serial format.)

The SCC in transparent mode can work with the TSA or NMSI. The SCC can support modem lines using the general-purpose I/O pins. The data can be transmitted and received with MSB or LSB first in each octet.

The SCC in transparent mode consists of separate transmit and receive sections whose operations are asynchronous with the CPU32+ core and may be either synchronous or asynchronous with respect to the other SCCs. Each clock can be supplied from the internal baud rate generator bank, DPLL output, or external pins.

7.10.21.1 TRANSPARENT CONTROLLER FEATURES. The transparent controller contains the following key features:

- Flexible Data Buffers
- Automatic SYNC Detection on Receive

  —16-Bit Pattern
  —8-Bit Pattern
  —4-Bit Pattern
  —External Sync Pin Support
- CRCs Can Optionally Be Transmitted and Received
- Reverse Data Mode
- Another Protocol Can Be Performed on the SCC's Other Half (Transmitter or Receiver) During Transparent Mode
- MC68302-Compatible Sync Options

7.10.21.2 TRANSPARENT CHANNEL FRAME TRANSMISSION PROCESSING. The transparent transmitter is designed to work with almost no intervention from the CPU32+ core. When this CPU32+ core enables the SCC transmitter in transparent mode, it will start transmitting idles. The SCC polls the first BD in the transmit channel's BD table. When there is a message to transmit, the SCC will fetch the data from memory, load the transmit FIFO, and wait for transmitter synchronization before starting to transmit the message.

Transmitter synchronization can be achieved using the $\overline{\text{CTS}}$ pin or waiting for the receiver to achieve synchronization, depending on the TXSY bit in the GSMR. See 7.10.21.4 Achieving Synchronization in Transparent Mode for more details. Once transmitter synchronization is achieved, transmission begins.

When a BD's data has been completely transmitted, the last in message (L) bit is checked. If the L-bit is set, the SCC writes the message status bits into the BD and clears the R-bit. It will then start transmitting idles until the next BD is ready. (Even if the next BD is already ready, some idles will still be transmitted.) The transmitter will only begin transmission again after it achieves synchronization.

When the end of the current BD has been reached and the L-bit is cleared (working in multibuffer mode), only the R-bit is cleared, and the transmitter moves immediately to the next buffer to begin it transmission, with no gap on the serial line between buffers. Failure to provide the next buffer in time results in a transmit underrun, causing the TXE bit in the transparent event register to be set.

In both cases, an interrupt is issued according to the interrupt (I) bit in the BD. By appropriately setting the I-bit in each BD, interrupts can be generated after the transmission of each buffer or a after a group of buffers have been transmitted. The SCC will then proceed to the next BD in the table.

Any whole number of bytes may be transmitted. If the REVD bit in the GSMR is set, each data byte will be reversed in its bit order before transmission, transmitting the MSB of each octet first.

An option is available to decrease the latency of the transmitter by decreasing the transmit FIFO size. This option is enabled by the TFL bit in the GSMR. The user, however, should note that this option can cause transmitter underruns at higher transmission speeds.

An optional CRC may be appended to each transparent frame if enabled in the Tx BD. The CRC pattern is chosen in the TCRC bits in the GSMR.

7.10.21.3 TRANSPARENT CHANNEL FRAME RECEPTION PROCESSING. When the CPU32+ core enables the SCC receiver in transparent mode, it will wait to achieve synchronization before beginning to receive data. The receiver can be synchronized to the data by a synchronization pulse or by a SYNC pattern. See 7.10.21.4 Achieving Synchronization in Transparent Mode for more details.

After a buffer is filled, the SCC clears the empty (E) bit in the BD and generates an interrupt if the interrupt (I) bit in the BD is set. It then moves to the next Rx BD in the table and begins moving data to its associated buffer. If the next buffer is not available when needed, a busy condition is signified by the setting of the BSY bit in the transparent event register, which can generate a maskable interrupt.

The receiver will revert to the hunt mode upon receiving the ENTER HUNT MODE command or recognizing an error condition such as a lack of receive buffers, detection of CD lost, or a receiver overrun.

If the REVD bit in the GSMR is set, each data byte will be reversed in its bit order before it is written to memory.

An option is available to decrease the latency of the receiver by decreasing the receive FIFO width. This option is enabled by the RFW bit in the GSMR. The user, however, should note that this option can cause receiver overruns at higher transmission speeds.

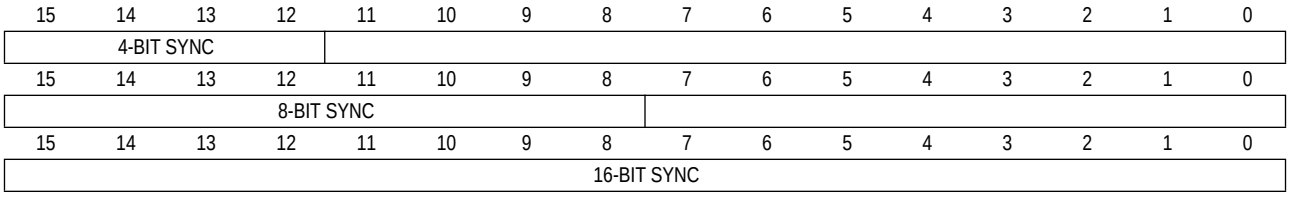
The receiver always checks the CRC of the frame that is received, according to the TCRC bits in the GSMR. If a CRC is not required, the resulting errors may be ignored.

7.10.21.4 ACHIEVING SYNCHRONIZATION IN TRANSPARENT MODE. Once the SCC transmitter is enabled for transparent operation in the GSMR, the Tx BD is prepared for the SCC, and the transmit FIFO has been preloaded by the SDMA channel, one additional process must occur before data can be transmitted—i.e., transmit synchronization.

Similarly, once the SCC receiver is enabled for transparent operation in the GSMR and the Rx BD is made empty for the SCC, one additional process must occur before data can be received—receive synchronization.

The synchronization process gives the user bit-level control over when the transmission and reception can begin. There are two basic methods: an in-line synchronization pattern and external synchronization signals.

7.10.21.4.1 In-Line Synchronization Pattern. The transparent channel can be programmed to transmit and receive a synchronization pattern if the SYNL bits in the GSMR are non-zero. The pattern is defined in the DSR. The length of the SYNC pattern is defined in the SYNL bits in the GSMR.



The receiver synchronizes on the synchronization pattern that is located in the DSR. For instance, if a 4-bit SYNC is selected, then reception begins as soon as these four bits are received, beginning with the first bit following the 4-bit SYNC.

The transmitter can synchronize on the receiver pattern if the RSYN bit in the GSMR is set. This effectively links the transmitter synchronization to the receiver synchronization.

External Synchronization Signals

If the SYNL bits in the GSMR are programmed to 00, an external signal is used to begin the sequence. The $\overline{\text{CTS}}$ pin is used for the transmitter, and the $\overline{\text{CD}}$ pin is used for the receiver. The $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ pins share two options: the pulse option and the sampling option.

The pulse option determines whether the $\overline{\text{CD}}$ pin or $\overline{\text{CTS}}$ pins need only be asserted once to begin reception/transmission or whether the $\overline{\text{CD}}$ pin or $\overline{\text{CTS}}$ pins must be asserted and stay asserted for the duration of the transparent frame. This is controlled by the CDP and CTSP bits in the GSMR. If the user expects a continuous stream of data without interruption, then the pulse operation should be used. However, if the user is trying to identify frames of transparent data, then the envelope mode of the $\overline{\text{CD}}$ and $\overline{\text{CTS}}$ pins should be used.

NOTE

The MC68302 transparent mode offered the EXSYN bit, which, when set, gave the pulse behavior.

The sampling option determines the delay between $\overline{CD}$ and $\overline{CTS}$ being asserted and the resulting action by the SCC. The $\overline{CD}$ or $\overline{CTS}$ pins may be assumed to be asynchronous to the data and then internally synchronized by the SCC, or the $\overline{CD}$ or $\overline{CTS}$ pins may be assumed to be synchronous to the data giving faster operation. This option allows the $\overline{RTS}$ pin of one SCC to be connected to the $\overline{CD}$ pin of another SCC (on another QUICC) and to have the data synchronized and bit aligned.

NOTE

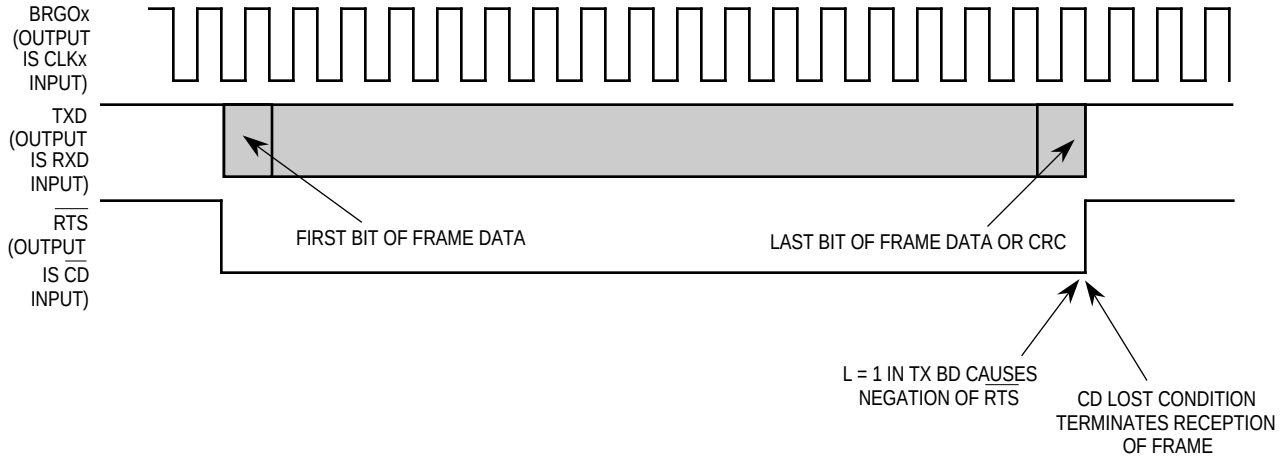
The MC68302 transparent mode only offered the asynchronous option.

Diagrams for the pulse/envelope and sampling options may be found in 7.10.11 SCC Timing Control.

Lastly, an option exists to link the transmitter synchronization to the receiver synchronization.

7.10.21.4.2 Transparent Synchronization Example. Figure 7-64 shows an example of synchronization using external signals.

NOTES:
1. Each QUICC generates its own transmit clocks. If the transmit and receive clocks are the same, it is possible for one QUICC to generate transmit and receive clocks for the other QUICC (for example, CLKx on QUICC 2 could be used to clock the transmitter and receiver).



NOTES:
1. CTS should be configured as always asserted in the port C parallel I/O or else connected to ground externally.
2. The required GSMR configurations are: DIAG = 00, CTSS = 1, CTSP is a don't care, CDS = 1, CDP = 0, TTX = 1, and TRX = 1. REVD and TCRC are application dependent.
3. The transparent frame will contain a CRC if the TC bit is set in the Tx BD.

Figure 7-64. Sending Transparent Frames Between QUICCs

QUICC1 and QUICC2 exchange transparent frames and synchronize each other using the RTS and CD pins. The CTS pin is not required since transmission may begin at any time. Thus, the RTS signal is directly connected to the other QUICC's CD pin. The RSYN option in GSMR is not used, and transmission and reception from each QUICC are independent.

7.10.21.5 TRANSPARENT MEMORY MAP. When configured to operate in transparent mode, the QUICC overlays the structure listed in Table 7-5 onto the protocol-specific area of the SCC parameter RAM listed in Table 7-10.

Table 7-10. Transparent-Specific Parameters

Address	Name	Width	Description
SCC Base + 30	**CRC_P**	Long	CRC Preset for Totally Transparent
SCC Base + 34	**CRC_C**	Long	CRC Constant for Totally Transparent Receiver

NOTE: The boldfaced items should be initialized by the user.

CRC_P**.** For the 16-bit CRC-CCITT, CRC_P should be initialized with $0000FFFF. For the 32-bit CRC-CCITT, CRC_P should be initialized with $FFFFFFFF. For the CRC-16, CRC_P should be initialized with ones ($0000FFFF) or zeros ($00000000).

CRC_C. For the 16-bit CRC-CCITT, CRC_C should be initialized with $0000F0B8. For the 32-bit CRC-CCITT, CRC_C should be initialized with $DEBB20E3. For the CRC-16 which is normally used with BISYNC, CRC_C should be initialized with $00000000.

NOTE

This value overlaps with the CRC constant (mask) for the HDLC-based protocols. This overlap is not detrimental since the CRC constant (mask) is only used for the receiver; thus, only one entry is required. Therefore, the user may choose HDLC transmitter with a transparent receiver or a transparent transmitter with an HDLC receiver.

7.10.21.6 TRANSPARENT COMMAND SET. The following transmit and receive commands are issued to the CR.

7.10.21.6.1 Transmit Commands. The following paragraphs describe the transparent transmit commands.

STOP TRANSMIT Command*.* After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the transmit enable mode and starts polling the first BD in the table every 64 clocks (immediately if the TOD bit in the TODR is set).

The STOP TRANSMIT command disables the transmission of frames on the transmit channel. If this command is received by the transparent controller during frame transmission, transmission of that buffer is aborted after a maximum of 64 additional bits are transmitted, and the transmit FIFO is flushed. The TBPTR is not advanced, no new BD is accessed, and no new buffers are transmitted for this channel. The transmitter will send idles.

GRACEFUL STOP TRANSMIT Command*.* The GRACEFUL STOP TRANSMIT command is used to stop transmission in an orderly way, rather than abruptly as performed by the regular STOP TRANSMIT command. It stops transmission after the current frame has completed transmission, or immediately if there is no frame being transmitted. (A transparent frame is not complete until a BD with the L-bit set has its associated buffer completely transmitted.) The GRA bit in the SCCE will be set once transmission has stopped. After transmission ceases, the transmit parameters, including BDs, may be modified. The TBPTR will point to the next Tx BD in the table. Transmission will begin once the R-bit of the next BD is set and the RESTART TRANSMIT command is issued.

RESTART TRANSMIT Command*.* The RESTART TRANSMIT command reenables the transmission of characters on the transmit channel. This command is expected by the transparent controller after a STOP TRANSMIT command, after a STOP TRANSMIT command and disabling the channel in its SCC mode register, after a GRACEFUL STOP TRANSMIT

command, or after a transmitter error (underrun or CTS lost). The transparent controller will resume transmission from the current TBPTR in the channel's Tx BD table.

INIT TX PARAMETERS Command. This command initializes all transmit parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the transmitter is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both transmit and receive parameters.

7.10.21.6.2 Receive Commands. The following paragraphs describe the transparent receive commands.

ENTER HUNT MODE Command*.* After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the receive enable mode and will use the first BD in the table.

The ENTER HUNT MODE command is used to force the transparent receiver to abort reception of the current frame and enter the hunt mode. In the hunt mode, the transparent controller waits for the synchronization sequence. After receiving the command, the current receive buffer is closed. Further data reception will use the next BD.

CLOSE Rx BD Command*.* The CLOSE Rx BD command is used to force the SCC to close the Rx BD, if it is currently being used, and to use the next BD for any subsequent data that is received. If the SCC is not in the process of receiving data, no action is taken by this command.

INIT RX PARAMETERS Command*.* This command initializes all the receive parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the receiver is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both receive and transmit parameters.

7.10.21.7 TRANSPARENT ERROR-HANDLING PROCEDURE. The SCC reports message reception and transmission error conditions using the channel BDs, the error counters, and the SCC event register.

7.10.21.7.1 Transmission Errors. The following paragraphs describe various types of transmission errors.

Transmitter Underrun*.* When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the UN bit in the BD, and generates the TXE interrupt (if enabled). The channel resumes transmission after the reception of the RESTART TRANSMIT command. Underrun can occur after a transmit frame for which the L-bit in the Tx BD was not set. In this case, only the TXE bit is set. Underrun cannot occur between transparent frames.

CTS Lost During Message Transmission*.* When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the CT bit in the BD, and generates the TXE interrupt if it is enabled. The channel will resume transmission after the reception of the RESTART TRANSMIT command.

7.10.21.7.2 Reception Errors. The following paragraphs describe various types of reception errors.

Overrun Error**.** The SCC maintains an internal FIFO for receiving data. The CP begins programming the SDMA channel (if the data buffer is in external memory) and updating the CRC when 8 or 32 bits (according to the RFW bit in the GSMR) are received in the FIFO. If a FIFO overrun occurs, the SCC writes the received data byte to the internal FIFO over the previously received byte. The previous character and its status bits are lost. Following this, the channel closes the buffer, sets the OV bit in the BD, and generates the RX interrupt (if enabled). The receiver then enters hunt mode immediately.

CD Lost During Message Reception**.** When this error occurs, the channel terminates message reception, closes the buffer, sets the CD bit in the BD, and generates the RX interrupt (if enabled). This error has the highest priority; the rest of the message is lost, and no other errors are checked in the message. The receiver then enters hunt mode immediately.

7.10.21.8 TRANSPARENT MODE REGISTER (PSMR). The PSMR is called the transparent mode register when an SCC is programmed for transparent mode. However, since all transparent mode selections are in the GSMR, this register is not used by the transparent controller. If transparent mode is only selected for the transmitter/receiver, then the transmitter/receiver may be programmed to support another protocol. In such a case, the PSMR may be used for that other protocol.

7.10.21.9 TRANSPARENT RECEIVE BUFFER DESCRIPTOR (RX BD). •The CP reports information about the received data for each buffer using an Rx BD. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:

1. Detecting an error
2. Detecting a full receive buffer
3. Issuing the ENTER HUNT MODE command
4. Issuing the CLOSE Rx BD command

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**E**	—	**W**	**I**	L	**F**	**CM**	—	DE	—	—	NO	—	CR	OV	CD
OFFSET + 2	DATA LENGTH															
OFFSET + 4	RX DATA BUFFER POINTER*															
OFFSET + 6																

NOTE: Entries in boldface must be initialized by the user.

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 CPU32+ 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 CPU32+ core should not write any fields of this Rx BD.

Bits 14, 10, 8, 6, 5, 3—Reserved

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 BD s in this table is programmable and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been used.
1 = When this buffer has been closed by the transparent controller, the RX bit in the transparent event register will be set. The RX bit can cause an interrupt if it is enabled.

L—Last in Frame

This bit is set by the transparent controller when this buffer is the last in a frame. This implies the negation of $\overline{CD}$ in envelope mode or the reception of an error, in which case one or more of the OV, CD, and DE bits are set. The transparent controller will write 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.

F—First in Frame

This bit is set by the transparent controller when this buffer is the first in a frame.

0 = The buffer is not the first in a frame.
1 = The buffer is the first in a frame.

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 overwritten automatically when the CP next accesses this BD. However, the E-bit will be cleared if an error occurs during reception, regardless of the CM bit.

DE—DPLL Error

This bit is set by the transparent controller when a DPLL error has occurred during the reception of this buffer. In Decoding modes where a transition is promised every bit, the DPLL error will be set when a missing transition occurs.

NO—Rx Nonoctet Aligned Frame

A frame that contained a number of bits not exactly divisible by eight was received.

CR—CRC Error indication bits

This frame contains a CRC error. The received CRC bytes are always written to the receive buffer.

OV—Overrun

A receiver overrun occurred during buffer reception.

CD—Carrier Detect Lost

The carrier detect signal was negated during buffer reception.

Data Length

The data length is the number of octets that the CP has written into this BD's data buffer. It is written only once by the CP as the buffer is closed.

NOTE

The actual amount of memory allocated for this buffer should be greater than or equal to the contents of the maximum receive buffer length register (MRBLR).

Rx Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, must be divisible by 4 (unless the RFW bit in the GSMR is set to 8-bits wide, in which case it may be even or odd). The buffer may reside in either internal or external memory.

7.10.21.10 TRANSPARENT TRANSMIT BUFFER DESCRIPTOR (TX BD). Data is presented to the CP for transmission on an SCC channel by arranging it in buffers referenced by the channel's Tx BD table. The CP confirms transmission or indicates error conditions using the BDs to inform the processor that the buffers have been serviced.

The status and control bits are prepared by the user before transmission and are set by the CP after the buffer has been transmitted.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**R**	—	**W**	**I**	**L**	**TC**	**CM**	—	—	—	—	—	—	—	UN	CT
OFFSET + 2	**DATA LENGTH**															
OFFSET + 4	**TX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: Entries in boldface must be initialized by the user.

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.

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 receive incoming data into the first BD in the table (the BD pointed to by TBASE). The number of Tx BDs in this table is programmable, and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.
1 = When this buffer is serviced by the CP, TX, or TXE bit in the transparent event register will be set. The TX and TXE bits can cause interrupts if they are enabled.

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) will be transmitted immediately following 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 transmitted, the transmitter will require synchronization before the next buffer will be transmitted.

TC—Transmit CRC

0 = No CRC sequence will be transmitted after this buffer.
1 = A frame check sequence as defined by the TCRC bits in the GSMR will be transmitted after the last byte of this buffer.

CM—Continuous Mode

0 = Normal operation.
1 = The R-bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this BD. However, the R-bit will be cleared if an error occurs during transmission, regardless of the CM bit.

UN—Underrun

The SCC encountered a transmitter underrun condition while transmitting the associated data buffer.

CT—CTS Lost

CTS was lost during frame transmission.

Data Length

The data length is the number of bytes that the CP should transmit from this BD’s data buffer. The data length, which should be greater than zero, may be even or odd. This value is never modified by the CP.

Tx Data Buffer Pointer

The transmit buffer pointer, which always points to the first byte of the associated data buffer, may be even or odd. The buffer may reside in either internal or external memory.

7.10.21.11 TRANSPARENT EVENT REGISTER (SCCE). The SCCE is called the transparent event register when the SCC is operating as a transparent controller. It is a 16-bit register used to report events recognized by the transparent channel and to generate interrupts. On recognition of an event, the transparent controller will set the corresponding bit in the transparent event register. Interrupts generated by this register may be masked in the transparent mask register.

The transparent event register is a memory-mapped register that may be read at any time. A bit is reset by writing a one (writing a zero does not affect a bit's value). More than one bit may be reset at a time. All unmasked bits must be reset before the CP will negate the internal interrupt request signal. This register is cleared at reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—			GLr	GLt	DCC	—	—	GRA	—	—	TXE	RCH	BSY	TX	RX

Bits 15–13, 9–8, 6–5—Reserved

GLr—Glitch on Rx

A clock glitch was detected by this SCC on the receive clock.

GLt—Glitch on Tx

A clock glitch was detected by this SCC on the transmit clock.

DCC—DPLL CS Changed

The carrier sense status as generated by the DPLL has changed state. The real-time status may be found in SCCS. This is not the $\overline{CD}$ pin status, which is reported elsewhere, and is only valid when the DPLL is used.

GRA—Graceful Stop Complete

A graceful stop, which was initiated by the GRACEFUL STOP TRANSMIT command, is now complete. This bit is set as soon the transmitter has finished transmitting any frame that was in progress when the command was issued. It will be set immediately if no frame was in progress when the command was issued.

TXE—Tx Error

An error (CTS lost or underrun) occurred on the transmitter channel.

RCH—Receive Character

A byte or long word has been received and written to the buffer. This depends on the setting of the RFW bit in the GSMR.

BSY—Busy Condition

A byte/long-word was received and discarded due to lack of buffers. The receiver will resume reception after an ENTER HUNT MODE command.

TX—Tx Buffer

A buffer has been transmitted. This bit is set no sooner than when the last bit of the last byte of the buffer begins its transmission, assuming the L-bit of the Tx BD is set. If the L-bit is not set, TX is set when the last byte of data is written to the transmit FIFO.

RX—Rx Buffer

A complete buffer has been received on the SCC channel. This bit is set no sooner than two serial clocks after the last bit of the last byte in which the buffer is received on the RXD pin.

7.10.21.12 TRANSPARENT MASK REGISTER (SCCM). The SCCM is referred to as the transparent mask register when the SCC is operating in transparent mode. It is a 16-bit read-write register that has the same bit format as the transparent event register. If a bit in the transparent mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

7.10.21.13 SCC STATUS REGISTER (SCCS). The SCCS is an 8-bit read-only register that allows the user to monitor real-time status conditions on the RXD line. The real-time status of the $\overline{CTS}$ and $\overline{CD}$ pins are part of the port C parallel I/O.

7	6	5	4	3	2	1	0
—	—	—	—	—	—	CS	—

CS—Carrier Sense (DPLL)

This bit shows the real-time carrier sense of the line as determined by the DPLL, if it is used.

0 = The DPLL does not sense a carrier.
1 = The DPLL does sense a carrier.

7.10.21.14 SCC TRANSPARENT EXAMPLE. The following list is an initialization sequence for an SCC transparent channel. The transmitter and receiver are both enabled, but operate independently of each other; they implement the connection shown on QUICC 2 in Figure 7-64. Both transmit and receive clocks are provided externally to QUICC 2 using the CLK7 pin. SCC4 is used. The transparent controller is configured with the $\overline{RTS4}$ and $\overline{CD4}$ pins active. $\overline{CTS4}$ is grounded internally by the configuration in port C. A 16-bit CRC-CCITT is sent with each transparent frame. The FIFOs are configured for fast operation.

1. The SDCR (SDMA Configuration Register) should be initialized to $0740, rather than being left at its default value of $0000.
2. Configure the port A pins to enable the TXD4 and RXD4 pins. Write PAPAR bits 6 and 7 with ones. Write PADIR bits 6 and 7 with zeros. Write PAODR bits 6 and 7 with zeros.
3. Configure the port C pins to enable $\overline{RTS4}$, $\overline{CTS4}$, and $\overline{CD4}$. Write PCPAR bit 3 with one and bit 11 with zero. Write PCDIR bits 3 and 11 with zero. Write PCSO bit 11 with one and bit 10 with zero.
4. Configure port A to enable the CLK7 pin. Write PAPAR bit 14 with a one. Write

PADIR bit 14 with a zero.

5. Connect the CLK7 pin to SCC4 using the SI. Write the R4CS bits in SICR to 110. Write the T4CS bits in SICR to 110.
6. Connect the SCC4 to the NMSI (i.e., its own set of pins). Clear the SC4 bit in the SICR.
7. Write $0740 to the SDCR to initialize the SDMA Configuration Register.
8. Write RBASE and TBASE in the SCC parameter RAM to point to the Rx BD and Tx BD in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM, and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.
9. Program the CR to execute the INIT RX & TX PARAMS command for this channel. For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
10. Write RFCR with $18 and TFCR with $18 for normal operation.
11. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 16 bytes, so MRBLR = $0010.
12. Write CRC_P with $0000FFFF to comply with the 16-bit CRC-CCITT.
13. Write CRC_C with $0000F0B8 to comply with the 16-bit CRC-CCITT.
14. Initialize the Rx BD. Assume the Rx data buffer is at $00001000 in main memory. Write $B000 to Rx_BD_Status. Write $0000 to Rx_BD_Length (not required—done for instructional purposes only). Write $00001000 to Rx_BD_Pointer.
15. Initialize the Tx BD. Assume the Tx data buffer is at $00002000 in main memory and contains five 8-bit characters. Write $BC00 to Tx_BD_Status. Write $0005 to Tx_BD_Length. Write $00002000 to Tx_BD_Pointer.
16. Write $FFFF to the SCCE to clear any previous events.
17. Write $0013 to the SCCM to enable the TXE, TX, and RX interrupts.
18. Write $08000000 to the CIMR to allow SCC4 to generate a system interrupt. (The CICR should also be initialized.)
19. Write $00001980 to GSMR_H4 to configure the transparent channel.
20. Write $00000000 to GSMR_L4 to configure the $\overline{\text{CTS}}$ and $\overline{\text{CD}}$ pins to automatically control transmission and reception (DIAG bits). Normal operation of the transmit clock is used (TCI is cleared). Notice that the transmitter (ENT) and receiver (ENR) have not been enabled yet.
21. Write $00000030 to GSMR_L4 to enable the SCC4 transmitter and receiver. This additional write ensures that the ENT and ENR bits will be enabled last.

NOTE

After 5 bytes have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after 16 bytes have been received. Any additional receive data beyond 16 bytes will cause

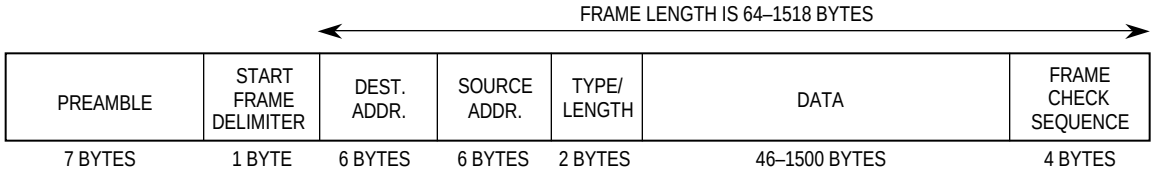
a busy (out-of-buffers) condition since only one Rx BD was prepared.

7.10.22 RAM Microcodes

Additional protocols may be added in the future through the use of RAM microcodes. See Appendix C RISC Microcode from RAM for more information.

7.10.23 Ethernet Controller

The Ethernet/IEEE 802.3 protocol is a widely used LAN that is based on the carrier sense multiple access/collision detect (CSMA/CD) approach. Ethernet and IEEE 802.3 frames are very similar and can co-exist on the same LAN. The two protocols are referred to synonymously as "Ethernet" in this manual unless specifically noted. Ethernet/IEEE 802.3 frames are based on the frame structure shown in Figure 7-65.



NOTE: The LSB of each octet is transmitted first.

Figure 7-65. Ethernet/802.3 Frame Format;

The frame begins with a 7-byte preamble of alternating ones and zeros. Since the frame is Manchester encoded, the preamble gives receiving stations a known pattern on which to lock. The start frame delimiter, which signifies the beginning of the frame, follows the preamble. 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; however, this addressing has never been significantly used in the industry.

The next field is the type field in Ethernet and the length field in IEEE 802.3. The type field is used to signify the protocol used in the rest of the frame (e.g., TCP/IP). The length field is used to specify the length of the data portion of the frame. For Ethernet and IEEE 802.3 frames to co-exist on the same LAN, the length field of the frame must always be unique from any type fields used in Ethernet. This has limited the length of the data portion of the frame to 1500 bytes, and therefore the total frame length to 1518 bytes.

The final 4 bytes of the frame are the FCS. This is the standard 32-bit CCITT-CRC polynomial used in many other protocols.

When a station wishes to transmit, it checks for activity on the LAN. When the LAN becomes silent for a specified period, the station begins transmission. During transmission, the station continually checks for collision on the LAN. If a collision is detected, the station forces a jam of all ones on its frame and ceases transmission. Collisions usually occur close to the beginning of a frame. The station then waits a random period of time (backoff) before attempting to transmit again. Once the backoff is complete, the station waits for silence on the LAN and

then begins retransmission on the LAN, called a retry. If the frame is not successfully transmitted within 15 retries, then an error is indicated.

A few key Ethernet timing parameters are as follows. The Ethernet of 10 Mbps gives 0.8 μs per byte. The preamble plus start frame delimiter is transmitted in 6.4 μs. The minimum interframe gap is 9.6 μs. The slot time is 52 μs.

7.10.23.1 ETHERNET ON QUICC—MC68EN360. The Ethernet protocol is available only on the Ethernet version of the QUICC, called the MC68EN360. The non-Ethernet version of the QUICC is the MC68360. The term "QUICC" is the overall device name that denotes all versions of the device.

The MC68EN360 is a superset of the MC68360, having the additional option allowing Ethernet operation on any of the four SCCs. Due to performance reason not all SCCs can be configured as Ethernet controller at the same time. The MC68EN360 is not restricted only to Ethernet operation. HDLC, UART, and other protocols may be used to allow dynamic switching between protocols. See Appendix A Serial Performance for available SCC performance.

When the MODE bits of the SCC GSMR select the Ethernet protocol, then that SCC performs the full set of IEEE 802.3/Ethernet CSMA/CD media access control and channel interface functions (see Figure 7-66).

The QUICC Ethernet controller requires an external serial interface adaptor (SIA) and transceiver function to complete the interface to the media. This function is implemented in the Motorola MC68160 enhanced Ethernet serial transceiver (EEST).

The QUICC+EEST solution provides a direct connection to the attachment unit interface (AUI) or twisted-pair (10BASE-T). The EEST provides a glueless interface to the QUICC, Manchester encoding and decoding, automatic selection of 10BASE-T versus AUI ports, 10BASE-T polarity detection and correction, LED drivers, and a low-power mode. For more information, refer to the MC68160 device description.

The QUICC Ethernet controller provides a number of features listed below. Although the QUICC contains DPLLs that allow Manchester encoding and decoding, these DPLLs were not designed for Ethernet rates. Therefore, the Ethernet controller on the QUICC bypasses the on-chip DPLLs and uses the external SIA on the EEST instead.

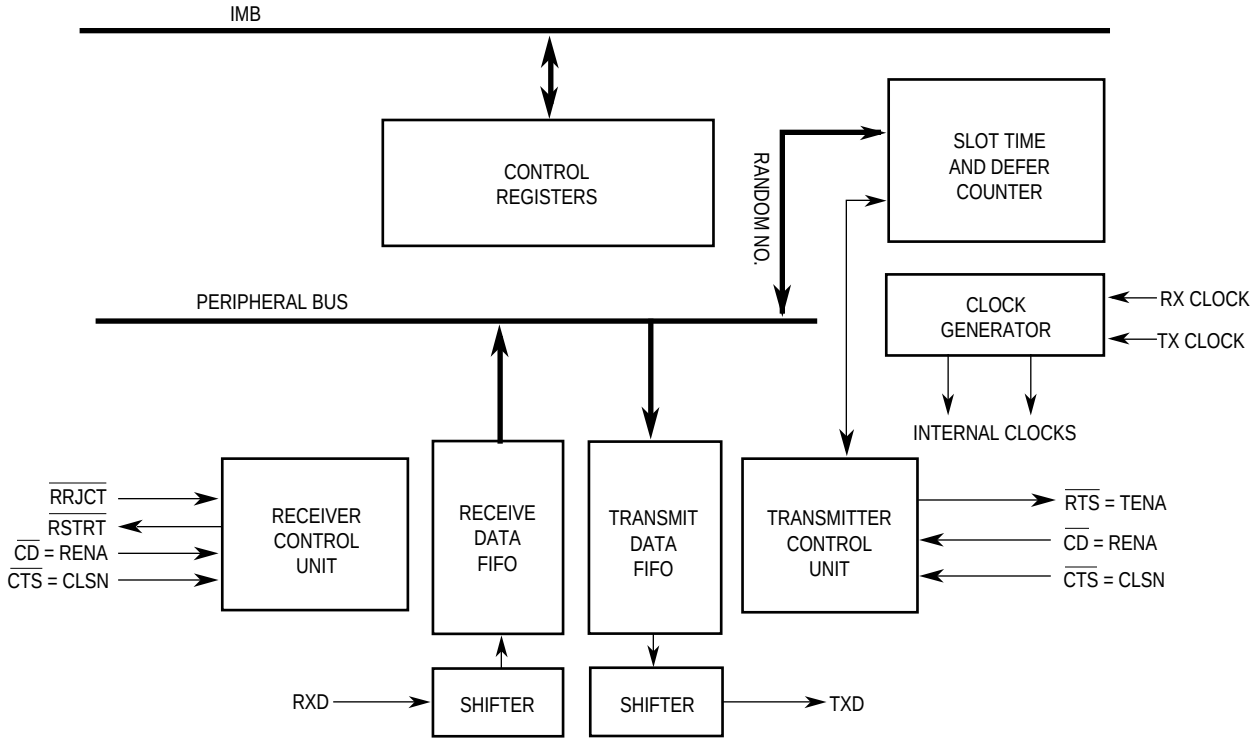


Figure 7-66. Ethernet Block Diagram

7.10.23.2 ETHERNET KEY FEATURES. The Ethernet contains the following key features:

- Performs MAC Layer Functions of Ethernet and IEEE 802.3
- Performs Framing Functions
  - —Preamble Generation and Stripping
  - —Destination Address Checking
  - —RC Generation and Checking
  - —Automatic “Short Frames” Padding 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 “Attempt Limit” Is Reached)
  - —Automatic Discard of Incoming Collided Frames
  - —Delay Transmission of New Frames for Specified Interframe Gap
- Bit Rates up to 10 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 Addresses, but Discards Frame If Reject Pin Asserted

- External CAM Support on Both Serial and System Bus Interfaces
- Up to Eight Parallel I/O Pins May Be Sampled and Appended to Any Frame
- 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

7.10.23.3 LEARNING ETHERNET ON THE QUICC. The following paragraphs detail the Ethernet functionality on the QUICC. However, they show the additions made to the standard SCC functionality to implement Ethernet. Therefore, the reader is encouraged to learn the basics of the SCCs and the overall architecture of the CPM before attempting to learn this section in great detail.

A first-time user of the QUICC who plans to use Ethernet on the QUICC should first read the following sections of this user manual.

1. 7.1 RISC Controller**,** 7.2 Command Set, and 7.3 Dual-Port RAM. The RISC controller is used to issue special commands to the Ethernet channel. The dual-port RAM is used to load Ethernet parameters and initialize BDs for use by the Ethernet channel.
2. 7.7 SDMA Channels discusses how SDMA channels are used to transfer data to/from the Ethernet channel and system memory.
3. 7.8.9 NMSI Configuration explains how clocks are routed to the SCCs through the bank of clocks.
4. 7.10.1 SCC Overview contains more detailed information on the SCCs that are appli-

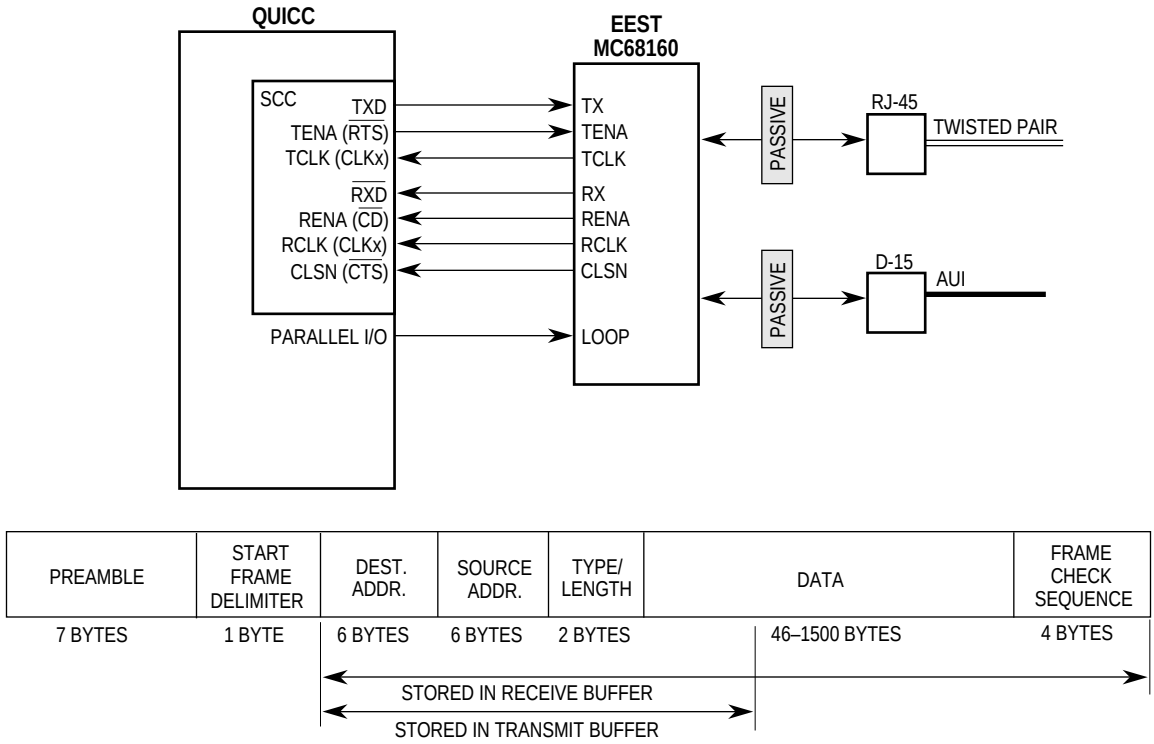
cable to all protocols. The reader does not need to read the SCC DPLL description since the on-chip DPLLs are not used in Ethernet.

5. 7.10.23 Ethernet Controller should be read next.
6. 7.15 CPM Interrupt Controller (CPIC) defines the interrupt priority of this SCC and how interrupts are generated to the CPU32+ core.
7. 7.14 Parallel I/O Ports shows how to configure the desired Ethernet pin functions to be active.

7.10.23.4 CONNECTING QUICC TO ETHERNET. Figure 7-67 shows the basic components and pins required to make the Ethernet connection between the QUICC and the EEST.

The QUICC Ethernet controller has seven basic pins that make up the interface to the external EEST chip:

1. Receive clock. Receive clock to the SCC (RCLK) may be either the CLK1, CLK2, CLK3, or CLK4 pin that is routed through the bank of clocks on the QUICC.
2. Transmit clock. Transmit clock to the SCC (TCLK) may be either the CLK1, CLK2, CLK3, or CLK4 pin that is routed through the bank of clocks on the QUICC. (The SCC RCLK and SCC TCLK should not be connected to the same CLKx pin since the EEST provides a separate receive and transmit clock signal).
3. Transmit data. This is the QUICC the TXD pin.
4. Receive data. This is the QUICC RXD pin.



NOTE: Short transmit frames are padded automatically by the QUICC.

Figure 7-67. Connecting the QUICC to Ethernet

The following pins take on new meanings when the Ethernet protocol is selected for the SCC:

1. Transmit Enable (TENA). The SCC's $\overline{\text{RTS}}$ pin changes to become TENA when the SCC is configured for Ethernet operation. The polarity of TENA is active high; whereas, the polarity of $\overline{\text{RTS}}$ is active low.
2. Receive Enable (RENA). The SCC's $\overline{\text{CD}}$ pin changes to become RENA when the SCC is configured for Ethernet operation. The polarity of RENA is active high; whereas, the polarity of $\overline{\text{CD}}$ is active low.
3. Collision (CLSN). The SCC's $\overline{\text{CTS}}$ pin changes to become CLSN when the SCC is configured for Ethernet operation. The polarity of CLSN is active high; whereas, the polarity of $\overline{\text{CTS}}$ is active low.

NOTE

The carrier sense signal is often referred to in Ethernet descriptions, because it defines whether the LAN is currently in use. Carrier sense is defined as RENA ORed with CLSN.

The EEST has similar names for its connection to the seven basic QUICC pins. In addition, the EEST contains a loopback pin to allow the QUICC to perform external loopback testing. This can be controlled by any available parallel I/O pin on the QUICC.

In addition, the QUICC has additional pins used to interface to an optional external content-addressable memory (CAM). These pins are described in 7.10.23.7 CAM Interface.

External to the EEST are the passive components (principally transformers) required to connect to AUI or twisted-pair media. For more information on the EEST connection circuits, refer to the MC68160 device description.

The QUICC stores every byte received after the start frame delimiter into system memory, using the SDMA channels. On transmit, the user provides the destination address, source address, type/length field, and the transmit data. The QUICC will automatically pad frames that have less than 46 bytes in the data field to meet the minimum frame requirements. In addition, the QUICC will append the FCS to the frame.

7.10.23.5 ETHERNET CHANNEL FRAME TRANSMISSION. The Ethernet transmitter is designed to work with almost no intervention from the host. When the host enables the transmitter, the Ethernet controller will poll the first Tx BD in the channel's Tx BD table. The poll occurs every 128 serial clocks. If the user has a frame ready to transmit, the TOD bit in the transmit-on-demand register may be set to eliminate waiting for the next poll to occur.

When there is a frame to transmit, the Ethernet controller will begin fetching the data from the data buffer, assert TENA to the EEST, and start transmitting the preamble sequence, the start frame delimiter, and then the frame information. However, the controller will defer the transmission if the line is busy (carrier sense is active). Before transmitting, the controller waits for carrier sense to become inactive. Once carrier sense becomes inactive, the controller determines if carrier sense stays inactive for 6.0 μs. If so, then the transmission will begin after waiting an additional 3.6 μs (i.e., 9.6 μs after carrier sense originally became inactive).

If a collision occurs during the transmit frame, the Ethernet controller follows the specified backoff procedures and attempts 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 (8 bytes if the transmit frame was long-word aligned) in internal RAM, so that they do not have to be retrieved from system memory in case of a collision. This improves bus utilization and latency in the case that the backoff timer output results in a need for an immediate retransmission. If a collision occurs during the transmission of the frame, the Ethernet controller will return to the first buffer for a retransmission. The only restriction is that the first buffer should contain at least 9 bytes.

When the end of the current BD has been reached and the L-bit in the Tx BD is set, the FCS (32-bit CRC) bytes are appended (if the TC bit is set in the Tx BD), and TENA is negated. This tells the EEST to generate the illegal Manchester encoding that signifies the end of the Ethernet frame.

Following the transmission of the CRC, the Ethernet controller writes the frame status bits into the BD and clears the R-bit. When the end of the current BD has been reached, and the L-bit is not set (i.e., a frame is comprised of multiple buffers), only the R-bit is cleared.

In either mode, an interrupt can be issued according to the I-bit in the Tx BD. The Ethernet controller will then proceed to the next Tx BD in the table. In this way, the user may be interrupted after each frame, after each buffer, or after a specific buffer has been transmitted.

The Ethernet controller has an option to add pad characters to short frames. If the PAD bit is set in the Tx BD, the frame will be padded up to the value of the minimum frame length register.

To rearrange the transmit queue before the CP has completed transmission of all frames, issue the GRACEFUL STOP TRANSMIT command. This technique can be useful for transmitting expedited data before previously linked buffers or for error situations. When the GRACEFUL STOP TRANSMIT command is issued, the Ethernet controller will stop immediately if no transmission is in progress, or continue transmission until the current frame has successfully completed transmission or terminates with a collision. When the Ethernet controller is given the RESTART TRANSMIT command, it resumes transmission.

The Ethernet controller transmits bytes LSB first.

7.10.23.6 ETHERNET CHANNEL FRAME RECEPTION. The Ethernet receiver is also designed to work with almost no intervention from the host. The Ethernet receiver can perform address recognition, CRC checking, short frame checking, maximum DMA transfer checking, and maximum frame length checking.

When the host enables the Ethernet receiver, it will enter hunt mode as soon as the RENA signal is asserted if CLSN is negated. In hunt mode, as data is shifted into the receive shift register one bit at a time, the contents of the register are compared to the contents of the SYN1 field in the data synchronization register. This compare function becomes valid a certain number of clocks after the start of the frame (depending on the NIB bits in the PSMR). If the two are not equal, the next bit is shifted in, and the comparison is repeated. If a double zero fault or double one fault is detected between bits 14 to 21 from the start of the frame, the frame is rejected. If a double zero fault is detected after 21 bits from the start of the frame and before detection the start frame delimiter, the frame is also rejected. When the registers match, the hunt mode is terminated, and character assembly begins.

When the receiver detects the first bytes of the frame, the Ethernet controller will perform address recognition functions on the frame (see 7.10.23.11 Ethernet Address Recognition). The receiver can receive physical (individual), group (multicast), and broadcast addresses. No Ethernet receive frame data is written to memory until the internal address recognition algorithm is complete, which improves bus utilization in the case of frames not addressed to this station.

The receiver can also work with an external CAM. See 7.10.23.7 CAM Interface for more details. In the case of an external CAM, frame reception continues normally unless the CAM specifically signals the frame to be rejected.

If a match is detected, the Ethernet controller will fetch the next Rx BD and, if it is empty, will start to transfer the incoming frame to the Rx BD’s associated data buffer. If a collision is detected during the frame, the Rx BDs associated with this frame are reused. Thus, no col-

lision frames are presented to the user except late collisions, which indicate serious LAN problems.

When the data buffer has been filled, the Ethernet controller clears the E-bit in the Rx BD and generates an interrupt if the I-bit is set. If the incoming frame exceeds the length of the data buffer, the Ethernet controller will fetch the next Rx BD in the table and, if it is empty, will continue to transfer the rest of the frame to this BD's associated data buffer.

The Rx BD length is determined in the MRBLR value in the SCC general-purpose parameter RAM. The user should program MRBLR to be at least 64 bytes.

During reception, the Ethernet controller will check for a frame that is too short or too long. When the frame ends (carrier sense is negated), 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. This enables software to correctly recognize the frame-too-long condition.

When the receive frame is complete, the Ethernet controller has the option to sample one byte from the port B parallel I/O (PB15–PB8) and append this byte to the end of the last Rx BD in the frame. For any of the PB15–PB8 pins that are defined as outputs, the contents of the PBDAT latch is read, rather than the pin itself. Although this capability is useful for CAM applications, it may be used whether or not an external CAM is present. The sampling occurs at the end of frame reception.

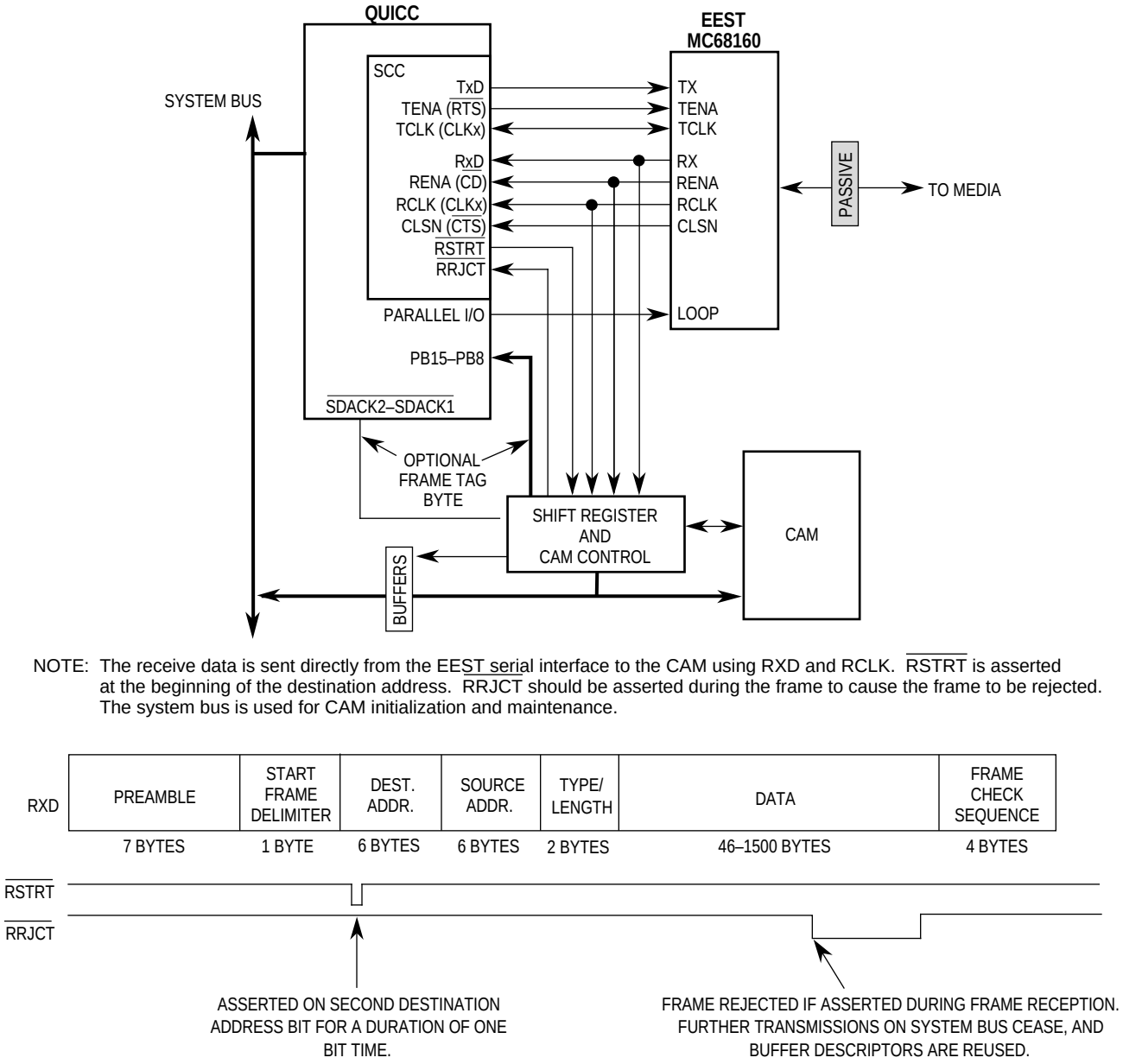
The Ethernet controller then sets the L-bit in the Rx BD, writes the other frame status bits into the Rx BD, and clears the E-bit. The Ethernet controller next generates a maskable interrupt, indicating that a frame has been received and is in memory. The Ethernet controller then waits for a new frame.

The Ethernet controller receives serial data LSB first.

7.10.23.7 CAM INTERFACE. The Ethernet controller has two options for connecting to an external CAM: a serial interface option and a system bus interface option. Actually, both options may be used at once (there is no mode bit to select them); however, they are described independently for clarity. To implement an option, the user only needs to enable the particular pins that are desirable.

Both options use a reject pin on the QUICC to signify that the current frame is to be discarded. The QUICC internal address recognition logic may be used in combination with an external CAM. See 7.10.23.11 Ethernet Address Recognition for more details.

The serial interface option is shown in Figure 7-66. The QUICC outputs a receive start ($\overline{\text{RSTRT}}$) signal when the start frame delimiter is recognized. The $\overline{\text{RSTRT}}$ signal is asserted for just one bit time on the second destination address bit.



NOTE: The receive data is sent directly from the EEST serial interface to the CAM using RXD and RCLK. $\overline{\text{RSTRT}}$ is asserted at the beginning of the destination address. RRJCT should be asserted during the frame to cause the frame to be rejected. The system bus is used for CAM initialization and maintenance.

Figure 7-68. QUICC Ethernet Serial CAM Interface

The CAM control logic uses $\overline{\text{RSTRT}}$, in combination with the RXD and RCLK signals, to store the destination address, source address, etc. and generate writes to the CAM for address recognition. In addition, the RENA signal supplied from the EEST may be used to abort the comparison if a collision occurs on the receive frame.

After the comparison occurs, the CAM control logic asserts the receive reject ($\overline{\text{RRJCT}}$) pin, if the current receive frame should be rejected. The QUICC Ethernet controller will then immediately stop writing data to system memory and will reuse the buffer(s) for the next frame. If the CAM wishes to accept the frame, the CAM control logic does nothing ($\overline{\text{RRJCT}}$ is not asserted). If $\overline{\text{RRJCT}}$ is asserted, it must be asserted prior to the end of the receive frame.

Additionally, the CAM control logic may wish to provide additional information on the PB15–PB8 pins. The QUICC Ethernet controller will write this additional byte to memory during the last SDMA write if the SIP bit is set in the PSMR. This information tag is sampled by the QUICC Ethernet controller as the last FCS byte is read from the receive FIFO. The information TAG should be provided by the CAM control logic no later than when RENA is negated at the end of a non-collision frame, and should be held stable on the PB15–PB8 pins until the $\overline{\text{SDACK2}}$–$\overline{\text{SDACK1}}$ pins signal that the tag byte is being written to memory.

The parallel interface option is shown in Figure 7-69. The QUICC outputs two signals every time it writes Ethernet frame data to system memory. The signals SDMA acknowledge ($\overline{\text{SDACK2}}$–$\overline{\text{SDACK1}}$), are asserted during all bus cycles on which Ethernet frame data is written to memory. (These signals are not used for other protocols.)

The CAM control logic uses these pins to enable the CAM writes simultaneously with system memory writes. In this way, the CAM captures the frame data at the same time that it is being written to system memory. The chief advantage of this approach is that the data is already in parallel form when it leaves the QUICC.

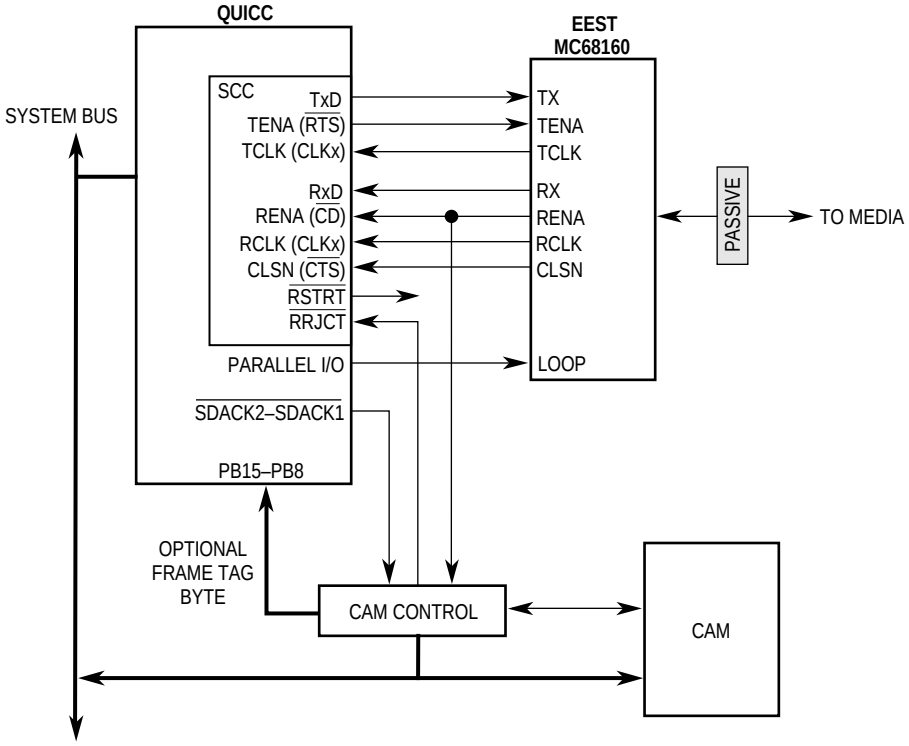
The $\overline{\text{SDACK2}}$–$\overline{\text{SDACK1}}$ signals are asserted during all bus cycles writes of the frame data. A certain $\overline{\text{SDACK2}}$–$\overline{\text{SDACK1}}$ combination specifically identifies the first 32-bits of the frame, another identifies all mid-frame data, and a third combination identifies the last 32-bit bus write of the frame (only if the tag byte is appended). The tag byte is appended from the sample of PB15–PB8 if the SIP bit is set in the PSMR. The tag byte will always be in byte 3 of the last 32-bit write. The Rx BD Data Length does not include tag byte in the length calculation.

If the system memory is 32 bits, then the QUICC 32-bit write will take one bus cycle. If the system memory is 16 bits or 8 bits, then the QUICC 32-bit write will take two or four bus cycles. In any case, the $\overline{\text{SDACK2}}$–$\overline{\text{SDACK1}}$ signals are valid on each bus cycle of a 32-bit write cycle and only during bus cycles associated with the Ethernet receiver.

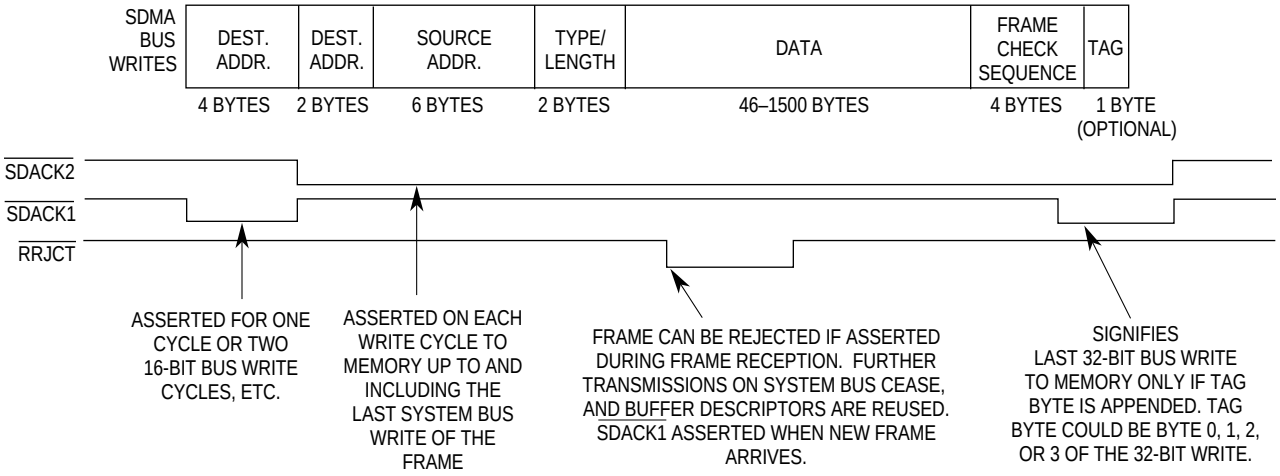
Additionally, the user may choose a unique function code (FC3–FC0) associated with the SDMA receive channel associated with the Ethernet SCC to have an alternate method of identifying accesses from this SCC.

NOTE

The tag byte is always written to byte 3 of the last SDMA write to the buffer, and is not necessarily appended to the last byte of the frame. The Rx BD Data Length does not show the length of the tag byte in the frame. Also the $\overline{\text{SDACK2-1}}$ signals will equal "00" whenever the frame length is not an even multiple of 4 (i.e., it does not depend on whether the tag byte is appended).



NOTE: The receive data is sent to the CAM as it is written to system memory. The SDACK2–SDACK1 signals are used to identify the destination address and any other frame bytes desired. The RSTRT signal is not required in this configuration, although it is still available.



NOTE: The diagram shows SDMA system bus writes, not data on the RXD pin. Other bus activity may occur between successive 32-bit writes. In such a case, the SDACK2–1 pins would not be asserted for other bus activity.

Figure 7-69. QUICC Ethernet Parallel CAM Interface;

7.10.23.8 ETHERNET MEMORY MAP. When configured to operate in Ethernet mode, the QUICC overlays the structure described in Table 7-5 onto the protocol-specific area of the SCC parameter RAM described in Table 7-11.

Table 7-11. Ethernet-Specific Parameters

Address	Name	Width	Description	User Writes
SCC Base + 30	**C_PRES**	Long	Preset CRC	$FFFFFFFF
SCC Base + 34	**C_MASK**	Long	Constant MASK for CRC	$DEBB20E3
SCC Base + 38	**CRCEC**	Long	CRC Error Counter	
SCC Base + 3C	**ALEC**	Long	Alignment Error Counter	
SCC Base + 40	**DISFC**	Long	Discard Frame Counter	
SCC Base + 44	**PADS**	Word	Short Frame PAD character	
SCC Base + 46	**RET_Lim**	Word	Retry Limit Threshold	15 dec
SCC Base + 48	RET_cnt	Word	Retry Limit Counter	
SCC Base + 4A	**MFLR**	Word	Maximum Frame Length Register	1518 dec
SCC Base + 4C	**MINFLR**	Word	Minimum Frame Length Register	64 dec
SCC Base + 4E	**MAXD1**	Word	Max DMA1 Length Register	1518 dec
SCC Base + 50	**MAXD2**	Word	Max DMA2 Length Register	1518 dec
SCC Base + 52	MAXD	Word	Rx Max DMA	
SCC Base + 54	**DMA_cnt**	Word	Rx DMA Counter	
SCC Base + 56	MAX_b	Word	Max BD Byte Count	
SCC Base + 58	**GADDR1**	Word	Group Address Filter 1	
SCC Base + 5A	**GADDR2**	Word	Group Address Filter 2	
SCC Base + 5C	**GADDR3**	Word	Group Address Filter 3	
SCC Base + 5E	**GADDR4**	Word	Group Address Filter 4	
SCC Base + 60	TBUF0.data0	Long	Save Area 0 - Current Frame	
SCC Base + 64	TBUF0.data1	Long	Save Area 1 - Current Frame	
SCC Base + 68	TBUF0.rba0	Long		
SCC Base + 6C	TBUF0.crc	Long		
SCC Base + 70	TBUF0.bcnt	Word		
SCC Base + 72	**PADDR1_L**	Word	Physical Address 1 (LSB)	
SCC Base + 74	**PADDR1_M**	Word	Physical Address 1	
SCC Base + 76	**PADDR1_H**	Word	Physical Address 1 (MSB)[2]	
SCC Base + 78	**P_Per**	Word	Persistence	
SCC Base + 7A	RFBD_ptr	Word	Rx First BD Pointer	
SCC Base + 7C	TFBD_ptr	Word	Tx First BD Pointer	
SCC Base + 7E	TLBD_ptr	Word	Tx Last BD Pointer	
SCC Base + 80	TBUF1.data0	Long	Save Area 0 - Next Frame	
SCC Base + 84	TBUF1.data1	Long	Save Area 1 - Next Frame	
SCC Base + 88	TBUF1.rba0	Long		
SCC Base + 8C	TBUF1.crc	Long		
SCC Base + 90	TBUF1.bcnt	Word		
SCC Base + 92	TX_len	Word	Tx Frame Length Counter	
SCC Base + 94	**IADDR1**	Word	Individual Address Filter 1	
SCC Base + 96	**IADDR2**	Word	Individual Address Filter 2	
SCC Base + 98	**IADDR3**	Word	Individual Address Filter 3	
SCC Base + 9A	**IADDR4**	Word	Individual Address Filter 4	
SCC Base + 9C	BOFF_CNT	Word	Backoff Counter	

Table 7-11. Ethernet-Specific Parameters

SCC Base + 9E	**TADDR_L**	Word	Temp Address (LSB)	
SCC Base + A0	**TADDR_M**	Word	Temp Address	
SCC Base + A2	**TADDR_H**	Word	Temp Address (MSB)[2]	

NOTE:
1. The boldfaced items should be initialized by the user.
2. The address should be wrtten in little endian, not Motorola big endian format (i.e., physical address 112233445566 should be written PADDR1_L= 6655, PADDR1_M=4433, PADDR1_H=2211.

C_PRES. For the 32-bit CRC-CCITT, C_PRES should be initialized with $FFFFFFFF.

C_MASK. For the 32-bit CRC-CCITT, C_MASK sh ould be initialized with $DEBB20E3.

CRCEC, ALEC, and DISFC. These 32-bit (modulo 2^{32}) counters are maintained by the CP. They may be initialized by the user while the channel is disabled. CRCEC is incremented for each received frame with a CRC error, except it does not includes frames not addressed to the user, 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 includes frames not addressed to the user, 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 this counter to be incremented.

PADS. Into this 16-bit register the user writes the pattern of the pad characters that should be sent when short frame padding is implemented. The byte pattern written to the register may be any value, but both the high and low bytes should be the same.

RET_Lim. The user writes the number of retries that should be made to transmit a frame into this 16-bit register. This value is typically 15 decimal. If the frame is not transmitted after this limit is reached, an interrupt may be generated. RET_cnt is a temporary down-counter used to count the number of retries made.

MFLR. The Ethernet controller checks the length of an incoming Ethernet frame against the user-defined value given in this 16-bit register. Typically this register is set to 1518 decimal. If this limit is exceeded, the remainder of the incoming frame is discarded, and the LG (Rx frame too long) bit is set in the last Rx BD belonging to that frame. The Ethernet controller will report the frame status and the frame length in the last Rx BD.

MFLR is defined as all the in-frame bytes between the start frame delimiter and the end of the frame (destination address, source address, length, LLC data, PAD, and FCS). DMA_cnt is a temporary down-counter used to track the frame length.

MINFLR. The Ethernet controller checks the length of an incoming Ethernet frame against the user-defined value given in this 16-bit register. Typically this register is set to 64 decimal. If the received frame length is less than the register value, then this frame is discarded unless the RSH (receive short frames) bit in the PSMR is set. If RSH is set, then the SH (Rx frame too short) bit is set in the last Rx BD belonging to that frame. For transmit operation, if the frame is too short, the Ethernet controller will add PADs to the transmitted frame

(according to the PAD bit in the Tx BD and the PAD value in the parameter RAM). PADs will be added to make the transmit frame MINFLR bytes in length.

MAXD1. This parameter gives the user the ability to stop system bus writes from occurring after a frame has exceeded a certain size. The value of this register is valid only if an address match was detected. The Ethernet controller checks the length of an incoming Ethernet frame against the user-defined value given in this 16-bit register. Typically, this register is set to 1518 decimal. If this limit is exceeded, the remainder of the incoming frame is discarded. The Ethernet controller waits to the end of the frame (or until MFLR bytes have been received) and reports the frame status and the frame length in the last Rx BD.

MAXD2. This parameter gives the user the ability to stop system bus writes from occurring after a frame has exceeded a certain size. The value of this register is valid in promiscuous mode when no address match was detected. The Ethernet controller checks the length of an incoming Ethernet frame against the user-defined value given in this 16-bit register. Typically, this register is set to 1518 decimal. If this limit is exceeded, the remainder of the incoming frame is discarded. The Ethernet controller waits to the end of the frame (or until MFLR bytes have been received) and reports the frame status and the frame length in the last Rx BD.

In a monitor station, MAXD2 can be programmed to a value much less than MAXD1 to receive entire frames addressed to this station, but receive only the headers of all other frames.

GADDR1–4. These four registers 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.

PADDR1. The user writes the 48-bit individual address of this station into this location. PADDR1_L is the lowest order word, and PADDR1_H is the highest order word.

P_Per**.** This parameter allows the Ethernet controller to be less aggressive in its behavior following a collision. Normally, this parameter should be set to $0000. To decrease the aggressiveness of the Ethernet controller, P_Per can be set to a value from 1 to 9, with 9 being the least aggressive. The P_Per value is added to the retry count in the backoff algorithm to reduce the probability of transmission on the next time slot.

NOTE

The use of P_Per is fully allowed within Ethernet/802.3 specifications. In a heavily congested Ethernet LAN, a less aggressive backoff algorithm used by multiple stations on the LAN increases the overall LAN throughput by reducing the probability of collisions.

The SBT bit in the PSMR offers another way to reduce the aggressiveness of the Ethernet controller.

IADDR1–4. These four registers are used in the hash table function of the individual addressing mode. The user may write zeros 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.

TADDR. This parameter allows the user to add and delete addresses from the individual and group hash tables. After placing an address in TADDR, the user would then issue the SET GROUP ADDRESS command. TADDR_L is the lowest order word, and TADDR_H is the highest order word.

7.10.23.9 ETHERNET PROGRAMMING MODEL. The host configures SCC to operate as an Ethernet controller by the MODE bits in the GSMR.

The receive errors (collision, overrun, nonoctet aligned frame, short frame, frame too long, and CRC error) are reported through the Rx BD. The transmit errors (underrun, heartbeat, late collision, retransmission limit, and carrier sense lost) are reported through the Tx BD.

Several bit fields in the GSMR must be programmed to special values for Ethernet. See the GSMR for more details. The user should program the DSR as shown below. The 6 bytes of preamble programmed in the GSMR, in combination with the programming of the DSR shown below, causes 8 bytes of preamble on transmit (including the 1-byte start delimiter with the value $D5).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYN2 = $D5								SYN1 = $55							

7.10.23.10 ETHERNET COMMAND SET. Ethernet:Ethernet Command SetThe following transmit and receive commands are issued to the CR.

NOTE

Before issuing the CP RESET command, configure the TENA ($\overline{RTS}$) pin to be an input. See step 3 of 7.10.23.23 SCC Ethernet Example. for more information.

7.10.23.10.1 Transmit Commands. The following paragraphs describe the Ethernet transmit commands.

STOP TRANSMIT Command**.** When used with the Ethernet controller, this command violates specified behavior of an Ethernet/IEEE 802.3 station. It should not be used.

GRACEFUL STOP TRANSMIT Command**.** The channel GRACEFUL STOP TRANSMIT command is used to stop transmission in an orderly way. It stops transmission after the current frame has completed transmission or undergoes a collision (immediately if there is no frame being transmitted). The GRA bit in the SCCE will be set once transmission has stopped. After transmission ceases, the Ethernet transmit parameters, including BDs, may be modified by the user. The TBPTR will point to the next Tx BD in the table. Transmission will begin once the R-bit of the next BD is set and the RESTART TRANSMIT command is issued.

NOTE

If the GRACEFUL STOP TRANSMIT command is issued and the current transmit frame ends in a collision, the TBPTR will point to the beginning of the collided frame, with the R-bit still set in the Tx BD (i.e., the frame will look as if it were never transmitted).

RESTART TRANSMIT Command*.* The RESTART TRANSMIT command enables the transmission of characters on the transmit channel. This command is expected by the Ethernet controller after a GRACEFUL STOP TRANSMIT command or after a transmitter error (underrun, retransmission limit reached, or late collision). The Ethernet controller will resume transmission from the current TBPTR in the channel's Tx BD table.

INIT TX PARAMETERS Command*.* This command initializes all the transmit parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the transmitter is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both transmit and receive parameters.

7.10.23.10.2 Receive Commands. The following paragraphs describe the Ethernet receive commands.

ENTER HUNT MODE Command*.* After a hardware or software reset and the enabling of the channel in the SCC mode register, the channel is in the receive enable mode and will use the first BD in the table.

The ENTER HUNT MODE command is generally used to force the Ethernet receiver to abort reception of the current frame and enter the hunt mode. In the hunt mode, 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 current receive buffer is closed, and the CRC calculation is reset. Further frame reception will use the next Rx BD.

CLOSE Rx BD Command*.* This command should not be used with the Ethernet controller.

INIT RX PARAMETERS Command*.* This command initializes all the receive parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the receiver is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both receive and transmit parameters.

7.10.23.10.3 SET GROUP ADDRESS Command. The SET GROUP ADDRESS command is used to set a bit in one of the 64 bits of the four individual/group address hash filter registers (GADDR1–4 or IADDR1–4). 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 RISC controller 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 zero in the I/G bit implies an individual address, and a one in the I/G bit implies a group address.

This command may 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 should be disabled, the hash table registers should be cleared, and the SET GROUP ADDRESS command must be executed for the remaining desired addresses. This is required because the hash table may have mapped multiple addresses to the same hash table bit.

7.10.23.11 ETHERNET ADDRESS RECOGNITION. The Ethernet controller can filter the received frames based on different addressing types: physical (referred to as individual), group (referred to as multicast), broadcast (an 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 7-70.

In the physical type of address recognition, the Ethernet controller will compare the destination address field of the received frame with the physical address that the user programs in the PADDR1. Alternatively, the user may perform address recognition on multiple individual addresses using the IADDR1–4 hash table. See 7.10.23.12 Hash Table Algorithm for more information.

In the group type of address recognition, the Ethernet controller will determine whether the group address is a broadcast address. If broadcast addresses are enabled, then the frame is accepted. If the group address is not a broadcast address, then the user may perform address recognition on multiple group addresses using the GADDR1–4 hash table. See 7.10.23.12 Hash Table Algorithm for more information.

In the promiscuous mode, the Ethernet controller will receive all the incoming frames regardless of their address, unless the $\overline{\text{RRJCT}}$ pin is asserted.

If an external CAM is used for address recognition, then the user should select the promiscuous mode, and the frame can be rejected by assertion of the $\overline{\text{RRJCT}}$ pin during the reception of the frame. The on-chip address recognition functions may be used in addition to the external CAM address recognition functions.

NOTE

If the external CAM is used to store addresses that should be rejected, rather than accepted, then the use of the $\overline{\text{RRJCT}}$ pin by the CAM should be logically inverted.

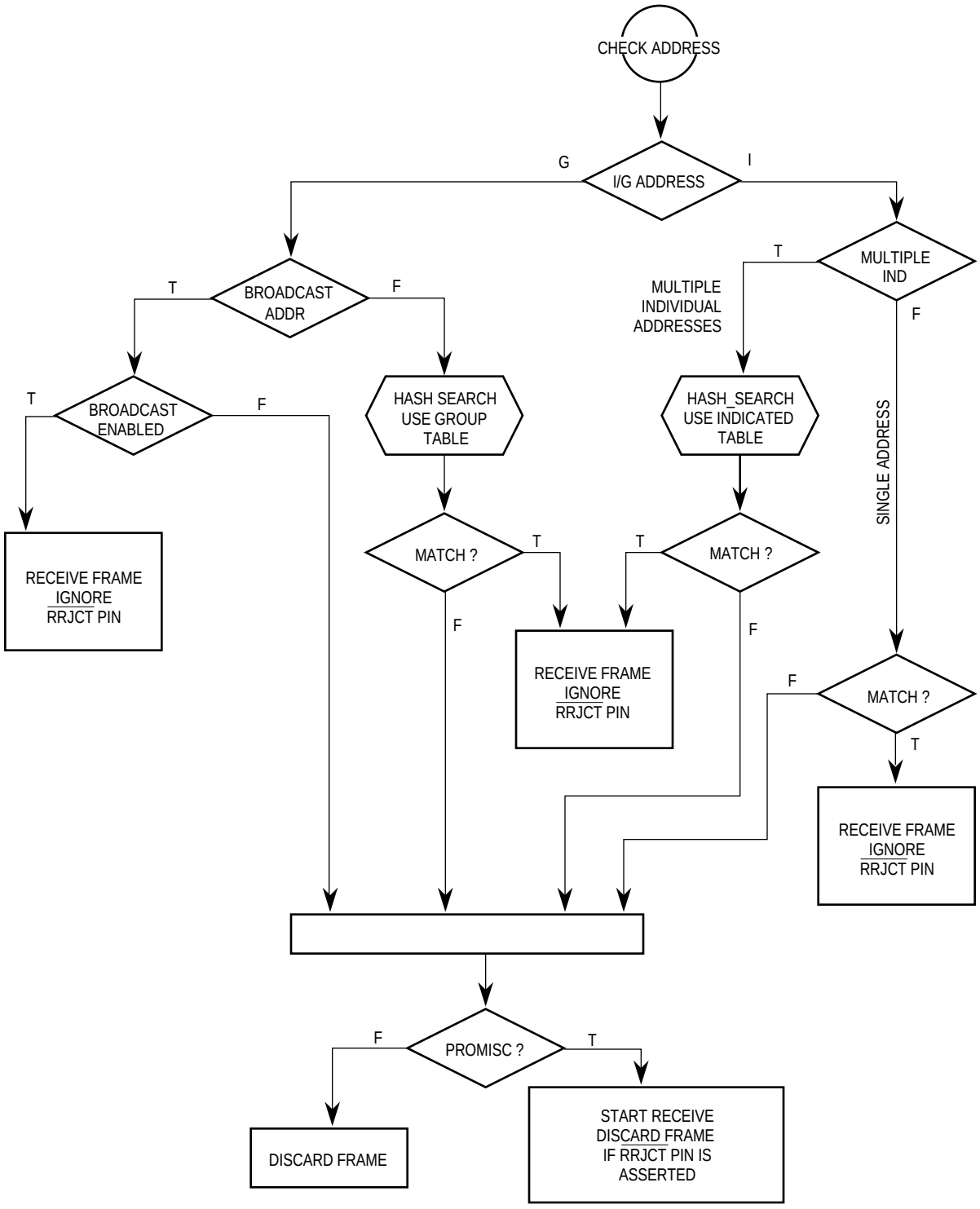


Figure 7-70. Ethernet Address Recognition Flowchart

7.10.23.12 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. The 64 bins are represented by 64 bits stored in GADDR1–4 or IADDR1–4.

When the SET GROUP ADDRESS command is executed, the Ethernet controller maps the selected 48-bit address into one of the 64 bits. This is performed 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 four GADDRs or IADDRs, and bits 29–26 of the CRC result select the bit within the selected register.

When a frame is received by the Ethernet controller, 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. 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 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

Better performance is achieved by using the group hash table and individual hash table at the same time. 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 will be set, preventing only a small fraction of the frames from reaching memory. In such instances, an external CAM is advised if the extra bus utilization cannot be tolerated. See 7.10.23.7 CAM Interface for more details.

NOTE

The hash tables cannot be used to reject frames that match a set of entered addresses because unintended addresses will be mapped to the same bit in the hash table. Thus, an external CAM must be used to implement this function.

7.10.23.13 INTERPACKET GAP TIME. The minimum interpacket gap time for back-to-back transmission is 9.6 μs. The receiver can receive back-to-back frames with this minimum spacing. In addition, after the backoff algorithm, the transmitter will wait for carrier sense to be negated before retransmitting the frame. The retransmission will begin 9.6 μs after carrier sense is negated if carrier sense stays negated for at least 6.4 μs.

7.10.23.14 COLLISION HANDLING. If a collision occurs during frame transmission, the Ethernet controller will continue the transmission for at least 32 bit times, transmitting a JAM pattern consisting of 32 ones. If the collision occurs during the preamble sequence, the JAM pattern will be sent after the end of the preamble sequence.

If a collision occurs within 64 byte times, the retry process is initiated. The transmitter will wait a random number of slot times. A slot time is 512 bit times (52 μs). If collision occurs

after 64 byte times, then no retransmission is performed, and the buffer is closed with an LC error indication.

If a collision occurs during frame reception, the reception is stopped. This error will be reported in the BD only if the length of this frame is greater than or equal to the MINFLR or if the RSH mode is enabled in the PSMR.

7.10.23.15 INTERNAL AND EXTERNAL LOOPBACK. Both internal and external loopback are supported by the Ethernet controller. In loopback 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 the LPB bit in the PSMR and the DIAG bits in the GSMR. Because of the full-duplex nature of the loopback operation, the performance of the other SCCs will be degraded.

Internal loopback disconnects the SCC from the SI. The receive data is connected to the transmit data, and the receive clock is connected to the transmit clock. Both FIFOs are used. The transmitted data from the transmit FIFO is received immediately into the receive FIFO. There is no heartbeat check in this mode. In this mode TENA should be configured to be a general purpose output.

(PCPAR[0]=0, PCDIR[0]=1, PCDAT[0]=0) and the HBC bit in the PSMR should be 0.

In external loopback operation, the Ethernet controller listens for receive data from the EEST at the same time that it is transmitting.

7.10.23.16 ETHERNET ERROR-HANDLING PROCEDURE. The Ethernet controller reports frame reception and transmission error conditions using the channel BDs, the error counters, and the Ethernet event register.

7.10.23.16.1 Transmission Errors. The following paragraphs describe various types of Ethernet transmission errors.

Transmitter Underrun*.* If this error occurs, the channel sends 32 bits that ensure a CRC error, terminates buffer transmission, closes the buffer, sets the UN bit in the Tx BD, and sets TXE in the Ethernet event register. The channel will resume transmission after reception of the RESTART TRANSMIT command.

Carrier Sense Lost During Frame Transmission*.* When this error occurs and no collision is detected in this frame, the channel sets the CSL bit in the Tx BD, sets TXE in the Ethernet event register, and continues the buffer transmission normally. No retries are performed as a result of this error.

Retransmission Attempts Limit Expired*.* When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the RL bit in the Tx BD, and sets TXE. The channel will resume transmission after reception of the RESTART TRANSMIT command.

Late Collision*.* When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the LC bit in the Tx BD, and sets TXE. The channel will resume transmission after reception of the RESTART TRANSMIT command.

NOTE

The definition of what constitutes a late collision is made in the LCW bit in the PSMR.

Heartbeat*.* Some transceivers have a self-test feature called “heartbeat” or “signal quality error.” To signify a good self-test, the transceiver indicates a collision to the QUICC within 20 clocks after completion of a frame transmitted by the Ethernet controller. This indication of a collision does not imply a real collision error on the network, but is rather an indication that the transceiver still seems to be functioning properly. This is called the heartbeat condition.

If the HBC bit is set in the Ethernet mode register and the heartbeat condition is not detected by the QUICC after a frame transmission, then a heartbeat error occurs. When this error occurs, the channel closes the buffer, sets the HB bit in the Tx BD, and generates the TXE interrupt if it is enabled.

7.10.23.16.2 Reception Errors. The following paragraphs describe various types of Ethernet reception errors.

Overrun Error*.* The Ethernet controller maintains an internal FIFO for receiving data. If a receiver FIFO overrun occurs, the channel writes the received data byte to the internal FIFO over the previously received byte. The previous data byte and the frame status are lost. The channel closes the buffer, sets the OV bit in the Rx BD, sets RXF in the Ethernet event register, and increments the discarded frame counter (DISFC). The receiver then enters the hunt mode.

Busy Error. A frame was received and discarded due to lack of buffers. The channel sets BSY in the Ethernet event register and increments the discarded frame counter (DISFC).

Nonoctet Error (Dribbling Bits)*.* The Ethernet controller can handle up to seven dribbling bits when the receive frame terminates nonoctet aligned. The Ethernet controller checks the CRC of the frame on the last octet boundary. If there is a CRC error, then the frame nonoctet aligned (NO) error is reported, the RXF bit is set, and the alignment error counter (ALEC) is incremented. If there is no CRC error, then no error is reported.

CRC Error*.* When a CRC error occurs, the channel closes the buffer, sets the CR bit in the Rx BD, and sets the RXF bit. The channel also 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 the CRC error may be ignored if checking is not required.

7.10.23.17 ETHERNET MODE REGISTER (PSMR). The Ethernet mode register is a 16-bit, memory-mapped, read-write register that controls the SCC operation. The term Ethernet mode register refers to the PSMR of the SCC when that SCC is configured for Ethernet. This register is cleared at reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HBC	FC	RSH	IAM	CRC		PRO	BRO	SBT	LPB	SIP	LCW	NIB			FDE

HBC—Heartbeat Checking

0 = No heartbeat checking is performed. Do not wait for a collision after transmission.
1 = Wait 20 transmit clocks (2 μs) for a collision asserted by the transceiver after transmission. The HB bit in the Tx BD is set if the heartbeat is not heard within 20 transmit clocks.

FC—Force Collision

0 = Normal operation.
1 = Force collision. The channel forces a collision on transmission of every transmit frame. The QUICC should be configured in loopback operation when using this feature, which allows the QUICC collision logic to be tested by the user. It will result in the retry limit being exceeded for each transmit frame.

RSH—Receive Short Frames

0 = Discard short frames (less than MINFLR in length)
1 = Receive short frames

IAM—Individual Address Mode

0 = Normal operation. A single 48-bit physical address stored in PADDR1 is checked on receive.
1 = The individual hash table is used to check all individual addresses that are received.

CRC—CRC Selection

00 = Reserved.
01 = 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.

PRO—Promiscuous

0 = Check the destination address of the incoming frames.
1 = Receive the frame regardless of its address, unless the $\overline{\text{RRJCT}}$ pin is asserted during the frame reception.

BRO—Broadcast Address

0 = Receive all frames containing the broadcast address.
1 = Reject all frames containing the broadcast address unless PRO = 1.

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 may be used in combination with the SBT bit (or in place of the SBT bit), if desired.

LPB—Loopback Operation

0 = Normal Operation.
1 = Loopback operation. The channel is configured into internal or external loopback operation as determined by the DIAG bits in the GSMR. For external loopback, the DIAG bits should be configured for normal operation. For internal loopback, the DIAG bits should be configured for loopback operation.

SIP—Sample Input Pins

0 = Normal operation.
1 = After the frame is received, the value on the PB15–PB8 pins is sampled and written to the end of the last receive buffer of the frame. This value is called a tag byte. If the frame is discarded, the Ethernet:tag byte is also discarded.

LCW—Late Collision Window

0 = The definition of a late collision is any collision that occurs 64 or more bytes from the preamble.
1 = The definition of a late collision is any collision that occurs 56 or more bytes from the preamble.

NIB—Number of Ignored Bits

This parameter determines how soon after RENA assertion that the Ethernet controller should begin looking for the start frame delimiter. In most situations, the user would select 22 bits.

000 = Begin searching for the SFD 13 bits after the assertion of RENA.
001 = Begin searching for the SFD 14 bits after the assertion of RENA.
010 = Begin searching for the SFD 15 bits after the assertion of RENA.
011 = Begin searching for the SFD 16 bits after the assertion of RENA.
100 = Begin searching for the SFD 21 bits after the assertion of RENA.
101 = Begin searching for the SFD 22 bits after the assertion of RENA.
110 = Begin searching for the SFD 23 bits after the assertion of RENA.
111 = Begin searching for the SFD 24 bits after the assertion of RENA.

FDE—Full Duplex Ethernet (Bit 0 of PSMR)

0 = Disable full duplex ethernet mode.
1 = Enable full duplex ethernet.

NOTE

When this bit is set to 1 the LPB bit must also be set to 1.

7.10.23.18 ETHERNET RECEIVE BUFFER DESCRIPTOR (RX BD). The Ethernet controller uses the Rx BD to report information about the received data for each buffer. Figure 7-71 shows an Ethernet Rx BD example.

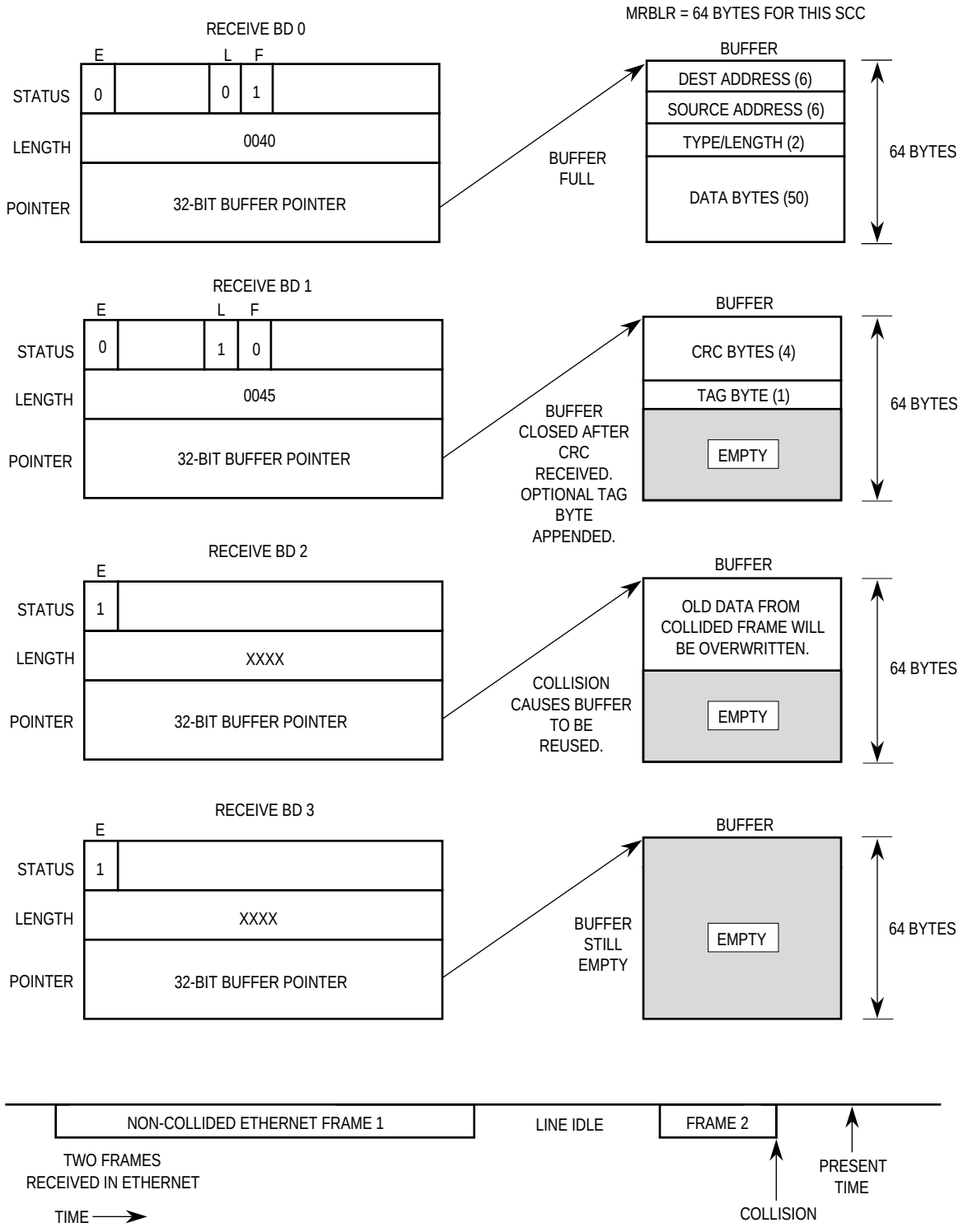


Figure 7-71. Ethernet Rx BD Example

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**E**	—	**W**	**I**	L	F	—	—	—	—	LG	NO	SH	CR	OV	CL
OFFSET + 2	DATA LENGTH															
OFFSET + 4	**RX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: Entries in boldface must be initialized by the user.

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 CPU32+ 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 Rx 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 CPU32+ core should not write any fields of this Rx BD.

Bits 14, 9–6—Reserved

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 BD s in this table is programmable and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been used.

1 = The RXB bit or RXF bit in the Ethernet event register will be set when this buffer has been used by the Ethernet controller. These two bits may cause interrupts if they are enabled.

L—Last in Frame

This bit is set by the Ethernet controller when this buffer is the last in a frame. This implies the end of the frame or reception of an error, in which case one or more of the CL, OV, CR, SH, NO, and LG bits are set. The Ethernet controller will write the number of frame octets to the data length field.

0 = The buffer is not the last in a frame.

1 = The buffer is the last in a frame.

F—First in Frame

This bit is set by the Ethernet controller when this buffer is the first in a frame.

0 = The buffer is not the first in a frame.

1 = The buffer is the first in a frame.

M—Miss

This bit is set by the Ethernet controller for frames that were accepted in promiscuous mode, but were flagged as a "miss" by the internal address recognition. Thus, while in pro-

miscuous mode, the user can use the Miss bit to quickly determine whether the frame was destined to this station. This bit is valid only if the L bit is set.

0 = The frame was received because of an address recognition hit.
1 = The frame was received because of promiscuous mode.

LG—Rx Frame Length Violation

A frame length greater than the maximum defined for this channel was recognized (only the maximum-allowed number of bytes is written to the data buffer).

NO—Rx Nonoctet Aligned Frame

A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error.

SH—Short Frame

A frame length that was less than the minimum defined for this channel was recognized.

This indication is possible only if the RSH bit is set in the PSMR.

CR—Rx CRC Error

This frame contains a CRC error.

OV—Overrun

A receiver overrun occurred during frame reception.

CL—Collision

This frame was closed because a collision occurred during frame reception. This bit will be set only if a late collision occurred or if the RSH bit is enabled in the PSMR. The late collision definition is determined by the LCW bit in the PSMR.

Data Length

The data length is the number of octets written by the CP into this BD's data buffer. It is written by the CP once as the buffer is closed.

When this BD is the last BD in the frame (L = 1), the data length contains the total number of frame octets (including four bytes for CRC).

NOTE

The actual amount of memory allocated for this buffer should be greater than or equal to the contents of the MRBLR.

Rx Data Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, may reside in either internal or external memory. This pointer must be divisible by 4.

7.10.23.19 ETHERNET TRANSMIT BUFFER DESCRIPTOR (TX BD). Data is presented to the Ethernet controller for transmission on an SCC channel by arranging it in buffers ref-

erenced by the channel's Tx BD table. The Ethernet controller confirms transmission or indicates error conditions using the BDs to inform the host that the buffers have been serviced.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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																

NOTE: Entries in boldface must be initialized by the user.

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.

PAD—Short Frame Padding

This bit is valid only when the L-bit is set; otherwise, it is ignored.

0 = Do not add PADs to short frames.

1 = Add PADs to short frames. Pad bytes will be inserted until the length of the transmitted frame equals the MINFLR. The PAD bytes are stored in PADs in the parameter RAM.

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 receive incoming data into the first BD in the table (the BD pointed to by TBASE). The number of Tx BD s in this table is programmable and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

NOTE

The TxBD table must contain more than one BD in Ethernet mode.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.

1 = The TXB bit or TXE bit will be set in the Ethernet event register after this buffer has been serviced. These bits can cause interrupts if they are enabled.

L—Last

0 = This is not the last buffer in the transmit frame.

1 = This is the last buffer in the current transmit frame.

TC—Tx CRC

This bit is 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.

The following status bits are written by the Ethernet controller after it has finished transmitting the associated data buffer.

DEF—Defer Indication

This frame had a collision before being successfully sent. Useful for channel statistics.

HB—Heartbeat

The collision input was not asserted within 20 transmit clocks following the completion of transmission. This bit cannot be set unless the HBC bit is set in the PSMR.

LC—Late Collision

A collision has occurred after the number of bytes defined with the LCW bit in the PSMR (either 56 or 64) have been transmitted. The Ethernet controller will terminate the transmission.

RL—Retransmission Limit

The transmitter has failed Retry Limit + 1 attempts to successfully transmit a message due to repeated collisions on the medium.

RC—Retry Count

These four bits indicate the number of retries required before this frame was successfully transmitted. If RC = 0, then the frame was transmitted correctly the first time. If RC = 15 and RET_Lim = 15 in the parameter RAM, then 15 retries were required. If RC = 15 and RET_Lim > 15 in the parameter RAM, then 15 or more retries were required.

UN—Underrun

The Ethernet controller encountered a transmitter underrun condition while transmitting the associated data buffer.

CSL—Carrier Sense Lost

Carrier sense was lost during frame transmission.

Data Length

The data length is the number of octets the Ethernet controller should transmit from this BD's 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, may be even or odd. The buffer may reside in either internal or external memory. This value is never modified by the CP.

7.10.23.20 ETHERNET EVENT REGISTER (SCCE). The SCCE is called the Ethernet event register when the SCC is operating as an Ethernet controller. It is a 16-bit register used to report events recognized by the Ethernet channel and to generate interrupts. On recognition of an event, the Ethernet controller will set the corresponding bit in the Ethernet event register. Interrupts generated by this register may be masked in the Ethernet mask register. An example of interrupts that may be generated in the HDLC protocol is given in Figure 7-72.

The Ethernet event register is a memory-mapped register that 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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	—	—	—	—	—	—	—	GRA	—	—	TXE	RXF	BSY	TXB	RXB

Bits 15–8, 6–5—Reserved

GRA—Graceful Stop Complete

A graceful stop, which was initiated by the GRACEFUL STOP TRANSMIT command, is now complete. This bit is set as soon the transmitter has finished transmitting any frame that was in progress when the command was issued. It will be set immediately if no frame was in progress when the command was issued.

TXE—Tx Error

An error occurred on the transmitter channel.

RXF—Rx Frame

A complete frame was received on the Ethernet channel.

BSY—Busy Condition

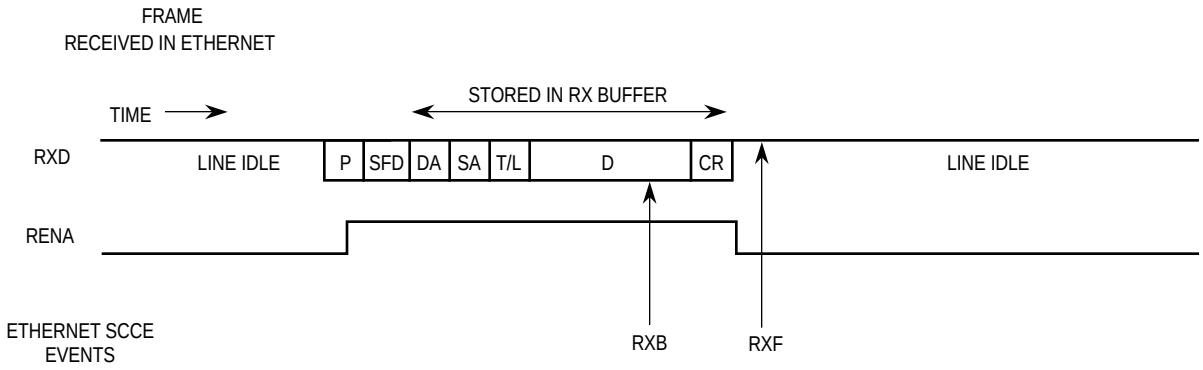
A frame was received and discarded due to lack of buffers.

TXB—Tx Buffer

A buffer has been transmitted on the Ethernet channel.

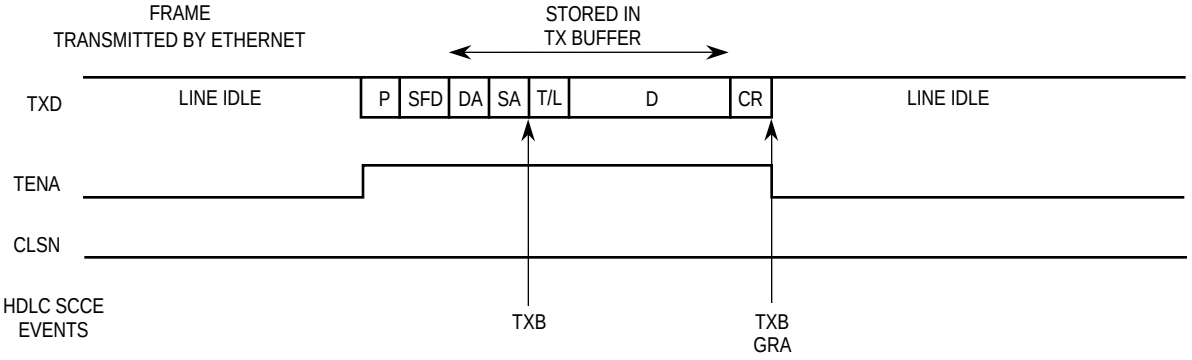
RXB—Rx Buffer

A buffer that was not a complete frame has been received on the Ethernet channel.



NOTES:
1. RXB event assumes receive buffers are 64 bytes each.
2. The RENA events, if required, must be programmed in the port C parallel I/O, not in the SCC itself.
3. The RXF interrupt may occur later than RENA due to receive FIFO latency.

LEGEND:
P = Preamble, SFD = Start Frame Delimiter, DA and SA = Source/Destination Address, T/L = Type/Length, D = Data, and CR = CRC bytes.



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 port C parallel I/O, not in the SCC itself.

Figure 7-72. Ethernet Interrupt Events Example;

7.10.23.21 ETHERNET MASK REGISTER (SCCM). The SCCM is referred to as the Ethernet mask register when the SCC is operating as an Ethernet controller. It is a 16-bit read-write register that has the same bit formats as the Ethernet event register. If a bit in the Ethernet mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

7.10.23.22 ETHERNET STATUS REGISTER (SCCS). This register is not valid for the Ethernet protocol. The current state of the RENA and CLSN signals may be read in port C.

7.10.23.23 SCC ETHERNET EXAMPLE. The following list is an initialization sequence for an Ethernet channel. SCC1 is used. The CLK1 pin is used for the Ethernet receiver, and the CLK2 pin is used for the Ethernet transmitter.

1. The SDCR (SDMA Configuration Register) should be initialized to $0740, rather than being left at its default value of $0000.
2. Configure the port A pins to enable the TXD1 and RXD1 pins. Write PAPAR bits 0 and 1 with ones. Write PADIR bits 0 and 1 with zeros. Write PAODR bit 1 with zero.
3. Configure the port C pins to enable $\overline{\text{CTS1}}$ (CLSN) and $\overline{\text{CD1}}$ (RENA). Write PCPAR bits 4 and 5 with zeros. Write PCDIR bits 4 and 5 with zero. Write PCSO bits 4 and 5 with ones.
4. Do not enable the $\overline{\text{RTS1}}$ (TENA) pin yet because the pin is still functioning as $\overline{\text{RTS}}$ (inactive in the high state), and transmission on the LAN could accidentally begin.
5. Configure port A to enable the CLK1 and CLK2 pins. Write PAPAR bits 8 and 9 with a ones. Write PADIR bits 8 and 9 with zeros.
6. Connect the CLK1 and CLK2 pins to SCC1 using the SI. Write the R1CS bits in SICR to 101. Write the T1CS bits in SICR to 100.
7. Connect the SCC1 to the NMSI (i.e., its own set of pins). Clear the SC1 bit in the SICR.
8. Write RBASE and TBASE in the SCC parameter RAM to point to the Rx BD and Tx BD in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM, and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.
9. Program the CR to execute the INIT RX & TX PARAMS command for this channel. For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
10. Write RFCR with $18 and TFCR with $18 for normal operation.
11. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 1520 bytes, so MRBLR = $05F0. (In this example, the user wants to receive an entire frame into one buffer, so the MRBLR value is simply chosen to be the first value larger than 1518 that is evenly divisible by 4).
12. Write C_PRES with $FFFFFFFF to comply with 32-bit CCITT-CRC.
13. Write C_MASK with $DEBB20E3 to comply with 32-bit CCITT-CRC.
14. Clear CRCEC, ALEC, and DISFC for the sake of clarity.
15. Write PAD with $8888 for the PAD value.
16. Write RET_Lim with $000F.
17. Write MFLR with $05EE to make the maximum frame size 1518 bytes.
18. Write MINFLR with $0040 to make the minimum frame size 64 bytes.
19. Write MAXD1 and MAXD2 with $05EE to make the maximum DMA count 1518 bytes.

20. Clear GADDR1–GADDR4. The group hash table is not used.
21. Write PADDR1_H with $0000, PADDR1_M with $0000, and PADDR1_L with $0040 to configure the physical address.
22. Write P_Per with $0000. It is not used.
23. Clear IADDR1–IADDR4. The individual hash table is not used.
24. Clear TADDR_H, TADDR_M, and TADDR_L for the sake of clarity.
25. Initialize the Rx BD. Assume the Rx data buffer is at $00001000 in main memory. Write $B000 to Rx_BD_Status. Write $0000 to Rx_BD_Length (not required—done for instructional purposes only). Write $00001000 to Rx_BD_Pointer.
26. Initialize the Tx BD. Assume the Tx data frame is at $00002000 in main memory and contains fourteen 8-bit characters (destination and source addresses plus the type field). Write $FC00 to Tx_BD_Status. Add PAD to the frame and generate a CRC. Write $000D to Tx_BD_Length. Write $00002000 to Tx_BD_Pointer.
27. Write $FFFF to the SCCE to clear any previous events.
28. Write $001A to the SCCM to enable the TXE, RXF, and TXB interrupts.
29. Write $40000000 to the CIMR to allow SCC1 to generate a system interrupt. (The CICR should also be initialized.)
30. Write $00000000 to GSMR_H1 to enable normal operation of all modes.
31. Write $1088000C to GSMR_L1 to configure the $\overline{\text{CTS}}$ (CLSN) and $\overline{\text{CD}}$ (RENA) pins 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 QUICC's transmit data. TPL and TPP are set as required for Ethernet. The DPLL is not used with Ethernet. Notice that the transmitter (ENT) and receiver (ENR) have not been enabled yet.
32. Write $D555 to DSR
33. Set the PSMR1 to $0A0A to configure 32-bit CRC, promiscuous mode (receive all frames), and begin searching for the start frame delimiter 22 bits after RENA.
34. Enable the TENA pin ($\overline{\text{RTS}}$). Since the MODE bits in GSMR have been written to Ethernet, the TENA signal is low. Write PCPAR bit 0 with a one. Write PCDIR bit 0 with a zero.
35. Write $1088003C to GSMR_L1 to enable the SCC1 transmitter and receiver. This additional write ensures that the ENT and ENR bits will be enabled last.

NOTE

After 14 bytes and the 46 bytes of automatic pad (plus the 4 bytes of CRC) have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after a frame is received. Any additional receive data beyond 1520 bytes or a single frame will cause a busy (out-of-buffers) condition since only one Rx BD was prepared.

7.11 SERIAL MANAGEMENT CONTROLLERS (SMCS)

The SMC key features are as follows:

- Each SMC can implement the UART protocol on its own pins.
- Each SMC can implement a totally transparent protocol on a multiplexed line or on a nonmultiplexed line. This mode can also be used for a fast connection between QUICCs.
- Each SMC channel fully supports the C/I and Monitor channels of the GCI (IOM-2) in ISDN applications.
- Two SMCs fully support the two sets of C/I and Monitor channels in the SCIT channel 0 and channel. 1
- Full-Duplex operation.
- Local Loopback and Echo Capability for testing.

7.11.1 SMC Overview

The SMCs are two full-duplex ports that may be independently configured to support any one of three protocols: UART, transparent, or GCI.

The SMCs can support simple UART operation for such purposes as providing a debug/ monitor port in an application, allowing the four SCCs to be free for another purpose. The UART functionality of the SMCs is reduced as compared to the SCCs. The SMC clock can be derived from one of the four internal baud rate generators or from an external clock pin. The clock provided to the SMC should be a 16x clock.

The SMCs can also support totally transparent operation. In this mode, the SMC may be connected to a TDM channel (such as a T1 line) or directly to its own set of pins. The receive and transmit clocks can be derived from the TDM channel, the internal baud rate generators, or from an external clock. In either case, the clock provided to the SMCs should be a 1x clock. The transparent protocol also allows the use of an external synchronization pin for the transmitter and receiver. The transparent functionality of the SMCs is reduced as compared to the SCCs.

Finally, each SMC can support the C/I and monitor channels of the GCI bus (IOM-2). In this case, the SMC is connected to a TDM channel in the SI. See 7.8 Serial Interface with Time Slot Assigner for the details of configuring the GCI interfaces.

The SMCs support loopback and echo modes for testing.

NOTE

In the MC68302, the SMCs also provide support for the A and M bits of the IDL definition. Since the IDL definition has been modified to eliminate the A and M bits, the QUICC does not provide special SMC support for IDL; however, the A and M bits may still be routed to the SMC using the TSA, if desired. The SMC would be configured into transparent mode for this operation.

Refer to Figure 7-73 for the SMC block diagram. The SMC receiver and transmitter are double-buffered, as shown in the block diagram. This corresponds to an effective FIFO size (latency) of two characters.

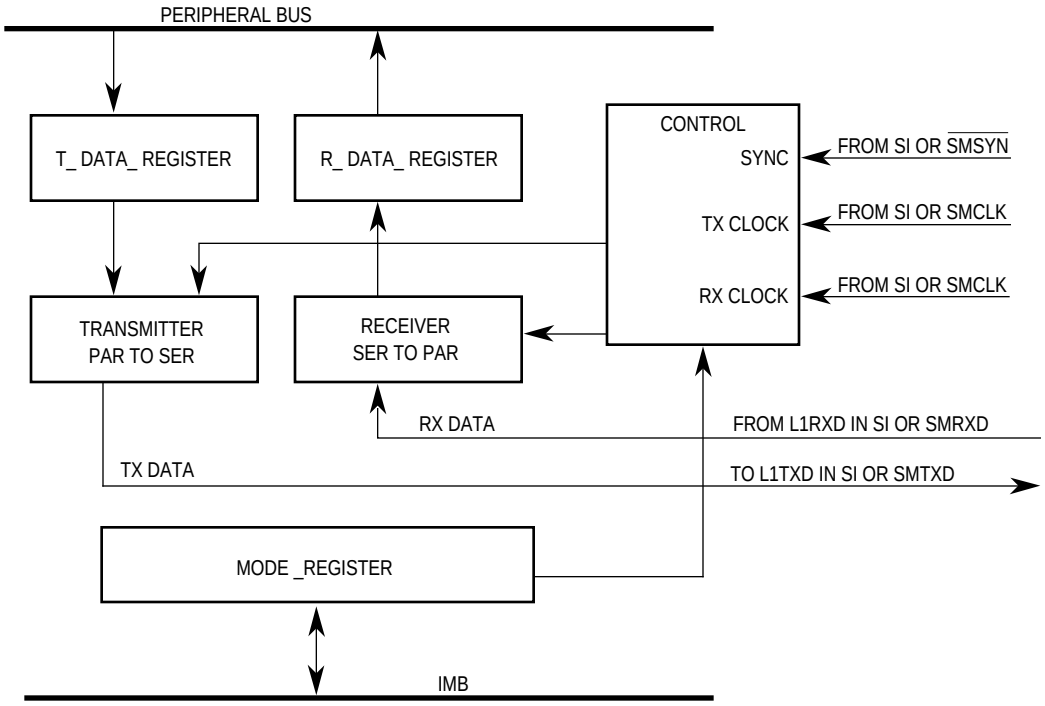


Figure 7-73. SMC Block Diagram

The receive data source for an SMC can be either the L1RXD pin if the SMC is connected to a TDM channel of the SI, or the SMRXD pin if the SMC is connected to the NMSI. The transmit data source can either be the L1TXD pin if the SMC is connected to a TDM, or the SMTXD pin if the SMC is connected to the NMSI.

If the SMC is connected to a TDM, the SMC receive clock and SMC transmit clock can be independent from each other as defined in the SI description. However, if the SMC is connected to the NMSI, the SMC receive clock and SMC transmit clock must be connected to a single clock source called SMCLK. SMCLK is an internal signal name for a clock that is generated from the bank of clocks defined in the SI description. SMCLK may originate from an external pin or one of the four internal baud rate generators. See 7.8.9 NMSI Configuration for more details.

If the SMC is connected to a TDM, it derives its synchronization pulse from the TSA as defined in the SI description. Otherwise, if the SMC is connected to the NMSI and the totally transparent protocol is selected, the SMC may use the $\overline{\text{SMSYN}}$ pin as a synchronization pin to determine when transmission and reception should begin. (The $\overline{\text{SMSYN}}$ pin is not used in the SMC UART mode.)

7.11.2 General SMC Mode Register (SMCMR)

The operating mode of each SMC port is defined by the 16-bit, memory-mapped, read-write SMCMR. See the specific SMC protocol for more information on this register.

7.11.3 SMC Buffer Descriptors

When the SMCs are configured to operate in GCI mode, the memory structure for the SMCs is pre-defined to be one word long for transmit and one word long for receive. These one-word structures are detailed later when the GCI operation is described in more detail.

However, in UART and transparent modes of operation, the SMCs have a memory structure that is like that of the SCCs. The data associated with the SMCs is stored in buffers. Each buffer is referenced by BD organized in a buffer descriptor ring located in the dual-port RAM (see Figure 7-74).

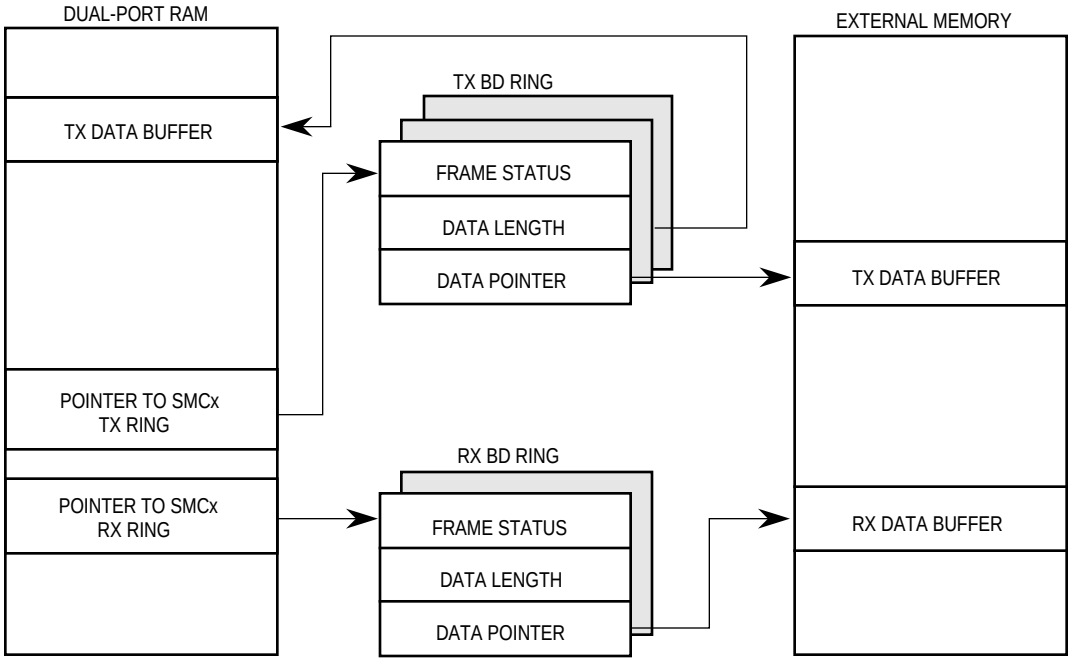


Figure 7-74. SMC Memory Structure

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 actual buffers may reside in either external memory or internal memory. Data buffers may reside in the parameter area of an SCC or SMC if that channel is not enabled.

7.11.4 SMC Parameter RAM

Each SMC parameter RAM area begins at the same offset from each SMC base area. The protocol-specific portions of the SMC parameter RAM are discussed in the specific protocol

descriptions. The part of the SMC parameter RAM that is the same for the UART and transparent SMC protocols is shown in Table 7-12. The following discussion does not apply to the GCI SMC protocol, which has its own parameter RAM.

Table 7-12. SMC UART and Transparent

Address	Name	Width	Description
SMC Base + 00	**RBASE**	Word	Rx Buffer Descriptors Base Address
SMC Base + 02	**TBASE**	Word	Tx Buffer Descriptors Base Address
SMC Base + 04	**RFCR**	Byte	Rx Function Code
SMC Base + 05	**TFCR**	Byte	Tx Function Code
SMC Base + 06	**MRBLR**	Word	Maximum Receive Buffer Length
SMC Base + 08	RSTATE	Long	Rx Internal State
SMC Base + 0C		Long	Rx Internal Data Pointer
SMC Base + 10	RBPTR	Word	Rx Buffer Descriptor Pointer
SMC Base + 12		Word	Rx Internal Byte Count
SMC Base + 14		Long	Rx Temp
SMC Base + 18	TSTATE	Long	Tx Internal State
SMC Base + 1C		Long	Tx Internal Data Pointer
SMC Base + 20	TBPTR	Word	Tx Buffer Descriptor Pointer
SMC Base + 22		Word	Tx Internal Byte Count
SMC Base + 24		Long	Tx Temp
SMC Base + 28			First Word of Protocol Specific Area
SMC Base + 36			Last Word of Protocol Specific Area

NOT E: The boldfaced items should be initialized by the user.

Certain parameter RAM values (marked in boldface) need to be initialized by the user before the SMC is enabled; other values are initialized/written by the CP. Once initialized, most parameter RAM values will not need to be accessed in user software since most of the activity is centered around the transmit and receive BDs, not the parameter RAM. However, if the parameter RAM is accessed by the user, the following restrictions should be noted. The parameter RAM can be read at any time. The parameter RAMvalues related to the SMC transmitter can only be written whenever the TEN bit in the SMC mode register is zero, after a STOP TRANSMIT and before a RESTART TRANSMIT command. The parameter RAM values related to the SMC receiver can only be written whenever the REN bit in the SMC mode register is zero or if the receiver has previously been enabled after an ENTER HUNT MODE command or CLOSE Rx BD command before the REN bit is set.

7.11.4.1 BD TABLE POINTER (RBASE, TBASE). The RBASE and TBASE entries define the starting location in the dual-port RAM for the set of BDs for receive and transmit functions of the SMC. This provides a great deal of flexibility in how BDs for an SMC are partitioned. By selecting RBASE and TBASE entries for all SMCs, and by setting the W-bit in the last BD in each BD list, the user may select how many BDs to allocate for the transmit and receive side of every SMC. The user must initialize these entries before enabling the corre-

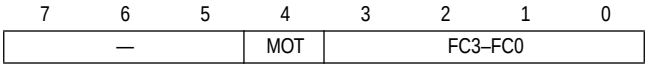
sponding channel. Furthermore, the user should not configure BD tables of two enabled SMCs to overlap, or erratic operation will occur.

NOTE

RBASE and TBASE should contain a value that is divisible by 8.

7.11.4.2 SMC FUNCTION CODE REGISTERS (RFCR, TFCR). There are four separate function code registers for the two SMC channels: two for receive data buffers (RFCRx) and two for transmit data buffers (TFCRx). The FC entry contains the value that the user would like to appear on the function code pins FC3–FC0 when the associated SDMA channel accesses memory. It also controls the byte-ordering convention to be used in the transfers.

Receive Function Code Register



Bits 7–5—Reserved

MOT—Motorola

This bit should be set by the user to achieve normal operation. MOT *must be set* if the data buffer is located in external memory and has a 16-bit wide memory port size.

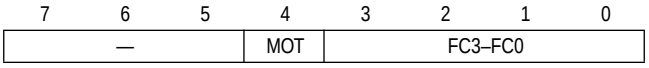
0 = DEC (and Intel) convention is used for byte ordering—swapped operation. It is also called little-endian byte ordering. The bytes stored in each buffer word are reversed as compared to the Motorola mode.

1 = Motorola byte ordering—normal operation. It is also called big-endian byte ordering. As data is received from the serial line and put into the buffer, the most significant byte of the buffer word contains data received earlier than the least significant byte of the same buffer word.

FC3–FC0—Function Code 3–0

These bits contain the function code value used during this SDMA channel's memory accesses. The user should write bit FC3 with a one to identify this SDMA channel access as a DMA-type access. Example: FC3–FC0 = 1000 (binary). Do not write the value 0111 (binary) to these bits.

Transmit Function Code Register



Bits 7–5—Reserved

MOT—Motorola

This bit should be set by the user to achieve normal operation. MOT *must be set* if the data buffer is located in external memory and has a 16-bit wide memory port size.

0 = 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 Motorola mode.

1 = Motorola 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.

FC3–FC0—Function Code 3–0

These bits contain the function code value used during this SDMA channel's memory accesses. The user should write bit FC3 with a one to identify this SDMA channel access as a DMA-type access. Example: FC3–FC0 = 1000 (binary). Do not write the value 0111 (binary) to these bits.

7.11.4.3 MAXIMUM RECEIVE BUFFER LENGTH REGISTER (MRBLR). Each SMC has one MRBLR to define the receive buffer length for that SMC. MRBLR defines the maximum number of bytes that the QUICC will write to a receive buffer on that SMC before moving to the next buffer. The QUICC may write fewer bytes to the buffer than MRBLR if a condition such as an error or end-of-frame occurs, but it will never write more bytes than the MRBLR value. It follows, then, that buffers supplied by the user for use by the QUICC should always be of size MRBLR (or greater) in length.

The transmit buffers for an SMC are not affected in any way by the value programmed into MRBLR. Transmit buffers may be individually chosen to have varying lengths, as needed. The number of bytes to be transmitted is chosen by programming the data length field in the Tx BD.

NOTES

MRBLR should not be changed dynamically while an SMC is operating. However, if it is modified in a single bus cycle with one 16-bit move (not two 8-bit bus cycles back-to-back), then a dynamic change in receive buffer length can be successfully achieved. This occurs when the CP moves control to the next Rx BD in the table. Thus, a change to MRBLR will not have an immediate effect. To guarantee the exact Rx BD on which the change will occur, the user should change MRBLR only while the SMC receiver is disabled.

The MRBLR value should be greater than zero, and should be even if the character length of the data is greater than 8 bits.

7.11.4.4 RECEIVER BUFFER DESCRIPTOR POINTER (RBPTR). The RBPTR for each SMC channel points to the next BD that the receiver will transfer 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 this pointer to the value programmed in the RBASE entry.

Although RBPTR need never be written by the user in most applications, it may be modified by the user when the receiver is disabled or when the user is sure that no receive buffer is currently in use.

7.11.4.5 TRANSMITTER BUFFER DESCRIPTOR POINTER (TBPTR). The TBPTR for each SMC channel points to the next BD that the transmitter will transfer 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 initializes this pointer to the value programmed in the TBASE entry. Although TBPTR need never be written by the user in most applications, it may be modified by the user when the transmitter is disabled or when the user is sure that no transmit buffer is currently in use (e.g., after a STOP TRANSMIT command is issued, or after a GRACEFUL STOP TRANSMIT command is issued, and the frame completes its transmission.)

7.11.4.6 OTHER GENERAL PARAMETERS. Additional parameters are listed in Table 7-12. These parameters do not need to be accessed by the user in normal operation, and are listed only because they may provide helpful information for experienced users and for debugging.

The Rx and Tx internal data pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.

The Tx internal byte count is a down-count value that is initialized with the Tx BD data length and decremented with every byte read by the SDMA channels. The Rx internal byte count is a down-count value that is initialized with the MRBLR value and decremented with every byte written by the SDMA channels.

NOTE

To extract data from a partially full receive buffer, the CLOSE Rx BD command may be used.

The Rx internal state, Tx internal state, Rx temp, Tx temp, and reserved areas are for RISC use only.

7.11.5 Disabling the SMCs on the Fly

If an SMC is not needed for a period of time, it may be disabled and re-enabled later. In this case, a sequence of operations is followed.

This sequence ensures that any buffers in use will be properly closed and that new data will be transferred to/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 (on-the-fly) 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. For instance, the baudrate generators allow on-the-fly changes.

NOTES

The modification of parameter RAM does not require a full disabling of the SMC. See the parameter RAM description for details on when parameter RAM values may be modified.

If the user desires to disable all SCCs, SMCs, and SPI, then the CR may be used to reset the entire CP with a single command.

7.11.5.1 SMC TRANSMITTER FULL SEQUENCE. For the SMC transmitter, the full disable and enable sequence is as follows:

1. STOP TRANSMIT command. This command is recommended if the SMC is currently in the process of transmitting data since it stops transmission in an orderly way. If the SMC is not transmitting (e.g., no Tx BDs are ready), then the STOP TRANSMIT command is not required. Furthermore, if the TBPTR will be overwritten by the user or the INIT TX PARAMETERS command will be executed, this command is not required.
2. Clear the TEN bit in the SMCMR. This disables the SMC transmitter and puts it in a reset state.
3. Make modifications. The user may make modifications to the SMC transmit parameters including the parameter RAM. If the user desires to switch protocols or restore the SMC transmit parameters to their initial state, the INIT TX PARAMETERS command may now be issued.
4. RESTART TRANSMIT command. This command is required if the INIT TX PARAMETERS command was not issued in step 3.
5. Set the TEN bit in the SMCMR. Transmission will now begin using the Tx BD pointed to by the TBPTR value as soon as the Tx BD R-bit is set.

7.11.5.2 SMC TRANSMITTER SHORTCUT SEQUENCE. A shorter sequence is possible if the user desires to reinitialize the transmit parameters to the state they had after reset. This sequence is as follows:

1. Clear the TEN bit in the SMCMR.
2. INIT TX PARAMETERS command. Any additional modifications may now be made.
3. Set the TEN bit in the SMCMR.

7.11.5.3 SMC RECEIVER FULL SEQUENCE. For the receiver, the full disable and enable sequence is as follows:

1. Clear the REN bit in the SMCMR. Reception will be aborted immediately. This disables the receiver of the SMC and puts it in a reset state.
2. Make modifications. The user may make modifications to the SMC receive parameters including the parameter RAM. If the user desires to switch protocols or restore the SMC receive parameters to their initial state, the INIT RX PARAMETERS command may now be issued.
3. CLOSE Rx BD command. This command is required if the INIT RX PARAMETERS command was not issued in step 2.
4. Set the REN bit in the SMCMR. Reception will now begin immediately using the Rx

BD pointed to by the RBPTR if the Rx BD E-bit is set.

7.11.5.4 SMC RECEIVER SHORTCUT SEQUENCE. A shorter sequence is possible if the user desires to reinitialize the receive parameters to the state they had after reset. This sequence is as follows:

1. Clear the REN bit in the SMCMR.
2. INIT RX PARAMETERS command. Any additional modifications may now be made.
3. Set the REN bit in the SMCMR.

7.11.5.5 SWITCHING PROTOCOLS. Sometimes the user desires to switch the protocol that the SMC is executing (for instance, UART to Transparent) without resetting the board or affecting any other SMC. This can be accomplished using only one command and a short number of steps:

1. Clear the TEN and REN bits in the SMCMR.
2. INIT TX AND RX PARAMETERS command. This one command initializes both transmit and receive parameters. Any additional modifications may now be made in the SMCMR to change the protocol, etc.
3. Set the SMCMR TEN and REN bits. The SMC is enabled with the new protocol.

7.11.6 Saving Power

When the TEN and REN bits of an SMC are cleared, that SMC consumes a minimal amount of power.

7.11.7 SMC as a UART

The following paragraphs describe the use of the SMC as a UART.

7.11.7.1 SMC UART KEY FEATURES. The SMC UART contains the following key features:

- 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

7.11.7.2 SMC UART COMPARISON. As compared to the UART modes supported by the SCCs, the SMCs generally offer less functionality and performance. This fits with their purpose of providing simple debug/monitor ports rather than full-featured UARTs. The SMC UARTs do not support the following features:

- $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ Pins
- Receive and Transmit Sections Being Clocked at Different Rates

- Fractional Stop Bits
- Built-In Multidrop Modes
- Freeze Mode for Implementing Flow Control
- Isochronous Operation (1x Clock)
- Interrupts upon Receiving Special Control Characters
- Ability To Transmit Data on Demand using the TODR
- SCCS Register To Determine Idle Status of the Receive Pin
- Other Features for the SCCs as Described in the GSMR

The SMCs in UART mode, however, do provide one feature not provided by the regular SCCs. The SMCs allow a data length option of up to 14 bits; whereas, the SCCs provide a data length up to 8 bits. See Figure 7-75 for the SMC UART frame format.

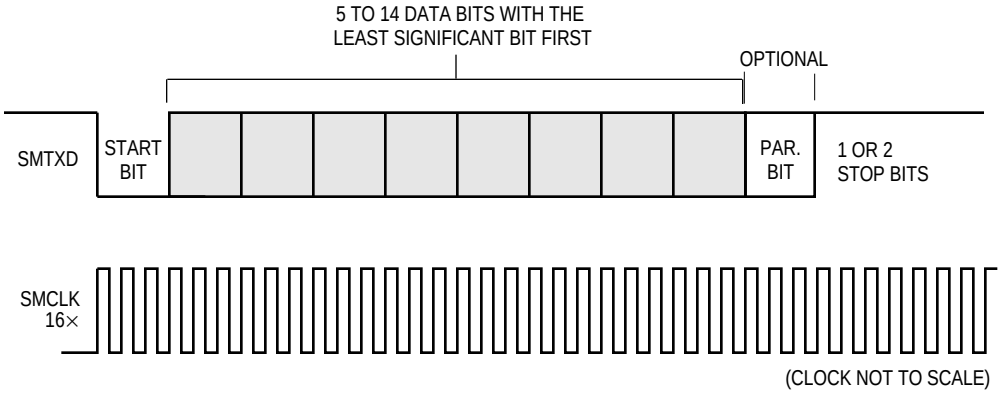


Figure 7-75. SMC UART Frame Format

7.11.7.3 SMC UART MEMORY MAP. When configured to operate in UART mode, the QUICC overlays the structure listed in Table 7-5 with the UART-specific parameters described in Table 7-13.

Table 7-13. SMC UART-Specific Parameter RAM

Address	Name	Width	Description
SMC Base + 28	**MAX_IDL**	Word	Maximum Idle Characters
SMC Base + 2A	IDLC	Word	Temporary Idle Counter
SMC Base + 2C	**BRKLN**	Word	Last Received Break Length
SMC Base + 2E	**BRKEC**	Word	Receive Break Condition Counter
SMC Base +30	**BRKCR**	Word	Break Count Register (Transmit)
SMC Base +32	R_mask	Word	Temporary Bit Mask

MAX_IDL. Once a character of data is received on the line, the UART controller begins counting any idle characters received. If a MAX_IDL number of idle characters is received before the next data character is received, an idle timeout occurs, and the buffer is closed. This, in turn, can produce an interrupt request to the CPU32+ core to receive the data from

the buffer. Thus, MAX_IDL provides a convenient way to demarcate frames in the UART mode. If the MAX_IDL functionality is not desired, the user should program MAX_IDL to $0000, and the buffer will never be closed, regardless of the number of idle characters received. A character of idle is calculated as the following number of bit times: 1 + data length (5 to 14) + 1 (if parity bit is used) + number of stop bits (1 or 2). Example: for 8 data bits, no parity, and 1 stop bit, the character length is 10 bits.

IDLC. This value is used by the RISC to store the current idle counter value in the MAX_IDL timeout process. IDLC is a down-counter; it does not need to be initialized or accessed by the user.

BRKLN. This value is used to store the length of the last break character received. This value is the length in bits of that character. Example: If the receive pin is low for 257 bit times, BRKLN will show the value $0101. BRKLN is accurate to within one character unit of bits. For example, for 8 data bits, no parity, 1 stop bit, and 1 start bit, BRKLN is accurate to within 10 bits.

BRKEC. This counter counts the number of break conditions that occurred on the line. Note that one break condition may last for hundreds of bit times, yet this counter is incremented only once during that period.

BRKCR. The SMC UART controller will send an a break character sequence whenever a STOP TRANSMIT command is given. The number of break characters sent by the UART controller is determined by the value in BRKCR. In the case of 8 data bits, no parity, 1 stop bit, and 1 start bit, each break character is 10 bits in length and consists of all zeros.

7.11.7.4 SMC UART TRANSMISSION PROCESSING. The UART transmitter is designed to work with almost no intervention from the CPU32+ core. When the CPU32+ core enables the SMC transmitter, it will start transmitting idles. The SMC immediately polls the first BD in the transmit channel's BD ring, and thereafter once every character time, depending on the character length (i.e., every 7 to 16 serial clocks). When there is a message to transmit, the SMC will fetch the data from memory and start transmitting the message.

When a BD's data has been completely written to the transmit FIFO, the SMC writes the message status bits into the BD and clears the R-bit. An interrupt is issued if the I-bit in the BD is set. If the next Tx BD is ready, the data from its data buffer will be appended to the previous data and transmitted out on the transmit pin, with no gaps occurring between buffers. If the next Tx BD is not ready, the SMC will start transmitting idles and wait for the next Tx BD to become ready.

By appropriately setting the I-bit in each BD, interrupts can be generated after the transmission of each buffer, a specific buffer, or each block. The SMC will then proceed to the next BD in the table.

If the CM bit is set in the Tx BD, the R-bit will not be cleared, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this data buffer. For instance, if a single Tx BD is initialized with the CM bit set and the W-bit set, the data buffer will be continuously transmitted until the user clears the R-bit of the BD.

7.11.7.5 SMC UART RECEPTION PROCESSING. When the CPU32+ core enables the SMC receiver in UART mode, it will enter hunt mode, waiting for the first character to arrive. Once the first character arrives, the first Rx BD is checked by the CP to see if it is empty. It then begins storing characters in the associated data buffer.

When the data buffer has been filled or the MAX_IDL timer has expired (assuming it was enabled), the SMC clears the E-bit in the BD and generates an interrupt if the I-bit in the BD is set. If the incoming data exceeds the length of the data buffer, the SMC will fetch the next BD in the table and, if it is empty, will continue to transfer data to this BD's associated data buffer.

If the CM bit is set in the Rx BD, the E-bit will not be cleared, allowing the associated data buffer to be overwritten automatically when the CP next accesses this data buffer.

7.11.7.6 SMC UART PROGRAMMING MODEL. An SMC configured as a UART uses the same data structure as in the other modes. The SMC UART data structure supports multi-buffer operation. The SMC UART allows the user to transmit break and preamble sequences. Overrun, parity, and framing errors are reported via the BDs. In its simplest form, the SMC UART can function in a character-oriented environment. Each character is transmitted with accompanying stop bits and parity (as configured by the user), and received into separate 1-byte buffers. Reception of each buffer may generate a maskable interrupt.

Many applications may want to take advantage of the message-oriented capabilities supported by the SMC UART by using linked buffers (in either receive or transmit). In this case, data is handled in a message-oriented environment; users can work on entire messages rather than operating on a character-by-character basis. A message may span several linked buffers. Each message can be both transmitted and received as a linked list of buffers without any intervention from the CPU32+, which achieves both ease in programming and significant savings in processor overhead.

In the message-oriented environment, the idle sequence is used as the message delimiter. The transmitter is able to generate an idle sequence before starting a new message, and the receiver is able to close a buffer upon detection of idle sequence.

7.11.7.7 SMC UART COMMAND SET. The following transmit and receive commands are issued to the CR.

7.11.7.7.1 Transmit Commands. The following paragraphs describe the SMC UART transmit commands.

STOP TRANSMIT Command. The channel STOP TRANSMIT command disables the transmission of characters on the transmit channel. If this command is received by the SMC UART controller during message transmission, transmission of that message is aborted. The SMC UART completes transmission of any data already transferred to its FIFO and shift register (up to two characters) and then stops transmitting data. The TBPTR is not advanced when this command is issued.

The SMC UART transmitter will transmit a programmable number of break sequences and then start to transmit idles. The number of break sequences (which may be zero) should be

written to the break count register before this command is given to the SMC UART controller.

RESTART TRANSMIT Command. The RESTART TRANSMIT command enables the transmission of characters on the transmit channel. This command is expected by the SMC UART controller after disabling the channel in its SMC mode register and after the STOP TRANSMIT command. The SMC UART controller will resume transmission from the current TBPTR in the channel's Tx BD table.

INIT TX PARAMETERS Command. This command initializes all the transmit parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the transmitter is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both transmit and receive parameters.

7.11.7.7.2 Receive Commands. The following paragraphs describe the UART receive commands.

ENTER HUNT MODE Command. This command should not be used for an SMC UART channel. The CLOSE RX BD command may be used instead.

CLOSE RX BD Command. The CLOSE RX BD command is used to force 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 subsequent data that is received. If the SMC is not in the process of receiving data, no action is taken by this command.

iNIT RX PARAMETERS Command. This command Initializes all the receive parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the receiver is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both receive and transmit parameters.

7.11.7.8 SEND BREAK (TRANSMITTER). A break is an all-zeros character without stop bits. A break is sent by issuing the STOP TRANSMIT command. The SMC UART completes transmission of any outstanding data and then sends a character with consecutive zeros (the number of zero bits in this character is the sum of the character length, plus the number of start, parity, and stop bits). The SMC UART transmits a programmable number of break characters according to the break count register and then reverts to idle or sends data if the RESTART TRANSMIT command was given before completion. At the completion of the break, the transmitter sends at least one character of idle before transmitting any data to guarantee recognition of a valid start bit.

7.11.7.9 SENDING A PREAMBLE (TRANSMITTER). A preamble sequence gives the programmer a convenient way of ensuring that the line goes idle before starting a new message. The preamble sequence length is constructed of consecutive ones of one character length. If the preamble bit in a BD is set, the SMC will send a preamble sequence before transmitting that data buffer. Example: 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 may be transmitted without any delay occurring on the transmit pin between the two transmit buffers.

7.11.7.10 SMC UART ERROR-HANDLING PROCEDURE. The SMC UART reports character reception error conditions via the channel buffer descriptors, and the SMC UART event register. There are no transmission errors for the SMC UART controller.

7.11.7.10.1 Overrun Error. The SMC UART maintains a two-character length FIFO for receiving data (shift register plus data register). The data will be 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. Then the channel writes the received character to the buffer, closes the buffer, sets the OV bit in the BD, and generates the RX interrupt if it is enabled. Reception then continues normally.

NOTE

The SMC UART may occasionally get an overrun when the line is at idle. The user should ignore an overrun error when the line is known to be at idle.

7.11.7.10.2 Parity Error. When a parity error occurs, the channel writes the received character to the buffer, closes the buffer, sets the PR bit in the BD, and generates the RX interrupt if it is enabled. Reception then continues normally.

7.11.7.10.3 Idle Sequence Receive. An idle is detected when one character consisting of all ones is received. Once an idle is received, the channel counts the number of consecutive idle characters received. If the count reaches the MAX_IDL value, the buffer is closed, and an RX interrupt is generated. If no receive buffer is open, this event does not generate an interrupt or any status information. The idle counter is reset every time a character is received.

7.11.7.10.4 Framing Error. A framing error is detected by the SMC UART controller when a character is received with no stop bit. When this error occurs, the channel writes the received character to the buffer, closes the buffer, sets the FR bit in the BD, and generates the RX interrupt if it is enabled. When this error occurs, parity is not checked for this character.

7.11.7.10.5 Break Sequence. A break sequence is detected by the SMC UART receiver when an all-zero's character with a framing error is received. When a break sequence is received, the channel will increment the BRKEC and generate a maskable BRK interrupt in the SMC UART event register. The channel will also measure the length of the break sequence and store this value in the BRKLN counter. If the channel was in the middle of buffer processing when the break was received, the buffer will be closed with the BR bit in the Rx BD set, and the RX interrupt will be generated if it is enabled.

7.11.7.11 SMC UART MODE REGISTER (SMCMR). The operating mode of an SMC is defined by the SMCMR. The SMCMR is a 16-bit, memory-mapped, read-write register. The register is cleared at reset. The function of bits 7–0 is common to each SMC protocol. The function of bits 15–8 varies according to the protocol selected by the SM bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	CLEN				SL	PEN	PM	—		SM		DM		TEN	REN

Bits 15, 7, 6—Reserved

These bits should be cleared by the user.

CLEN—Character Length

The CLEN value should be programmed with the total number of bits in the character minus one. The total number of bits in the character is calculated as 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).

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. Thus, CLEN should be programmed to 9.

The number of data bits in the character may range from 5 to 14 bits. If the data bit length is less than 8 bits, the MSBs of each byte in memory are not used on transmit and are written with zeros on receive. If the data bit length is more than 8 bits, the MSBs of each 16-bit word in memory are not used on transmit and are written with zeros on receive.

NOTES

The total number of bits in the character must never exceed 16. Thus, if a 14-bit data length is chosen, SL must be set to one stop bit, and parity should not be enabled. If a 13-bit data length is chosen and parity is enabled, SL must be set to one stop bit.

The values 0 to 3 should not be written to CLEN, or erratic behavior may result.

SL—Stop Length

0 = One stop bit
1 = Two stop bits

PEN—Parity Enable

0 = No parity
1 = Parity is enabled for the transmitter and receiver as determined by the PM bit.

PM—Parity Mode

0 = Odd parity
1 = Even parity

SM—SMC Mode

00 = GCI or SCIT support
01 = Reserved
10 = UART (must be selected for SMC UART operation)
11 = Totally transparent operation

DM—Diagnostic Mode

00 = Normal operation
01 = Local loopback mode
10 = Echo mode
11 = Reserved

TEN—SMC Transmit Enable

0 = SMC transmitter disabled
1 = SMC transmitter enabled

REN—SMC Receive Enable

0 = SMC receiver disabled
1 = SMC receiver enabled

7.11.7.12 SMC UART RECEIVE BUFFER DESCRIPTOR (RX BD). •The CP reports information concerning the received data on a per-buffer basis via Rx BDs. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:

1. Detection of an error during message processing
2. Detection of a full receive buffer
3. Reception of a programmable number of consecutive idle characters

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**E**	—	**W**	**I**	—	—	**CM**	ID	—	—	BR	FR	PR	—	OV	CD
OFFSET + 2	DATA LENGTH															
OFFSET + 4	**RX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE: Entries in boldface must be initialized by the user.

An example of the UART Rx BD process is shown in Figure 7-76. This figure shows the resulting state of the Rx BDs after receipt of 10 characters, an idle period, and five characters—one with a framing error. The example assumes that MRBLR = 8 in the SMC parameter RAM.

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 CPU32+ 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 CPU32+ core should not write any fields of this Rx BD.

Bits 14, 11, 10, 7, 6, 2—Reserved

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.

I—Interrupt

0 = No interrupt is generated after this buffer has been filled.
1 = The RX bit in the event register will be set when this buffer has been completely filled by the CP, indicating the need for the CPU32+ core to process the buffer. The RX bit can cause an interrupt if it is enabled.

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 overwritten automatically when the CP next accesses this BD. However, the E-bit will be cleared if an error occurs during reception, regardless of the CM bit.

The following status bits are written by the CP after the received data has been into the associated data buffer.

ID—Buffer Closed on Reception of Idles

The buffer was closed due to the reception of the programmable number of consecutive idle sequences.

BR—Buffer Closed on Reception of Break

The buffer was closed due to the reception of a break sequence.

FR—Framing Error

A character with a framing error was received and is located in the last byte of this buffer. A framing error is a character without a stop bit. A new receive buffer will be used for further data reception.

PR—Parity Error

A character with a parity error was received and is located in the last byte of this buffer. A new receive buffer will be used for further data reception.

OV—Overrun

A receiver overrun occurred during message reception.

Data Length

Data length is the number of octets that the CP has written into this BD’s data buffer. It is written only 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.

Rx Data Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, must be even. The buffer may reside in either internal or external memory.

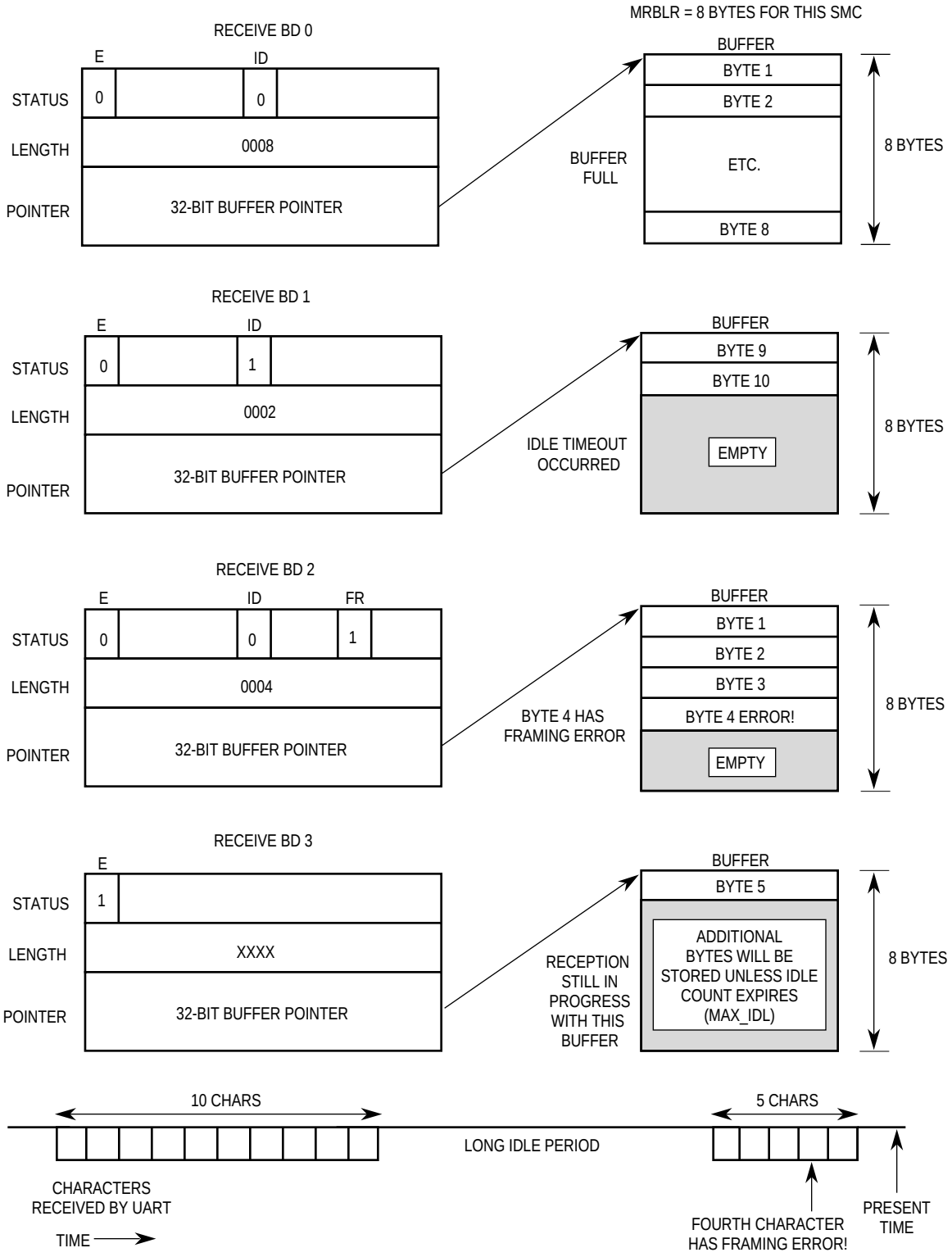


Figure 7-76. SMC UART Rx BD Example

7.11.7.13 SMC UART TRANSMIT BUFFER DESCRIPTOR (TX BD). Data is presented to the CP for transmission on an SMC channel by arranging it in buffers referenced by the

channel's Tx BD ring. The CP confirms transmission or indicates error conditions via the BDs to inform the processor that the buffers have been serviced.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**R**	—	**W**	**I**	—	—	**CM**	**P**	—	—	—	—	—	—	—	—
OFFSET + 2	**DATA LENGTH**															
OFFSET + 4	TX DATA BUFFER POINTER															
OFFSET + 6																

NOTE : Entries in boldface must be initialized by the user.

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.

Bits 14, 11, 10, 7–0—Reserved

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 receive incoming data into the first BD in the table (the BD pointed to by TBASE). The number of Tx BDs in this table is programmable and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.

1 = The TX bit in the SMC UART event register will be set when this buffer has been serviced. TX can cause an interrupt if it is enabled.

CM—Continuous Mode

0 = Normal operation.

1 = The R-bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this BD.

P—Preamble

0 = No preamble sequence is sent.

1 = The UART will send one all-ones character before sending the data so that the other end will detect 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 will be sent.

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. The data length may be equal to zero with the P-bit set, and only a preamble will be sent.

If the number of data bits in the UART character is greater than 8, then the data length should be even. Example: to transmit three UART characters of 8-bit data, 1 start, and 1 stop, the data length field should be initialized to 3. However, to transmit three UART characters of 9-bit data, 1 start, and 1 stop, the data length field should be initialized to 6, since the three 9-bit data fields occupy three words in memory (the 9 LSBs of each word).

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 (unless the number of actual data bits in the UART character is greater than 8 bits, in which case the transmit buffer pointer must be even.) For instance, the pointer to 8-bit data, 1 start, and 1 stop characters may be even or odd, but the pointer to 9-bit data, 1 start, and 1 stop characters must be even. The buffer may reside in either internal or external memory.

7.11.7.14 SMC UART EVENT REGISTER (SMCE). When the UART protocol is selected, the SMCE register is called the SMC UART event register. It is an 8-bit register used to report events recognized by the SMC UART channel and to generate interrupts. On recognition of an event, the UART will set the corresponding bit in the SMC UART event register.

The SMC UART event register is a memory-mapped register that 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.

An example of the timing of various events in the SMC UART event register is shown in Figure 7-77.

7	6	5	4	3	2	1	0
—	BRKe[1]	—	BRK	—	BSY	TX	RX

NOTES:
1: Only available on REV C mask or later. NOT Available on REV A or B.
Rev A mask is C63T
Rev B mask are C69T, and F35G
Current Rev C mask are E63C, E68C and F15W

Bits 7, 5, 3—Reserved.

BRKe—Break End

The end of break sequence was detected. This indication will be no sonner than after one idle bit is received following a break sequence.

BRK—Break Character Received

A break character was received. If a very long break sequence occurs, this interrupt will occur only once after the first all-zeros character is received.

BSY—Busy Condition

A character was received and discarded due to lack of buffers. This bit is be set no sooner than the middle of the last stop bit of the first receive character for which there is no available buffer. Reception continues when an empty buffer is provided.

TX—Tx Buffer

A buffer has been transmitted over the UART channel. This bit is set once the transmit data of the last character in the buffer was written to the transmit FIFO. The user must wait two character times to be sure that the data was completely sent over the transmit pin.

RX—Rx Buffer

A buffer has been received and its associated Rx BD is now closed. This bit is set no sooner than the middle of the last stop bit of the last character that was written to the receive buffer.

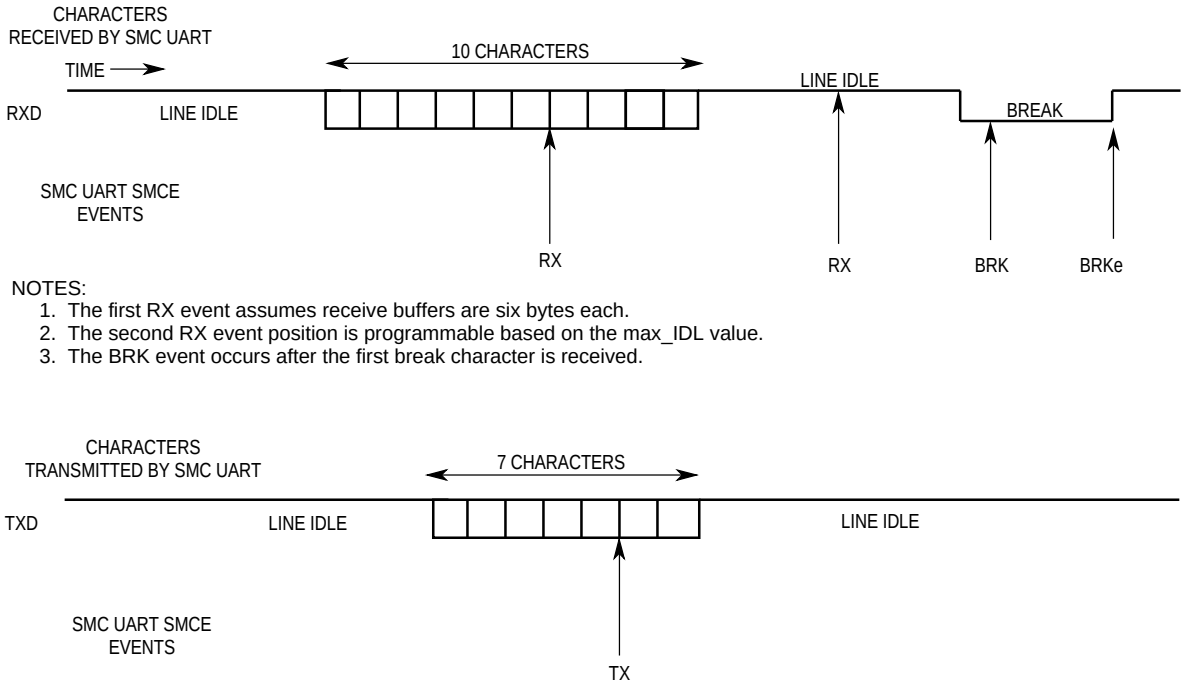


Figure 7-77. SMC UART Interrupts Example

7.11.7.15 SMC UART MASK REGISTER (SMCM). The SMCM is referred to as the SMC UART mask register when the SMC is operating as a UART. It is an 8-bit read-write register with the same bit format as the SMC UART event register. If a bit in the SMC UART mask register is a one, the corresponding interrupt in the event register will be enabled. If the bit is zero, the corresponding interrupt in the event register will be masked. This register is cleared upon reset.

7.11.8 SMC UART Example

The following list is an initialization sequence for 9600 baud, 8 data bits, no parity, and 1 stop bit operation of an SMC UART assuming a 25-MHz system frequency. BRG1 and SMC1 are used.

1. The SDCR (SDMA configuration register) should be initialized to $0740, rather than being left at its default value of $0000.
2. Configure the port B pins to enable the SMTXD1 and SMRXD1. Write PBPAR bits 6 and 7 with ones. Write PBDIR bits 6 and 7 with zeros. Write PBODR bits 6 and 7 with zeros.
3. Configure the BRG1. Write BRGC1 with $010144. The DIV16 bit is not used, and the divider is 162 (decimal). The resulting BRG1 clock is 16x the desired bit rate of the UART.
4. Connect the BRG1 clock to SMC1 using the SI. Write the SMC1 bit in SIMODE with a 0. Write the SMC1CS bits in SIMODE with 000.
5. Write RBASE and TBASE in the SMC parameter RAM to point to the Rx BD and Tx BD in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.
6. Program the CR to execute the INIT RX & TX PARAMS command for this channel. For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
7. Write RFCR with $18 and TFCR with $18 for normal operation.
8. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 16 bytes, so MRBLR = $0010.
9. Write MAX_IDL with $0000 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 for the sake of clarity.
11. Set BRKCR to $0001, so that if a STOP TRANSMIT command is issued, one break character will be sent.
12. Initialize the Rx BD. Assume the Rx data buffer is at $00001000 in main memory. Write $B000 to Rx_BD_Status. Write $0000 to Rx_BD_Length (not required—done for instructional purposes only). Write $00001000 to Rx_BD_Pointer.
13. Initialize the Tx BD. Assume the Tx data buffer is at $00002000 in main memory

and contains five 8-bit characters. Write $B000 to Tx_BD_Status. Write $0005 to Tx_BD_Length. Write $00002000 to Tx_BD_Pointer.

14. Write $FF to the SMCE to clear any previous events.
15. Write $17 to the SMCM to enable all possible SMC interrupts.
16. Write $00000010 to the CIMR to allow SMC1 to generate a system interrupt. (The CICR should also be initialized.)
17. Write $4820 to SMCMR to configure normal operation (not loopback), 8-bit characters, no parity, 1 stop bit. Notice that the transmitter and receiver have not been enabled yet.
18. Write $4823 to SMCMR to enable the SMC transmitter and receiver. This additional write ensures that the TEN and REN bits will be enabled last.

NOTE

After 5 bytes have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after 16 bytes have been received. Any additional receive data beyond 16 bytes will cause a busy (out-of-buffers) condition since only one Rx BD was prepared.

7.11.9 SMC Interrupt Handling

The following list describes what would normally occur within an interrupt handler for the SMC:

1. Once an interrupt occurs, read the SMCE to see which sources have caused interrupts. The SMCE bits would normally be cleared at this time.
2. Process the Tx BD to reuse it if the TX bit was set in SMCE. Extract data from the Rx BD if the RX bit was set in SMCE. To transmit another buffer, simply set the Tx BD R-bit.
3. Clear the SMC1 bit in the CISR.
4. Execute the RTE instruction.

7.11.10 SMC as a Transparent Controller

The following paragraphs describe using the SMC as a transparent controller.

7.11.10.1 SMC TRANSPARENT CONTROLLER KEY FEATURES. The SMC transparent controller contains the following key features:

- Flexible Data Buffers
- May Be Used To Connect to a TDM Bus using the TSA in the SI
- May Transmit and Receive Transparently on Its Own Set of Pins using a Sync Pin 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

7.11.10.2 SMC TRANSPARENT COMPARISON. As compared to the transparent modes supported by the SCCs, the SMCs offer less functionality. This fits with their purpose of providing simpler functions and slower speeds. The SMC transparent controller does not support the following features:

- Independent Transmit and Receive Clocks Unless Connected to a TDM Channel of the SI
- CRC Generation and Checking
- Full $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ Pins—Supports One $\overline{\text{SMSYN}}$ Pin Only
- Ability To Transmit Data on Demand using the TODR
- Receiver/Transmitter in Transparent Mode While Receiver/Transmitter Executes Another Protocol
- 4-, 8-, or 16-Bit SYNC Recognition
- Internal DPLL Support
- Other Features for the SCCs As Described in the GSMR

The SMCs in transparent mode, however, do provide one feature not provided by the regular SCCs. The SMCs allow a data character length option of 4 to 16 bits; whereas, the SCCs provide a data character length of just 8 or 32 bits (determined by the RFW bit in the GSMR).

7.11.10.3 SMC TRANSPARENT MEMORY MAP. There is no protocol-specific parameter RAM for the SMC when it is used as a transparent controller. Only the general SMC parameter RAM is used, which is discussed in 7.11.4 SMC Parameter RAM.

Table 7-14. SMC Transparent-Specific Parameter RAM

Address	Name	Width	Description
SMC Base + 28	Reserved	Word	Reserved
SMC Base + 2A	Reserved	Word	Reserved
SMC Base + 2C	Reserved	Word	Reserved
SMC Base + 2E	Reserved	Word	Reserved
SMC Base + 30	Reserved	Word	Reserved

7.11.10.4 SMC TRANSPARENT TRANSMISSION PROCESSING. The transparent transmitter is designed to work with almost no intervention from the CPU32+ core. When the CPU32+ enables the SMC transmitter in transparent mode, it will start transmitting idles. The SMC immediately polls the first BD in the transmit channel's BD ring, and thereafter once every character time, depending on the character length (i.e., every 4 to 16 serial clocks). When there is a message to transmit, the SMC controller will fetch the data from memory and start transmitting the message once synchronization is achieved.

Synchronization can be achieved in two ways. 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 to occur before transmission begins. Data will only be transmitted during the time slots defined by the TSA. Secondly, when working with its own set of pins (nonmultiplexed mode), the transmitter will start transmission when the $\overline{\text{SMSYNx}}$ line is asserted (falling edge).

When a BD's data has been completely written to the transmit FIFO, the L-bit is checked. If the L-bit is set, the SMC writes the message status bits into the BD and clears the R-bit. It will then start transmitting idles. When the end of the current BD has been reached and the L-bit is not set (multibuffer mode), only the R-bit 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 the transmission of each buffer, a specific buffer, or each block. The SMC will then proceed to the next BD in the table.

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 transmitting idles.

If the CM bit is set in the Tx BD, the R-bit will not be cleared, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this data buffer. For instance, if a single Tx BD is initialized with the CM bit set and the W-bit set, the data buffer will be continuously transmitted until the user clears the R-bit of the BD.

7.11.10.5 SMC TRANSPARENT RECEPTION PROCESSING. When the CPU32+ core enables the SMC receiver in transparent mode, it will wait for synchronization before receiving data. Once synchronization is achieved, the receiver will transfer the incoming data into memory according to the first Rx BD in the ring.

Synchronization can be achieved in two ways. 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 will only be received during the time slots defined by the TSA. Secondly, when working with its own set of pins (nonmultiplexed mode), the receiver will start reception when the $\overline{\text{SMSYNx}}$ line is asserted (falling edge).

When the data buffer has been filled, the SMC clears the E-bit in the BD and generates an interrupt if the I-bit in the BD is set. If the incoming data exceeds the length of the data buffer, the SMC will fetch the next BD in the table and, if it is empty, will continue to transfer data to this BD's associated data buffer.

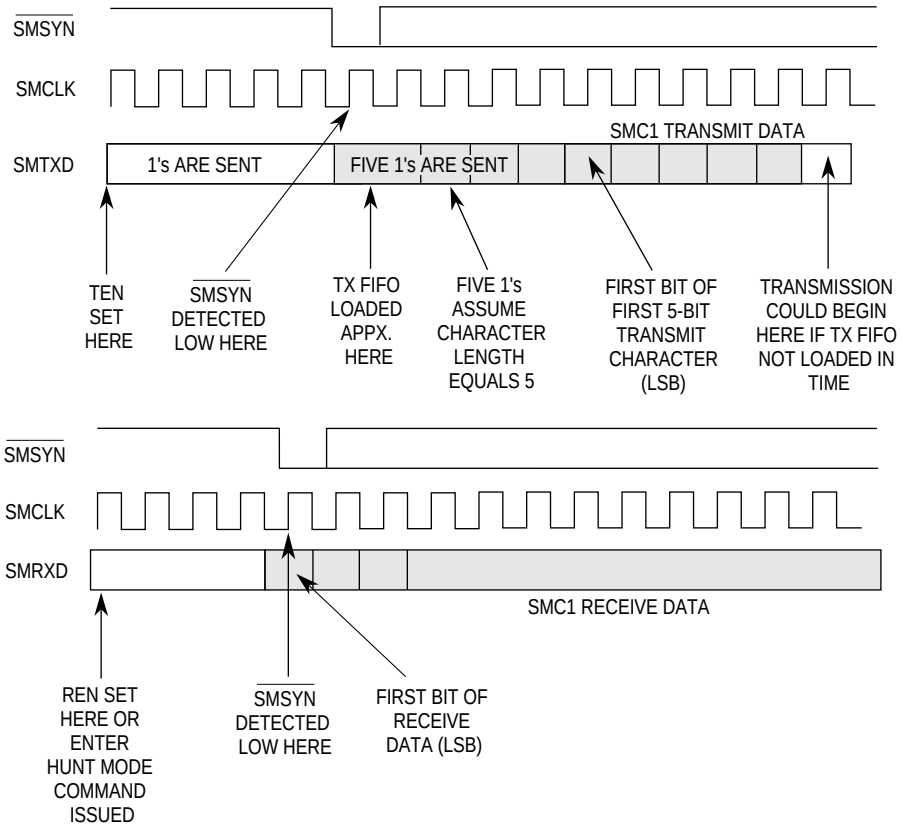
If the CM bit is set in the Rx BD, the E-bit will not be cleared, allowing the associated data buffer to be overwritten automatically when the CP next accesses this data buffer.

7.11.10.6 USING THE $\overline{\text{SMSYNx}}$ PIN FOR SYNCHRONIZATION. The $\overline{\text{SMSYNx}}$ pin offers a method to synchronize the SMC channel externally. This method differs somewhat from the synchronization options available in the SCCs and should be studied carefully. See Figure 7-78 for an example.

NOTE

Regardless of whether the transmitter or receiver uses the $\overline{\text{SMSYNx}}$ signal, the $\overline{\text{SMSYNx}}$ signal must make glitch-free transitions from high to low or low to high. Glitches on $\overline{\text{SMSYNx}}$ may cause errant behavior of the SMC.

Once the REN bit is set in SMCMR, the first rising edge of SMCLK that detects the $\overline{\text{SMSYNx}}$ pin as low causes the SMC receiver to achieve synchronization. Data will begin to be received (latched) on the same rising edge of SMCLK that latched $\overline{\text{SMSYNx}}$. This will be the first bit of data received. The receiver will never lose synchronization again, regardless of the state of $\overline{\text{SMSYNx}}$, until the REN bit is cleared by the user.



NOTES:
1. SMCLK is an internal clock derived from an external CLKPIN 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 7-78. Synchronization with the $\overline{\text{SMSYNx}}$ Pin

Once the TEN bit is set in SMCMR, the first rising edge of SMCLK that detects the $\overline{\text{SMSYNx}}$ pin as low causes the SMC transmitter to achieve synchronization. The SMC transmitter will begin transmitting ones asynchronously from the falling edge of $\overline{\text{SMSYNx}}$. After one character of ones is transmitted, if the transmit FIFO is loaded (i.e., the Tx BD was ready with

data), data will begin to be transmitted on the next falling edge of SMCLK after one character of ones is transmitted. If the transmit FIFO is loaded at some later time, the data will begin transmission after some multiple number of all-ones characters is transmitted. The transmitter will never lose synchronization again, regardless of the state of $\overline{\text{SMSYNx}}$, until the TEN bit is cleared by the user or the ENTER HUNT MODE command is issued.

If both the REN and TEN bits are set in SMCMR, the first falling edge of the $\overline{\text{SMSYNx}}$ pin causes both the transmitter and receiver to achieve synchronization. To re-synchronize the transmitter, the SMC transmitter may be disabled and reenabled, and the $\overline{\text{SMSYNx}}$ pin can be used again to re-synchronize just the transmitter. See 7.11.5 Disabling the SMCs on the Fly for a description of how to safely disable and reenable the SMC (simply clearing TEN and setting TEN may not be sufficient). The receiver may be re-synchronized in a similar fashion.

7.11.10.7 USING THE TSA FOR SYNCHRONIZATION. The TSA offers a method to synchronize the SMC channel internally without using the $\overline{\text{SMSYNx}}$ pin. This behavior is similar to that of the $\overline{\text{SMSYNx}}$ pin, except that the synchronization event is not the falling edge of the $\overline{\text{SMSYNx}}$ pin, but rather the first time slot for this SMC receiver/transmitter following the frame sync indication. See 7.8 Serial Interface with Time Slot Assigner for further information on configuring time slots for the SMCs and SCCs.

The TSA allows the SMC receiver and transmitter to be enabled simultaneously, yet synchronized separately, a capability not provided by the $\overline{\text{SMSYNx}}$ pin. See Figure 7-79 for an example of synchronization using the TSA.

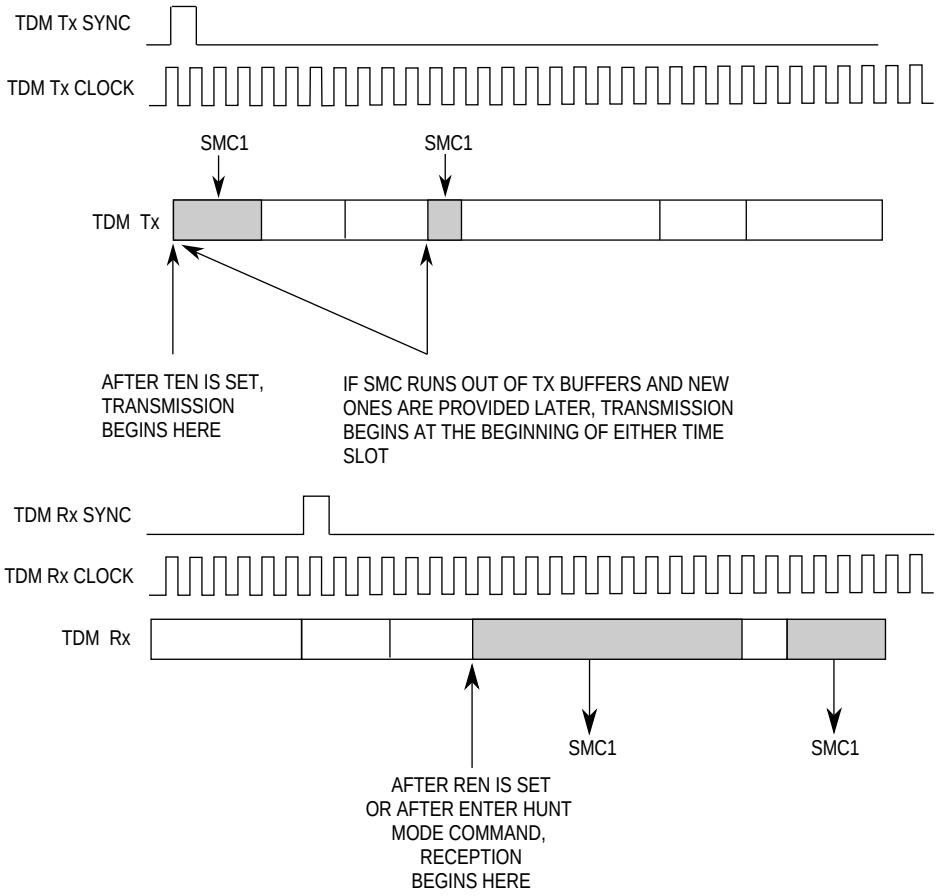


Figure 7-79. Synchronization with the TSA

Once the REN bit is set in SMCMR, the first time slot after frame sync causes the SMC receiver to achieve synchronization. Data will begin to be received immediately, but only during the defined receive time slots. The receiver will continue to receive data during its defined time slots until the REN bit is cleared by the user. If the ENTER HUNT MODE command is executed, the receiver will lose synchronization, close the current buffer, and resynchronize to the first time slot after the frame sync.

Once the TEN bit is set in SMCMR, the SMC waits for the transmit FIFO to be loaded, before attempting to achieve synchronization. Once the transmit FIFO is loaded, synchronization and transmission begin on the first bit of the first time slot after the frame sync. Idles (ones) are transmitted until data begins transmission.

If the SMC runs out of transmit buffers and a new transmit buffer is provided later, idles will be transmitted during the gap between data buffers, and data transmission from the later data buffer will begin at the beginning of an SMC time slot, but not necessarily the first time slot after the frame sync. Thus, if the user wishes to maintain a certain bit alignment beginning with the first time slot, the user should always make sure that at least one Tx BD is always ready and that no underruns occur. Otherwise, the SMC transmitter should be disabled and reenabled. See 7.11.5 Disabling the SMCs on the Fly for a description of how to

safely disable and reenable the SMC (simply clearing TEN and setting TEN may not be sufficient).

7.11.10.8 SMC TRANSPARENT COMMAND SET. The following transmit and receive commands are issued to the CR.

7.11.10.8.1 Transmit Commands. The following paragraphs describe the transparent transmit commands.

STOP TRANSMIT Command. After a hardware or software reset and the enabling of the channel in the SMC mode register, the channel is in the transmit enable mode and starts polling the first BD in the table.

The STOP TRANSMIT command disables the transmission of frames on the transmit channel. If this command is received by the transparent controller during frame transmission, transmission of that buffer is aborted after the contents of the FIFO are transmitted (up to 2 characters). The TBPTR is not advanced to the next BD, no new BD is accessed, and no new buffers are transmitted for this channel. The transmitter will send idles until the RESTART TRANSMIT command is given.

RESTART TRANSMIT Command. The RESTART TRANSMIT command is used to begin or resume transmission from the current TBPTR in the channel's Tx BD table. When this command is received by the channel, it will start polling the R-bit in this BD. This command is expected by the SMC after a STOP TRANSMIT command and the disabling of the channel in its mode register, or after a transmitter error (underrun) occurs.

INIT TX PARAMETERS Command. This command initializes all the transmit parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the transmitter is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both transmit and receive parameters.

7.11.10.8.2 Receive Commands. The following paragraphs describe the transparent receive commands.

ENTER HUNT MODE Command. This command 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 subsequent data that is received. If the SMC is not in the process of receiving data, the buffer is not closed. Additionally, this command causes the receiver to wait for a re-synchronization, before further reception continues.

CLOSE Rx BD Command. The CLOSE Rx BD command is used to force 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 subsequent data that is received. If the SMC is not in the process of receiving data, no action is taken by this command.

INIT RX PARAMETERS Command. Initializes all the receive parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the receiver is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both receive and transmit parameters.

7.11.10.9 SMC TRANSPARENT ERROR-HANDLING PROCEDURE. The SMC reports message reception and transmission error conditions using the channel BDs and the SMC event register.

7.11.10.9.1 Transmission Error (Underrun). When this error occurs, the channel terminates buffer transmission, closes the buffer, sets the UN bit in the BD, and generates the TXE interrupt if it is enabled. The channel resumes transmission after the reception of the RESTART TRANSMIT command. Underrun cannot occur between frames.

7.11.10.9.2 Reception Error (Overrun). The SMC maintains an internal FIFO for receiving data (shift register plus data register). The CP begins programming the SDMA channel (if the data buffer is in external memory) when the first character is received into the FIFO. If a FIFO overrun occurs, the SMC writes the received data character to the internal FIFO over the previously received character. The previous character and its status bits are lost. Following this, the channel closes the buffer, sets the OV bit in the BD, and generates the RX interrupt if it is enabled. Reception then continues normally.

7.11.10.10 SMC TRANSPARENT MODE REGISTER (SMCMR). The operating mode of an SMC is defined by the SMCMR. The SMCMR is a 16-bit, memory-mapped, read-write register. The register is cleared at reset. The function of bits 7–0 is common to each SMC protocol. The function of bits 15–8 varies according to the protocol selected by the SM bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	CLEN				—		REVD	—		SM		DM		TEN	REN

Bits 15, 10, 9, 7, 6—Reserved

CLEN—Character Length

CLEN is programmed with a value from 3 to 15 to obtain 4 to 16 bits per character. If the character length is less than 8 bits, the MSBs of the byte in buffer memory are not used on transmit and are written with zeros on receive. If the character length is more than 8 bits, but less than 16 bits, the MSBs of the word in buffer memory will not be used on transmit and will be written with zeros on receive.

NOTES

The values 0 to 2 should not be written to CLEN, or erratic behavior may result.

Larger character lengths increase the potential performance of the SMC channel and lower the performance impact on other channels. For instance, the use of 16-bit characters, rather than 8-bit characters, is encouraged if 16-bit characters are acceptable in the end application.

REVD—Reverse Data

0 = Normal mode
1 = Reverse the character bit order; the MSB is transmitted first.

SM—SMC Mode

00 = GCI or SCIT support
01 = Reserved
10 = UART
11 = Totally transparent operation (must be selected for SMC transparent operation)

DM—Diagnostic Mode

00 = Normal operation
01 = Local loopback mode
10 = Echo mode
11 = Reserved

TEN—SMC Transmit Enable

0 = SMC transmitter disabled
1 = SMC transmitter enabled

NOTES

Once the SMC transmit enable bit is cleared, the bit must not be reenabled for at least 3 serial clocks.

REN—SMC Receive Enable

0 = SMC receiver disabled
1 = SMC receiver enabled

7.11.10.11 SMC TRANSPARENT RECEIVE BUFFER DESCRIPTOR (RX BD). The CP reports information about the received data for each buffer using Rx BDs. The CP closes the current buffer, generates a maskable interrupt, and starts to receive data into the next buffer after one of the following events:

1. Detecting an overrun error
2. Detecting a full receive buffer
3. Issuing the ENTER HUNT MODE command

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**E**	—	**W**	**I**	—	—	**CM**	—	—	—	—	—	—	—	OV	—
OFFSET + 2	DATA LENGTH															
OFFSET + 4	**RX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE : Entries in boldface must be initialized by the user

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 CPU32+ 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 CPU32+ core should not write any fields of this Rx BD.

Bits 14, 11, 10, 8–2, 0—Reserved

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.

I—Interrupt

0 = No interrupt is generated after this buffer has been filled.
1 = The RX bit in the event register will be set when this buffer has been completely filled by the CP, indicating the need for the CPU32+ core to process the buffer. The RX bit can cause an interrupt if it is enabled.

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 overwritten automatically when the CP next accesses this BD. However, the E-bit will be cleared if an error occurs during reception, regardless of the CM bit.

The following status bit is written by the CP after the received data has been placed into the associated data buffer.

OV—Overrun

A receiver overrun occurred during message reception.

Data Length

The data length is the number of octets that the CP has written into this BD's data buffer. It is written only once by the CP as the buffer is closed.

NOTE

The actual amount of memory allocated for this buffer should be greater than or equal to the contents of the MRBLR.

Rx Data Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, must be even. The buffer may reside in either internal or external memory.

7.11.10.12 SMC TRANSPARENT TRANSMIT BUFFER DESCRIPTOR (TX BD). Data is presented to the CP for transmission on an SMC channel by arranging it in buffers referenced by the channel's Tx BD table. The CP confirms transmission or indicates error conditions using the BDs to inform the processor that the buffers have been serviced.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**R**	—	**W**	**I**	**L**	—	**CM**	—	—	—	—	—	—	—	UN	—
OFFSET + 2	**DATA LENGTH**															
OFFSET + 4	**TX DATA BUFFER POINTER**															
OFFSET + 6																

NOTE : Entries in boldface must be initialized by the user

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.

Bits 14, 10, 8–2, 0—Reserved

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 receive incoming data into the first BD in the table (the BD pointed to by TBASE). The number of Tx BDs in this table is programmable and is determined by the W-bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.

1 = The TX or TXE bit in the event register will be set when this buffer has been serviced. TX and TXE can cause interrupts if they are enabled.

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) will be transmitted immediately following 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 transmitted, the transmitter will require synchronization before the next buffer will be transmitted.

CM—Continuous Mode

0 = Normal operation.

1 = The R-bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this BD. However, the R-bit will be cleared if an error occurs during transmission, regardless of the CM bit.

UN—Underrun

The SMC encountered a transmitter underrun condition while transmitting the associated data buffer.

Data Length

The data length is the number of octets that the CP should transmit from this BD's data buffer. This value is never modified by the CP.

The data length may be even or odd; however, if the number of bits in the transparent character is greater than 8, the data length should be even. 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, since the three 9-bit characters occupy three words in memory (the 9 LSBs of each word).

Tx Data Buffer Pointer

The transmit buffer pointer, which always points to the first byte of the associated data buffer, may be even or odd (unless the 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 may be even or odd, but the pointer to 9-bit transparent characters must be even. The buffer may reside in either internal or external memory.

7.11.10.13 SMC TRANSPARENT EVENT REGISTER (SMCE). SMCE is referred to as the SMC transparent event register when the SMC is programmed for transparent mode. It is an 8-bit register used to report events recognized by the SMC channel and to generate interrupts. On recognition of an event, the SMC controller sets the corresponding bit in the SMCE. Interrupts generated by this register may be masked in the SMC mask register.

The SMCE is a memory-mapped register that 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.

7	6	5	4	3	2	1	0
—	—	—	TXE	—	BSY	TX	RX

Bits 7–5, 3—Reserved

TXE—Tx Error

An underrun error occurred on the transmitter channel.

BSY—Busy Condition

A character was received and discarded due to lack of buffers. Reception will begin after a new buffer is provided. The user may wish to execute an ENTER HUNT MODE command to cause the receiver to wait for re-synchronization.

TX—Tx Buffer

A buffer has been transmitted. If the L-bit of the Tx BD is set, this bit is set when the last data character begins to be transmitted; the user must wait one character time to be sure

that the data was completely sent over the transmit pin. If the L-bit of the Tx BD is cleared, this bit is set when the last data character is written to the transmit FIFO; the user must wait two character times to be sure that the data was completely sent over the transmit pin.

RX—Rx Buffer

A buffer has been received on the SMC channel and its associated Rx BD is now closed. This bit is set after the last character was written to the buffer.

7.11.10.14 SMC TRANSPARENT MASK REGISTER (SMCM). The SMCM is referred to as the SMC transparent mask register when the SMC is operating in transparent mode. It is an 8-bit read-write register that has the same bit format as the transparent event register. If a bit in the SMCM is a one, the corresponding interrupt in the transparent event register will be enabled. If the bit is zero, the corresponding interrupt in the transparent event register will be masked. This register is cleared upon reset.

7.11.11 SMC Transparent NMSI Example

The following list is an initialization sequence for operation of the SMC1 transparent channel over its own set of pins. The transmit and receive clocks are provided from the CLK3 pin (no baud rate generator is used), and the $\overline{\text{SMSYNx}}$ pin is used to obtain synchronization. (The SMC UART example shows an example of configuring the baud rate generator.)

1. The SDCR (SDMA Configuration Register) should be initialized to $0740, rather than being left at its default value of $0000.
2. Configure the port B pins to enable the SMTXD1, SMRXD1, and SMSYN1. Write PBPAR bits 6, 7, and 8 with ones. Write PBDIR bits 6, 7, and 8 with zeros. Write PBODR bits 6, 7, and 8 with zeros.
3. Configure the port A pins to enable CLK3. Write PBPAR bit 10 and PADIR bit 10 with a one. Write PBDIR bit 10 with a zero. The other functions of this pin are the timers or the TSA.
   These alternate functions cannot be used on this pin.
4. Connect the CLK3 clock to SMC1 using the SI. Write the SMC1 bit in SIMODE with a 0. Write the SMC1CS bits in SIMODE with 110.
5. Write RBASE and TBASE in the SMC parameter RAM to point to the Rx BD and Tx BD in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.
6. Program the CR to execute the INIT RX & TX PARAMS command for this channel." For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
7. Write RFCR with $18 and TFCR with $18 for normal operation.
8. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 16 bytes, so MRBLR = $0010.
9. Initialize the Rx BD. Assume the Rx data buffer is at $00001000 in main memory.

Write $B000 to Rx_BD_Status. Write $0000 to Rx_BD_Length (not required—done for instructional purposes only). Write $00001000 to Rx_BD_Pointer.

10. Initialize the Tx BD. Assume the Tx data buffer is at $00002000 in main memory and contains five 8-bit characters. Write $B000 to Tx_BD_Status. Write $0005 to Tx_BD_Length. Write $00002000 to Tx_BD_Pointer.
11. Write $FF to the SMCE to clear any previous events.
12. Write $13 to the SMCM to enable all possible SMC interrupts.
13. Write $00000010 to the CIMR to allow SMC1 to generate a system interrupt. (The CICR should also be initialized.)
14. Write $3830 to SMCMR to configure 8-bit characters, non-reversed data, and normal operation (not loopback). Notice that the transmitter and receiver have not been enabled yet.
15. Write $3833 to SMCMR to enable the SMC transmitter and receiver. This additional write ensures that the TEN and REN bits will be enabled last.

NOTE

After 5 bytes have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after 16 bytes have been received. Any additional receive data beyond 16 bytes will cause a busy (out-of-buffers) condition since only one Rx BD was prepared.

7.11.12 SMC Transparent TSA Example

The following list is an initialization sequence for operation of the SMC1 transparent channel over the TSA. It is assumed that the TSA and the TDM pins already have been set up to route time slot data to the SMC transmitter and receiver. (7.8 Serial Interface with Time Slot Assigner shows examples of how to configure the TSA.) The transmit and receive clocks and synchronization signals are provided internally from the TSA.

1. Write RBASE and TBASE in the SMC parameter RAM to point to the Rx BD and Tx BD in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.
2. Program the CR to execute the INIT RX & TX PARAMS command for this channel." For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
3. Write RFCR with $18 and TFCR with $18 for normal operation.
4. Write MRBLR with the maximum number of bytes per receive buffer. For this case,assume 16 bytes, so MRBLR = $0010.
5. Initialize the Rx BD. Assume the Rx data buffer is at $00001000 in main memory. Write $B000 to Rx_BD_Status. Write $0000 to Rx_BD_Length (not required—done for instructional purposes only). Write $00001000 to Rx_BD_Pointer.
6. Initialize the Tx BD. Assume the Tx data buffer is at $00002000 in main memory and

contains five 8-bit characters. Write $B000 to Tx_BD_Status. Write $0005 to Tx_BD_Length. Write $00002000 to Tx_BD_Pointer.

7. Write $FF to the SMCE to clear any previous events.
8. Write $13 to the SMCM to enable all possible SMC interrupts.
9. Write $00000010 to the CIMR to allow SMC1 to generate a system interrupt. (The CICR should also be initialized.)
10. Write $3830 to SMCMR to configure 8-bit characters, non-reversed data, and normal operation (not loopback). Notice that the transmitter and receiver have not been enabled yet.
11. Write $3833 to SMCMR to enable the SMC transmitter and receiver. This additional write ensures that the TEN and REN bits will be enabled last.

NOTE

After 5 bytes have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after 16 bytes have been received. Any additional receive data beyond 16 bytes will cause a busy (out-of-buffers) condition since only one Rx BD was prepared.

7.11.13 SMC Interrupt Handling

The following list describes what would normally occur within an interrupt handler for the SMC.

1. Once an interrupt occurs, read the SMCE to see which sources have caused interrupts. The SMCE bits would normally be cleared at this time.
2. Process the Tx BD to reuse it if the TX bit was set in SMCE. Extract data from the Rx BD if the RX bit was set in SMCE. To transmit another buffer, simply set the Tx BD R-bit.
3. Clear the SMC1 bit in the CISR.
4. Execute the RTE instruction.

7.11.14 SMC as a GCI Controller

The SMC can be used to control the C/I and to monitor channels of the GCI frame. When using the SCIT configuration of GCI, one SMC can handle SCIT channel 0, and the other SMC can handle SCIT channel 1. The main features are as follows:

- Each SMC Channel 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 SCIT Channel 0 and Channel 1
- Full-Duplex Operation
- Local Loopback and Echo Capability for Testing

NOTE

The SMCs on the QUICC differ from the SMCs on the MC68302. On the QUICC, a single SMC handles both the monitor and C/I fields of a GCI channel. On the MC68302, one SMC handles the monitor field, and another SMC is required for the C/I field. Additionally, the MC68302 cannot use SCIT channel 1; whereas, the QUICC can.

To use the SMC GCI channels properly, the TSA in the SI must be configured to route the monitor and C/I channels to the desired SMC. See 7.8 Serial Interface with Time Slot Assigner for more details on how to program this configuration. The following SMC discussion assumes that this time-slot routing is programmed properly.

7.11.14.1 SMC GCI MEMORY MAP. The GCI parameter RAM area begins at the same offset from each SMC base area.

The SMC GCI mode has a very different set of parameter RAM than the SMC UART or SMC transparent modes. In the SMC GCI mode, the general-purpose parameter RAM contains the BDs, rather than pointers to the BDs. Contrast Table 7-15 with Table 7-12 to see these differences. Additionally, the SMC in GCI mode contains no protocol-specific parameter RAM.

Table 7-15. SMC GCI Parameter RAM

Address	Name	Width	Description
SMC Base + 00	M_RxBD	Word	Monitor Channel Rx BD
SMC Base + 02	M_TxBD	Word	Monitor Channel Tx BD
SMC Base + 04	CI_RxBD	Word	C/I Channel Rx BD
SMC Base + 06	CI_TxBD	Word	C/I Channel Tx BD
SMC Base + 08	RSTATE	Long	Rx & Tx Internal State
SMC Base + 0C	M_RxD	Word	Monitor Rx Data
SMC Base + 0E	M_TxD	Word	Monitor Tx Data
SMC Base + 10	CI_RxD	Word	C/I Rx Data
SMC Base + 12	CI_TxD	Word	C/I Tx Data

NOTES:
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 areas for RISC use only.

7.11.14.1.1 SMC Monitor Channel Transmission. The monitor channel 0 is used for data exchange with a layer 1 device (e.g., reading and writing internal registers and transferring of the S and Q bits). The monitor channel 1 is used for programming and controlling voice/data modules such as CODECs.

The CPU32+ core writes the data byte into the SMC Tx BD. The SMC will transmit the data on the monitor channel. The SMC transmitter can be programmed to work in one of two modes:

Monitor Channel Protocol

In this mode, the SMC transmits the data and handles the A and E control bits according to the GCI monitor channel protocol. When using the monitor channel protocol, the user may issue the TIMEOUT command to solve deadlocks in case of errors in the A and E bit states on the data line.

7.11.14.1.2 SMC Monitor Channel Reception. The SMC receiver can be programmed to work in one of two modes:

Monitor Channel Protocol

In this mode, the SMC receives the data and handles the A and E control bits according to the GCI monitor channel protocol. When a received data byte is stored by the CP in the SMC Rx BD, a maskable interrupt is generated.

When using the monitor channel protocol, the user may issue the TRANSMIT ABORT REQUEST command. The QUICC will then transmit an abort request on the E-bit.

7.11.14.2 SMC C/I CHANNEL HANDLING. The C/I channel (in SCIT configuration, C/I channel 0) is used to control the layer 1 device. The layer 2 device in the TE sends commands and receives indication to/from the upstream layer 1 device via 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 (e.g., CODECs).

7.11.14.2.1 SMC C/I Channel Transmission. The CPU32+ core writes the data byte into the SMC C/I Tx BD. The SMC will transmit the data continuously on the C/I channel to the physical layer device.

7.11.14.2.2 SMC C/I Channel Reception. The SMC receiver continuously monitors the C/I channel. When a change in the data is recognized and this value is received in two successive frames, it will be interpreted as valid data. This is referred to as the double last-look method. The received data byte is stored by the CP in the C/I Rx BD, and a maskable interrupt is generated. If the SMC is configured to support SCIT channel 1, the double last-look method is not used.

7.11.14.3 SMC COMMANDS IN GCI MODE. The following commands are issued to the CR.

INIT TX AND RX PARAMETERS Command. This command initializes the transmit and receive parameters in the parameter RAM to their reset state. This command is especially useful when switching protocols on a given serial channel.

TRANSMIT ABORT REQUEST Command. This receiver command may be issued when the QUICC implements the monitor channel protocol. When issued, the QUICC sends an abort request on the A-bit.

TIMEOUT Command. This transmitter command may be issued when the QUICC implements the monitor channel protocol. It is issued because the device is not responding or because GCI A-bit errors are detected. When issued, the QUICC sends an abort request on the E-bit.

7.11.14.4 SMC GCI MODE REGISTER (SMCMR). The operating mode of an SMC is defined by the SMCMR. The SMCMR is a 16-bit, memory-mapped, read-write register. The register is cleared at reset. The functions of bits 7–0 are common to each SMC protocol. The functions of bits 15–8 vary according to the protocol selected by the SM bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	CLEN				ME	—	C#	—		SM		DM		TEN	REN

Bit 15, 9, 7, 6—Reserved

These bits should be cleared by the user.

CLEN—Character Length

This value is used to define the total number of bits in the C/I and monitor channels of the SCIT channel 0 or channel 1. CLEN ranges from 0 to 15 and specifies values from 1 to 16 bits. CLEN should be written with 13 for the SCIT channel 0 or GCI (8 data bits, plus A and E bits, plus 4 C/I bits = 14 bits). CLEN should be written with 15 for the SCIT channel 1 (8 data, bits, plus A and E bits, plus 6 C/I bits = 16 bits).

ME—Monitor Enable

0 = The SMC does not support the monitor channel.
1 = The SMC supports the monitor channel with either the transparent or monitor channel protocol as defined in the MP bit.

C#—SCIT Channel Number

0 = SCIT channel 0
1 = SCIT channel 1 (required for Siemens ARCOFI and SGS S/T chips)

SM—SMC Mode

00 = GCI or SCIT support (required for SMC GCI or SCIT operation)
01 = Reserved
10 = UART
11 = Totally transparent operation

DM—Diagnostic Mode

00 =Normal operation
01 =Local loopback mode
10 = Echo mode
11 = Reserved

TEN—SMC Transmit Enable

0 = SMC transmitter disabled
1 = SMC transmitter enabled

REN—SMC Receive Enable

0 = SMC receiver disabled
1 = SMC receiver enabled

7.11.14.5 SMC MONITOR CHANNEL RX BD. The CP reports information about the monitor channel receive byte using this BD.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E	L	ER	MS	—		—	—	DATA							

E—Empty

0 = This bit is cleared by the CP to indicate that data byte associated with this BD is now available to the CPU32+ core.
1 = This bit is set by the CPU32+ core to indicate that the data byte associated with this BD has been read.

When the SMC implements the monitor channel protocol, the SMC will wait until this bit is set by the CPU32+ core before acknowledging the monitor channel data. In the transparent mode, additional received data bytes will be discarded until the E-bit is set by the CPU32+ core.

L—Last (EOM)

This bit is valid only when the SMC implements the monitor channel protocol. This bit is set when the end-of-message (EOM) indication is received on the E-bit.

NOTE

When this bit is set, the data byte is not valid.

ER—Error Condition

This bit is valid only when the SMC implements the monitor channel protocol. This bit is set when an error condition occurs on the monitor channel protocol. (A new byte is transmitted before the SMC acknowledges the previous byte.)

MS—Data Mismatch

This bit is valid only when the SMC implements the monitor channel protocol. This bit is set when two different consecutive bytes are received and is 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 Rx BD.

Bits 11–10—Reserved

These bits should be cleared by the user.

DATA—Data Field

The data field contains the monitor channel data byte received by the SMC.

7.11.14.6 SMC MONITOR CHANNEL TX BD. The CP reports the information about the monitor channel transmit byte using this BD.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	L	AR	—			—	—	DATA							

R—Ready

0 = This bit is cleared by the CP after transmission. The Tx BD is now available to the CPU32+ core.

1 = This bit is set by the CPU32+ core to indicate that the data byte associated with this BD is ready for transmission.

L—Last (EOM)

This bit is valid only when the SMC implements the monitor channel protocol. When this bit is set, the SMC will first transmit the buffer's data and then transmit the end-of-message (EOM) indication on the E-bit.

AR—Abort Request

This bit is valid only when the SMC implements the monitor channel protocol. This bit is set by the SMC when an abort request is received on the A-bit. The SMC transmitter will transmit the EOM on the E-bit after an abort request is received.

Bits 12–10—Reserved

These bits should be cleared by the user.

DATA—Data Field

The data field contains the data to be transmitted by the SMC on the monitor channel.

7.11.14.7 SMC C/I CHANNEL RECEIVE BUFFER DESCRIPTOR (RX BD). The CP reports information about the C/I channel receive byte using this BD.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E	—							C/I DATA						—	

E—Empty

0 = This bit is cleared by the CP to indicate that data byte associated with this BD is now available to the CPU32+ core.

1 = This bit is set by the CPU32+ core to indicate that the data byte associated with this BD has been read.

NOTE

Additional data received will be discarded until the E-bit is set.

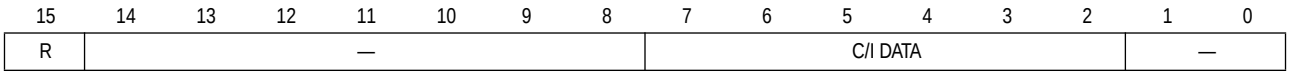
Bits 14–8,1-0—Reserved

These bits should be cleared by the user.

C/I DATA—Command/Indication Data Bits

C/I DATA is a 4-bit data field for C/I channel 0 and a 6-bit data field for C/I channel 1. It contains the data received from the C/I channel. For C/I channel 0, bits 5-2 contain the 4-bit data field, and bits 7 and 6 are always written with zeros. For C/I channel 1, bits 7-2 contain the 6-bit data field..

7.11.14.8 SMC C/I CHANNEL TRANSMIT BUFFER DESCRIPTOR (TX BD). The CP reports information about the C/I channel transmit byte using the BD.



R—Ready

0 = This bit is cleared by the CP after transmission to indicate that the BD is now available to the CPU32+ core.
1 = This bit is set by the CPU32+ core to indicate that the data associated with this BD is ready for transmission.

Bits 14–6—Reserved

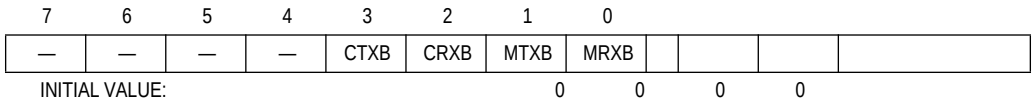
These bits should be cleared by the user.

C/I DATA—Command/Indication Data Bits

C/I DATA is a 4-bit data field for C/I channel 0 and a 6-bit data field for C/I channel 1. It contains the data to be transmitted onto the C/I channel. For C/I channel 0, bits 5-2 contain the 4-bit data field, and bits 7 and 6 are always written with zeros. For C/I channel 1, bits 7-2 contain the 6-bit data field.

7.11.14.9 SMC EVENT REGISTER (SMCE). The SMCE is an 8-bit register used to report events recognized by the SMC channel and to generate interrupts. On recognition of event, the SMC sets its corresponding bit in this register. Interrupts generated by this register may be masked in the SMC mode register.

The SMCE is a memory-mapped register that 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 to the CPM interrupt controller. This register is cleared at reset.



Bits 7–4—Reserved

CTXB—C/I Channel Buffer Transmitted

The C/I transmit buffer became empty.

CRXB—C/I Channel Buffer Received

The C/I receive buffer is full.

MTXB—Monitor Channel Buffer Transmitted

The monitor transmit buffer became empty.

MRXB—Monitor Channel Buffer Received

The monitor receive buffer is full.

7.11.14.10 SMC MASK REGISTER (SMCM). The SMCM is an 8-bit, memory-mapped, read-write register. It has the same bit format as the SMC event register. If a bit in the SMCM is a one, the corresponding interrupt in the SMC event register will be enabled. If the bit is zero, the corresponding interrupt in the SMC event register will be masked. The SMCM is clear upon reset.

7.12 SERIAL PERIPHERAL INTERFACE (SPI)

The SPI allows the QUICC to exchange data between other QUICC chips, the MC68302, the M68HC11 and M68HC05 microcontroller families, and a number of peripheral devices such as EEPROMs, real-time clock devices, A/D converters, and ISDN devices.

7.12.1 Overview

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 transmitted and received simultaneously. Refer to Figure 7-80 for the SPI block diagram.

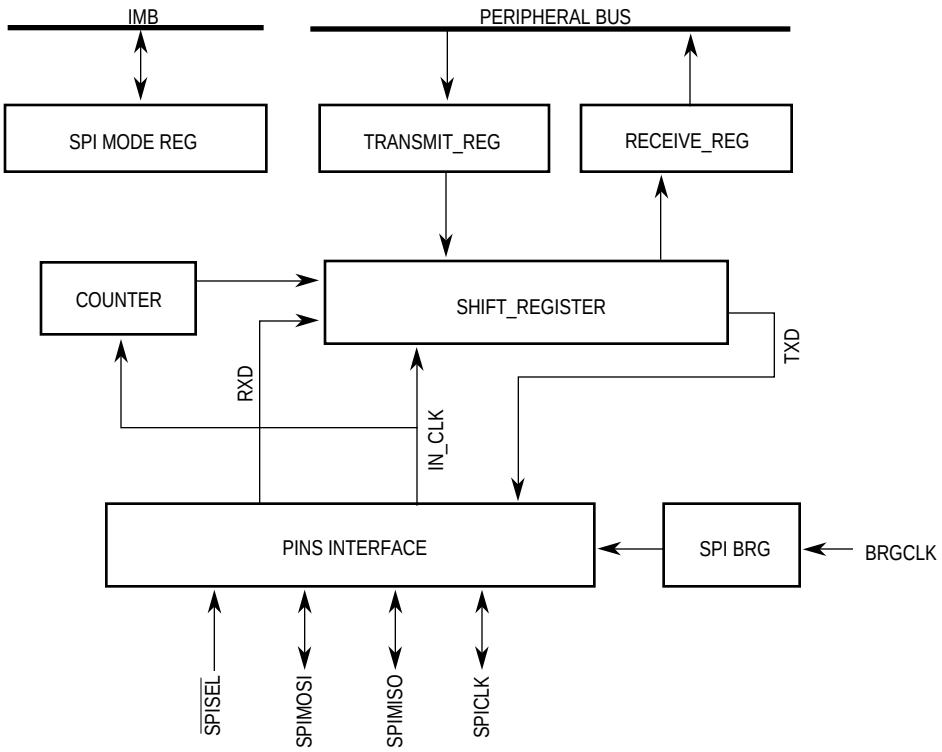


Figure 7-80. SPI Block Diagram

NOTE

The SPI is a superset of the MC68302 serial communications port (SCP).

The SPI receiver and transmitter are double-buffered as shown in the block diagram. This corresponds to an effective FIFO size (latency) of 2 characters.

Note that the LSB of the SPI is labeled as data bit 0 on the serial line; whereas, other devices, such as the MC145554 CODEC, may label the MSB as data bit 0. The QUICC SPI bit 7 (MSB) is shifted out first.

When the SPI is not enabled in the SPMODE, it consumes minimal power.

7.12.2 SPI Key Features

The SPI contains the following key features:

- Four-Wire Interface (SPIMOSI, SPIMISO, SPICLK, and $\overline{\text{SPISEL}}$)
- Full-Duplex Operation
- Works with Data Characters from 4 to 16 bits in length
- Supports Back-to-Back Character Transmission and Reception
- Master or Slave SPI Modes Supported

- Multi-Master Environment Support
- Continuous Transfer Mode for auto scanning of a peripheral
- Supports Clock Rates up to 6.25 MHz in Master Mode and up to 12.5 MHz in Slave Mode (assuming a 25-MHz system clock)
- Independent Programmable Baud Rate Generator
- Programmable Clock Phase and Polarity
- Open-Drain Output Pins support multi-master configuration
- Local Loopback Capability for Testing

7.12.3 SPI Clocking and Pin Functions

The SPI can be configured as a master for the serial channel, meaning that it generates both the enable and clock signals, or as slave, meaning that the enable and clock signals are inputs to the SPI. The SPI also supports operation in a multi-master environment.

When the SPI is a master, the SPI baud rate generator is used to generate the SPI transmit and receive clocks. The SPI baud rate generator takes its input from the BRGCLK.

The BRGCLK is generated in the clock synthesizer of the QUICC specifically for the SPI baud rate generator and the other four baud rate generators in the CPM. BRGCLK defaults to the system frequency (25 MHz). However, the clock synthesizer in the SIM60 has an option to divide the BRGCLK by 1, 4, 16, or 64 before it leaves the clock synthesizer. Whatever the resulting frequency of BRGCLK, the user may use that BRGCLK frequency as the input to the SPI baud rate generator.

NOTE

User should note the maximum clock rate dose not equal maximum data rate. See Appendix A Serial Performance for more detail.

The ability to reduce the frequency of BRGCLK before it leaves the clock synthesizer is useful for two reasons. First, in a low-power mode, the baud rate generator clocking could be a significant factor in overall QUICC power consumption. Thus, if none of the QUICC baud rate generators need to generate high frequencies nor require a high resolution in the user application, a lower frequency BRGCLK may be input to the baud rate generators. Secondly, the user may wish to dynamically change the general system clock frequency in the clock synthesizer (SLOW GO mode), while still having the baud rate generator run at the original frequency. The BRGCLK allows this option also.

The SPI master-in slave-out (SPIMISO) pin is an input in master mode and an output in slave mode. The SPI master-out slave-in (SPIMOSI) pin is an output in master mode and an input in slave mode. The reason the pins names SPIMOSI and SPIMISO change functionality between master and slave mode is to support a multi-master configuration that allows communication from any SPI to any other SPI with the same hardware configuration.

When the SPI is working as a master, SPICLK is the clock output signal that shifts in the received data from the SPIMISO pin and shifts out the transmitted data to the SPIMOSI pin. Additionally, an SPI master device must provide a slave select signal output to enable the SPI slave devices. This may be implemented using one of the QUICC's general-purpose I/O pins. The $\overline{\text{SPISEL}}$ pin should not be asserted while the SPI is working as a master, or the SPI will indicate an error.

When the SPI is working as a slave, SPICLK is the clock input signal that shifts in the received data from the SPIMOSI pin and shifts out the transmitted data to the SPIMISO pin. The $\overline{\text{SPISEL}}$ pin provided by the QUICC is the enable input to the SPI slave.

When the SPI is working in a multi-master environment, the $\overline{\text{SPISEL}}$ pin is still an input and is used to detect an error condition when more then one master is operating.

SPICLK is a gated clock (i.e., the clock only toggles while data is being transferred). The user can select any of four combinations of SPICLK phase and polarity using two bits in the SPI mode register (SPMODE).

The SPI pins can also be configured as open-drain pins to support a multi-master configuration where the same SPI pin can be driven by the QUICC or an external SPI device.

7.12.4 SPI Transmit/Receive Process

The following paragraphs discuss SPI master, slave, and multi-master operation.

7.12.4.1 SPI MASTER MODE. When the SPI functions in master mode, the SPI transmits a message to the peripheral (SPI slave), which in turn sends back a simultaneous reply. When the QUICC works with more than one slave, it can use the general-purpose parallel I/O pins to selectively enable different slaves.

To begin the data exchange, the CPU32+ core writes the data to be transmitted into a data buffer, configures a Tx BD with its R-bit set and configures one or more Rx BDs. The CPU32+ core should then set the STR bit in the SPCOM to start transmission of data. The data will begin transmission once the SDMA channel has loaded the transmit FIFO with data.

The SPI controller then generates programmable clock pulses on the SPICLK pin for each character and shifts the data out on the SPIMOSI pin. At the same time, the SPI shifts receive data in from the SPIMISO pin. This receive data is written into a receive buffer using the next available Rx BD. The SPI will continue transmitting and receiving characters until the transmit buffer has been completely transmitted or an error has occurred ($\overline{\text{SPISEL}}$ pin unexpectedly asserted). The CP then clears the R and E bits in the Tx BD and Rx BD, and issues a maskable interrupt to the CPM interrupt controller.

When multiple Tx BDs are ready for transmission, the Tx BD L-bit determines whether the SPI continues to transmit without waiting for the STR bit to be set again. If the L-bit is cleared, the data from the next Tx BD will begin its transmission following the transmission of data from the first Tx BD. In most cases, the user should see no delay on the SPIMOSI pin between buffers. If the L-bit is set, transmission will cease after data from this Tx BD has

completed transmission. In addition, the current Rx BD that is used to receive data is closed after the transmission completes, even if the receive buffer is not full. Thus, the user does not need to provide receive buffers of the same length as the transmit buffers.

If the SPI is the only master in a system, then the $\overline{\text{SPISEL}}$ pin can be used as a general-purpose I/O, and the internal $\overline{\text{SPISEL}}$ signal to the SPI will always be forced inactive internally, eliminating the possibility of a multi-master error.

7.12.4.2 SPI SLAVE MODE. When the SPI functions in slave mode, the SPI receives messages from an SPI master and, in turn, sends back a simultaneous reply. The $\overline{\text{SPISEL}}$ pin must be asserted before receive clocks will be recognized. Once $\overline{\text{SPISEL}}$ is asserted, the SPICLK pin becomes an input from the master to the slave. SPICLK may be any frequency from DC to the BRGCLK/2 (i.e., 12.5 MHz for a 25-MHz system).

Before the data exchange, the CPU32+ core writes the data to be transmitted into a data buffer, configures a Tx BD with its R-bit set, and configures one or more Rx BDs. The CPU32+ core should then set the STR bit in the SPCOM to enable the SPI to prepare the data for transmission and wait for the $\overline{\text{SPISEL}}$ pin to be asserted. Data is shifted out from the slave on the SPIMISO pin and shifted in through the SPIMOSI pin. A maskable interrupt is issued upon complete transmission or reception of a full buffer or after an error has occurred (receive overrun, transmit underrun, out of receive buffers, etc.). The SPI will then continue reception using the next Rx BD in the ring until it runs out of receive buffers or the $\overline{\text{SPISEL}}$ pin is negated.

Transmission will continue until no more data is available to be transmitted or the $\overline{\text{SPISEL}}$ pin is negated. If the $\overline{\text{SPISEL}}$ pin is negated prior to all the transmit data being transmitted, transmission will cease, but the Tx BD will remain open. Further transmission from that point will continue once the $\overline{\text{SPISEL}}$ pin is reasserted and SPICLK begins toggling. After completing transmission of characters in the Tx DB, the SPI will transmit ones if $\overline{\text{SPISEL}}$ is not negated.

7.12.4.3 SPI MULTI-MASTER OPERATION. The SPI can operate in a multi-master environment in which some SPI devices are connected on the same bus. In this configuration, the SPIMOSI, SPIMISO, and SPICLK pins of all SPIs are connected together, and the $\overline{\text{SPISEL}}$ input pins are connected separately. In this environment, only one SPI device can work as a master at a time; all the others must be slaves. When the SPI is configured as a master and its $\overline{\text{SPISEL}}$ input goes active (low), a multi-master error has occurred since more than one SPI device is currently a bus master. The SPI sets the MME bit in the event register, and a maskable interrupt is issued to the CPU32+ core. It also disables the SPI operation and the output drivers of the SPI pins.

The CPU32+ core should clear the EN bit the SPMODE before using the SPI again. After the problems are corrected, the MME bit should be cleared, and the SPI should be enabled with the same procedure as after a reset.

NOTE:

The user should note the maximum data rate supported on the SPI is 500Kbps. The SPI can transfer a single character at much

higher rates (6.25MHz in master mode and 12.5MHz in slave mode). If multiple characters are to be transferred, a gap should be inserted between transmission so that it will not exceed the maximum data rate.

7.12.5 SPI Programming Model

The following paragraphs describe the registers in the SPI.

7.12.5.1 SPI MODE REGISTER (SPMODE). SPMODE is a read-write register that controls both the SPI operation mode and the SPI clock source. SPMODE is cleared by reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—	LOOP	CI	CP	DIV16	REV	M/S	EN	LEN				PM3	PM2	PM1	PM0

Bit 15—Reserved

This bit should be cleared by the user.

LOOP—Loop Mode

When set, this bit selects the local loopback operation. The transmitter output is internally connected to the receiver input; the receiver and transmitter operate normally except that the received data is ignored. (Loopback mode does not invert the SPI data, as do some SPI-type devices such as the MC68302.)

0 = Normal operation.
1 = The SPI is in loopback mode.

CI—Clock Invert

The CI bit inverts the SPI clock polarity (refer to Figure 7-81 and Figure 7-82).

0 = The inactive state of SPICLK is low.
1 = The inactive state of SPICLK is high.

CP—Clock Phase

The CP bit selects one of two fundamentally different transfer formats (refer to Figure 7-81 and Figure 7-82).

0 = SPICLK begins toggling at the middle of the data transfer.
1 = SPICLK begins toggling at the beginning of the data transfer.

DIV16—Divide by 16

The DIV16 bit selects the clock source for the SPI baud rate generator when configured as an SPI master. In slave mode, the clock source is the SPICLK pin.

0 = Use the BRGCLK as the input to the SPI baud rate generator.
1 = Use the BRGCLK/16 as the input to the SPI baud rate generator.

REV—Reverse Data

The REV bit determines the receive and transmit character bit order.

0= Reverse data—LSB of character transmitted and received first.
1= Normal operation—MSB of character transmitted and received first.

M/$\overline{S}$—Master/Slave

The M/$\overline{S}$ bit configures the SPI to work as a master or a slave.

0 = SPI is a slave.
1 = SPI is a master.

EN—Enable SPI

The EN bit enables the SPI operation. Note that SPIMOSI, SPIMISO, SPICLK, and $\overline{SPISEL}$ should be configured to connect to the SPI as described in 7.14.7 Port B Registers. When the EN bit is cleared, the SPI is in a reset state and consumes minimal power—the SPI baud rate generator is not functioning and the input clock is disabled.

0 = SPI is disabled.
1 = SPI is enabled.

NOTE

Other bits of the SPMODE should not be modified by the user while EN is set.

LEN—Character Length

The LEN field specifies how many bits are in a character. The value 0000 corresponds to 1 bit, and the value 1111 corresponds to 16 bits. Acceptable values are in the range of 4 to 16 bits inclusive. Programming a value less than 4 bits may cause erratic behavior.

If the LEN value is less than or equal to a byte, there will be LEN number of valid bits in every byte (8 bits) in memory. If the LEN value is greater than a byte, there will be LEN number of valid bits in every word (16 bits) in memory (See the following example).

PM3–PM0—Prescale Modulus Select

These four bits specify the divide ratio of the prescale divider in the SPI clock generator. The BRGCLK is divided by 4 * ([PM3–PM0] + 1) giving a clock divide ratio of 4 to 64. The clock has a 50% duty cycle.

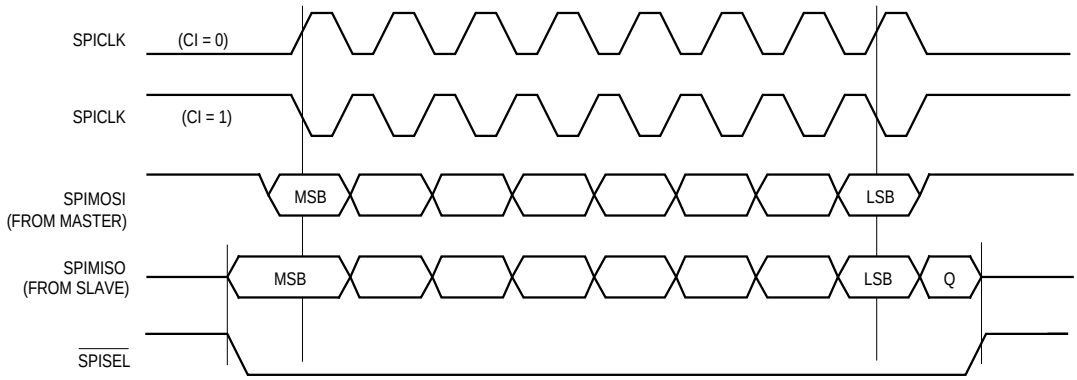
SPI Examples with Different LEN Values

These examples use LEN to illustrate using the bits described above.To help map the process, let g through v be binary symbols, x indicates a deleted bit, __ indicates original byte boundaries, and _ indicates original nybble (4-bit) boundaries - both are used to aid readability and to help understand the process. Once the data string image is determined, it is always transmitted byte by byte with the lsb first.

Let the memory contain the following binary image:

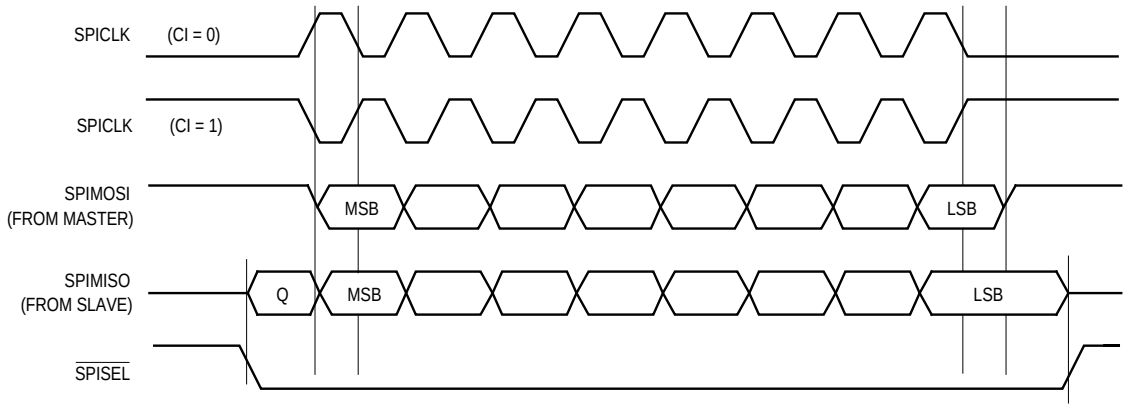
```
            msb             ghij_klmn __opqr_stuv             lsb
Example 1:
with LEN=4 (data size=5), the following data is selected:
            msb             xxxj_klmn__xxxr_stuv              lsb
with REV=0, the data string image is:
            msb             j_klmn__r_stuv                    lsb
the order of the string appearing on the line, a byte at a time is:
                           nmlk_j__vuts_r
with REV=1, the string has each byte reversed
the data string image is:
        msb             nmlk_j__vuts_r                        lsb
```

```
the order of the string appearing on the line, a byte at a time is:
                          j_klmn__r_stuv
Example 2:
with LEN=7 (data size=8), the following data is selected:
           msb             ghij_klmn__opqr_stuv             lsb
the data string selected is:
           msb             ghij_klmn__opqr_stuv             lsb
with REV=0, the string transmitted, a byte at a time, with lsb first is:
                          nmlk_jihg__vuts_rqpo
with REV=1, the string is byte reversed and transmitted, a byte at a time,
       with lsb first:
                          ghij_klmn__opqr_stuv
Example 3:
with LEN=0cH(12), (data size=0dH(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:
                          vuts_r__nmlk_jihg
with REV=1, the string is WORD reversed
                          nmlk_jihg__vuts_r
and transmitted, a byte at a time, with lsb first:
                          ghij_klmn__r_stuv
```



NOTE: Q = Undefined Signal

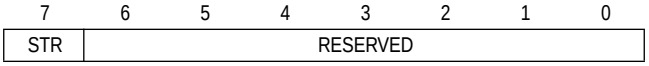
Figure 7-81. SPI Transfer Format with CP = 0



NOTE: Q = Undefined Signal

Figure 7-82. SPI Transfer Format with CP = 1

7.12.5.2 SPI COMMAND REGISTER (SPCOM). The SPCOM is an 8-bit read-write register that is used to start SPI operation.



Bits 6–0—Reserved.

These bits should be written with zeros by the user.

STR—Start Transmit

When the SPI is configured as a master, setting the STR bit to one causes the SPI controller to start the transmission and reception of data from/to the SPI transmit/receive buffers (if they are configured as ready by the user).

When the SPI is configured as a slave, setting the STR bit to one when the SPI is idle (between transfers) causes the SPI to load the transmit data register from the SPI transmit buffer and start transmission as soon as the next SPI input clocks and select signal are received.

The STR bit is cleared automatically after one system clock cycle.

7.12.5.3 SPI PARAMETER RAM MEMORY MAP. The SPI parameter RAM area (see Table 7-16) begins at the SPI base address. This area is used for the general SPI parameters. The user will notice that it is similar to the SCC general-purpose parameter RAM.

Table 7-16. SPI Parameter RAM Memory Map

Address	Name	Width	Description
SPI Base + 00	**RBASE**	Word	Rx BD Base Address
SPI Base+ 02	**TBASE**	Word	Tx BD Base Address
SPI Base+ 04	**RFCR**	Byte	Rx Function Code
SPI Base+ 05	**TFCR**	Byte	Tx Function Code
SPI Base+ 06	**MRBLR**	Word	Maximum Receive Buffer Length
SPI Base+ 08	RSTATE	Long	Rx Internal State
SPI Base+ 0C		Long	Rx Internal Data Pointer
SPI Base+ 10	RBPTR	Word	Rx BD Pointer
SPI Base+ 12		Word	Rx Internal Byte Count
SPI Base+ 14		Long	Rx Temp
SPI Base+ 18	TSTATE	Long	Tx Internal State
SPI Base+ 1C		Long	Tx Internal Data Pointer
SPI Base+ 20	TBPTR	Word	Tx BD Pointer
SPI Base+ 22		Word	Tx Internal Byte Count
SPI Base+ 24		Long	Tx Temp

NOTE: The items in boldface should be initialized by the user.

Certain parameter RAM values (marked in boldface) need to be initialized by the user before the SPI is enabled; other values are initialized by the CP. Once initialized, the parameter

RAM values will not normally need to be accessed by user software. They should only be modified when no SPI activity is in progress.

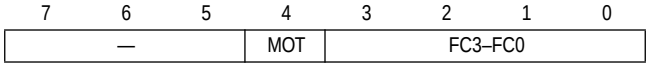
7.12.5.3.1 BD Table Pointer (RBASE, TBASE). The RBASE and TBASE entries define the starting location in the dual-port RAM for the set of BDs for receive and transmit functions of the SPI. This provides a great deal of flexibility in how BDs for an SPI are partitioned. By setting the W-bit in the last BD in each BD list, the user may select how many BDs to allocate for the transmit and receive side of the SPI. The user must initialize these entries before enabling the SPI. Furthermore, the user should not configure BD tables of the SPI to overlap any other serial channel's BDs, or erratic operation will occur.

NOTE

RBASE and TBASE should contain a value that is divisible by 8.

7.12.5.3.2 SPI Function Code Registers (RFCR, TFCR). The FC entry contains the value that the user would like to appear on the function code pins (FC3–FC0), when the associated SDMA channel accesses memory. It also controls the byte-ordering convention to be used in the transfers.

Receive Function Code Register



Bits 7–5—Reserved

These bits should be set to zero by the user.

MOT—Motorola

This bit should be set by the user to achieve normal operation. MOT *must be set* if the data buffer is located in external memory and has a 16-bit wide memory port size.

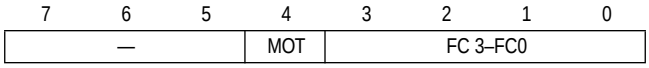
0 = DEC and Intel convention is used for byte ordering—swapped operation. It is also called little-endian byte ordering. The bytes stored in each buffer word are reversed as compared to the Motorola mode.

1 = Motorola byte ordering—normal operation. It is also called big-endian byte ordering. As data is received from the serial line and put into the buffer, the most significant byte of the buffer word contains data received earlier than the least significant byte of the same buffer word.

FC3–FC0—Function Code 3–0

These bits contain the function code value used during this SDMA channel's memory accesses. The user should write bit FC3 with a one to identify this SDMA channel access as a DMA-type access. Example: FC3–FC0 = 1000. To keep interrupt acknowledge cycles unique in the system, do not write the value 0111 binary to these bits.

Transmit Function Code Register



Bits 7–5—Reserved.

These bits should be set to zero by the user.

MOT—Motorola

This bit should be set by the user to achieve normal operation. MOT *must be set* if the data buffer is located in external memory and has a 16-bit wide memory port size.

0 = 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 Motorola mode.

1 = Motorola 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.

FC3–FC0—Function Code 3–0

These bits contain the function code value used during this SDMA channel's memory accesses. The user should write bit FC3 with a one to identify this SDMA channel access as a DMA-type access. Example: FC3-FC0 = 1000. To keep interrupt acknowledge cycles unique in the system, do not write the value 0111 (binary) to these bits.

7.12.5.3.3 Maximum Receive Buffer Length Register (MRBLR). The SPI has one MRBLR to define the receive buffer length for that SPI. MRBLR defines the maximum number of bytes that the QUICC will write to a receive buffer on that SPI before moving to the next buffer. The QUICC may write fewer bytes to the buffer than the MRBLR value if a condition such as an error or end-of-frame occurs, but it will never write more bytes than the MRBLR value. It follows, then, that buffers supplied by the user for use by the QUICC should always be of size MRBLR (or greater) in length.

The transmit buffers for an SPI are not affected in any way by the value programmed into MRBLR. Transmit buffers may be individually chosen to have varying lengths, as needed. The number of bytes to be transmitted is chosen by programming the data length field in the Tx BD.

NOTES

MRBLR is not intended to be changed dynamically while an SPI is operating. However, if it is modified in a single bus cycle with one 16-bit move (NOT two 8-bit bus cycles back-to-back), then a dynamic change in receive buffer length can be successfully achieved. This takes place when the CP moves control to the next Rx BD in the table. Thus, a change to MRBLR will not have an immediate effect. To guarantee the exact Rx BD on which the change will occur, the user should change MRBLR only while the SPI receiver is disabled.

The MRBLR value should be greater than zero and should be even if the character length of the data is greater than eight bits.

7.12.5.3.4 Receiver Buffer Descriptor Pointer (RBPTR). The RBPTR for each SPI channel points to the next BD that the receiver will transfer 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 this pointer to the value programmed in the RBASE entry. Although RBPTR need never be written by the user in most applications, it may be modified by the user when the receiver is disabled or when the user is sure that no receive buffer is currently in use.

7.12.5.3.5 Transmitter Buffer Descriptor Pointer (TBPTR). The TBPTR for each SPI channel points to the next BD that the transmitter will transfer data from when it is in idle state or to the current BD during frame transmission. After a reset or when the end of BD table is reached, the CP initializes this pointer to the value programmed in the TBASE entry. Although TBPTR need never be written by the user in most applications, it may be modified by the user when the transmitter is disabled or when the user is sure that no transmit buffer is currently in use.

7.12.5.3.6 Other General Parameters. Additional parameters are listed in Table 7-16. These parameters do not need to be accessed by the user in normal operation, and are listed only because they may provide helpful information for experienced users and for debugging.

The Rx and Tx internal data pointers are updated by the SDMA channels to show the next address in the buffer to be accessed.

The Tx internal byte count is a down-count value that is initialized with the Tx BD data length and decremented with every byte read by the SDMA channels. The Rx internal byte count is a down-count value that is initialized with the MRBLR value and decremented with every byte written by the SDMA channels.

NOTE

To extract data from a partially full buffer, the CLOSE Rx BD command may be used.

The Rx internal state, Tx internal state, Rx temp, Tx temp, and reserved areas are for RISC use only.

7.12.5.4 SPI COMMANDS. The following transmit and receive commands are issued to the CR.

7.12.5.4.1 INIT TX PARAMETERS Command. This command initializes all transmit parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the transmitter is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both transmit and receive parameters.

7.12.5.4.2 CLOSE Rx BD Command. The CLOSE Rx BD command is used to force the SPI controller to close the current Rx BD, if it is currently being used, and to use the next BD for any subsequent data that is received. If the SPI controller is not in the process of receiving data, no action is taken by this command.

7.12.5.4.3 INIT RX PARAMETERS Command. This command initializes all the receive parameters in this serial channel's parameter RAM to their reset state. This command should only be issued when the receiver is disabled. Note that the INIT TX AND RX PARAMETERS command may also be used to reset both receive and transmit parameters.

7.12.5.5 SPI BUFFER DESCRIPTOR RING. The data associated with the SPI is stored in buffers, which are referenced by BDs organized in a BD ring located in the dual-port RAM (see Figure 7-83). This ring has the same basic configuration as those used by the SCCs and SMCs.

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 actual buffers may reside in either external memory or internal memory. Data buffers may reside in the parameter area of an SCC if it is not enabled.

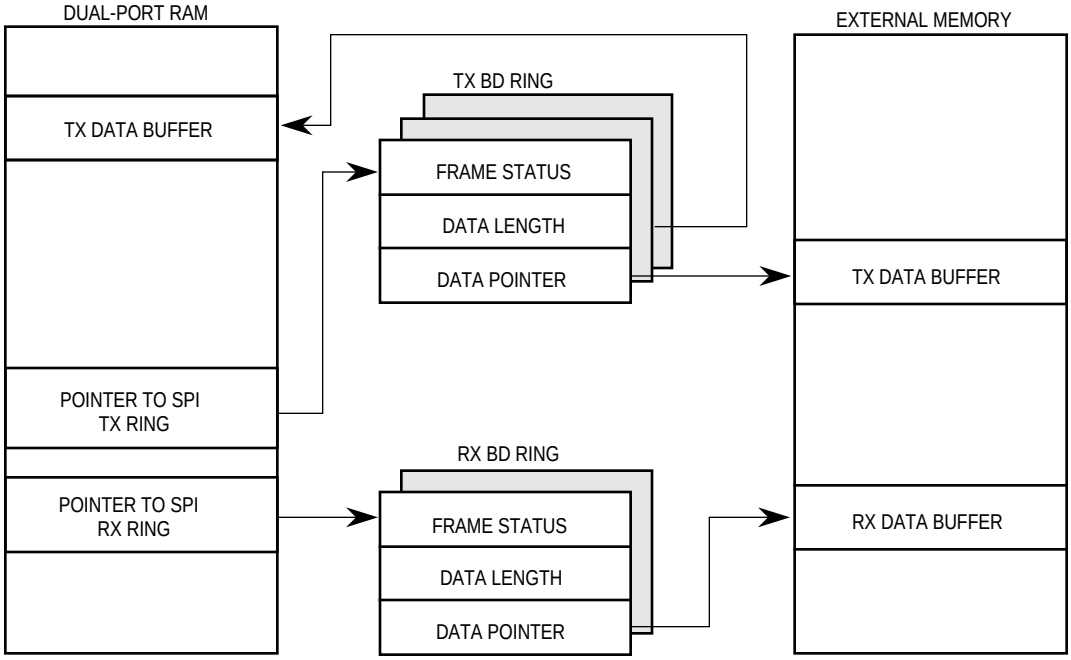


Figure 7-83. SPI Memory Structure

7.12.5.5.1 SPI Receive Buffer Descriptor (Rx BD). 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 will close the buffer when the SPI is configured as a slave and the $\overline{\text{SPISEL}}$ pin goes to an inactive state, indicating that the reception process is terminated.

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.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	—	W	I	L	—	CM	—	—	—	—	—	—	—	OV	ME
OFFSET + 2	DATA LENGTH															
OFFSET + 4	RX DATA BUFFER POINTER															
OFFSET + 6																

The following bits should be written by the CPU32+ core before enabling the SPI.

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 CPU32+ 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 CPU32+ core should not write any fields of this Rx BD.

Bits 14, 10, 8–2—Reserved

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.

I—Interrupt

0 = No interrupt is generated after this buffer has been filled.

1 = The RXB bit in the SPI event register will be set when this buffer has been completely filled by the CP, indicating the need for the CPU32+ core to process the buffer. The RXB bit can cause an interrupt if it is enabled.

CM—Continuous Mode

This bit is valid only when the SPI is configured as a master; it should be written as a zero in slave 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 overwritten automatically 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 CPU overhead.

The following status bits are written by the SPI after the received data has been placed into the associated data buffer.

L—Last

This bit is set by the SPI controller when the buffer is closed due to negation of the $\overline{\text{SPISEL}}$ pin. This can only occur when the SPI is a slave; otherwise, the ME bit is set.

0 = This buffer does not contain the last character of the message.
1 = This buffer contains the last character of the message.

OV—Overrun

A receiver overrun occurred during reception. This error can only occur when the SPI is a slave.

ME—Multi-Master Error

This buffer was closed because the $\overline{\text{SPISEL}}$ pin was asserted when the SPI was operating as a master. This indicates a synchronization problem between multiple masters on the SPI bus.

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.

Rx Data Buffer Pointer

The receive buffer pointer, which always points to the first location of the associated data buffer, must be even. The buffer may reside in either internal or external memory.

7.12.5.5.2 SPI Transmit Buffer Descriptor (Tx BD). Data to be transmitted with the SPI is presented to the CP by arranging it in buffers referenced by the Tx BD ring. The first word of the Tx BD contains status and control bits.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	R	—	W	I	L	—	CM	—	—	—	—	—	—	—	UN	ME
OFFSET + 2	DATA LENGTH															
OFFSET + 4	TX DATA BUFFER POINTER															
OFFSET + 6																

The following bits should be prepared by the user before transmission.

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.

Bits 14, 10, 8–2—Reserved

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 receive incoming data into the first BD in the table (the BD pointed to by TBASE). The number of Tx BDs in this table is programmable, and is determined only by the W-bit and the overall space constraints of the dual-port RAM.

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.

L—Last

0 = This buffer does not contain the last character of the message.
1 = This buffer contains the last character of the message.

CM—Continuous Mode

This bit is valid only when the SPI is configured as a master; it should be written as a zero in slave mode.

0 = Normal operation.
1 = The R-bit is not cleared by the CP after this BD is closed, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this BD.

The following status bits are written by the SPI after it has finished transmitting the associated data buffer.

UN—Underrun

The SPI encountered a transmitter underrun condition while transmitting the associated data buffer. This error condition is valid only when the SPI is configured as a slave.

ME—Multi-Master Error

This buffer was closed because the $\overline{\text{SPISEL}}$ pin was asserted when the SPI was operating as a master. This indicates a synchronization problem between multiple masters on the SPI bus.

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.

If the number of data bits in the character is greater than 8, then the data length should be even. Example: to transmit three characters of 8-bit data, 1 start, and 1 stop, the data length field should be initialized to 3. However, to transmit three characters of 9-bit data, the data length field should be initialized to 6 since the three 9-bit data fields occupy three words in memory (the 9 LSBs of each word).

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 (unless the number of actual data bits in the character is greater than 8 bits, in which case the transmit buffer pointer must be even). The buffer may reside in either internal or external memory.

7.12.5.6 SPI EVENT REGISTER (SPIE). The SPIE is an 8-bit register used to report events recognized by the SPI and to generate interrupts. Upon recognition of an event, the SPI sets its corresponding bit in the SPIE. Interrupts generated by this register may be masked in the SPI mask register.

The SPIE is a memory-mapped register that 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.

7	6	5	4	3	2	1	0
—	—	MME	TXE	—	BSY	TXB	RXB

Bits 7, 6, 3—Reserved

MME—Multi-Master Error

The SPI detected that the $\overline{\text{SPISEL}}$ pin was asserted externally while the SPI was in master mode.

TXE—Tx Error

An error occurred during transmission (underrun in SPI slave mode).

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.

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. The user must wait two character times to be sure that the data was completely sent over the transmit pin.

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.

7.12.5.7 SPI MASK REGISTER (SPIM). The SPIM is an 8-bit read-write register that has the same bit formats as the SPI event register. If a bit in the SPIM is one, the corresponding interrupt in the SPIE is enabled. If the bit is zero, the corresponding interrupt in the SPIE will be masked. This register is cleared at reset.

7.12.6 SPI Master Example

The following list is an initialization sequence for a high-speed use of the SPI as a master.

1. The SDCR (SDMA Configuration Register) should be initialized to $0740, rather than being left at its default value of $0000.
2. Write RBASE and TBASE in the SPI parameter RAM to point to the Rx BD and Tx BD in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.

NOTE

In the case of multi-master operation, the $\overline{\text{SPISEL}}$ pin should also be enabled to internally connect to the SPI.

3. Configure a parallel I/O pin to operate as the SPI select pin if needed. Supposing PB0 is chosen, write PBODR bit 0 with a zero, PBDIR bit 0 with a one, and PBPAR bit 0 with a zero. Write PBDAT bit 0 with a zero to constantly assert the select pin.
4. Write RBASE and TBASE in the SPI parameter RAM to point to the Rx BD and Tx BD in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.
5. Program the CR to execute the INIT RX & TX PARAMS command for this channel. For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
6. Write RFCR with $18 and TFCR with $18 for normal operation.
7. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 16 bytes, so MRBLR = $0010.
8. Initialize the Rx BD. Assume the Rx data buffer is at $00001000 in main memory. Write $B000 to Rx_BD_Status. Write $0000 to Rx_BD_Length (not required—done for instructional purposes only). Write $00001000 to Rx_BD_Pointer.
9. Initialize the Tx BD. Assume the Tx data buffer is at $00002000 in main memory and contains five 8-bit characters. Write $B800 to Tx_BD_Status. Write $0005 to Tx_BD_Length. Write $00002000 to Tx_BD_Pointer.
10. Write $FF to the SPIE to clear any previous events.
11. Write $37 to the SPIM to enable all possible SPI interrupts.

12. Write $00000020 to the CIMR to allow the SPI to generate a system interrupt. (The CICR should also be initialized.)
13. Write $0370 to SPMODE to enable normal operation (not loopback), master mode, SPI enabled, 8-bit characters, and the fastest speed possible.
14. Write PBDAT bit 0 with zero to assert the SPI select pin.
15. Set the STR bit in the SPCOM to start the transfer.

NOTE

After 5 bytes have been transmitted, the Tx BD is closed. Additionally, the receive buffer is closed after 5 bytes have been received because the L-bit of the Tx BD was set.

7.12.7 SPI Slave Example

The following list is an initialization sequence for use of the SPI as a slave. It is very similar to the SPI master example except that the $\overline{\text{SPISEL}}$ pin is used, rather than a general-purpose I/O pin.

1. The SDCR (SDMA Configuration Register) should be initialized to $0740, rather than being left at its default value of $0000.
2. Configure the port B pins to enable the SPIMOSI, SPIMISO, $\overline{\text{SPISEL}}$, and SPICLK pins. Write PBPAR bits 0, 1, 2, and 3 with ones. Write PBDIR bits 0, 1, 2, and 3 with ones. Write PBODR bits 0, 1, 2, and 3 with zeros.
3. Write RBASE and TBASE in the SPI parameter RAM to point to the Rx BD and Tx BD in the dual-port RAM. Assuming one Rx BD at the beginning of dual-port RAM and one Tx BD following that Rx BD, write RBASE with $0000 and TBASE with $0008.
4. Program the CR to execute the INIT RX & TX PARAMS command for this channel. For instance, to execute this command for SCC1, write $0001 to the CR. This command causes the RBPTR and TBPTR parameters of the serial channel to be updated with the new values just programmed into RBASE and TBASE.
5. Write RFCR with $18 and TFCR with $18 for normal operation.
6. Write MRBLR with the maximum number of bytes per receive buffer. For this case, assume 16 bytes, so MRBLR = $0010.
7. Initialize the Rx BD. Assume the Rx data buffer is at $00001000 in main memory. Write $B000 to Rx_BD_Status. Write $0000 to Rx_BD_Length (not required—done for instructional purposes only). Write $00001000 to Rx_BD_Pointer.
8. Initialize the Tx BD. Assume the Tx data buffer is at $00002000 in main memory and contains five 8-bit characters. Write $B800 to Tx_BD_Status. Write $0005 to Tx_BD_Length. Write $00002000 to Tx_BD_Pointer.
9. Write $FF to the SPIE to clear any previous events.
10. Write $37 to the SPIM to enable all possible SPI interrupts.
11. Write $00000020 to the CIMR to allow the SPI to generate a system interrupt. (The

CICR should also be initialized.)

12. Write $0170 to SPMODE to enable normal operation (not loopback), master mode, SPI enabled, and 8-bit characters. The SPI baud rate generator speed is ignored because the SPI is in slave mode.
13. Set the STR bit in the SPCOM to enable the SPI to be ready once the master begins the transfer.

NOTE

If the master transmits 3 bytes and negates the $\overline{\text{SPISEL}}$ pin, the Rx BD will be closed, but the Tx BD will remain open. If the master transmits 5 or more bytes, the Tx BD will be closed after the 5th byte. If the master transmits 16 bytes and negates the $\overline{\text{SPISEL}}$ pin, the Rx BD will be closed with no errors, and no out-of-buffers error will occur. If the master transmits more than 16 bytes, the Rx BD will be closed (completely full), and the out-of-buffers error will occur after the 17th byte is received.

7.12.8 SPI Interrupt Handling

The following list describes what would normally occur within an interrupt handler for the SPI.

1. Once an interrupt occurs, the SPIE should be read by the user to see which sources have caused interrupts. The SPIE bits would normally be cleared at this time.
2. Process the Tx BD to reuse it and the Rx BD to extract the data from it. To transmit another buffer, simply set the Tx BD R-bit, the Rx BD E-bit, and the STR bit in SPCOM.
3. Clear the SPI bit in the CISR.
4. Execute the RTE instruction.

7.13 PARALLEL INTERFACE PORT (PIP)

The PIP is a function of the CPM that allows data to be transferred to and from the QUICC over 8 or 16 parallel data pins. The pins of the PIP are multiplexed with the 18-bit port B parallel I/O port. The PIP supports the Centronics interface and a fast parallel connection between QUICCs. When the PIP is used, the SMC2 channel is not available.

7.13.1 PIP Key Features

The PIP contains the following key features:

- 18 General-Purpose I/O Pins
- Three Handshake Modes
- Programmable Handshake Timing Attributes
- Supports Centronics and Receiver/Transmitter Interface
- Allows Bidirectional Centronics (P1284) Operation To Be Implemented
- Supports Fast Connection Between QUICCs
- Can Be Controlled by the CPU32+ Core or by the CPM RISC

7.13.2 PIP Overview

The PIP is shown in Figure 7-84. The PIP may be operated as an 18-bit general-purpose I/O port or in one of three handshake modes:

- 8- or 16-Bit Strobed I/O Port with Two Interlocked Handshake Signals
- 8- or 16-Bit Strobed I/O Port with Two Pulsed Handshake Signals
- 8- or 16-Bit Transparent I/O Port with No Handshake Signals

When in one of the handshake modes, the PIP is controlled either by the RISC controller or the CPU32+ core. When the PIP is under RISC control, data is prepared by the CPU32+ core (or other host processor) using the same general BD structures as are used for the SCCs. Thus, the PIP can transfer or receive blocks of characters without interrupting the host processor. The data block may span several linked buffers; therefore, an entire block may be received or transmitted without intervention from the CPU32+ core. When the PIP is under CPU32+ core control (or the control of an external processor), the PIP is controlled directly by the core one byte/word at a time.

When the interlocked or pulsed handshake modes are used, the PIP offers programmable timing attributes such as setup time, pulse width, etc. The interlocked handshake mode supports level-sensitive handshake control signals. The pulsed handshake mode supports edge-sensitive handshakes like those used for the Centronics interface.

The PIP mode of operation may be configured independently for two groups of port B pins: PB7–PB0 and PB17–PB8. This configuration allows an 8-bit PIP data port to be defined, rather than a full 16-bit data port.

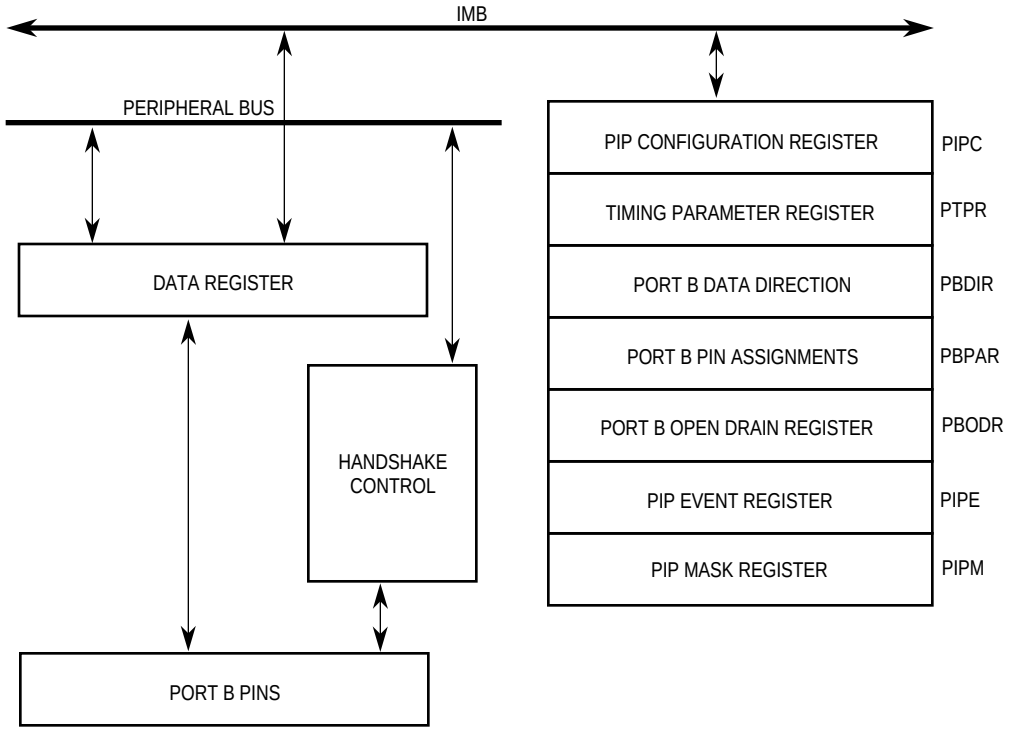


Figure 7-84. PIP Block Diagram

The PIP shares several registers with the SMC2 serial channel. SMC2 is not available and should not be enabled if the PIP is used. If SMC2 is enabled, erratic behavior will occur.

7.13.3 General-Purpose I/O Pins (Port B)

In this configuration, the PIP is not used, but rather operates as general-purpose parallel I/O port B. See 7.14.7 Port B Registers for more details.

7.13.4 Interlocked Data Transfers

In the interlocked handshake mode, the PIP may be configured as a transmitter or a receiver. This configuration allows a fast connection between QUICCs, and may be used for the P1284-protocol advanced byte transfer mode.

The interlocked handshake mode may be controlled by the RISC or the CPU32+ core. Operation using the RISC requires BDs and parameter RAM initialization very similar to the other serial channels. Data is then stored in the buffers using one of the SDMA channels (one of the available channels from SMC2). Operation by the CPU32+ core is performed by software-controlled reads and writes from/to the PIP data register upon interrupt request.

NOTE

At the time of writing, RISC operation of the PIP has not been fully defined. The user should use the CPU32+ core operation mode, until such time as RISC microcode becomes available or the full PIP microcode is available in the RISC internal ROM. Please contact the local Motorola sales representative to obtain the current status of the PIP RISC microcode. In the following description, the RISC reads and writes of the data register are replaced by CPU32+ core reads and writes.

When configured as a transmitter, the STBO pin (PB16) is used as a strobe output ($\overline{STB}$) handshake control signal, and the STBI pin (PB17) is used as an acknowledge ($\overline{ACK}$) input. When configured as a receiver, the PIP generates the $\overline{ACK}$ signal on the STBO pin and inputs the $\overline{STB}$ signal on the STBI pin.

Bits PB16 and PB17 in the port B data direction register (PBDIR) and the port B data register (PBDAT) corresponding to STBO and STBI are not valid and are ignored by the PIP in the interlocked handshake mode.

When the PIP is in this mode and is configured as a transmitter, the RISC controller loads data into the output latch when it receives a request to begin transfers from the host processor (see Figure 7-85). Once data is loaded, after a programmable setup time, the $\overline{STB}$ signal is asserted (low). Then when $\overline{ACK}$ is sampled as low, the data is transmitted, followed by the $\overline{STB}$ being negated (high). $\overline{STB}$ remains high until new data is loaded into the output latch and $\overline{ACK}$ is negated (high).

When the PIP is configured as a receiver, input data is latched when the $\overline{STB}$ signal is sampled as low. The $\overline{ACK}$ signal is then asserted. $\overline{ACK}$ will be negated (high) when the data has been removed from the input latch.

Thus, to connect to QUICCs using this interface, connect the STBO pin of each QUICC to the STBI pin of the other and connect the desired data pins (either PB8–PB15 or PB0–PB15 are connected between QUICCs).

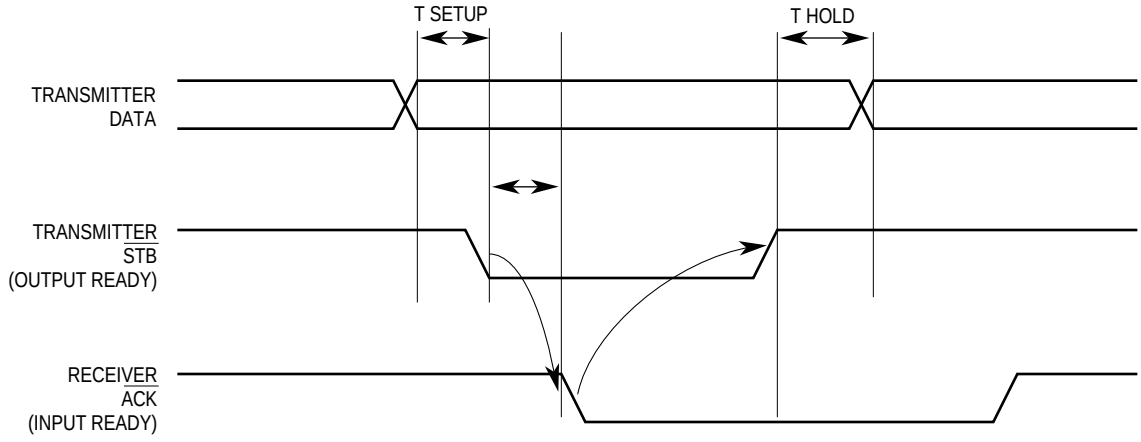


Figure 7-85. Interlock Handshake Mode

7.13.5 Pulsed Data Transfers

In the pulsed handshake mode, the PIP may be configured as a transmitter or a receiver. This configuration allows a Centronics-compatible interface to be implemented.

The pulsed handshake mode may be controlled by the RISC or the CPU32+ core. Operation using the RISC requires BDs and parameter RAM initialization very similar to the other serial channels. Data is then stored in the buffers using one of the SDMA channels (one of the available channels from SMC2). Operation by the CPU32+ core is performed by software-controlled reads and writes from/to the PIP data register upon interrupt request.

NOTE

At the time of writing, RISC operation of the PIP has not been fully defined. The user should use the CPU32+ core operation mode until as RISC microcode becomes available or the full PIP microcode is available in the RISC internal ROM. Please contact the local Motorola sales representative to obtain the current status of the PIP RISC microcode. In the following description, the RISC reads and writes of the data register are replaced by CPU32+ core reads and writes.

When configured as a transmitter, the STBO pin (PB16) is used as a strobe output ($\overline{STB}$) handshake control signal, and the STBI pin (PB17) is used as an acknowledge ($\overline{ACK}$) input. When configured as a receiver, the PIP generates the $\overline{ACK}$ signal on the STBO pin and inputs the $\overline{STB}$ signal on the STBI pin.

Bits PB16 and PB17 in the port B data direction register (PBDIR) and the port B data register (PBDAT) corresponding to STBO and STBI are not valid and are ignored by the PIP when the pulsed handshake mode is selected.

When configured as a transmitter, the PIP generates the $\overline{STB}$ signal when data is ready in the PIP's output latch and the previous transfer has been acknowledged (see Figure 7-86). The setup time and the strobe pulse width are user programmable. When configured as a receiver, the PIP uses the $\overline{STB}$ signal to latch the input data and acknowledges the transfer with the $\overline{ACK}$ signal. The timing of the $\overline{ACK}$ signal is user programmable.

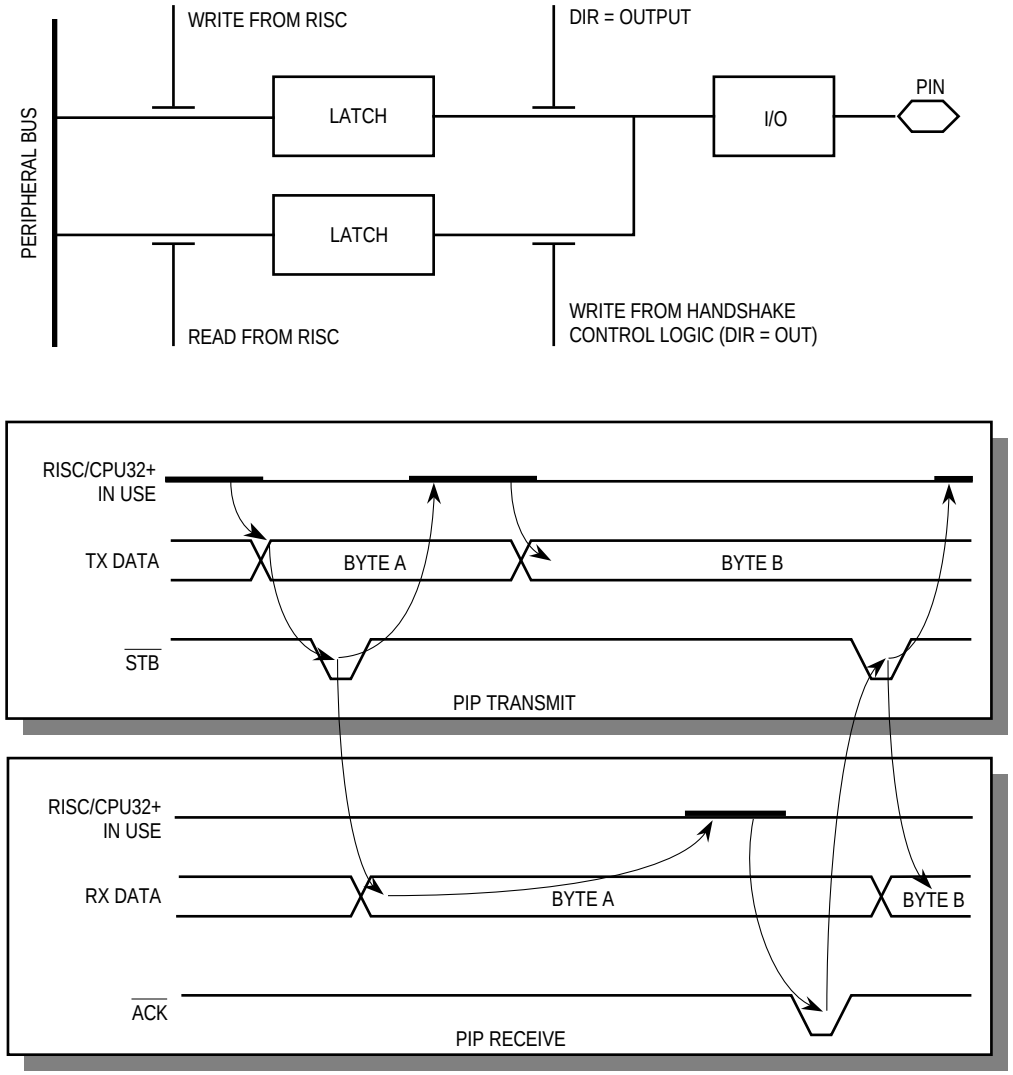


Figure 7-86. Pulsed Handshake Full Cycle

7.13.5.1 BUSY SIGNAL. In the pulsed handshake mode, the PIP receiver can generate an additional BUSY handshake signal, which is useful to implement the Centronics reception interface (see Figure 7-87). The BUSY signal is an output indication of a transfer in service. It is asserted by the Centronics receiver as soon as the data is latched into the PIP data register. The timing of BUSY negation in relation to the $\overline{ACK}$ signal is user programmable. Two bits in the PIP configuration register enable the assertion and negation of the BUSY signal via the host processor software.

The BUSY signal is multiplexed onto PB0; therefore, it is not possible to use the BUSY signal with a full 16-bit PIP interface. BUSY can be used with the standard 8-bit PIP interface to implement Centronics functions.

When in the pulsed handshake mode, the PIP transmitter may be configured to ignore the BUSY signal or to suspend the assertion of the $\overline{\text{STB}}$ output until the receiver's BUSY signal is negated.

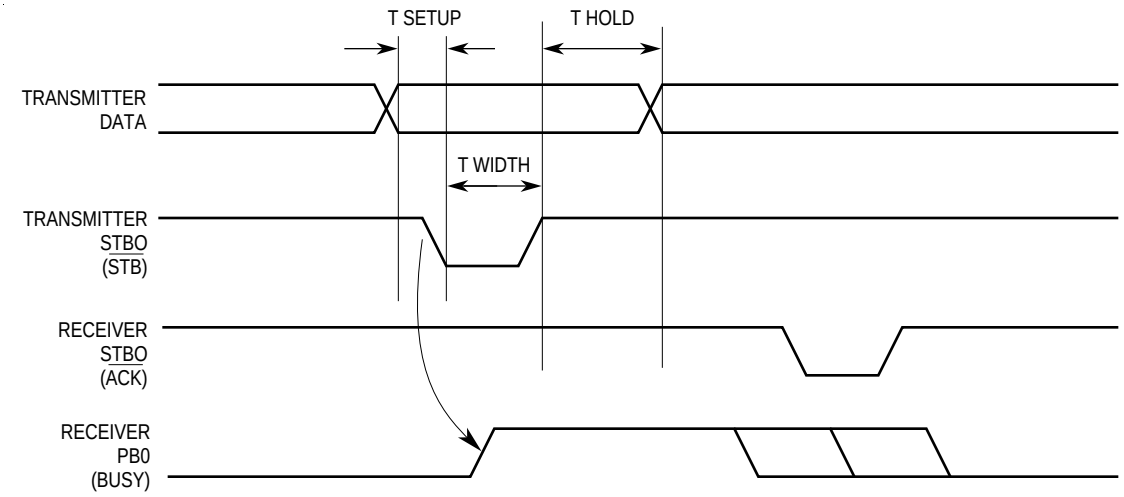


Figure 7-87. Pulsed Handshake Busy Signal

7.13.5.2 PULSED HANDSHAKE TIMING. The pulsed handshake mode transmitter timing is shown in Figure 7-88. In the pulsed handshake mode, four Centronics receive timing options select the relative timing of the BUSY signal to the $\overline{\text{ACK}}$ signal.

The timing parameters for the pulsed handshake mode are governed by two user-programmable timing parameters, TPAR1 and TPAR2. Each parameter defines an interval from 1 to 256 system clocks. Figure 7-89 through Figure 7-92 show the definition of TPAR1 and TPAR2 in the four receive modes.

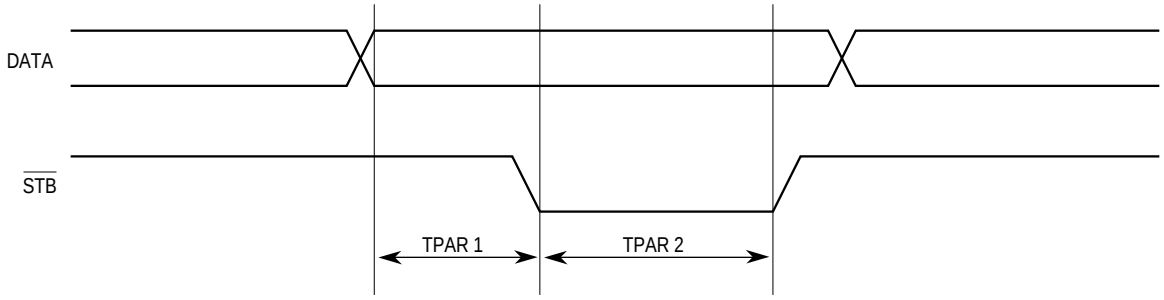


Figure 7-88. Centronics Transmitter Timing

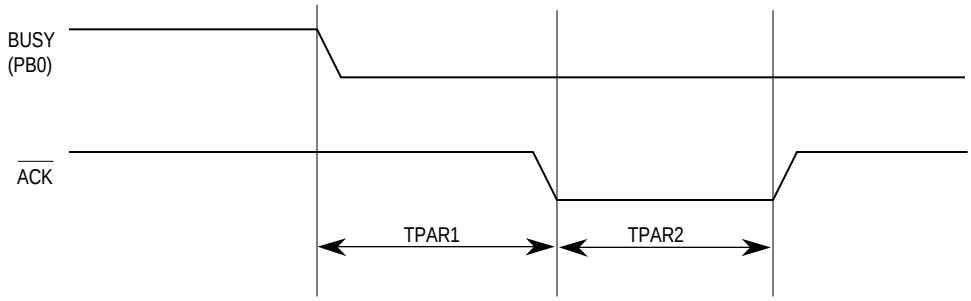


Figure 7-89. Centronics Receiver Timing Mode 0

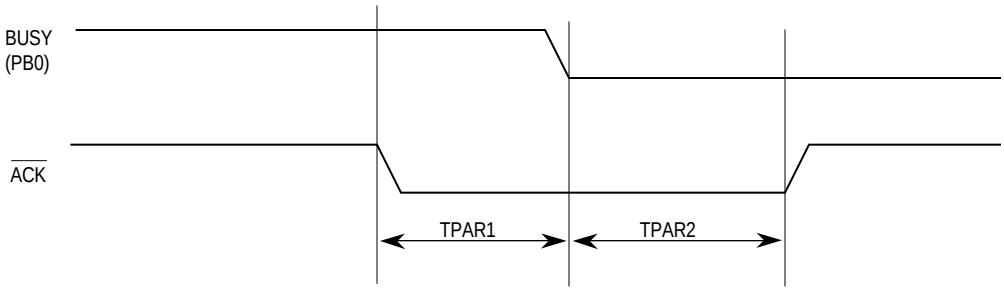


Figure 7-90. Centronics Receiver Timing Mode 1

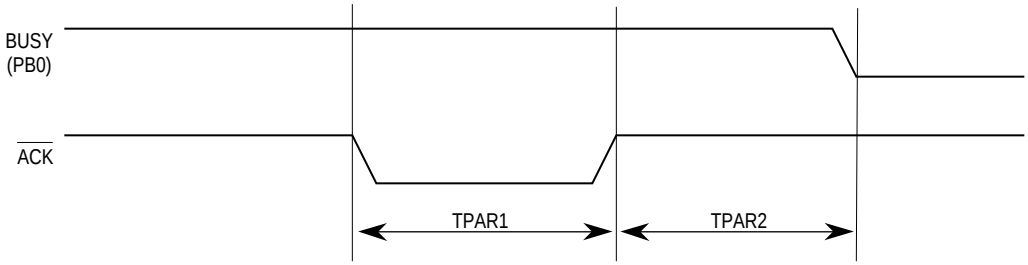


Figure 7-91. Centronics Receiver Timing Mode 2

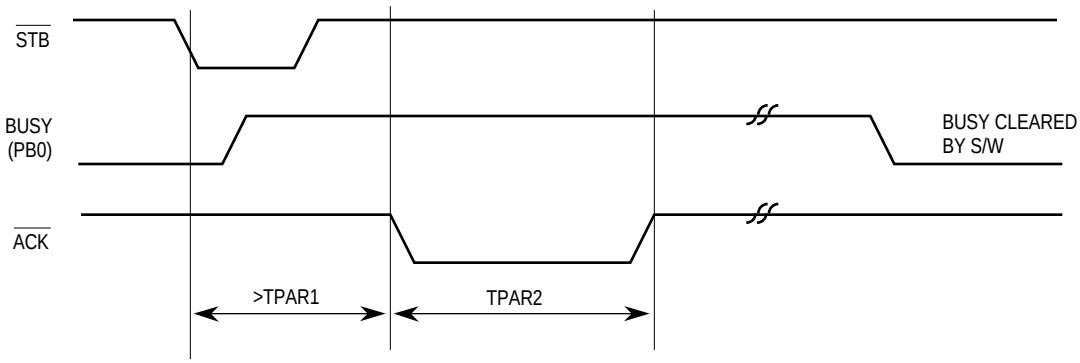


Figure 7-92. Centronics Receiver Timing Mode 3

7.13.6 Transparent Data Transfers

In the transparent handshake mode, the PIP may be configured as a transmitter or a receiver. This configuration has only one handshake pin.

The transparent mode is controlled only by the RISC. Operation using the RISC requires BDs and parameter RAM initialization very similar to the other serial channels. Data is then stored in the buffers using one of the SDMA channels (one of the available channels from SMC2).

NOTE

At the time of writing, this operation of the PIP has not been fully defined. This PIP operation may be implemented by the CPU32+ core, using the port B parallel I/O registers and any port C interrupt pin.

In this mode, the B17 pin falling edge generates the request to the RISC, which causes the RISC to receive/transmit data. The direction of the pins is controlled by the port B data direction register (PBDIR). The transparent handshake mode is shown in Figure 7-93.

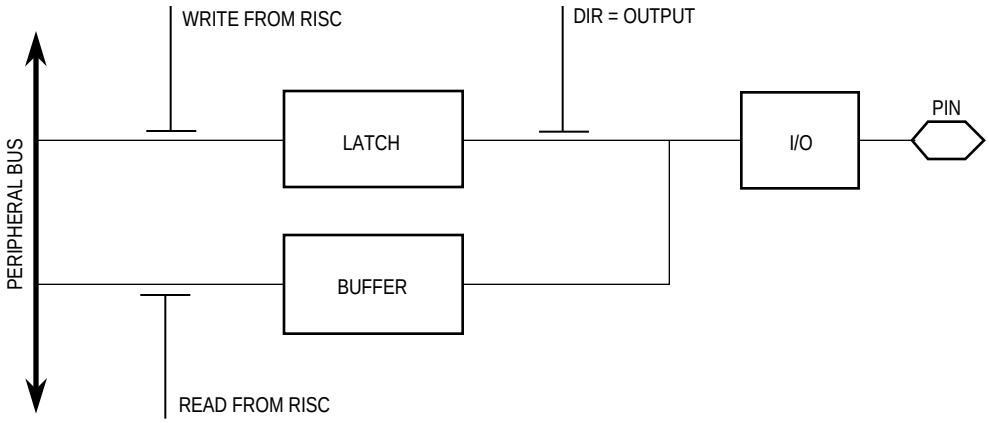


Figure 7-93. PIP Transparent Handshake Mode

7.13.7 Programming Model

The following paragraphs describe the PIP registers and parameter RAM.

7.13.7.1 PARAMETER RAM. At the time of writing, RISC operation on the PIP has not been fully defined. The user should use the CPU32+ core operation mode until the RISC microcode becomes available or the full PIP microcode becomes available in the RISC internal ROM. Please contact the local Motorola sales representative to obtain the current status of the PIP RISC microcode.

7.13.7.2 PIP CONFIGURATION REGISTER (PIPC). The PIPC is a 16-bit read-write register that is cleared at reset. The PIPC determines all PIP options.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STR	—			SACK	CBSY	SBSY	EBSY	TMOD		MODL		MODH		HSC	T/R

STR —Start (Valid for a Transmitter Only)

This bit is valid only when the T/R bit is set to 1 (transmitter). Setting this bit to 1 causes the RISC controller to poll the Tx BD. Thus, the user should prepare a Tx BD and set its R- bit before setting STR. STR is cleared after one system clock.

Bits 14–12—Reserved

SACK—Set Acknowledge

When set, this bit will assert the receiver's $\overline{\text{ACK}}$ output (low voltage), regardless of the receiver's state. SACK should be used when implementing the IEEE P1284 Bidirectional Centronics protocol.

CBSY—Clear BUSY

This bit is used by host software to force the BUSY signal low for a Centronics receiver. When CBSY is set, the BUSY signal will output at 0 (low voltage). CBSY is cleared after the PIP negates the BUSY signal.

NOTE

The T/R bit should be set to 0 (receiver) if CBSY is used.

SBSY—Set BUSY

This bit is used by host software to force the BUSY signal high for a Centronics receiver. When SBSY is set, the BUSY signal will output a 1 (high voltage). SBSY is cleared after the PIP asserts the BUSY signal.

NOTE

The T/R bit should be set to 0 (receiver) if SBSY is used. Also, EBSY would normally be set by the user before SBSY is set. (If EBSY is cleared, the PIP ignores the $\overline{\text{STB}}$ signal until CBSY is set in software.)

EBSY—Enable BUSY (Receiver)

This bit has a different definition depending on whether T/R is set to receiver or transmitter.

When T/R = 0 (PIP is a receiver), the definition is as follows:

0 = Disable BUSY signal generation on PB0 for the receiver.
1 = Enable the BUSY output signal on PB0. EBSY will only take effect if bit 0 of PBPAR is 0 to configure this pin to belong to the PIP and bit 0 of PBDIR is 1 to make this pin an output.

When T/R = 1 (PIP is a transmitter), the definition is as follows:

0 = Ignore the BUSY signal input on PB0 for the transmitter.
1 = Assertion of $\overline{\text{STB}}$ is conditioned by BUSY is negation. $\overline{\text{STB}}$ will not be asserted until the BUSY signal, input on PB0, is negated. EBSY will only take effect if bit 0 of PBPAR is 0 to configure this pin to belong to the PIP and bit 0 of PBDIR is 0 to make this pin an input.

NOTE

The programming of MODL has no effect on the BUSY pin if EBSY is set.

TMOD—Timing Mode (Centronics Receiver)

These bits are only valid when T/R is set to receive and MODH is set to pulsed handshake mode. Otherwise they are ignored.

00 = Centronics receiver timing mode 0 (BUSY is negated before $\overline{\text{ACK}}$ is asserted).
01 = Centronics receiver timing mode 1 (BUSY is negated after $\overline{\text{ACK}}$ assertion but before $\overline{\text{ACK}}$ negation).
10 = Centronics receiver timing mode 2 (BUSY is negated after $\overline{\text{ACK}}$ negation).
11 = Centronics receiver timing mode 3 (BUSY is negated by host software).

MODL—Mode Low

These bits determine the mode of the PIP's lower 8 pins (PB7–PB0).

00 = Port B general-purpose I/O mode (under host control).
01 = Transparent handshake mode (under RISC or host control).
1x = Mode of operation is controlled by MODH.

NOTE

The BUSY pin (PB0) is not affected by MODL programming if EBSY is set.

MODH—Mode High

These bits determine the mode of the PIP's upper 10 pins (PB17–PB8). MODH may be changed when the RISC processor is not currently receiving or transmitting data.

00 = Port B general-purpose I/O (under host control).
01 = Transparent handshake mode (under RISC or host control).
10 = Interlocked handshake mode (under RISC or host control).
11 = Pulsed handshake mode (under RISC or host control).

HSC—Host Control

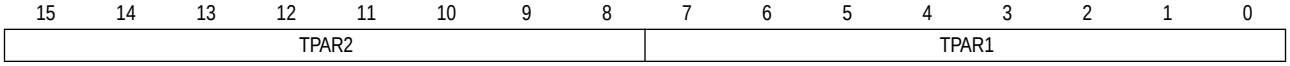
0 = The PIP data transfers are controlled by the RISC in the CPM, using the PIP parameter RAM, BDs, and SDMA channels.
1 = The PIP data transfers are controlled by the host software (i.e., CPU32+ or other external processor in the system).

T/R—Transmit/Receive Select

This bit selects transmitter or receiver operation for the PIP when it is using the interlocked, pulsed, or transparent handshake modes.

0 = Data is input to the PIP.
1 = Data is output from the PIP.

7.13.7.3 PIP TIMING PARAMETERS REGISTER (PTPR). The PTPR is a 16-bit read-write register that is cleared at reset. The PTPR holds two timing parameters, TPAR1 and TPAR2, which are used in the pulsed handshake modes for both a PIP transmitter and a receiver.



TPAR1—Timing Parameter 1

This 8-bit value defines the number of system clocks for TPAR1 in the transmitter or receiver pulsed handshake mode. The value $00 corresponds to 1 QUICC general system clock, and the value $FF corresponds to 256 QUICC general system clocks. A general system clock defaults to 40 ns, assuming a 25-MHz QUICC system.

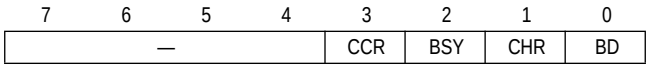
TPAR2—Timing Parameter 2

This 8-bit value defines the number of system clocks for TPAR2 in the transmitter or receiver pulsed handshake mode. The value $00 corresponds to 1 QUICC general system clock, and the value $FF corresponds to 256 QUICC general system clocks. A general system clock defaults to 40 ns, assuming a 25-MHz QUICC system.

7.13.7.4 PIP BUFFER DESCRIPTORS. BDs for the receiver and transmitter that support PIP operation were still in preparation at the time of writing.

7.13.7.5 PIP EVENT REGISTER (PIPE). The PIPE is an 8-bit register used to report events recognized by the PIP and to generate interrupts. It shares the same address as the SMC2 event register; thus, SMC2 cannot be used simultaneously with the PIP. Upon recognition of an event, the PIP sets its corresponding bit in the PIPE. Interrupts generated by this register may be masked in the PIP mask register.

The PIPE is a memory-mapped register that 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.



Bits 7–4—Reserved

CCR—Control Character Received

A control character was received (with reject (R) = 1) and stored in the receive control character register.

BSY—Busy condition

A data byte/word was not received/transmitted by the PIP due to lack of buffers.

CHR—Character Received/Transmitted

A data character was transmitted or received. This event may be used to generate interrupts to the CPU32+ core if the PIP was programmed to be controlled by host software.

BD—Rx/Tx Buffer

A complete buffer has been received/transmitted on the PIP channel. The channel closes the receive buffer due to one of the following events:

- Reception of a user-defined control character (and reject (R) = 0 for that character in the Centronics control characters table).
- Data length termination (the receive buffer was filled, or the transmit buffer has finished transmitting).
- Reception of a programmable silence period.

7.13.7.6 PIP MASK REGISTER (PIPM). The PIPM is an 8-bit read-write register. It shares the same address as the SMC2 mask register; thus, SMC2 cannot be used simultaneously with the PIP. Each bit in the PIPM corresponds a bit in the PIPE. If a bit in the PIPM is a one, the corresponding interrupt in the PIPE will be enabled. If the bit is a zero, the corresponding interrupt in the PIPE will be masked. This register is cleared at reset.

7.13.8 Centronics Controller Overview

Centronics is a parallel peripheral interface bus that is generally used as a communication channel between a host computer and printing equipment. The interface uses an 8 bit data bus, handshake signals that control the data exchange, and some status lines that reflect the peripheral device status. Traditionally, the direction of data transfer is from the host computer to the peripheral device. New standards, such as IEEE P1284, allow reverse channel operation (i.e data can be transferred from the peripheral to the host.)

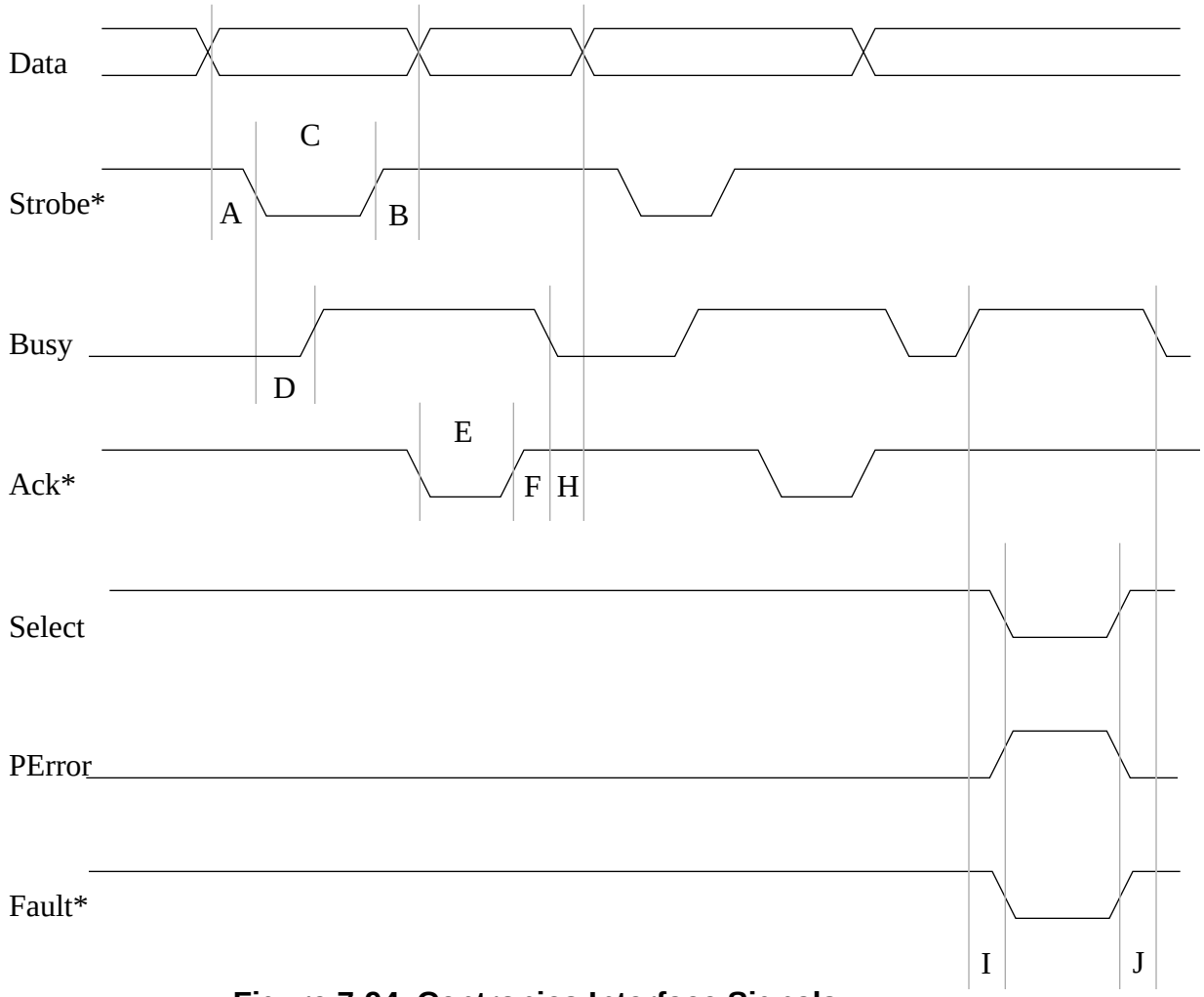
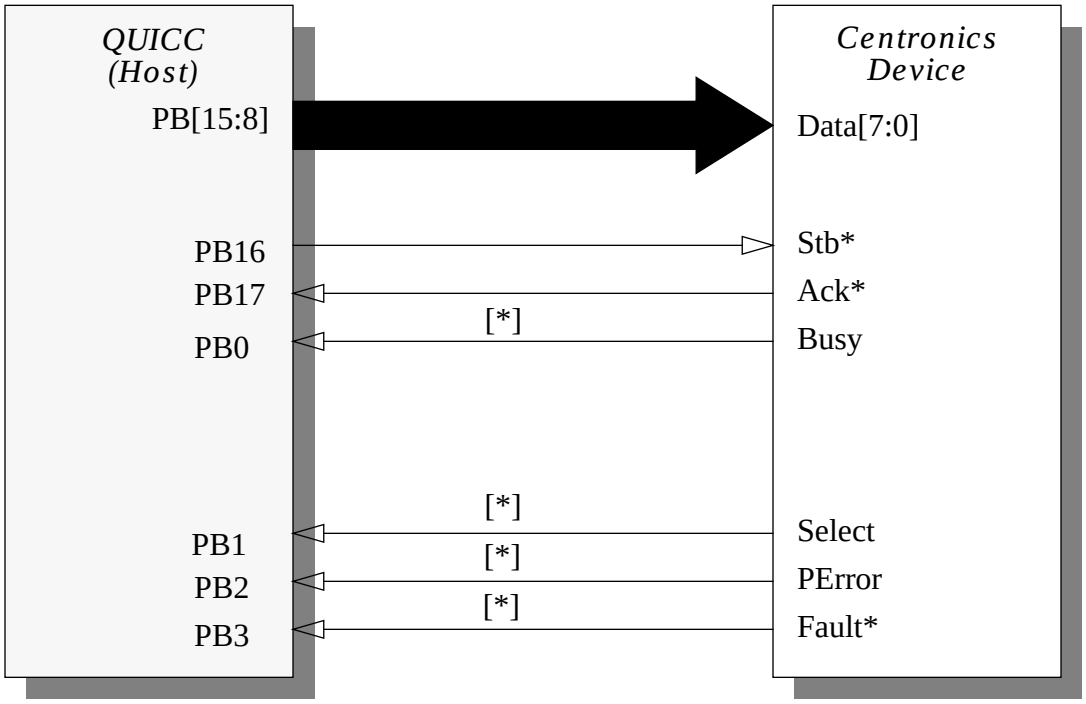


Figure 7-94. Centronics Interface Signals

The Centronics controller can be operated as a host port (transmitter), peripheral port (receiver), or can support bidirectional data transfer using some s/w support and a combination of the two modes.



[*] - optional **Figure 7-95. Centronics Transmitter Configuration**

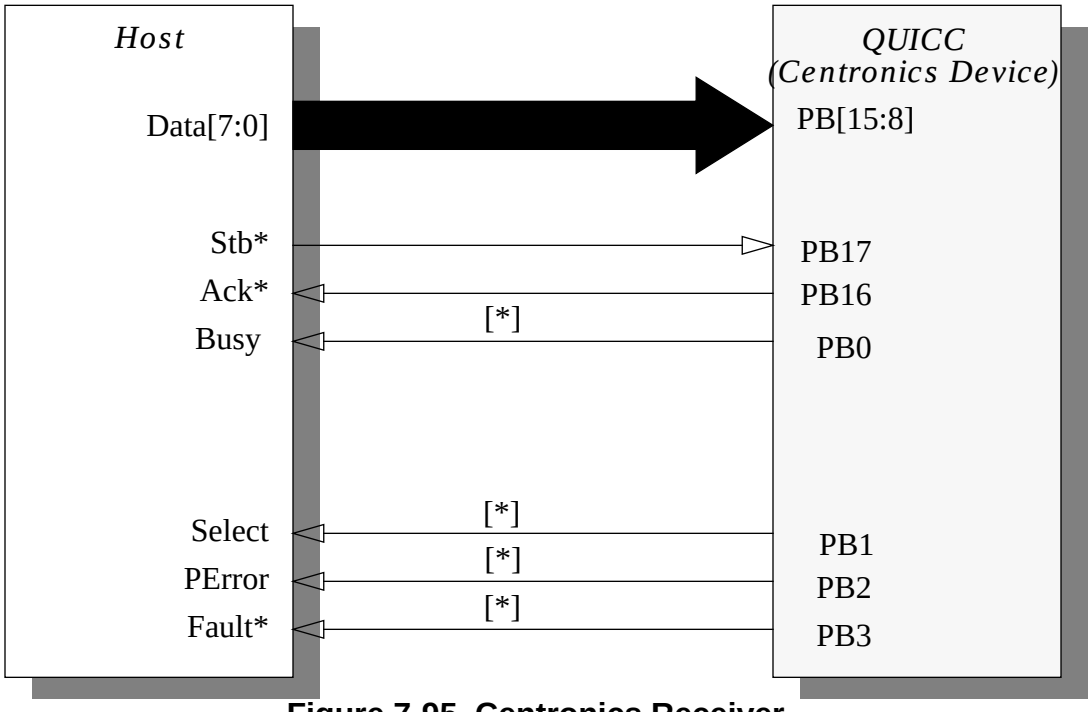


Figure 7-95. Centronics Receiver

7.13.8.1 CENTRONICS CONTROLLER KEY FEATURES. Super-set of the Centronics standard

- 8-bit or 16-Bit Data Transfer

- Supports Closed Loop Handshake for Higher Data Transfer Rates
- Supports Centronics Transmitter and Receiver Operating Modes
- Supports Bidirectional Centronics (P1284)
- Flexible Message-Oriented Data Structure
- Flexible Control Character Comparison (Receiver)
- Flexible Timing Modes
- Programmable timing parameters

7.13.8.2 CENTRONICS CHANNEL TRANSMISSION. The Centronics transmitter supports the same general data structure that is used by the SCCs for other protocols. When the STR bit in the PIP configuration register is set, the Centronics controller will process the next buffer descriptor (BD) in the Centronics transmitter BD table. If the BD is ready, the Centronics transmitter will fetch the data from the memory and start sending it to the printer. If the status mask bits are set in the SMASK register, the printer status line (Select, PError and $\overline{\text{Fault}}$) will be checked before each transfer. In this case, the user should configure PB1,2,3 pins as general purpose inputs and connect them to Select, PError, and Fault respectively.

For each transfer the Centronics controller will output the data on the Centronics interface data lines and will generate the strobe pulse if previous data was acknowledged and the minimum setup time was met. The strobe pulse width and the setup time parameters are programmed by the PIP Timing Parameter Register (PTPR). A single data frame may span several BDs. A maskable interrupt can be generated after the processing of each BD.

7.13.8.3 CENTRONICS TRANSMITTER MEMORY MAP. When configured to operate in Centronics Transmit mode, the QUICC overlays the structure illustrated in Table 7-17 with the SMC2 parameter RAM area.

Table 7-17. Centronics Transmitter Parameter RAM

Address	Name	Width	Description
PIP Base+00	Res	Word	Reserved
PIP Base+02	TBASE	Word	Tx Buffer Descriptors Base Address
PIP Base+04	CFCR	Byte	Centronics Function Code
PIP Base+05	SMASK	Byte	Status Mask
PIP Base+06	Res	Word	Reserved
PIP Base+08	Res	Long	Reserved
PIP Base+0C	Res	Long	Reserved
PIP Base+10	Res	Word	Reserved
PIP Base+12	Res	Word	Reserved
PIP Base+14	Res	Long	Reserved
PIP Base+18	TSTATE	Long	Tx Internal State
PIP Base+1C	T_PTR	Long	Tx Internal Data Pointer
PIP Base+20	TBPTR	Word	Tx Buffer Descriptor Pointer
PIP Base+22	T_CNT	Word	Tx Internal Byte Count
PIP Base+24	TTEMP	Long	Tx Temp

Certain parameter RAM values above (marked in **bold face**) need to be initialized by the user before the PIP is enabled; the others are initialized/written by the CP. Once initialized, most parameter RAM values will not need to be accessed in user software since most of the activity is centered around the transmit buffer descriptors, not the parameter RAM.

7.13.8.4 BUFFER DESCRIPTOR TABLE POINTER (TBASE). The TBASE entry defines the starting location in the dual-port RAM for the PIP transmitter's set of buffer descriptors. This provides a great deal of flexibility in how BDs are partitioned. By programming the TBASE entry and by setting the "wrap" bit in the last BD , the user may select how many BDs to allocate for the transmit function. The user must initialize TBASE before enabling the channel.

NOTE

TBASE should contain a value that is divisible by 8.

7.13.8.5 STATUS MASK REGISTER (SMASK). The status mask register controls which of the printer status lines will be checked before each transfer..

7	6	5	4	3	2	1	0
0	0	0	0	F	PE	S	0

S—Select Error

0 = The Select status will be ignored
1 = The Select status line will be checked during transmission

PE—Printer Error

0 = The PError status will be ignored
1 = The PError status line will be checked during transmission

F—Fault

0 = The Fault status will be ignored
1 = The Fault status line will be checked during transmission

7.13.8.6 CENTRONICS FUNCTION CODE REGISTER (CFCR). The FC entry contains the value that the user would like to appear on the function code pins (FC3-0) when the associated SDMA channel accesses memory. It also controls the byte ordering convention to be used in the transfers.

7	6	5	4	3	2	1	0
RES	RES	RES	MOT				

FC3-0—Function Code 3-0

These bits contain the function code value used during this SDMA channel's memory accesses. It is suggested that the user write bit FC3 with a one to identify this SDMA channel

access as a DMA-type access. Example: FC3-FC0 = 1000 (binary). Do not write the value 0111 (binary) to these bits.

MOT—Motorola

This bit should be set by the user to achieve normal operation.

0 = DEC (and Intel) convention is used for byte ordering. Swapped operation. Also called little-endian byte ordering. The bytes stored in each buffer word are reversed as compared to the Motorola mode.

1 = Motorola byte ordering. Normal operation. Also called big-endian byte ordering. As data is received from the serial line and put into the buffer, the most significant byte of the buffer word contains data received earlier than the least significant byte of the same buffer word.

Res—Reserved. Should be set to zero by the user.

7.13.8.7 TRANSMITTER BUFFER DESCRIPTOR POINTER (TBPTR). The transmitter buffer descriptor pointer (TBPTR) points to the next BD that the transmitter will transfer data from when it is in IDLE state, or to the current BD during frame transmission. After a reset or when the end of BD table is reached, the CP initializes this pointer to the value programmed in the TBASE entry. Although TBPTR need never be written by the user in most applications, it may be modified by the user when the transmitter is disabled, or when the user is sure that no transmit buffer is currently in use (i.e. after the STOP TRANSMIT command is issued.)

7.13.8.8 CENTRONICS TRANSMITTER PROGRAMMING MODEL. The host configures the PIP to operate as a Centronics controller by programming the PIP Configuration register (PIPC). Timing attributes (minimum data setup time and strobe pulse width) are set by programming the PIP Timing Parameters register (PTPR). The transmit errors are reported through the Tx BD.

7.13.8.9 CENTRONICS TRANSMITTER COMMAND SET. The Centronics transmitter uses SMC2 transmit commands (i.e, same opcodes and channel number)

7.13.8.9.1 STOP TRANSMIT Command. The channel STOP TRANSMIT command disables the transmission of frames on the transmit channel. If this command is received by the Centronics controller during frame transmission, transmission of that buffer is aborted and the TBPTR is not advanced to the next BD. No new BD is accessed, and no new buffers are transmitted for this channel. The transmitter will idle until the RESTART TRANSMIT command is given.

7.13.8.9.2 RESTART TRANSMIT Command. The RESTART TRANSMIT command is used to begin or resume transmission from the current Tx BD Pointer (TBPTR) in the channel's Tx BD table. When this command is received by the channel following by the STR bit in the PIP Configuration register (PIPC) being set, it will start processing the current BD. This command is expected by the Centronics controller after a STOP TRANSMIT command, after the disabling of the channel in its mode register, or after a transmitter error occurs.

7.13.8.9.3 INIT TX PARAMETERS Command. Initializes all the transmit parameters in this Centronics channel's parameter RAM to their reset state. This command should only be issued when the transmitter is disabled.

7.13.8.10 TRANSMISSION ERRORS.

7.13.8.10.1 Buffer Descriptor Not Ready. This error occurs if the centronics transmitter is active (STR bit in the PIP mode register was asserted by the host) and the current BD that should be processed by the Centronics controller is not ready (R bit in the BD = 0). When this condition occurs, the TXE (transmit error) interrupt will be set. The channel will resume transmission after the s/w prepares the BD and asserts the STR bit.

7.13.8.10.2 Printer Off-Line Error . This error occurs if the printer is off-line (Select line is negated) and if the printer status check option is enabled. The S bit will be set in the BD and TX Error (TXE) interrupt will be set. The channel will resume transmission after Restart Transmit command.

7.13.8.10.3 Printer Fault. This error occurs if the printer has a Fault condition (Fault* line asserted) and if the printer status check option is enabled. The F bit will be set in the BD and TX Error (TXE) interrupt will be set. The channel will resume transmission after Restart Transmit command.

7.13.8.10.4 Paper Error. This error occurs if the printer has an error in its paper path (PError line asserted) and if the printer status check option is enabled. The PE bit will be set in the BD and TX Error (TXE) interrupt will be set. The channel will resume transmission after Restart Transmit command

7.13.8.10.5 Centronics Transmitter Buffer Descriptor. The CP confirms transmission (or indicates error conditions) via the buffer descriptors to inform the processor that the buffers have been serviced.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**R**	—	**W**	**I**	**L**	—	**CM**	—	—	—	—	—	F	PE	S	—
OFFSET + 2	**DATA LENGTH**															
OFFSET + 4	**TX DATA BUFFER POINTER**															
OFFSET + 6																

R—Ready

0 = The data buffer associated with this BD is not currently 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.

W—Wrap (Final BD in Table)

0 = This is not the last buffer descriptor in the Tx BD Table.
1 = This is the last buffer descriptor in the Tx 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 TBASE). The number of Tx BDs in this table is programmable, and is determined only by the wrap bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been serviced.
1 = The TX bit in the PIP event register will be set when this buffer has been serviced by the CP, which can cause an interrupt.

L—Last

0 = This buffer is not the last buffer of the frame.
1 = This buffer is the last buffer of the frame.

CM—Continuous Mode

0 = Normal Operation.
1 = The R-bit is not cleared by the CP after this buffer is closed, allowing the associated data buffer to be retransmitted automatically when the CP next accesses this BD. However, the R bit will be cleared if an error occurs during transmission

F—Fault

0 = The Fault status remained negated during transmission
1 = The Fault status was asserted during transmission

PE—Printer Error

0 = The PError status remained negated during transmission
1 = The PError status was asserted during transmission

S—Select Error

0 = The Select status remained asserted during transmission
1 = The Select status was negated during transmission

7.13.8.11 CENTRONICS TRANSMITTER EVENT REGISTER (PIPE) . When the Centronics Transmitter protocol is selected, the SMC2 event register is called the Centronics Transmitter event register. It is an 8-bit register which is used to report events recognized by the Centronics channel and generate interrupts. On recognition of an event, the Centronics controller will set its corresponding bit in the Centronics event register.

The Centronics event register is a memory-mapped register that 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.

7	6	5	4	3	2	1	0
–	–	–	TXE	–	–	CHR	TX

TXE—Transmit Error

An error condition was detected. This error status is reported in the buffer descriptor.

CHR—Character Transmitted

Acknowledgment that the last character was strobed into the receiver input latch (STB was asserted by the transmitter) and a new character was written to the data register.

TX—Tx Buffer

A buffer has been transmitted over the Centronics channel. This bit is set only after the last character of the buffer was strobed into the receiver input latch (STB was asserted by the transmitter).

7.13.8.12 CENTRONICS CHANNEL RECEPTION. The Centronics receiver supports the same general data structure that is used by the SCCs for other protocols. Upon receiving a character from the Centronics interface, the receiver will check if the current buffer descriptor (BD) in the Centronics receiver BD table is ready for use. If the BD is ready, the Centronics receiver will compare the character against a user defined control character table. If no match was found, the character will be written to the BD's associated buffer. If a match was found, the character will be either written to the receive buffer (upon which the buffer is closed and a new receive buffer taken) or rejected, depending on the R bit in the Control Character Table. If rejected, the character is written to the Received Control Character Register (RCCR) in internal RAM and a maskable interrupt is generated. A maskable interrupt will be generated at the completion of the BD processing. A single received data frame may span several BDs.

For each transfer, the Centronics controller will generate ACK and BUSY handshake signals on the Centronics interface. The ACK pulse width and the timing of BUSY with respect to the ACK signal are determined by the setting in the PIP Timing Parameter Register (PTPR).

7.13.8.13 CENTRONICS RECEIVER MEMORY MAP. When configured to operate in Centronics receive mode, the QUICC overlays the structure illustrated in Table 7-17 with the SMC2 parameter RAM area.

Table 7-18. Centronics Receiver Parameter RAM

Address	Name	Width	Description
PIP Base+00	RBASE	Word	Rx Buffer Descriptors Base Address
PIP Base+02	Res	Word	Reserved
PIP Base+04	CFCR	Byte	Centronics Function Code
PIP Base+05	Res	Byte	Reserved
PIP Base+06	MRBLR	Word	Maximum Receive Buffer Length
PIP Base+08	RSTATE	Long	Rx Internal State
PIP Base+0C	R_PTR	Long	Rx Internal Data Pointer
PIP Base+10	RBPTR	Word	Rx Buffer Descriptor Pointer

Table 7-18. Centronics Receiver Parameter RAM

Address	Name	Width	Description
PIP Base+12	R_CNT	Word	Rx Internal Byte Count
PIP Base+14	RTEMP	Long	Rx Temp
PIP Base+18	Res	Word	Reserved
PIP Base+1C	Res	Word	Reserved
PIP Base+20	Res	Word	Reserved
PIP Base+22	Res	Word	Reserved
PIP Base+24	Res	Long	Reserved
PIP Base+28	MAX_SL	Word	Maximum Silence period
PIP Base+2a	SL_CNT	Word	Silence counter
PIP Base+2c	CHARCTER1	Word	CONTROL character 1
PIP Base+2E	CHARCTER2	Word	CONTROL character 2
PIP Base+30	CHARCTER3	Word	CONTROL character 3
PIP Base+32	CHARCTER4	Word	CONTROL character 4
PIP Base+34	CHARCTER5	Word	CONTROL character 5
PIP Base+36	CHARCTER6	Word	CONTROL character 6
PIP Base+38	CHARCTER7	Word	CONTROL character 7
PIP Base+3A	CHARCTER8	Word	CONTROL character 8
PIP Base+3C	RCCM	Word	Receive Control Character Mask
PIP Base+3E	RCCR	Word	Receive Character Control Register

Certain parameter RAM values above (marked in **bold face**) need to be initialized by the user before the PIP is enabled; the others are initialized/written by the CP. Once initialized, most parameter RAM values will not need to be accessed in user software since most of the activity is centered around the transmit buffer descriptors, not the parameter RAM.

7.13.8.14 BUFFER DESCRIPTOR TABLE POINTER (RBASE). The RBASE entry defines the starting location in the dual-port RAM for the PIP receiver's set of buffer descriptors. This provides a great deal of flexibility in how BDs are partitioned. By programming the RBASE entry and by setting the "wrap" bit in the last BD, the user may select how many BDs to allocate for the receive function. The user must initialize RBASE before enabling the channel.

NOTE

.RBASE should contain a value that is divisible by 8.

7.13.8.15 CENTRONICS FUNCTION CODE REGISTER (CFCR). The FC entry contains the value that the user would like to appear on the function code pins (FC3-0) when the associated SDMA channel accesses memory. It also controls the byte ordering convention to be used in the transfers.

7	6	5	4	3	2	1	0
RES	RES	RES	MOT	FC3–FC0			

FC3-0 —Function Code 3-0

These bits contain the function code value used during this SDMA channel's memory accesses. It is suggested that the user write bit FC3 with a one to identify this SDMA channel access as a DMA-type access. Example: FC3-FC0 = 1000 (binary). Do not write the value 0111 (binary) to these bits.

MOT—Motorola

This bit should be set by the user to achieve normal operation.

0 = DEC (and Intel) convention is used for byte ordering. Swapped operation. Also called little-endian byte ordering. The bytes stored in each buffer word are reversed as compared to the Motorola mode.

1 = Motorola byte ordering. Normal operation. Also called big-endian byte ordering. As data is received from the serial line and put into the buffer, the most significant byte of the buffer word contains data received earlier than the least significant byte of the same buffer word.

Res—Reserved. Should be set to zero by the user.

7.13.8.16 RECEIVER BUFFER DESCRIPTOR POINTER (RBPTR). The receiver buffer descriptor pointer (RBPTR) points to the next BD that the receiver will transfer data to when it is in IDLE state, or to the current BD during frame reception. After a reset or when the end of BD table is reached, the CP initializes this pointer to the value programmed in the RBASE entry. Although RBPTR need never be written by the user in most applications, it may be modified by the user when the receiver is disabled.

7.13.8.17 CENTRONICS RECEIVER PROGRAMMING MODEL. The host configures the PIP to operate as a Centronics controller by programming the PIP Configuration register (PIPC). Timing attributes (ACK pulse width and the timing between ACK and BUSY) are set by programming the PIP Timing Parameters register (PTPR). The receive errors are reported through the Rx BD.

7.13.8.18 CENTRONICS CONTROL CHARACTERS. The Centronics receiver has the capability to recognize special control characters. These characters may be used when the Centronics functions in a message oriented environment. Up to eight control characters may be defined by the user in the Control Characters Table. Each of these characters may be either written to the receive buffer (upon which the buffer is closed and a new receive buffer taken) or rejected. If rejected, the character is written to the Received Control Character Register (RCCR) in internal RAM and a maskable interrupt is generated. This method is useful for notifying the user of the arrival of control characters that are not part of the received messages.

The Centronics receiver uses a table of 16-bit entries to support control character recognition. Each entry consists of the control character, a valid bit, and a reject character bit.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	E	R							CHARACTER1							
OFFSET + 2	E	R							CHARACTER2							
OFFSET + 4	E	R							CHARACTER3							
OFFSET + 6																

OFFSET + E	E	R							CHARACTER8
OFFSET + 10	1	1							RCCM
OFFSET +1 2									RCCR

CHARACTER1-8—Control Character Values

These fields define control characters that should be compared to the incoming character. For less than 8 bits characters, the msb bits should be zero.

E—End of Table

0 = This entry is valid. The lower 8 bits will be checked against the incoming character.
1 = The entry is not valid. This must be the last entry in the Control Characters Table.

NOTE

In tables with 8 control characters this bit is always 0.

R—Reject character

0 = The character is not rejected but written into the receive buffer. The buffer is then closed and a new receive buffer is used if there is more data in the message. A maskable (I Bit in the Receive BD) interrupt is generated.
1 = If this character is recognized it will not be written to the receive buffer. Instead, it is written to the Received Control Characters Register (RCCR) and a maskable interrupt is generated. The current buffer is not closed when a control character is received with R set.

RCCM—Received Control Character Mask

The value in this register is used to mask the comparison of CHARACTER1 through CHARACTER8. The lower eight bits of RCCM correspond to the lower eight bits of CHARACTER1-8, and are decoded as follows.

0 = Mask this bit in the comparison of the incoming character, and CHARACTER1 through CHARACTER8.
1 = The address comparison on this bit proceeds normally. No masking takes place.

NOTE

The two most significant bits (bit 15 and bit 14) of RCCM must be set, or erratic operation may occur during the control character recognition process.

RCCR—Received Control Character Register

Upon a control character match for which the Reject bit is set, the Centronics will write the control character into the RCCR and generate a maskable interrupt. The core must process the interrupt and read the RCCR before a second control character arrives. Failure to do so will result in the Centronics overwriting the first control character.

7.13.8.19 CENTRONICS SILENCE PERIOD. The Centronics controller may be programmed to close the receive data buffer after a programmable silence period. The length of the silence period is determined by the MAX_SL register value. The centronics controller will decrement the MAX_SL value every 1024 system clocks. If it reaches zero before any data received, the receive buffer will be closed automatically. Setting MAX_SL value to zero disables this function.

7.13.8.20 CENTRONICS RECEIVER COMMAND SET.

7.13.8.20.1 INIT RX PARAMETERS Command. Initializes all the receive parameters in the Centronics parameter RAM to their reset state. This command should only be issued when the receiver is disabled.

7.13.8.20.2 CLOSE RX BD Command. The CLOSE RX BD command is used to force the Centronics controller to close the current receive BD if it is currently being used and to use the next BD in the list for any subsequent data that is received. If the Centronics controller is not in the process of receiving data, no action is taken by this command.

7.13.8.21 RECEIVER ERRORS.

7.13.8.21.1 Buffer Descriptor Busy. This error occurs if a character was received from the Centronics interface and the current BD that should be processed by the Centronics controller is not empty (E bit in the BD = 0). The channel will resume reception after the s/w prepares the BD.

7.13.8.22 CENTRONICS RECEIVE BUFFER DESCRIPTOR. The CP confirms transmission (or indicates error conditions) via the buffer descriptors to inform the processor that the buffers have been serviced.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OFFSET + 0	**E**	—	**W**	**I**	C	—	**CM**	SL	—	—	—	—	—	—	—	—
OFFSET + 2	DATA LENGTH															
OFFSET + 4	**TX DATA BUFFER POINTER**															
OFFSET + 6																

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 core is free to examine or write to any fields of this Rx BD. The CP will not use this BD again while the empty 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 CPU32+ core should not write any fields of this Rx BD.

W—Wrap (Final BD in Table)

0 = This is not the last buffer descriptor in the Rx BD Table.
1 = This is the last buffer descriptor 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 wrap bit and the overall space constraints of the dual-port RAM.

I—Interrupt

0 = No interrupt is generated after this buffer has been filled.
1 = The RX bit in the Centronics event register will be set when this buffer has been completely filled by the CP, indicating the need for the CPU32+ core to process the buffer. The RX bit can cause an interrupt if it is enabled.

C—Control Character

0 = This buffer does not contain a control character.
1 = This buffer contains a control character. The last byte in the buffer is one of the user defined control characters.

CM—Continuous Mode

0 = Normal Operation.
1 = The E-bit is not cleared by the CP after this buffer is closed, allowing the associated data buffer to be overwritten automatically when the CP next accesses this BD.

SL—Silence

The buffer was closed due to the expiration of the programmable silence period timer (defined in MAX_SL).

7.13.8.23 CENTRONICS RECEIVER EVENT REGISTER (PIPE). When the Centronics Receiver protocol is selected, the SMC2 event register is called the Centronics Receiver event register. It is an 8-bit register which is used to report events recognized by the Centronics channel and generate interrupts. On recognition of an event, the Centronics controller will set its corresponding bit in the Centronics event register.

7	6	5	4	3	2	1	0
–	–	–	–	CCR	BSY	CHR	RX

The Centronics event register is a memory-mapped register that 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.

CCR—Control Character Received

A control character was received (with reject (R) character = 1) and stored in the Receive Control Character Register (RCCR).

BSY—Busy Condition

A character was received and discarded due to lack of buffers. Reception continues as soon as an empty buffer is provided.

CHR—Character Received

A character was received from the Centronics channel and was written to the receive buffer.

RX—Rx Buffer

A buffer has been received on the Centronics channel.

7.13.9 Port B Registers

The PIP is associated with parallel I/O port B. The basic operation of the port is shown in Figure 7-96. The registers are described in the port B description; however, the registers as they relate to the PIP are mentioned in the following paragraphs.

7.13.9.1 PORT B ASSIGNMENT REGISTERS (PBPAR). The PBPAR is an 18-bit, memory-mapped, read-write register. To use port B pins as PIP pins, the corresponding PBPAR bits MUST BE CLEARED, and the MODH and MODL bits in the PIP configuration register may be configured as desired.

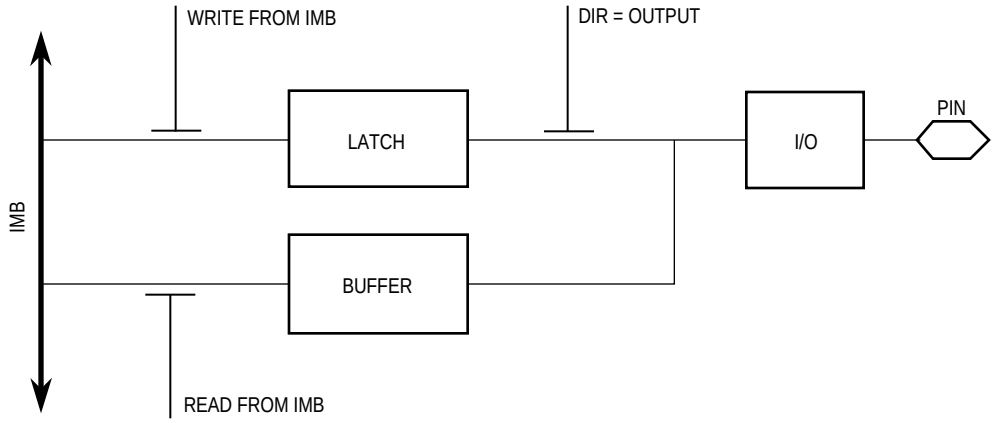


Figure 7-96. Port B General-Purpose I/O

7.13.9.2 DATA DIRECTION REGISTER (PBDIR). The PBDIR is an 18-bit, memory-mapped, read-write register. The description of PBDIR in 7.14.7 Port B Registers is also valid for the PIP.

7.13.9.3 DATA REGISTER (PBDAT). PBDAT functions as the PIP data register when the PIP is operational. This register is used to receive/transmit PIP data when the PIP is under host software control. The description of PBDAT in 7.14.7 Port B Registers is also valid for the PIP.

7.13.9.4 OPEN-DRAIN REGISTER (PBODR). The description of PBODR in 7.14.7 Port B Registers is also valid for the PIP.

7.14 PARALLEL I/O PORTS

The CPM supports three general-purpose I/O ports: A, B, and C.

7.14.1 Parallel I/O Key Features

The parallel I/O ports contain the following key features:

- Port A Is 16 Bits
- Port B Is 18 Bits
- Port C Is 12 Bits
- All Ports Are Bidirectional
- All Ports Have Alternate On-Chip Peripheral Functions
- All Ports Are Three-Stated at System Reset
- All Pin Values May Be Read While Pin Is Connected to an On-Chip Peripheral
- Port A and Port B Offer Open-Drain Capability
- Port C Offers 12 Interrupt Input Pins

7.14.2 Parallel I/O Overview

Each pin in the I/O ports may be configured as a general-purpose I/O pin or as a dedicated peripheral interface pin. Port A is shared with the SCC RXD and TXD pins, the bank of clocks pins, and some TDM pins. Port B is shared with the PIP and other functions such as the IDMA, SMC, and SPI pins. Port C is shared with the $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{CD}}$ pins of the SCCs as well as some TDM pins. Port C is unique in that its pins may generate interrupts to the CPM interrupt controller.

Each pin may be configured as an input or output and has a latch for data output. Each pin may be read or written at any time. Each pin may be configured as general-purpose I/O or as a dedicated peripheral pin.

Port A and port B have pins that can be configured as open-drain—that is, the pin may be configured in a wired-OR configuration on the board. The pin drives a zero voltage, but three-states when driving a high voltage.

NOTES

The port pins do not have internal pullup resistors.

Due to the significant flexibility of the QUICC's CPM, many dedicated peripheral functions are multiplexed onto ports A, B, and C. The functions are grouped in such a way as to maximize the usefulness of the pins in the greatest number of QUICC applications. The reader may not obtain a full understanding of the pin assignment capability described in this section until attaining an understanding of the CPM peripherals themselves.

7.14.3 Port A Pin Functions

Refer to Table 7-19 for the default description of all port A pin options. The pins marked in **boldface** can have open-drain capability. Each of the 16 port A pins is independently configured as a general-purpose I/O pin if the corresponding port A pin assignment register

(PAPAR) bit is cleared. Each pin is configured as a dedicated on-chip peripheral pin if the corresponding PAPAR bit is set.

When the port a 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 A data direction register (PADIR). The port A I/O pin is configured as an input if the corresponding PADIR bit is cleared; it is configured as an output if the corresponding PADIR bit is set. All PAPAR bits and PADIR bits are cleared on total system reset, configuring all port A pins as general-purpose input pins.

Table 7-19. Port A Pin Assignment

Signal	Pin Function			
	PAPAR = 0	PAPAR = 1		Input to On-Chip Peripherals
		PADIR = 0	PADIR = 1	
PA0	PORT A0	RXD1	RXD4[1]	GND
PA1	PORT A1	TXD1	TXD4[1]	—
PA2	PORT A2	RXD2	—	GND
PA3	PORT A3	TXD2	—	—
PA4	PORT A4	RXD3	L1TXDB	Undefined
PA5	PORT A5	TXD3	L1RXDB	GND
PA6	PORT A6	RXD4	L1TXDA	Undefined
PA7	PORT A7	TXD4	L1RXDA	L1RXDA = GND
PA8	PORT A8	CLK1/TIN1/L1RCLKA	BRGO1	CLK1/TIN1/L1RCLKA = BRGO1
PA9	PORT A9	CLK2	TOUT1	CLK2 = GND
PA10	PORT A10	CLK3/TIN2/L1TCLKA	BRGO2	CLK3/TIN2/L1TCLKA = BRGO2
PA11	PORT A11	CLK4	TOUT2	CLK4 = CLK8
PA12	PORT A12	CLK5/TIN3	BRGO3	CLK5/TIN3 = BRGO3
PA13	PORT A13	CLK6/L1RCLKB	TOUT3	CLK6/L1RCLKB = GND
PA14	PORT A14	CLK7/TIN4	BRGO4	CLK7/TIN4 = BRGO4
PA15	PORT A15	CLK8/L1TCLKB	TOUT4	CLK8/L1TCLKB = GND

NOTES:
1: Only available on REV C mask or later. NOT Available on REV A or B. And when PA6 or PA7 is not used as TXD4 or RXD4 functions
Rev A mask is C63T
Rev B mask are C69T, and F35G
Current Rev C mask are E63C, E68C and F15W

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

If an input to a peripheral is not supplied from a pin, then a default value is supplied to the on-chip peripheral as listed in Table 7-19.

7.14.4 Port A Registers

Port A has four memory-mapped, read-write, 16-bit control registers.

7.14.4.1 PORT A OPEN-DRAIN REGISTER (PAODR). The PAODR indicates a normal or wired-OR configuration of the port pins. Six of the PAODR bits can be open-drain to correspond to those pins that have serial channel output capability. The other bits are always zero. PAODR is cleared at system reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	OD7	OD6	OD5	OD4	OD3	0	OD1	0

For each ODx bit, the definition is as follows:

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 actively driven low, but in three-stated otherwise.

7.14.4.2 PORT A DATA REGISTER (PADAT). A read of PADAT returns the data at the pin, independent of whether the pin is defined as an input or an 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 PADIR is latched, and if that bit in the PADIR is configured as an output, the value latched for that bit will be driven onto its respective pin. PADAT can be read or written at any time. PADAT is not initialized and is undefined at reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

7.14.4.3 PORT A DATA DIRECTION REGISTER (PADIR). PADIR is cleared at system reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR15	DR14	DR13	DR12	DR11	DR10	DR9	DR8	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0

For each DRx bit, the definition is as follows:

0 = The corresponding pin is an input.
1 = The corresponding pin is an output.

7.14.4.4 PORT A PIN ASSIGNMENT REGISTER (PAPAR). PAPAR is cleared at system reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DD15	DD14	DD13	DD12	DD11	DD10	DD9	DD8	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0

For each DDx bit, the definition is as follows:

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 may be determined by other bits such as those is the PADIR.

7.14.5 Port A Examples

The following paragraphs discuss various ways some of the port A pins can be configured. Figure 7-97 and Figure 7-98 show block diagrams of the PA0 and PA1 pins.

PA0 can be configured as a general-purpose I/O pin, but not an open-drain pin. It may also be the RXD1 pin for SCC1 in the NMSI mode. If PA0 is configured as a general-purpose I/O pin, then the RXD1 input is internally grounded. If SCC1 is connected to a TDM or is not used, then PA0 may be used as general-purpose I/O.

PA1 can be configured as a general-purpose I/O pin, either open-drain or not. It may also be the TXD1 pin for SCC1 in the NMSI mode. If TXD1 is configured as an output on PA1 and the OD1 bit is set in PAODR, then TXD1 will be output from SCC1 as an open-drain output. If PA1 is configured as a general-purpose I/O pin, then the TXD1 output is not connected externally. If SCC1 is connected to a TDM or is not used, then PA1 may be used as a general-purpose I/O.

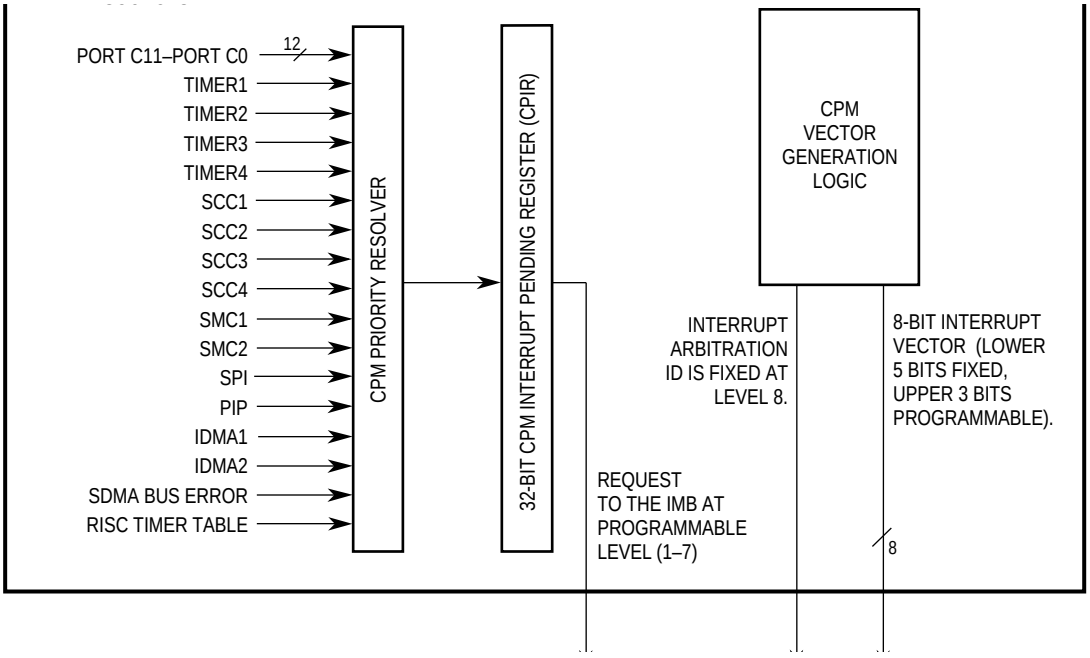


Figure 7-97. Parallel Block Diagram for PA0

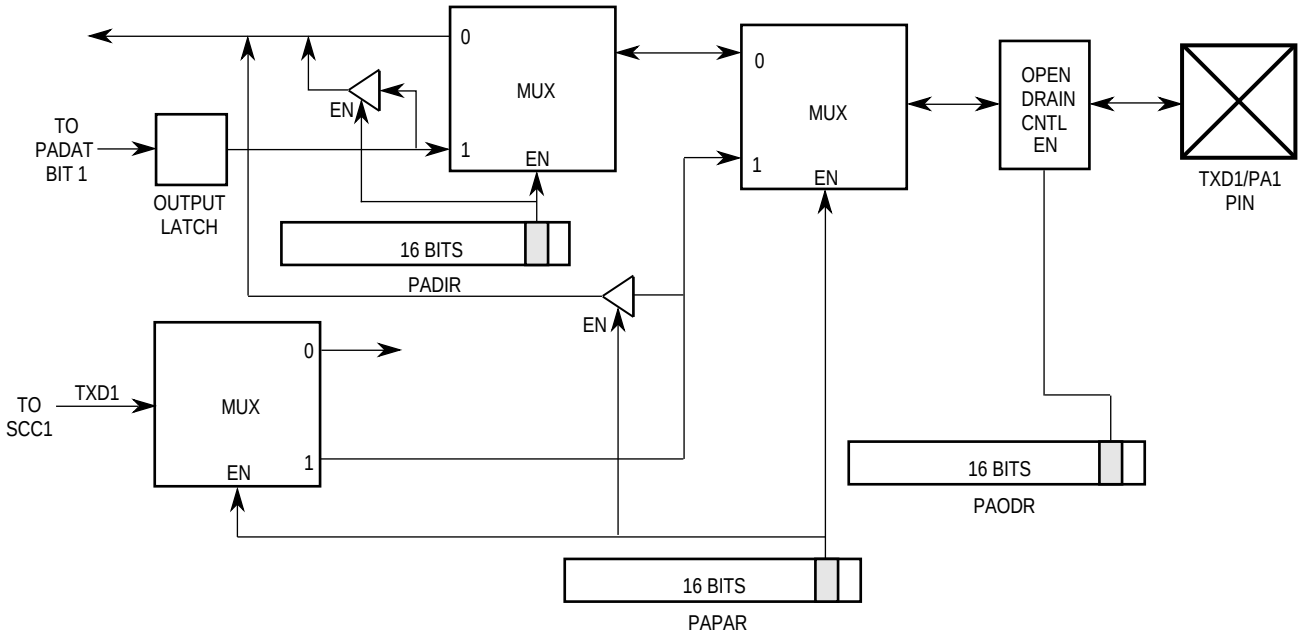


Figure 7-98. Parallel Block Diagram for PA1

PA4 can be configured as a general-purpose I/O pin, and as an open-drain pin. It may also be the RXD3 pin for SCC3 in the NMSI mode if the PADIR bit is a zero. It may also be the L1TXDB pin for TDMb if the PADIR bit is a one. If PA4 is configured as a general-purpose I/O pin, then the RXD3 input is not defined. If SCC3 is connected to a TDM or is not used, then PA4 may be used as general-purpose I/O.

PA8 can be configured as a general-purpose I/O pin, but not an open-drain pin. If the corresponding PADIR bit is a zero, it may also be the CLK1 pin (part of the bank of clocks in the SI), the TIN1 pin (input to timer 1), or the L1RCLKA pin (receive clock to TDMa) or all three at once. There is no selection between these three inputs in port A, because the connections are made separately in the SI and the timer mode registers. If the PADIR bit is a one, this pin may also be the BRGO1 pin (output from BRG1). If the PA8 pin is a general-purpose I/O pin, then the input to the on-chip peripheral (CLK1, TIN1, or L1RCLKa) is internally connected to BRGO1. See 7.8 Serial Interface with Time Slot Assigner for more details on the use of the CLK1 and L1RCLKA pins.

PA11 can be configured as a general-purpose I/O pin, but not an open-drain pin. If the PADIR bit is a zero, PA11 may also be the CLK4 pin (part of the bank of clocks). If the PADIR bit is a one, PA11 may also be the $\overline{\text{TOUT2}}$ pin (output from timer 2). If the PA11 pin is a general-purpose I/O pin, then the input to the on-chip CLK4 function is the value supplied on the CLK8 pin. This interesting option is useful because not all CLK pins can be routed to all serial channels in all situations. The ability to send a clock from CLK8 to CLK4 can increase the flexibility of this assignment process. See 7.8 Serial Interface with Time Slot Assigner for more details.

7.14.6 Port B Pin Functions

Refer to Table 7-20 for the description of all port B pin options. All port B pins except PB17 and PB16 may be open-drain. Port B pins are independently configured as a general-purpose I/O pins if the corresponding bit in the port B pin assignment register (PBPAR) is cleared. They are configured as dedicated on-chip peripheral pins if the corresponding PBPAR bit is set.

When acting as a general-purpose I/O pin, the signal direction for that pin is determined by the corresponding control bit in the port B data direction register (PBDIR). The port I/O pin is configured as an input if the corresponding PBDIR bit is cleared; it is configured as an output if the corresponding PBDIR bit is set. All PBPAR bits and PBDIR bits are cleared on total system reset, configuring all port B pins as general-purpose input pins.

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

All of the port B pins have more than one option. These options include on-chip peripheral functions relating to the IDMA, Ethernet CAM interface, SPI, SMC1, SMC2, TDMa, and TDMb.

Port B is also multiplexed with the PIP. The PIP is a CPM parallel port that can implement fast parallel interfaces, such as Centronics. For a functional description of the dedicated pin functions of the PIP, refer to 7.13 Parallel Interface Port (PIP).

NOTES

If the user does not use the PIP, the description in this section is sufficient to describe the features of port B, and the PIP description does not need to be studied.

The PIP STRBI and STRBO pins are not listed in Table 7-20. See 7.13 Parallel Interface Port (PIP) for instructions on how to enable them.

PB3–PB5 and PB16 have an unusual property in that their on-chip peripheral functions (BRGO4, BRGO3, BRGO2, and BRGO1) are repeated in port A. This gives an alternate way to output the BRGO pins if other functions are used on port A. PB12–PB15 have an unusual property in that their on-chip peripheral functions (such as $\overline{\text{RTSx}}$ or L1ST1) are repeated in port C. This gives an alternate location to output these pins if other functions on port C are used.

Table 7-20. Port B Pin Assignment

Signal	Pin Function			
	PBPAR = 0	PBPAR = 1		Input to On-Chip Peripherals
		PBDIR = 0	PBDIR = 1	
PB0	PORT B0	$\overline{\text{RRJCT1}}$	SPISEL	V_{DD}
PB1	PORT B1	$\overline{\text{RSTR2}}$	SPICLK	SPICLK = GND
PB2	PORT B2	$\overline{\text{RRJCT2}}$	SPIMOSI	V_{DD}
PB3	PORT B3	BRGO4	SPIMISO	SPIMISO = SPIMOSI
PB4	PORT B4	BRGO1	DREQ1	$\overline{\text{DREQ1}}$ = GND
PB5	PORT B5	BRGO2	DACK1	—
PB6	PORT B6	SMTXD1	DONE1	$\overline{\text{DONE1}}$ = V_{DD}
PB7	PORT B7	SMRXD1	DONE2	V_{DD}
PB8	PORT B8	SYMSYN1	DREQ2	GND
PB9	PORT B9	SYMSYN2	DACK2	$\overline{\text{SYMSYN2}}$ = GND
PB10	PORT B10	SMTXD2	L1CLKOB	—
PB11	PORT B11	SMRXD2	L1CLKOA	SMRXD2 = GND
PB12	PORT B12	L1ST1	RTS1	—
PB13	PORT B13	L1ST2	RTS2	—
PB14	PORT B14	L1ST3	$\overline{\text{RTS3}}$/L1RQB	—
PB15	PORT B15	L1ST4	$\overline{\text{RTS4}}$/L1RQA	—
PB16	PORT B16	—	BRGO3	—
PB17	PORT B17	—	RSTRT1	—

NOTE

The user may freely configure any of the previous functions to be output onto two pins at once, although there is typically no advantage in doing this (except in the case of a large fanout, where it is advantageous to share the load between two pins).

7.14.7 Port B Registers

Port B has four memory-mapped, read-write, 16-bit control registers.

7.14.7.1 PORT B OPEN-DRAIN REGISTER (PBODR). The PBODR is a 16-bit register that indicates a normal or wired-OR configuration of the port pins. (Bits 17 and 16 of PBODR do not exist.) PBODR is cleared at system reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0

For each ODx bit, the definition is as follows:

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 actively driven low, but is three-stated otherwise.

NOTE

SMTxD1, DONE1 and DONE2 can not be set as open drain driver regardless of the setting of this register.

7.14.7.2 PORT B DATA REGISTER (PBDAT). A read of PBDAT returns the data at the pin, independent of whether the pin is defined as an input or an 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 PBDAT is latched, and if that bit in the PBDIR is configured as an output, the value latched for that bit will be driven onto its respective pin. PBDAT can be read or written at any time. PBDAT is not initialized and is undefined at reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
														17	16
														D17	D16

7.14.7.3 PORT B DATA DIRECTION REGISTER (PBDIR). PBDIR is cleared at system reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR15	DR14	DR13	DR12	DR11	DR10	DR9	DR8	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0
														17	16
														DR17	DR16

For each DRx bit, the definition is as follows:

0 = The corresponding pin is an input.
1 = The corresponding pin is an output.

7.14.7.4 PORT B PIN ASSIGNMENT REGISTER (PBPAR). PBPAR is cleared at system reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DD15	DD14	DD13	DD12	DD11	DD10	DD9	DD8	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0
														17	16
														DD17	DD16

For each DDx bit, the definition is as follows:

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 may be determined by other bits such as these in the PBDIR.

7.14.8 Port B Example

PB0 can be configured as a general-purpose I/O pin or as an open-drain pin. It may also be the receiver reject pin for the SCC1 Ethernet CAM interface ($\overline{\text{RRJCT1}}$) or the SPI select input ($\overline{\text{SPISEL}}$). If the PB0 pin is not configured to connect to the $\overline{\text{RRJCT}}$ signal or the $\overline{\text{SPISEL}}$ signal, then the SCC and/or SPI receives V_{DD} on that signal.

NOTE

In the description of the PIP, the PB0 pin, as well as other port B pins, can also be used as PIP functions. However, the PIP does not affect the operation of port B unless it is enabled. Therefore, the PIP description does not need to be studied by users of port B unless the PIP will be used in the application.

7.14.9 Port C Pin Functions

Port C consists of 12 general-purpose I/O pins with interrupt capability on each pin. Refer to Table 7-21 for the description of all port C pin options.

Table 7-21. Port CPin Assignment

Signal	PCPAR = 0		PCPAR = 1		Input to On-Chip Peripherals
	PCDIR = 1 or PCSO = 0	PCDIR = 0 and PCSO = 1	PCDIR = 0	PCDIR = 1	
PC0	Port C0	—	RTS1	L1ST1	—
PC1	Port C1	—	RTS2	L1ST2	—
PC2	Port C2	—	$\overline{\text{RTS3}}$/L1RQB	L1ST3	—
PC3	Port C3	—	$\overline{\text{RTS4}}$/L1RQA	L1ST4	—
PC4	Port C4	CTS1	—		GND
PC5	Port C5	CD1	TGATE1		GND
PC6	Port C6	CTS2	—		GND
PC7	Port C7	CD2	TGATE2		GND
PC8	Port C8	CTS3	L1TSYNCB	SDACK2	$\overline{\text{CTS3}}$ and/or L1TSYNCB = GND
PC9	Port C9	CD3	L1RSYNCB		GND
PC10	Port C10	CTS4	L1TSYNCA	SDACK1	$\overline{\text{CTS4}}$ and/or L1TSYNCA = GND
PC11	Port C11	CD4	L1RSYNCA		GND

All PCDIR bits and PCPAR bits are cleared on total system reset, configuring all port pins as general-purpose input pins. Note that the global CIMR is also cleared on total system reset so that, if any PCIO pin is left floating, it will not cause a false interrupt.

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

To configure a port C pin an a general-purpose output pin, use the following steps. Note that when the pin is configured as an output, port C interrupts are not possible.

1. Write the corresponding PCPAR bit with a zero.
2. Write the corresponding PCDIR bit with a one.
3. Write the corresponding PCSO bit with a zero (for the sake of clarity).
4. The corresponding PCINT bit is a don’t care.
5. Write the pin value using the PCDAT.

To configure a port C pin as a general-purpose input pin that does not generate an interrupt, use the following steps:

1. Write the corresponding PCPAR bit with a zero.
2. Write the corresponding PCDIR bit with a zero.
3. Write the corresponding PCSO bit with a zero.
4. The corresponding PCINT bit is a don’t care.
5. Write the corresponding CIMR bit with a zero to prevent interrupts from being generated to the CPU32+ core.
6. Read the pin value using the PCDAT.

When a port C pin is configured as a general-purpose I/O input, a change according to the port C interrupt register (PCINT) will cause an interrupt request signal to be sent to the CPM interrupt controller. Each port C line can be programmed to assert an interrupt request upon a high-to-low change or any change. Each port C line asserts a unique interrupt request to the CPM interrupt pending register and has a different internal interrupt priority level within the CPM interrupt controller. See 7.15 CPM Interrupt Controller (CPIC) for more details. Each request can be masked independently in the CPM interrupt mask register.

To configure a port C pin an a general-purpose input pin that generates an interrupt, use the following steps:

1. Write the corresponding PCPAR bit with a zero.
2. Write the corresponding PCDIR bit with a zero.
3. Write the corresponding PCSO bit with a zero.

4. Set the PCINT bit to determine which edges cause interrupts.
5. Write the corresponding CIMR bit with a one to allow interrupts to be generated to the CPU32+ core.
6. Read the pin value using the PCDAT.

NOTE

These steps correspond to the “software operation” mode of the SCM DIAG bits on the MC68302.

The port C lines associated with the $\overline{\text{CDx}}$ and $\overline{\text{CTSx}}$ pins have a mode of operation where the pin may be internally connected to the SCC but may 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 simultaneously uses the $\overline{\text{CTS}}$ and/or $\overline{\text{CD}}$ pin to automatically control operation. This allows the user to fully implement protocols V.24, X.21, and X.21 bis (with the assistance of other general-purpose I/O lines).

To configure a port C pin as a $\overline{\text{CTS}}$ or $\overline{\text{CD}}$ pin that is connected to the SCC and also generates interrupts, use the following steps:

1. Write the corresponding PCPAR bit with a zero.
2. Write the corresponding PCDIR bit with a zero.
3. Write the corresponding PCSO bit with a one.
4. Set the PCINT bit to determine which edges cause interrupts.
5. Write the corresponding CIMR bit with a one to allow interrupts to be generated to the CPU32+ core.
6. The pin value may be read at any time using PCDAT.

NOTE

After connecting the $\overline{\text{CTS}}$ or $\overline{\text{CD}}$ pins to the SCC, the user must also choose the “normal operation” mode in DIAG bits of the general SCC mode register (GSMR) to enable and disable SCC transmission and reception with these pins.

7.14.10 Port C Registers

The user interfaces with port C via five registers. The port C interrupt control register (PCINT) indicates how changes on the pin cause interrupts when interrupts are generated with that pin. The port C special options register (PCSO) indicates whether certain port C pins have the ability to be connected to on-chip peripherals while simultaneously being able to generate an interrupt. The other three port C registers also exist on the other ports: PCDAT, PCDIR, and PCPAR. Since port C does not have open-drain capability, it does not contain an open-drain register.

7.14.10.1 PORT C DATA REGISTER (PCDAT). When read, PCDAT always reflects the current status of each line.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

7.14.10.2 PORT C DATA DIRECTION REGISTER (PCDIR). PCDIR is a 16-bit register that is cleared at system reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	DR11	DR10	DR9	DR8	DR7	DR6	DR5	DR4	DR3	DR2	DR1	DR0

7.14.10.3 PORT C PIN ASSIGNMENT REGISTER (PCPAR). PCPAR is a 16-bit register that is cleared at system reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	DD11	DD10	DD9	DD8	DD7	DD6	DD5	DD4	DD3	DD2	DD1	DD0

7.14.10.4 PORT C SPECIAL OPTIONS (PCSO). PCSO is a 16-bit read-write register. Each defined bit in the PCSO corresponds to a port C line (PC11–PC4 and PC1–PC0). The PCSO is cleared at reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—				CD4	CTS4	CD3	CTS3	CD2	CTS2	CD1	CTS1	—			

Bits 15–12, 3–2—Reserved

These bits should be written with zeros.

CDx—Carrier Detect

0 = PCx is a general-purpose interrupt I/O pin. (The SCC's internal CDx signal is always asserted.) If PCDIR configures this pin as an input, this pin can generate an interrupt to the CPU32+ core, as controlled by the PCINT bits.

1 = PCx is connected to the corresponding SCC signal input in addition to being a general-purpose interrupt pin.

CTSx—Clear-To-Send

0 = PCx is a general-purpose interrupt I/O pin. (The SCC's internal CTSx signal is always asserted.) If PCDIR configures this pin as an input, this pin can generate an interrupt to the CPU32+ core, as controlled by the PCINT bits.

1 = PCx is connected to the corresponding SCC signal input in addition to being a general-purpose interrupt pin.

EXTx— External Request to the RISC

0 = PCx is a general-purpose interrupt I/O pin, with the direction controlled in PCDIR. If PCDIR configures this pin as an input, this pin can generate an interrupt to the CPU32+ core, as controlled by the PCINT bits.

1 = PCx becomes an external request to the RISC controller instead of being a general-purpose interrupt pin. The corresponding PCINT bits control when a request is generated.

NOTE

EXTx should only be set, if the user is instructed to do so, during the initialization of a Motorola-supplied RAM microcode.

7.14.10.5 PORT C INTERRUPT CONTROL REGISTER (PCINT). PCINT is a 16-bit read-write register. Each defined bit in the PCINT corresponds to a port C line to determine whether the corresponding port C line will assert an interrupt request upon a high-to-low change or any change on the pin. The PCINT is cleared at reset.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
—				EDM11	EDM10	EDM9	EDM8	EDM7	EDM6	EDM5	EDM4	EDM3	EDM2	EDM1	EDM0

Bits 15–12—Reserved.

These bits should be written with zeros.

EDMx—Edge Detect Mode for Line x

The corresponding port C line (PCx) will assert 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.

7.15 CPM INTERRUPT CONTROLLER (CPIC)

The CPIC is the focal point for all interrupts associated with the CPM. The CPIC accepts and prioritizes all the internal and external interrupt requests from all functional blocks associated with the CPM. It is also responsible for generating a vector during the CPU interrupt acknowledge cycle.

The CPIC contains has the following key features:

- Twenty-Eight Interrupt Sources (16 Internal and 12 External)
- Sources May Be Assigned to a Programmable Interrupt Level (1–7)
- Programmable Priority Between SCCs
- Two Priority Schemes for the SCCs
- Programmable Highest Priority Request
- Fully Nested Interrupt Environment
- Unique Vector Number for Each Interrupt Source

7.15.1 Overview

An overview of the QUICC interrupt structure is shown in Figure 7-99. The upper half of the figure shows the CPIC. The CPIC receives interrupts from internal sources such as the four SCCs, the two SMCs, the SPI, the two IDMA controllers, the PIP, the general-purpose timers, and the port C parallel I/O pins. The CPIC allows masking of each interrupt source. When multiple events within a CPM sub-block can cause the interrupt, each event is also maskable in that CPM sub-block.

All CPM sub-block interrupt sources are prioritized, and bits are set in the CPM interrupt pending register (CIPR). All 28 interrupt sources within the CIPR are assigned one programmable priority level (1–7) before the request for an interrupt is sent to the IMB. (On the MC68302, all interrupt sources are fixed at priority level 4; however, on the QUICC, the interrupt level is programmable to be 1–7.)

Within the CPM interrupt level, the 28 sources are assigned a priority structure. On the MC68302, the interrupts have a fixed priority structure; however, on the QUICC, some flexibility is given to the user concerning the relative priority of the 28 interrupt sources. This flexibility is in two areas: 1) the ability to modify the relative priority of the SCCs and 2) the ability to choose any interrupt source to be the highest of the 28 sources.

Once an unmasked interrupt source is pending in the CIPR, the CPIC sends an interrupt request to the IMB. This request is at level 1, 2, 3, 4, 5, 6, or 7. The CPIC then waits for an interrupt acknowledge cycle to occur on the bus.

Once an interrupt cycle occurs at the interrupt level that matches the CPIC interrupt request, an interrupt arbitration begins on the IMB. The interrupt arbitration process is designed to choose between multiple requests at the same level. For instance, if the CPM request is at level 4, but an external peripheral is simultaneously requesting service on the $\overline{\text{IRQ4}}$ pin, an interrupt arbitration process is required to decide who wins the interrupt. (The interrupt arbitration process does not affect users who can assign all interrupt sources in the system to a unique interrupt level 1–7.)

In the interrupt arbitration process, the module places its arbitration ID on the IMB. The arbitration ID ranges in value from 0–15. The CPIC arbitration ID is always fixed at 8. The higher arbitration value always wins.

NOTE

The other source of interrupts on the QUICC is the SIM60, which has a programmable arbitration ID (initially 15). Thus, if a SIM sub-block is programmed to the same interrupt level, then a higher SIM60 arbitration ID selects whether the SIM60 has a higher interrupt priority than the CPM.

Assuming that the CPM wins the arbitration process, the CPM places its 8-bit vector on the bus, corresponding to the sub-block with the highest current priority. The vector is composed of two parts. The three MSBs of the interrupt vector come from a 3-bit field in the CPM inter-

rupt configuration register (CICR). The five LSBs are fixed in the CPIC, and are unique for each CPIC interrupt source.

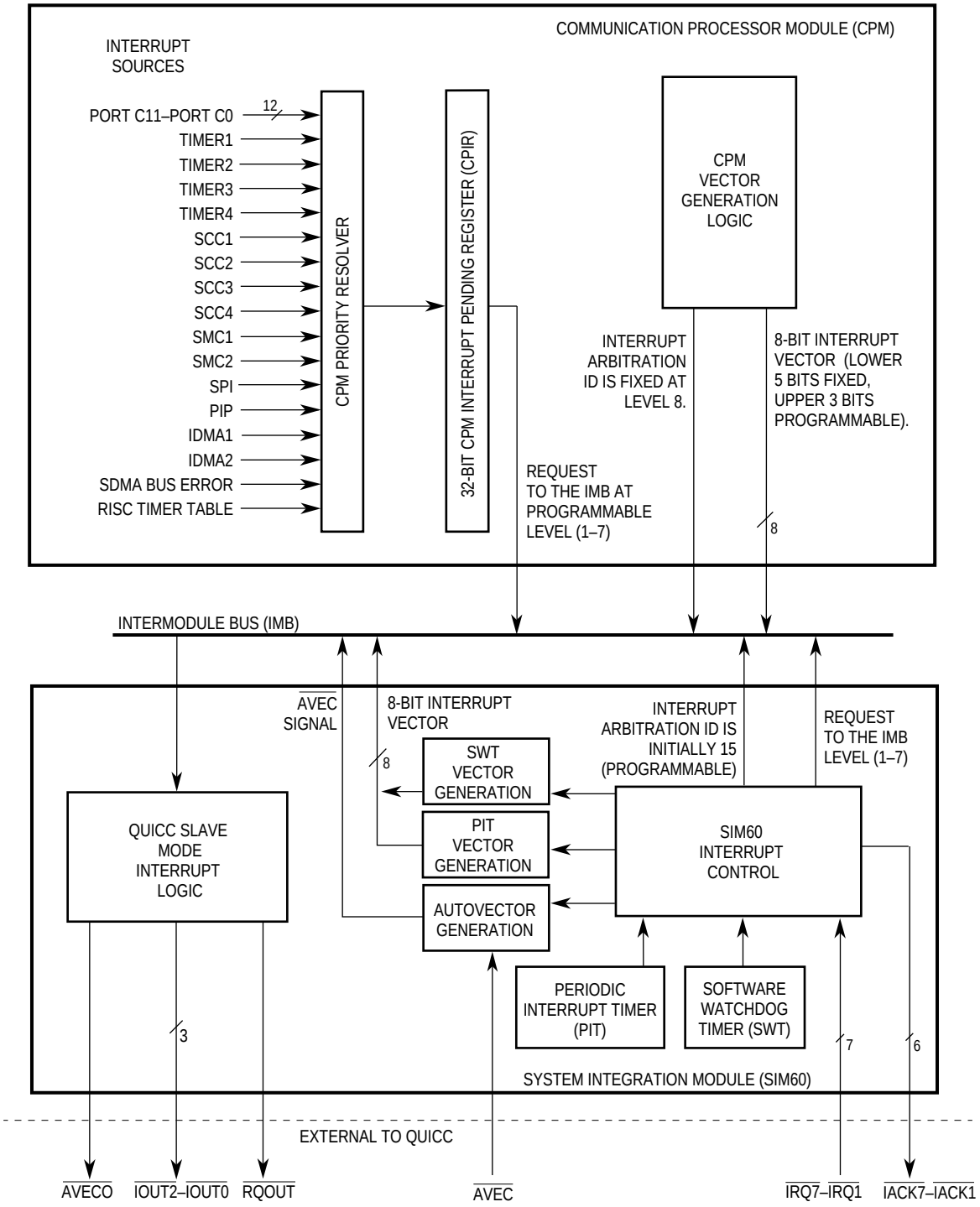


Figure 7-99. QUICC Interrupt Structure

7.15.2 CPM Interrupt Source Priorities

The CPIC has 28 interrupt sources that assert just one programmable interrupt request level to the CPU32+ core. The priority between all interrupt sources is shown in Table 7-22. There is some flexibility in the relative ordering of the interrupts in the table, but, in general, the relative priorities are fixed in the descending order shown in the table. An interrupt from the parallel I/O line PC0 has the highest priority, and an interrupt from the parallel I/O line PC11 has the lowest priority. A single interrupt priority number is associated with each table entry.

Note the lack of SDMA interrupt sources. The SDMA-related interrupts are reported through each individual SCC, SMC, or SPI channel. The only true SDMA interrupt source is the SDMA channel's bus error entry that is reported when a bus error occurs during an SDMA access.

There are two methods to add flexibility to the table of CPM interrupt priorities: the SCC's relative priority option and the highest priority option.

7.15.2.1 SCC RELATIVE PRIORITY. The relative priority between the four SCCs is programmable and can be dynamically changed. In Table 7-22 there is no entry for SCC1, SCC2, SCC3, or SCC4, but rather there are entries for SCCa, SCCb, SCCc, and SCCd because each of the SCCs can be mapped to any of these locations. This is programmed in the CICR and may be dynamically changed. The user can utilize this on-the-fly capability to implement a rotating priority.

In addition, the grouping of the locations of the SCCa, SCCb, SCCc, and SCCd entries has two options: group and spread. In the group scheme, the SCCs are all grouped together at the top of the priority table, ahead of most of the other CPM interrupt sources. This scheme is ideal for applications where all SCCs function at a very high data rate and interrupt latency is very important. In the spread scheme, the SCC priorities are spread over the table so that other sources can have lower interrupt latencies than the SCCs. This scheme is also programmed in the CICR, but it may not be dynamically modified.

7.15.2.2 HIGHEST PRIORITY INTERRUPT. In addition to the SCC relative priority option, the user may choose one interrupt source to be of highest priority. This highest priority interrupt is still within the same interrupt level as the rest of the CPIC interrupts, but is simply serviced prior to any other interrupt in the table. If the highest priority feature is not used, select PC0 to be the highest priority interrupt, and no modifications to the standard interrupt priority order will occur.

This highest priority source is dynamically programmable in the CICR. This allows the user to change a normally low priority source into a high priority source for a specified period of time.

Table 7-22. Prioritization of CPM Interrupt Sources

Number	Priority Level	Interrupt Source Description	Multiple Events
1F	Highest	Parallel I/O–PC0	No
1E		SCCa (Grouped and Spread)	Yes
1D		SCCb (Grouped)	Yes
1C		SCCc (Grouped)	Yes
1B		SCCd (Grouped)	Yes
1A		Parallel I/O–PC1	No
19		Timer 1	Yes
18		Parallel I/O–PC2	No
17		Parallel I/O–PC3	No
16		SDMA Channel Bus Error	Yes
15		IDMA1	Yes
14		IDMA2	Yes
13		SCCb (Spread)	Yes
12		Timer 2	Yes
11		RISC Timer Table	Yes
10		Reserved	Yes
F		Parallel I/O–PC4	No
E		Parallel I/O–PC5	No
D		SCCc (Spread)	Yes
C		Timer 3	Yes
B		Parallel I/O–PC6	No
A		Parallel I/O–PC7	No
9		Parallel I/O–PC8	No
8		SCCd (Spread)	Yes
7		Timer 4	Yes
6		Parallel I/O–PC9	No
5		SPI	Yes
4		SMC1	Yes
3		SMC2/PIP	Yes
2		Parallel I/O–PC10	No
1		Parallel I/O–PC11	No
0	Lowest	Reserved	—

7.15.2.3 NESTED INTERRUPTS. The CPIC supports a fully nested interrupt environment that allows a higher priority interrupt (from another CPM source) to suspend a lower priority interrupt's service routine. This nesting is achieved by the CPM interrupt in-service register (CISR).

The CPIC prioritizes all interrupt sources based upon their assigned priority level. The highest priority interrupt request is presented to the CPU32+ core for servicing. After the vector number corresponding to this interrupt is passed to the CPU32+ core during an interrupt acknowledge cycle, that interrupt request is cleared. If there are remaining interrupt requests, they are then prioritized, and another interrupt request may be presented to the CPU32+ core.

The 3-bit mask in the CPU32+ status register ensures that a subsequent interrupt request at a higher interrupt priority level will suspend handling of a lower priority interrupt. The mask indicates the current processor priority, and interrupts are inhibited for all priority levels less than or equal to the current processor priority.

The CISR and the mask register in the CPU32+ core can be used together to allow a higher priority interrupt within the same interrupt level to be presented to the CPU32+ core before the servicing of a lower priority interrupt is completed. Each bit in the CISR corresponds to a CPM interrupt source. During an interrupt acknowledge cycle for a CPM interrupt, the in-service bit in the CISR is set by the CPIC for that interrupt source. The setting of the bit prevents any subsequent CPM interrupt requests at this priority level or lower (within the CPIC interrupt table), until the servicing of the current interrupt has completed and the in-service bit is cleared by the user. (Pending interrupts for these sources are still set in the CPIC during this time).

Thus, in the interrupt service routine for the CPM interrupts, the user can lower the core's mask to the next lower level (the level being serviced minus 1) to allow higher priority interrupts within this level to generate an interrupt request. This capability provides nesting of interrupt requests for CPM interrupt level sources in a similar manner as the CPU32+ core's interrupt mask provides nesting of interrupt requests for the seven interrupt priority levels.

7.15.3 Masking Interrupt Sources in the CPM

By programming the CPM interrupt mask register (CIMR), the user may mask the CPM interrupts to prevent an interrupt request to the CPU32+ core. Each bit in the CIMR corresponds to one of the CPM interrupt sources. To enable an interrupt, write a one to the corresponding CIMR bit.

When a masked CPM interrupt source has a pending interrupt request, the corresponding bit in the CIPR is still set, even though the interrupt is not generated to the CPU32+ core. By masking all interrupt sources in the CIMR, the user may implement a polling interrupt servicing scheme for the CPM interrupts.

When a CPM interrupt source has multiple interrupting events, the user can individually mask these events by programming a mask register within that block. Table 7-22 indicates the interrupt sources that have multiple interrupting events. Figure 7-100 shows an example of how the masking occurs, using an SCC as an example.

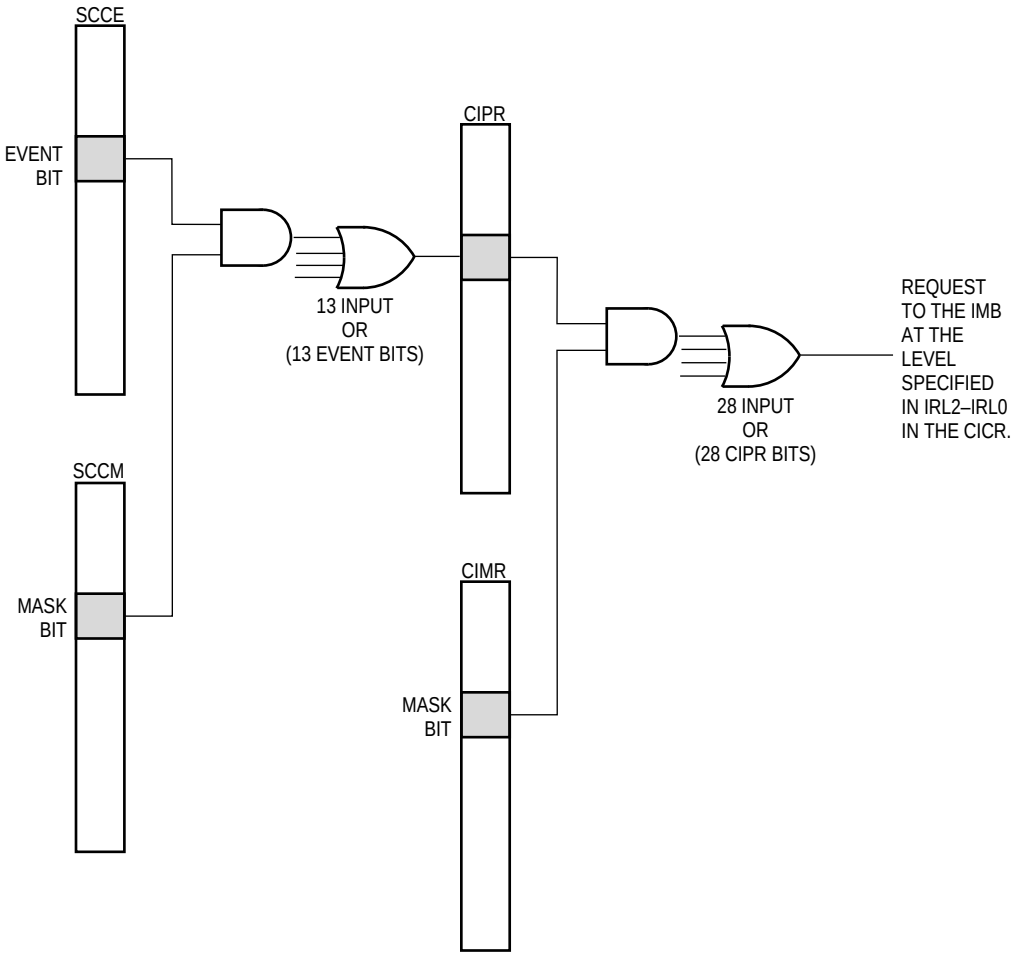


Figure 7-100. Interrupt Request Masking

7.15.4 Interrupt Vector Generation and Calculation

Pending unmasked CPM interrupts are presented to the CPU32+ core in order of priority. The CPU32+ core responds to an interrupt request by initiating an interrupt acknowledge cycle to receive a vector number, which allows the core to locate the interrupt's service routine. For CPM interrupts, the CPIC passes an interrupt vector corresponding to the highest priority, unmasked, pending interrupt. The CPM always generates a vector during an interrupt acknowledge cycle at its interrupt level, regardless of whether the QUICC is in normal mode or slave mode.

The three MSBs of the interrupt vector number are programmed by the user in the CIMR. These three bits are concatenated with five bits generated by the CPIC to provide an 8-bit vector number to the CPU32+ core. The CPIC's encoding of the five low-order bits of the interrupt vector is listed in Table 7-22.

Note that the interrupt vector table is the same as the CPM interrupt priority table except for two differences. First, the lower five bits of the SCC vectors are fixed; they are not affected by the SCC group or spread mode or the relative priority order of the SCCs. Second, an error

Table 7-23. Encoding the Interrupt Vector

Interrupt Number	Interrupt Source Description	Lower 5 Bits of Vector
1F	Parallel I/O—PC0	11111
1E	SCC1	11110
1D	SCC2	11101
1C	SCC3	11100
1B	SCC4	11011
1A	Parallel I/O—PC1	11010
19	Timer 1	11001
18	Parallel I/O—PC2	11000
17	Parallel I/O—PC3	10111
16	SDMA Channel Bus Error	10110
15	IDMA1	10101
14	IDMA2	10100
13	Reserved	10011
12	Timer 2	10010
11	RISC Timer Table	10001
10	Reserved	10000
F	Parallel I/O—PC4	01111
E	Parallel I/O—PC5	01110
D	Reserved	01101
C	Timer 3	01100
B	Parallel I/O—PC6	01011
A	Parallel I/O—PC7	01010
9	Parallel I/O—PC8	01001
8	Reserved	01000
7	Timer 4	00111
6	Parallel I/O—PC9	00110
5	SPI	00101
4	SMC1	00100
3	SMC2 / PIP	00011
2	Parallel I/O—PC10	00010
1	Parallel I/O—PC11	00001
0	Error	00000

vector exists as the last entry in this table. The error vector is issued by the CPM if an interrupt was requested by the CPM but was masked by the user prior to being serviced by CPU32+ core and if no other pending interrupts for the CPM are present. The user should provide an error interrupt service routine, even if it is simply an RTE instruction.

The following list gives an example of how to find the beginning of the interrupt handler from the interrupt vector. SCC1 is used as an example.

1. Formulate the 8-bit vector. The three MSBs come from VBA2–VBA0 in the CICR. Assume these are programmed to 101b. The five LSBs have a fixed value of 11110b (see Table 7-22). Thus, the 8-bit vector is 10111110b. This is the value presented on the bus during an interrupt acknowledge cycle.
2. Multiply by 4 to get the offset address of the vector in the vector table. Thus, the offset address is 1011111000b = $2F8.
3. Determine the full vector address. In a CPU32+ system, the offset is added to the vector base register in the CPU32+. Assuming that the vector base register = $80000000, the final vector address is $800002F8.
4. Determine the location of the interrupt handler. At location $800002F8, the address of the interrupt handler is stored. If the long word at location $800002F8 contains $80001000, then the first instruction of the SCC1 interrupt handler will be found at $80001000.

7.15.5 CPIC Programming Model

The user interfaces with the CPIC via four registers. The CICR defines the overall CPM interrupt attributes. The CIPR indicates which CPM interrupt sources require interrupt service. The CIMR allows the user to prevent any CPM interrupt source from generating an interrupt request. The CISR allows a fully nested environment capability for interrupt requests within the CPM interrupt level.

7.15.5.1 CPM INTERRUPT CONFIGURATION REGISTER (CICR). The 24-bit read-write CICR defines the request level for the CPM interrupts, the priority between the SCCs, the highest priority interrupt, and the vector base address. The CICR, which can be dynamically changed by the user, is cleared at reset.

23	22	21	20	19	18	17	16	15	14	13	12
SCdP		SCcP		SCbP		SCaP		IRL2	IRL1	IRL0	HP4
11	10	9	8	7	6	5	4	3	2	1	0
IHP3	HP2	HP1	HP0	VBA2	VBA1	VBA0	—				SPS

SCdP—SCCd Priority Order

These two bits define which SCC will assert its request in the SCCd priority position. The user should not program the same SCC to more than one priority position (a, b, c, or d). These bits may be changed dynamically.

00 = SCC1 will assert its request in the SCCd position.
01 = SCC2 will assert its request in the SCCd position.
10 = SCC3 will assert its request in the SCCd position.
11 = SCC4 will assert its request in the SCCd position.

SCcP—SCCc Priority Order

These two bits define which SCC will assert its request in the SCCc priority position. The user should not program the same SCC to more than one priority position (a, b, c, or d). These bits may be changed dynamically.

00 = SCC1 will assert its request in the SCCc position.
01 = SCC2 will assert its request in the SCCc position.
10 = SCC3 will assert its request in the SCCc position.
11 = SCC4 will assert its request in the SCCc position.

SCbP—SCCb Priority Order

These two bits define which SCC will assert its request in the SCCb priority position. The user should not program the same SCC to more than one priority position (a, b, c, or d). These bits may be changed dynamically.

00 = SCC1 will assert its request in the SCCb position.
01 = SCC2 will assert its request in the SCCb position.
10 = SCC3 will assert its request in the SCCb position.
11 = SCC4 will assert its request in the SCCb position.

SCaP—SCCa Priority Order

These two bits define which SCC will assert its request in the SCCa priority position. The user should not program the same SCC to more than one priority position (a, b, c, or d). These bits may be changed dynamically.

00 = SCC1 will assert its request in the SCCa position.
01 = SCC2 will assert its request in the SCCa position.
10 = SCC3 will assert its request in the SCCa position.
11 = SCC4 will assert its request in the SCCa position.

IRL2–IRL0—Interrupt Request Level

The IRL field contains the priority request level of the interrupt from the CPM that is sent to the CPU32+ core. Level 7 indicates a nonmaskable interrupt; level 0 indicates that all CPM interrupts are disabled. The IRL field, therefore, acts as a master enable for the CPM interrupts in addition to specifying the interrupt priority level. The IRL field is initialized to zero during reset to prevent the CPM from generating an interrupt until this register has been initialized. Value $4 is a good value to choose for the IRL field in most systems.

NOTES

In systems with multiple QUICCs sharing the same system bus, assign these bits to a different request level in each QUICC.

If QUICC is in slave mode (CPU32+ disabled), then the external $\overline{\text{IRQx}}$ pin corresponding to the value programmed in IRL2–IRL0 should not be used. (For example, if IRL2–IRL0 has the value $5, then $\overline{\text{IRQ5}}$ on this QUICC should not be used externally.) This also applies to the programmable interrupt timer and software watchdog in the SIM60 of the slave QUICC.

HP4–HP0—Highest Priority

These bits specify the 5-bit interrupt number of the single CPIC interrupt source that is to be advanced to the highest priority in the table. These bits may be dynamically modified. To keep the original priority order intact, simply program these bits to 11111.

VBA2–VB0—Vector Base Address

These three bits are concatenated with five bits provided by the CPIC for each specific interrupt source to form an 8-bit interrupt vector number. If these bits are not written, the uninitialized vector (value $0F) is provided for all CPM sources. These bits should not be dynamically modified.

Bits 4–1—Reserved

SPS—Spread Priority Scheme

This bit, which selects the relative SCC priority scheme, may not be changed dynamically.

0 = Grouped. The SCCs are grouped in priority at the top of the table.
1 = Spread. The SCCs are spread in priority throughout the table.

7.15.5.2 CPM INTERUPT PENDING REGISTER (CIPR). Each bit in the 32-bit read-write CIPR corresponds to a CPM interrupt source. When a CPM interrupt is received, the CPIC sets the corresponding bit in the CIPR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	171	16
PC0	SCC1	SCC2	SCC3	SCC4	PC1	TIMER1	PC2	PC3	SDMA	IDMA1	IDMA2	—	TIMER2	R–TT	—
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC4	PC5	—	TIMER3	PC6	PC7	PC8	—	TIMER4	PC9	SPI	SMC1	SMC2 / PIP	PC10	PC11	—

In a vectored interrupt scheme, the CIPR clears the CIPR bit when the vector number corresponding to the CPM interrupt source is passed during an interrupt acknowledge cycle, unless an event register exists for that interrupt source. (Event registers exist for interrupt sources that have multiple source events. For example, the SCCs have multiple events that can cause an SCC interrupt.)

In a polled interrupt scheme, the user must periodically read the CIPR. When a pending interrupt is handled, the user clears the corresponding bit in the CIPR. (However, if an event register exists, the unmasked event register bits should be cleared instead, causing the CIPR bit to be cleared.) To clear a bit in the CIPR, the user writes a one to that bit. Since the user can only clear bits in this register, bits written as zeros will not be affected. The CIPR is cleared at reset.

NOTES

The SCC CIPR bit positions are NOT changed according to the relative priority between SCCs (as determined by the SCxP and SPS bits in the CICR).

No bit in the CIPR is set if the error vector is issued.

7.15.5.3 CPM INTERRUPT MASK REGISTER (CIMR). Each bit in the 32-bit read-write CIMR corresponds to a CPM interrupt source. The user masks an interrupt by clearing the corresponding bit in the CIMR and enables an interrupt by setting the corresponding bit in the CIMR. When a masked CPM interrupt occurs, the corresponding bit in the CIPR is still set, regardless of the CIMR bit, but no interrupt request is passed to the CPU32+ core.

If a CPM interrupt source is requesting interrupt service when the user clears its CIMR bit, the request will cease. If its CIMR bit is later set by the user, a previously pending interrupt request will be processed by the CPU32+ core, according to its assigned priority. The CIMR can be read by the user at any time. The CIMR is cleared at reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	171	16
PC0	SCC1	SCC2	SCC3	SCC4	PC1	TIMER1	PC2	PC3	SDMA	IDMA1	IDMA2	—	TIMER2	R–TT	—
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC4	PC5	—	TIMER3	PC6	PC7	PC8	—	TIMER4	PC9	SPI	SMC1	SMC2 / PIP	PC10	PC11	—

NOTES

The SCC CIMR bit positions are NOT affected by the relative priority between SCCs (as determined by the SCxP and SPS bits in the CICR).

To clear bits that were set by multiple interrupt events, the user must clear all the unmasked events in the corresponding event register.

If a bit in the CIMR is masked at the same time that the corresponding CIPR bit causes an interrupt request to the IMB, then the interrupt is not processed, but the error vector is issued if the interrupt acknowledge cycle occurs with no other CPM interrupts pending. Thus, the user should always include an error vector routine, even if it just contains the RTE instruction.

The error vector cannot be masked.

7.15.5.4 CPM INTERRUPT IN-SERVICE REGISTER (CISR). Each bit in the 32-bit read-write CISR corresponds to a CPM interrupt source. In a vectored interrupt environment, the CPIC sets the CISR bit when the vector number corresponding to the CPM interrupt source is passed during an interrupt acknowledge cycle. The user's interrupt service routine must clear this bit after servicing is complete. (If an event register exists for this peripheral, its bits would normally be cleared as well.) To clear a bit in the CISR, the user writes a one to that bit. Since the user can only clear bits in this register, bits written as zeros will not be affected. The CISR is cleared at reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	171	16
PC0	SCC1	SCC2	SCC3	SCC4	PC1	TIMER1	PC2	PC3	SDMA	IDMA1	IDMA2	—	TIMER2	R–TT	—
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC4	PC5	—	TIMER3	PC6	PC7	PC8	—	TIMER4	PC9	SPI	SMC1	SMC2 / PIP	PC10	PC11	—

This register may be read by the user to determine which interrupt requests are currently in progress (i.e., the interrupt handler started execution) for each CPM interrupt source. More than one bit in the CISR may be a one if higher priority CPM interrupts are allowed to interrupt lower priority level interrupts within the same CPM interrupt level. For example, the TIMER2 interrupt routine could interrupt the handling of the TIMER3 routine, using a special nesting technique described earlier. During this time, the user would see both the TIMER3 and the TIMER2 bits simultaneously set in the CISR.

NOTES

The SCC CISR bit positions are NOT affected by the relative priority between SCCs (as determined by the SCxP and SPS bits in the CICR).

If the error vector is taken, no bit in the CISR is set. All undefined bits in the CISR return zeros when read.

The user can control the extent to which CPM interrupts may interrupt other CPM interrupts by selectively clearing the CISR. A new interrupt will be processed if it has a higher priority than the higher priority interrupt having its CISR bit set. Thus, if an interrupt routine lowers the 3-bit mask in the CPU32+ core to the CPM level minus one and also clears its CISR bit at the beginning of the interrupt routine, a lower priority interrupt can interrupt the higher one, as long as the lower priority interrupt is of higher priority than any other CISR bits that are currently set.

7.15.6 Interrupt Handler Examples

The following examples illustrate proper interrupt handling of CPM interrupts. Nesting of interrupts within the CPM interrupt level is not shown in the following examples.

7.15.6.1 EXAMPLE 1—PC6 INTERRUPT HANDLER. In this example, the CPIC hardware clears the PC6 bit in the CIPR during the interrupt acknowledge cycle. This is an example of a handler for an interrupt source without multiple events.

1. Vector to interrupt handler.
2. Handle event associated with a change in the state of the port C6 pin.
3. Clear the PC6 bit in the CISR.
4. Execute the RTE instruction.

7.15.6.2 EXAMPLE 2—SCC1 INTERRUPT HANDLER. In this example, the CIPR bit SCC1 remains set as long as one or more unmasked event bits remain in the SCCE1 register. This is an example of a handler for an interrupt source with multiple events. Note that the bit in CIPR does not need to be cleared by the handler, but the bit in CISR does need to be cleared.

1. Vector to interrupt handler.
2. Immediately read the SCC1 event register (SCCE1) into a temporary location.

3. Decide which events in the SCCE1 will be handled in this handler and clear those bits as soon as possible. (SCCE bits are cleared by writing ones.)
4. Handle events in the SCC1 Rx or Tx BD tables.
5. Clear the SCC1 bit in the CISR.
6. Execute the RTE instruction. If any unmasked bits in SCCE1 remain at this time (either not cleared by the software or set by the QUICC during the execution of this handler), this interrupt source will be made pending again immediately following the RTE instruction.

SECTION 8
SCAN CHAIN TEST ACCESS PORT

The QUICC provides a dedicated user-accessible test access port (TAP) that is JTAG compatible.

The QUICC TAP contains one additional signal not available with the MC68340 TAP—the test reset ($\overline{\text{TRST}}$) signal. This signal provides an asynchronous reset to the TAP.

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, implemented utilizing static logic design, is independent of the device system logic. The QUICC implementation provides the capability to:

1. Perform boundary scan operations to test circuit-board electrical continuity.
2. Bypass the QUICC for a given circuit-board test by effectively reducing the boundary scan register to a single cell.
3. Sample the QUICC system pins during operation and transparently shift out the result in the boundary scan register.
4. Disable the output drive to pins during circuit-board testing.

NOTE

Certain precautions must be observed to ensure that the IEEE 1149.-like test logic does not interfere with nontest operation. See 8.6 Non-Scan Chain Operation for details.

In addition to the scan-test logic, the QUICC contains a signal that can be used to three-state all QUICC output signals. This signal, called three-state ($\overline{\text{TRIS}}$), is sampled during system reset when the QUICC is not in slave mode.

8.1 OVERVIEW

An overview of the QUICC scan chain implementation is shown in Figure 8-1. The QUICC implementation includes a TAP controller, a 3-bit instruction register, and two test registers (a 1-bit bypass register and a 196-bit boundary scan register). This implementation includes a dedicated TAP consisting of the following signals:

- TCK—a test clock input to synchronize the test logic.
- TMS—a test mode select input (with an internal pullup 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 pullup resistor) that is sampled on the rising edge of TCK.

- TDO—a three-stateable test data output that is actively driven in the shift-IR and shift-DR controller states. TDO changes on the falling edge of TCK.
- $\overline{\text{TRST}}$—an asynchronous reset with an internal pullup resistor that provides initialization of the TAP controller and other logic required by the standard.

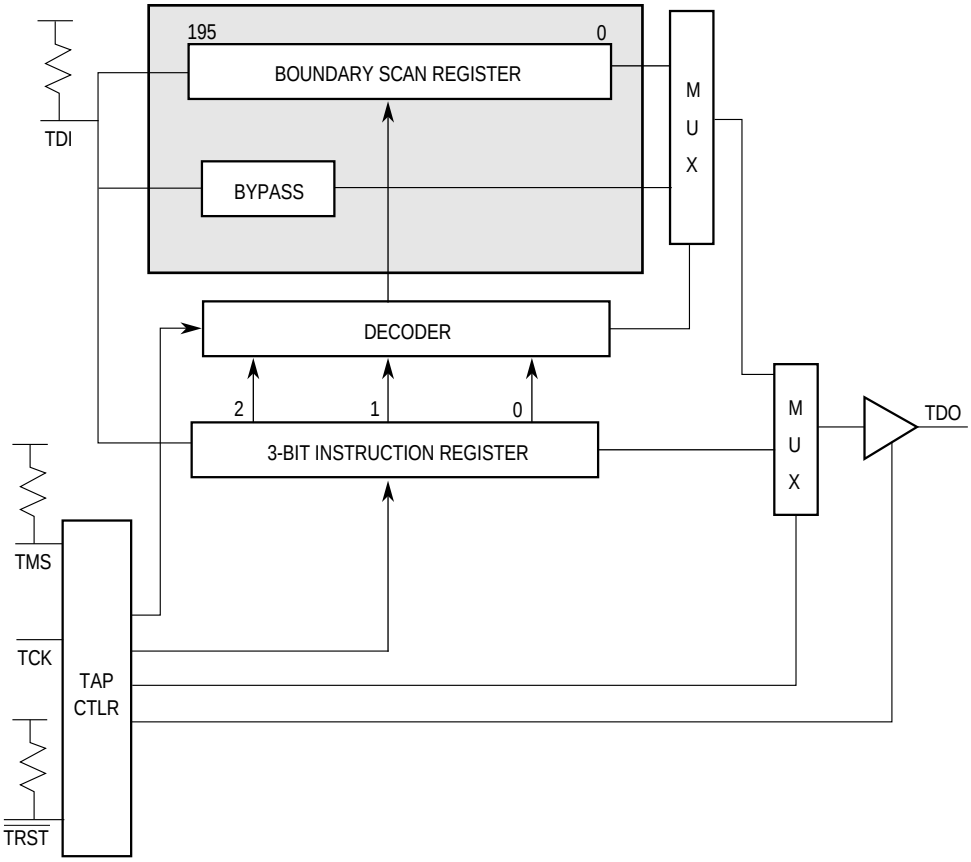


Figure 8-1. Test Logic Block Diagram

8.2 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 state machine is shown in Figure 8-2. The value shown adjacent to each arc represents the value of the TMS signal sampled on the rising edge of the TCK signal.

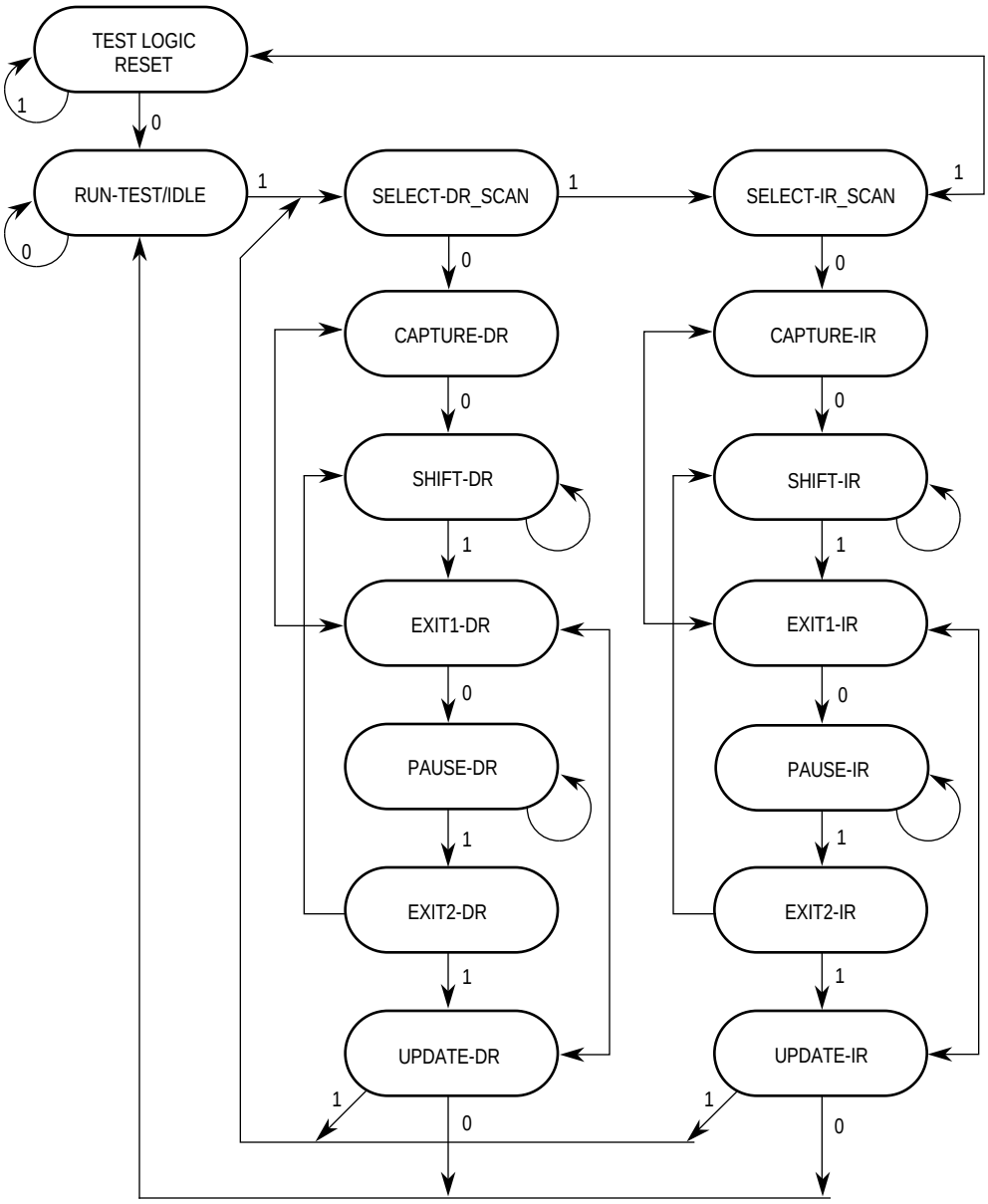


Figure 8-2. TAP Controller State Machine

8.3 BOUNDARY SCAN REGISTER

The QUICC scan chain implementation has a 196-bit boundary scan register. This register contains bits for all device signal and clock pins and associated control signals. The XTAL, and XFC pins are associated with analog signals and are not included in the boundary scan register.

All QUICC bidirectional pins have a single register bit in the boundary scan register for pin data and are controlled by an associated control bit in this register. Twenty-five bits in the boundary scan register define the output enable signal for associated groups of bidirectional and three-stateable pins. The control bits and their bit positions are listed in Table 8-1.

Table 8-1. Boundary Scan Control Bits

Name	Bit Number	Name	Bit Number	Name	Bit Number
g1.cntl	2	g10.cntl	32	pbhl.ctl	155
g2.cntl	5	add.cntl	59	pblh.ctl	159
g3.cntl	8	addh.cntl	83	pchh.ctl	170
g4.cntl	13	g11.cnt	89	pchl.ctl	174
g5.cntl	16	db.ctl	110	pclh.ctl	179
g6.cntl	18	pahh.ctl	131	g12.cntl	188
g7.cntl	23	pahl.ctl	136	g13.cntl	191
g8.cntl	27	palh.ctl	141	g14.cntl	194
g9.cntl	29	pbhh.ctl	150		

The boundary scan bit definitions are listed in Table 8-2.

The first column in the table defines the bit's ordinal position in the boundary scan register. The shift register cell nearest TDO (i.e., first to be shifted out) is defined as bit 0; the last bit to be shifted out is 195.

The second column references one of the four QUICC cell types depicted in Figure 8-3 through , which describe the cell structure for each type.

The third column lists the pin name for all pin-related cells or defines the name of bidirectional control register bits.

The fourth column lists the pin type for convenience, where TS-Output indicates a three-stateable output pin, I/O indicates a bidirectional pin, and OD-I/O denotes an open-drain bidirectional pin.

The last column indicates the associated boundary scan register control bit for bidirectional, three-state, and open-drain output pins.

Bidirectional pins include a single scan cell for data (IO.Cell) as depicted in Figure 8-6. These bits are controlled by the cell shown in Figure 8-5. The value of the control bit determines whether the bidirectional pin is an input or an output. One or more bidirectional data cells can be serially connected to a control cell as shown in Figure 8-7. Note that, when sampling the bidirectional data cells, the cell data can be interpreted only after examining the IO control cell to determine pin direction, and also note that the control cell captures the value of the following cell.

Table 8-2. Boundary Scan Bit Definition

Bit Num	Cell Type	Pin/Cell Name	Pin Type	Output CTLCell	Bit Num	Cell Type	Pin/Cell Name	Pin Type	Output CTLCell
0	I/O	reseth	I/O	g1.ctl	39	Output	Cs0	Output	—
1	I/O	Bkpt	I/O	g1.ctl	40	Output	Cs1	Output	—
2	IO.ctl	g1.ctl	—	—	41	Output	Cs2	Output	—
3	I/O	Irq6	I/O	g1.ctl	42	Output	Cs3	Output	—
4	I/O	Irq4	I/O	g1.ctl	43	Output	Cs4	Output	—
5	IO.ctl	g2.ctl	—	—	44	Output	Cs5	Output	—
6	I/O	Bgack	TS-O	g2.ctl	45	Output	Cs6	Output	—
7	I/O	Bg	I/O	g3.ctl	46	Output	Cs7	Output	—
8	IO.ctl	g3.ctl	—	—	47	Input	Irq7	Input	—
9	I/O	Br	$	g4.ctl	48	Input	Tris	Input	—
10	I/O	nc1	I/O	g4.ctl	49	I/O	Add0	I/O	add.ctl
11	I/O	Ifetch	I/O	g5.ctl	50	I/O	Add1	I/O	add.ctl
12	Output	oe	Output	—	51	I/O	Add2	I/O	add.ctl
13	IO.ctl	g4.ctl	—	—	52	I/O	Add3	I/O	add.ctl
14	I/O	bclro	I/O	g5.ctl	53	I/O	Add4	I/O	add.ctl
15	I/O	nc2	I/O	g5.ctl	54	I/O	Add5	I/O	add.ctl
16	IO.ctl	g5.ctl	—	—	55	I/O	Add6	I/O	add.ctl
17	I/O	Ipipe1	I/O	g5.ctl	56	I/O	Add7	I/O	add.ctl
18	IO.ctl	g6.ctl	—	—	57	I/O	Add8	I/O	add.ctl
19	I/O	As	I/O	g6.ctl	58	I/O	Add9	I/O	add.ctl
20	Output	pipe0I	Output	—	59	IO.ctl	add.ctl	—	—
21	I/O	Prty0	I/O	g7.ctl	60	I/O	Add10	I/O	add.ctl
22	I/O	Prty1	I/O	g7.ctl	61	I/O	Add11	I/O	add.ctl
23	IO.ctl	g7.ctl	—	—	62	I/O	Add12	I/O	add.ctl
24	I/O	Prty2	I/O	g7.ctl	63	I/O	Add13	I/O	add.ctl
25	I/O	Prty3	I/O	g7.ctl	64	I/O	Add14	I/O	add.ctl
26	I/O	Dsack1	TS-O	g8.ctl	65	I/O	Add15	I/O	add.ctl
27	IO.ctl	g8.ctl	—	—	66	I/O	Add16	I/O	add.ctl
28	I/O	Dsack0	TS-O	g8.ctl	67	I/O	Add17	I/O	add.ctl
29	IO.ctl	g9.ctl	—	—	68	I/O	Add18	I/O	add.ctl
30	I/O	nc3	I/O	g9.ctl	69	I/O	Add19	I/O	add.ctl
31	I/O	R/w	I/O	g9.ctl	70	I/O	Add20	I/O	add.ctl
32	IO.ctl	g10.ctl	—	—	71	I/O	Add21	I/O	add.ctl
33	I/O	Ds	I/O	g10.ctl	72	I/O	Add22	I/O	add.ctl
34	I/O	freeze	I/O	g10.ctl	73	I/O	Add23	I/O	add.ctl
35	Output	Cas0	Output	—	74	I/O	Add24	I/O	add.ctl
36	Output	Cas1	Output	—	75	I/O	Add25	I/O	add.ctl
37	Output	Cas2	Output	—	76	I/O	Add26	I/O	add.ctl
38	Output	Cas3	Output	—	77	I/O	Add27	I/O	add.ctl

Table 8-2. Boundary Scan Bit Definition

78	I/O	nc4	I/O	add.ctl	118	I/O	Data9	I/O	Db.ctl
79	Input	Modclk1	Input	—	119	I/O	Data8	I/O	Db.ctl
80	Input	Modclk0	Input	—	120	I/O	Data7	I/O	Db.ctl
81	I/O	Add28	I/O	Addh.ctl	121	I/O	Data6	I/O	Db.ctl
82	I/O	Add29	I/O	Addh.ctl	122	I/O	Data5	I/O	Db.ctl
83	IO.ctl	Addh.ctl	—	—	123	I/O	Data4	I/O	Db.ctl
84	I/O	Add30	I/O	Addh.ctl	124	I/O	Data3	I/O	Db.ctl
85	I/O	Add31	I/O	Addh.ctl	125	I/O	Data2	I/O	Db.ctl
86	I/O	Siz0	I/O	g11.ctl	126	I/O	Data1	I/O	Db.ctl
87	I/O	Siz1	I/O	g11.ctl	127	I/O	Data0	I/O	Db.ctl
88	I/O	Fc0	I/O	g11.ctl	128	I/O	pa15	I/O	pahh.ctl
89	IO.ctl	g11.ctl	—	—	129	I/O	pa14	I/O	pahh.ctl
90	I/O	Fc1	I/O	g11.ctl	130	I/O	pa13	I/O	pahh.ctl
91	I/O	Fc2	I/O	g11.ctl	131	IO.ctl	pahh.ctl	—	—
92	I/O	Fc3	I/O	g11.ctl	132	I/O	pa12	I/O	pahh.ctl
93	I/O	Data31	I/O	Db.ctl	133	I/O	pa11	I/O	pahh.ctl
94	I/O	Data30	I/O	Db.ctl	134	I/O	pa10	I/O	pahh.ctl
95	I/O	Data29	I/O	Db.ctl	135	I/O	pa9	I/O	pahl.ctl
96	I/O	Data28	I/O	Db.ctl	136	IO.ctl	pahl.ctl	—	—
97	I/O	Data27	I/O	Db.ctl	137	I/O	pa8	I/O	pahl.ctl
98	I/O	Data26	I/O	Db.ctl	138	I/O	pa7	I/O	pahl.ctl
99	I/O	Data25	I/O	Db.ctl	139	I/O	pa6	I/O	pahl.ctl
100	I/O	Data24	I/O	Db.ctl	140	I/O	pa5	I/O	pahl.ctl
101	I/O	Data23	I/O	Db.ctl	141	IO.ctl	palh.ctl	—	—
102	I/O	Data22	I/O	Db.ctl	142	I/O	pa4	I/O	palh.ctl
103	I/O	Data21	I/O	Db.ctl	143	I/O	pa3	I/O	palh.ctl
104	I/O	Data20	I/O	Db.ctl	144	I/O	pa2	I/O	palh.ctl
105	Output	Clko0	Output	—	145	I/O	pa1	I/O	palh.ctl
106	Output	Clko1	Output	—	146	I/O	pa0	I/O	palh.ctl
107	I/O	Data19	I/O	Db.ctl	147	I/O	pb17	I/O	pbhh.ctl
108	I/O	Data18	I/O	Db.ctl	148	I/O	pb16	I/O	pbhh.ctl
109	I/O	Data17	I/O	Db.ctl	149	I/O	pb15	I/O	pbhh.ctl
110	IO.ctl	Db.ctl	—	—	150	IO.ctl	pbhh.ctl	—	—
111	I/O	Data16	I/O	Db.ctl	151	I/O	pb14	I/O	pbhh.ctl
112	I/O	Data15	I/O	Db.ctl	152	I/O	pb13	I/O	pbhh.ctl
113	I/O	Data14	I/O	Db.ctl	153	I/O	pb12	I/O	pbhh.ctl
114	I/O	Data13	I/O	Db.ctl	154	I/O	pb11	I/O	pbhl.ctl
115	I/O	Data12	I/O	Db.ctl	155	IO.ctl	pbhl.ctl	—	—
116	I/O	Data11	I/O	Db.ctl	156	I/O	pb10	I/O	pbhl.ctl
117	I/O	Data10	I/O	Db.ctl	157	I/O	pb9	I/O	pbhl.ctl

Table 8-2. Boundary Scan Bit Definition

158	I/O	pb8	I/O	pbhl.ctl	177	I/O	pc4	I/O	pchl.ctl
159	I/O	pb7	I/O	pbhl.ctl	178	I/O	pc3	I/O	pclh.ctl
160	IO.ctl	pblh.ctl	—	—	179	IO.ctl	pclh.ctl	—	—
161	I/O	pb6	I/O	pblh.ctl	180	I/O	pc2	I/O	pclh.ctl
162	I/O	pb5	I/O	pblh.ctl	181	I/O	pc1	I/O	pclh.ctl
163	I/O	pb4	I/O	pblh.ctl	182	I/O	pc0	I/O	pclh.ctl
164	I/O	pb3	I/O	pblh.ctl	183	Input	$\overline{\text{Irq2}}$	Input	—
165	I/O	pb2	I/O	pblh.ctl	184	Input	$\overline{\text{Irq3}}$	Input	—
166	I/O	pb1	I/O	pblh.ctl	185	Input	$\overline{\text{Irq5}}$	Input	—
167	I/O	pb0	I/O	pblh.ctl	186	I/O	$\overline{\text{Irq1}}$	I/O	g12.ctl
168	I/O	pc11	I/O	pchh.ctl	187	I/O	$\overline{\text{Berr}}$	$	g12.ctl
169	IO.ctl	pchh.ctl	—	—	188	IO.ctl	g12.ctl	—	—
170	I/O	pc10	I/O	pchh.ctl	189	I/O	$\overline{\text{Halt}}$	$	g13.ctl
171	I/O	pc9	I/O	pchh.ctl	190	I/O	$\overline{\text{Resets}}$	$	g13.ctl
172	I/O	pc8	I/O	pchh.ctl	191	IO.ctl	g13.ctl	—	—
173	I/O	pc7	I/O	pchl.ctl	192	I/O	$\overline{\text{Rmc}}$	I/O	g14.ctl
174	IO.ctl	pchl.ctl	—	—	193	I/O	$\overline{\text{Avec}}$	TS-O	g14.ctl
175	I/O	pc6	I/O	pchl.ctl	194	IO.ctl	g14.ctl	—	—
176	I/O	pc5	I/O	pchl.ctl	195	Output	$\overline{\text{Perr}}$	$	g14.ctl

$=These pins are implemented differently in normal mode and in JTAG mode—open drain in normal mode, TS-O in JTAG mode.

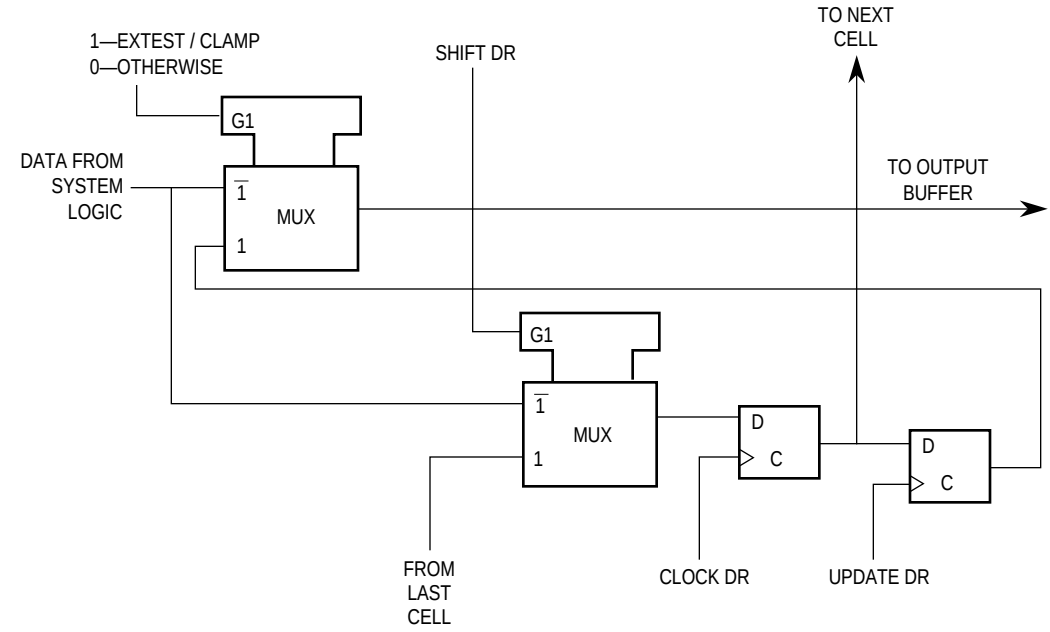


Figure 8-3. Output Latch Cell (O.Latch)

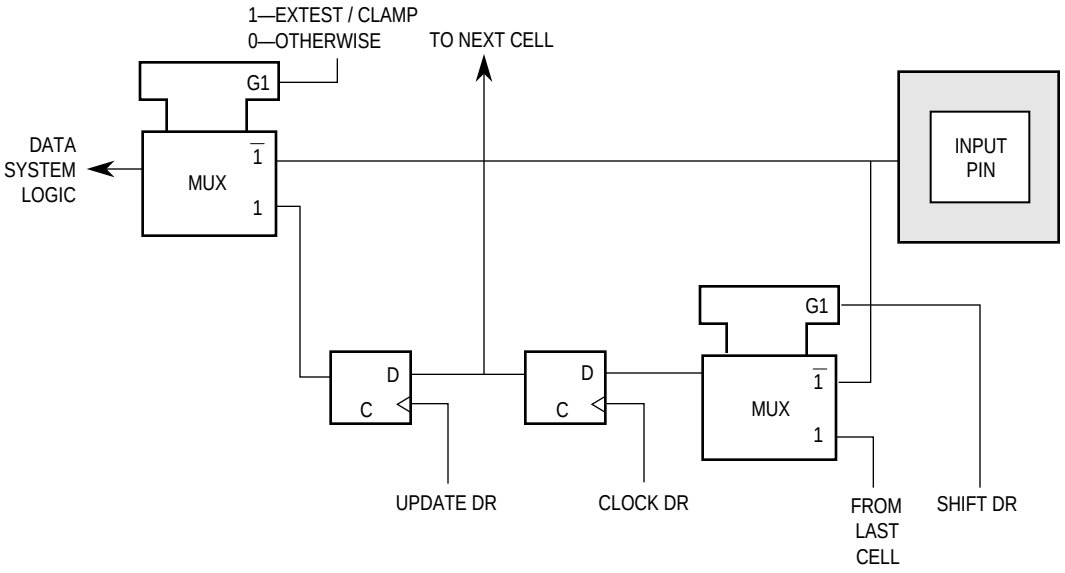


Figure 8-4. Input Pin Cell (I.Pin)

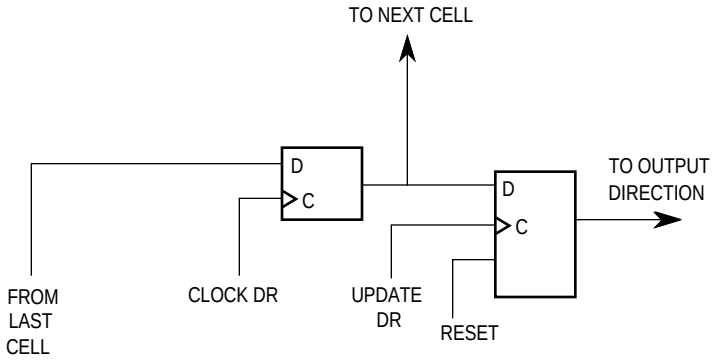


Figure 8-5. Control Cell (IO.Ctl)

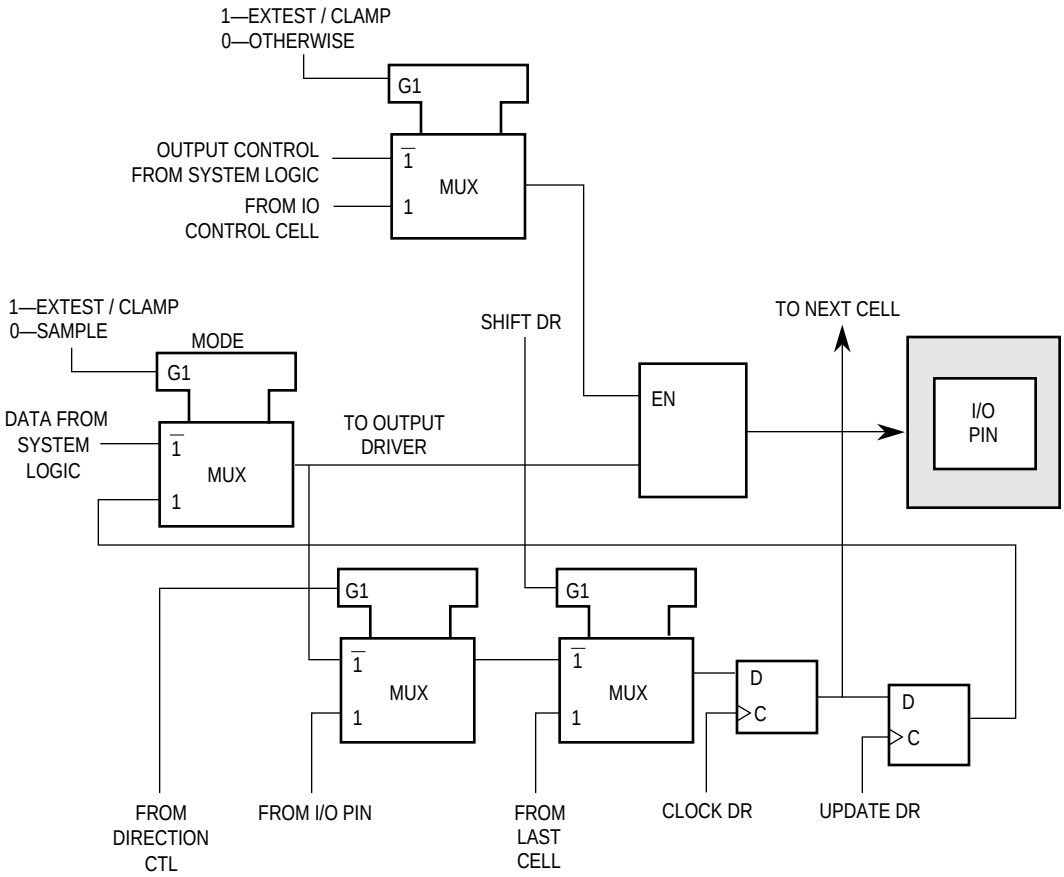
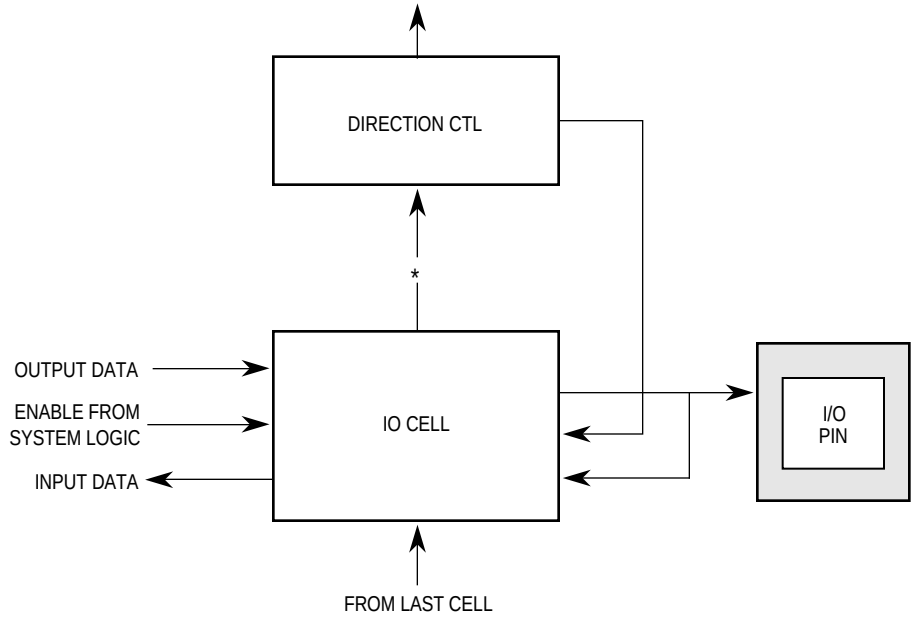


Figure 8-6. Bidirectional Data Cell (IO.Cell)



NOTE: More than one IO.Cell could be serially connected and controlled by a single IO.Ctl.

Figure 8-7. General Arrangement for Bidirectional Pins

8.4 INSTRUCTION REGISTER

The QUICC JTAG implementation includes the public instructions (EXTEST, SAMPLE/ PRELOAD, and BYPASS), and also supports the CLAMP instruction. One additional public instruction (HI-Z) provides the capability for disabling all device output drivers. The QUICC includes a 3-bit instruction register without parity consisting of a shift register with three parallel outputs. Data is transferred from the shift register to the parallel outputs during the update-IR controller state. The three bits are used to decode the five unique instructions listed in Table 8-3.

Table 8-3. Instruction Decoding

Code			Instruction
B2	B1	B0	
0	0	0	EXTEST
0	0	1	SAMPLE/PRELOAD
X	1	X	BYPASS
1	0	0	HI-Z
1	0	1	CLAMP and BYPASS

The parallel output of the instruction register is reset to all ones in the test-logic-reset controller state. Note 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.

8.4.1 EXTEST

The external test (EXTEST) instruction selects the 196-bit boundary scan register. EXTEST also asserts internal reset for the QUICC system logic to force a predictable benign internal state while performing external boundary scan operations.

By using the TAP, the register is capable of a) scanning user-defined values into the output buffers, b) capturing values presented to input pins, c) controlling the direction of bidirectional pins, and d) controlling the output drive of three-stateable output pins. For more details on the function and use of EXTEST, refer to the scan chaindocument.

8.4.2 SAMPLE/PRELOAD

The SAMPLE/PRELOAD instruction initializes the boundary scan register output cells prior to selection of EXTEST. This initialization ensures that known data will appear on the outputs when entering the EXTEST instruction. The SAMPLE/PRELOAD instruction also provides a means to obtain a snapshot of system data and control signals. In the case of the QUICC, this functionality is not supported.

NOTE

Since there is no internal synchronization between the scan chain clock (TCK) and the system clock (CLKO1), the user must

provide some form of external synchronization to achieve meaningful results.

Note that in the QUICC, the SAMPLE instruction is not functional.

8.4.3 BYPASS

The BYPASS instruction selects the single-bit bypass register as shown in Figure 8-8. This creates a shift register path from TDI to the bypass register and, finally, to TDO, circumventing the 196-bit boundary scan register. This instruction is used to enhance test efficiency when a component other than the QUICC becomes the device under test.

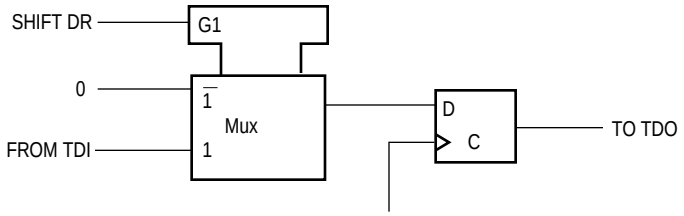


Figure 8-8. Bypass Register

When the bypass register is selected by the current instruction, the shift register stage is set to a logic zero on the rising edge of TCK in the capture-DR controller state. Therefore, the first bit to be shifted out after selecting the bypass register will always be a logic zero.

8.4.4 CLAMP

The CLAMP instruction selects the single-bit bypass register as shown in Figure 8-8, and the state of all signals driven from system output pins is completely defined by the data previously shifted into the boundary scan register (for example, using the SAMPLE/PRELOAD instruction).

8.4.5 HI-Z

The HI-Z instruction is provided as a manufacturer's optional public instruction to prevent having to backdrive the output pins during circuit-board testing. When HI-Z is invoked, all output drivers, including the two-state drivers, are turned off (i.e., high impedance). The instruction selects the bypass register.

NOTE

On the QUICC, the $\overline{\text{TRIS}}$ pin may also be used during system reset to perform the same function.

8.5 QUICC 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 QUICC output drivers are enabled into actively driven networks.

The QUICC includes on-chip circuitry to detect the initial application of power to the device. Power-on reset (POR), the output of this circuitry, is used to reset both the system and scan chainlogic. The purpose for applying POR to the scan chaincircuitry is to avoid the possibility of bus contention during power-on. The time required to complete device power-on is power-supply dependent. However, the scan chainTAP controller remains in the test-logic-reset state while POR is asserted. The TAP controller does not respond to user commands until POR is negated.

The QUICC features a low-power stop mode, which is invoked using a CPU instruction called LPSTOP. The interaction of the scan chaininterface with low-power stop mode is as follows:

1. The TAP controller must be in the test-logic-reset state to either enter or remain in the low-power stop mode. Leaving the TAP controller in the test-logic-reset state negates the ability to achieve low-power, but does not otherwise affect device functionality.
2. The TCK input is not blocked in low-power stop mode. To consume minimal power, the TCK input should be externally connected to V_{CC} or ground.
3. The TMS and TDI pins include on-chip pullup resistors. In low-power stop mode, these two pins should remain either unconnected or connected to V_{CC} to achieve minimal power consumption.

8.6 NON-SCAN CHAIN OPERATION

In non-scan chain operation, there are two constraints. First, the TCK input does not include an internal pullup resistor and should not be left unconnected to preclude mid-level inputs. The second constraint is to ensure that the scan chaintest logic is kept transparent to the system logic by forcing TAP into the test-logic-reset controller state, using either of two methods. During power-up, POR forces the TAP controller into this state. After power-up is concluded, TMS must be sampled as a logic one for five consecutive TCK rising edges. If TMS either remains unconnected or is connected to V_{CC}, then the TAP controller cannot leave the test-logic-reset state, regardless of the state of TCK.

SECTION 9
APPLICATIONS

9.1 MINIMUM SYSTEM CONFIGURATION

This section describes a basic minimum system configuration for the QUICC. It discusses the hardware and software issues of configurating the QUICC to support a basic system with a variety of ROM and RAM types.

9.1.1 QUICC Hardware Configuration

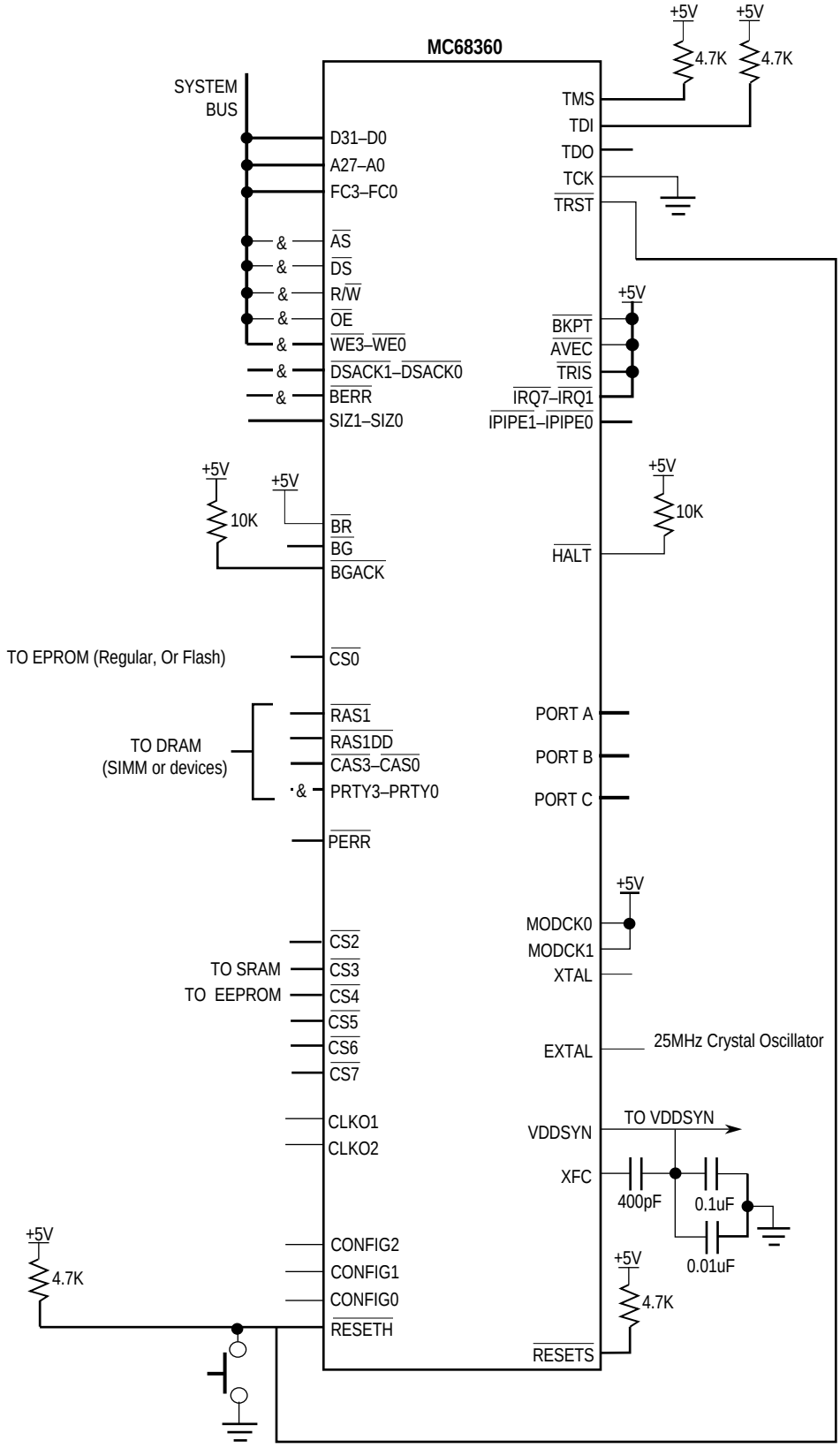
These paragraphs discuss the hardware issues relating to the configuration of the QUICC. Reference Figure 9-1 during these discussions. This configuration assumes a 32-bit data bus. Comments about the changes required for a 16-bit data bus solution are given at the end of the discussion.

9.1.1.1 QUICC BASIC ACCESSES. The basic connection is made through the data and address bus. All 32 data lines are used in this application, although not all memories are a full 32-bits wide.

Twenty-eight address lines are used, giving a 256-Mbytes address capability. It is possible to use all 32 address lines, but the QUICC would then lose its write enable lines ($\overline{WE3}$–$\overline{WE0}$). Since these lines are very useful in memory interfaces, they are used in the application.

The function code (FC3–FC0) and data strobe ($\overline{DS}$) lines are shown routed to the system bus, although the memories in this application do not require them.

Other pins not directly needed are $\overline{DSACKx}$, $\overline{BERR}$, SIZx, $\overline{PERR}$, $\overline{IPIPEx}$, and a number of chip selects. The $\overline{DSACKx}$ lines are not required because the on-chip wait state generator is used. The $\overline{BERR}$ pin is not needed because all bus errors are generated by internal monitor logic. The SIZx pins are not needed because the memory controller has programmable port sizes for each memory bank. The $\overline{PERR}$ pin is not needed because parity errors generate bus errors. The $\overline{IPIPEx}$ pins are only needed for emulator support. The additional chip selects can be used to add additional peripherals as required.



LEGEND:
& = Pullups recommended (<= 10K)

Figure 9-1. MC68360 Minimum System Configuration

9.1.1.2 CLOCKING STRATEGY. In this application, the system clock is generated from a 32.768-kHz crystal into the QUICC. The QUICC's internal phase-locked loop (PLL) then multiplies the frequency up to 25 MHz, and outputs 25 MHz on CLKO1 and 50 MHz on CLKO2. Neither CLKO pin is required for the application. It is recommended that the CLKO outputs be disabled in software to save power.

The use of a 32.768-kHz crystal is not a requirement in the application. A 4-MHz crystal or a 25-MHz external oscillator could have been used, if desired.

The QUICC clocking section allows for the clock oscillator to be kept running through the VDDSYN pin in a power-down situation. This section does not address low-power issues, however.

9.1.1.3 RESETTING THE QUICC. If a QUICC is configured to provide the global chip select, it will also provide an internal power-on reset generation. Thus, the reset support function is very simple. If a pushbutton switch is needed, it can be connected by an open-drain buffer to the hard reset ($\overline{\text{RESETH}}$) pin, once debounced. The soft reset ($\overline{\text{RESETS}}$) pin is not used in this design except to indicate that an internal QUICC soft reset is in progress.

9.1.1.4 INTERRUPTS. External interrupts may be brought into the QUICC through either the $\overline{\text{IRQx}}$ pins or parallel I/O pins. This design shows no external interrupts (the $\overline{\text{IRQ7}}$–$\overline{\text{IRQ1}}$ pins are pulled high), but this could be easily changed if desired. Without any external interrupts requiring autovector capability, the $\overline{\text{AVEC}}$ pin is also pulled high.

Internal interrupts from the QUICC may be generated in the SIM60 or the CPM. No additional hardware is required.

9.1.1.5 BUS ARBITRATION. This design assumes that no alternate bus masters exist in the system. Thus, $\overline{\text{BR}}$ is pulled high, and $\overline{\text{BGACK}}$ is not connected, but pulled high since it is an open-drain signal.

9.1.1.6 BREAKPOINT GENERATION. The QUICC can be used to generate a hardware breakpoint signal. The result of a breakpoint (either internally generated using the breakpoint address register or externally generated using the $\overline{\text{BKPT}}$ pin) is a CPU32+ breakpoint cycle. In this application, the $\overline{\text{BKPT}}$ pin is tied high and is not used.

9.1.1.7 BUS MONITOR FUNCTION. The QUICC can be programmed to monitor the bus for bus cycles that are not properly terminated. If $\overline{\text{AS}}$ is asserted but not negated, the cycle will terminate with the $\overline{\text{BERR}}$ pin being asserted.

9.1.1.8 SPURIOUS INTERRUPT MONITOR. The QUICC will watch for spurious interrupt cycles on the levels that it supports internally. If such a condition occurs, $\overline{\text{BERR}}$ will be asserted by the QUICC.

9.1.1.9 SOFTWARE WATCHDOG. If desired, the QUICC software watchdog can be used to generate a level 7 interrupt or a system reset. In this application, the software watchdog is configured in software to generate a reset. No additional hardware is required.

9.1.1.10 DOUBLE BUS FAULT. The QUICC double bus fault monitor may be used in the design. No additional hardware is required.

9.1.1.11 JTAG AND THREE-STATE. The QUICC provides a JTAG test access port, commonly known as JTAG. This interface uses five pins: TMS, TDI, TDO, TCK, and $\overline{\text{TRST}}$. TMS and TDI are left unconnected because they have internal pullups. The JTAG port is disabled in this application; however, the capability could be easily added.

When the QUICC is in master mode, it provides a pin ($\overline{\text{TRIS}}$) that allows all outputs on the device to be three-stated. This pin is simply pulled high in this application.

9.1.1.12 QUICC SERIAL PORTS. The functions on QUICC parallel I/O ports A, B, and C may be used as desired in this application, and have no bearing on the design as shown. However, any unused parallel I/O pins should be configured as outputs so they are not left floating.

9.1.2 Memory Interfaces

In this application, a number of memory arrays have been developed for EPROM, flash EPROM, SRAM, EEPROM, and DRAM. Each memory interface can be attached to the system bus as desired.

One issue not discussed is the decision of whether external buffers are needed on the system bus. This issue depends on the number of memory arrays used in the design and the layout (i.e., capacitance) of the system bus. This issue is left to the user for his particular design.

Another issue left to the user is the number of wait states used with each memory system. This depends on the memory speed, whether external buffers are used, and the loading on the system bus pins. (The QUICC provides capacitance de-rating figures to calculate the effect of additional or less capacitance on the AC Timing Specifications.)

9.1.2.1 QUICC MEMORY INTERFACE PINS. In this design, a number of QUICC pins are made available to the memory arrays (see Figure 9-1). Eight chip select or $\overline{\text{RAS}}$ pins are available in the system. In this design, $\overline{\text{CS0}}$ is used for any of the EPROM arrays since this is the global (boot) chip select. $\overline{\text{RAS1}}$ is used for the DRAM arrays because of its double-drive capability. $\overline{\text{CS2}}$/$\overline{\text{RAS2}}$ is not used in the design and is available for other purposes, such as a second DRAM bank. $\overline{\text{CS3}}$ is for SRAM; $\overline{\text{CS4}}$ is for EEPROM. $\overline{\text{CS5}}$, $\overline{\text{CS6}}$, and $\overline{\text{CS7}}$ are unused.

Parity may be supported for both SRAM and DRAM arrays using the 4-byte parity lines PRTY3–PRTY0. In this design, it is shown with only a DRAM. The QUICC is configured in software to generate a bus error when a parity error occurs.

The QUICC provides the address multiplexing for the DRAM arrays internally, which is configured in software. Therefore, the address multiplex pin is not needed, and it can be used as its other function—an output enable ($\overline{\text{OE}}$) pin. The DRAM arrays require the four $\overline{\text{CAS3}}$–$\overline{\text{CAS0}}$ pins provided by the QUICC. The QUICC also provides four write enable ($\overline{\text{WE}}$) pins to select the correct byte during write operations.

9.1.2.2 REGULAR EPROM. Figure 9-2 shows the glueless interface to standard EPROM in the system. In this case, an 8-bit boot EPROM is used. All accesses to the EPROM, even word or long-word length, will be partitioned into multiple byte accesses to the EPROM.

The fact that the CONFIG2–CONFIG0 pins are pulled high causes these pins to default to the 111 condition, selecting the CPU32+ core to be enabled, the $\overline{CS0}$ pin to select a byte port size, and the MBAR to be located in its normal address location. This is the most commonly used configuration for the CONFIGx pins. The pullups are used to allow for some of the alternate functions of the CONFIGx pins to be used in later applications.

During initialization, $\overline{CS0}$ should be programmed to respond to a 128-Kbyte area in this design.

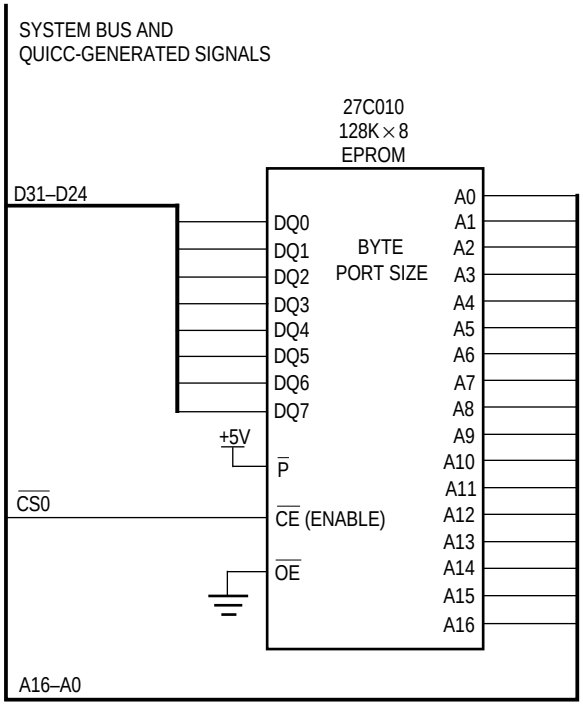


Figure 9-2. Glueless Interface to Standard EPROM

9.1.2.3 FLASH EPROM. Figure 9-3 shows the glueless interface to a flash EPROM device. It is identical to the regular EPROM except that it allows for write operations as well. This design assumes that the write operations are $\overline{CE}$ controlled, rather than $\overline{WE}$ controlled. Most flash EPROM manufacturers now support this alternative timing method.

The fact that the CONFIGx pins are pulled high causes these pins to default to the 111 condition, selecting the CPU32+ core to be enabled, the $\overline{CS0}$ pin to select a byte port size, and the MBAR to be located in its normal address location. This allows a byte-sized EPROM to be used without any external glue logic.

During initialization, $\overline{CS0}$ should be programmed to respond to a 256-Kbyte area in this design.

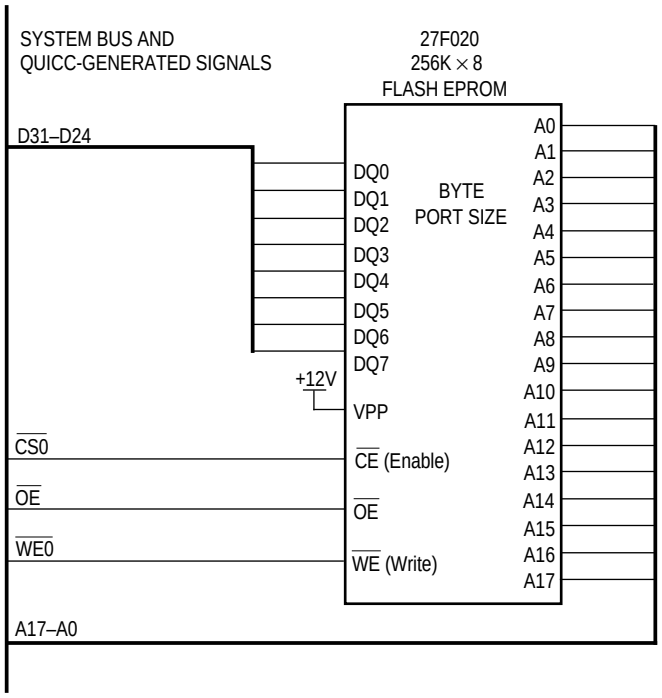


Figure 9-3. Glueless Interface to Flash EPROM

9.1.2.4 SRAM. Figure 9-4 shows the glueless interface to an SRAM. In this case, a full 32-bit-wide SRAM bank is assumed, but an 8- or 16-bit-wide bank would also be possible at the expense of performance. The $\overline{CS3}$ pin should be programmed to respond to a 128-Kbyte area in this design.

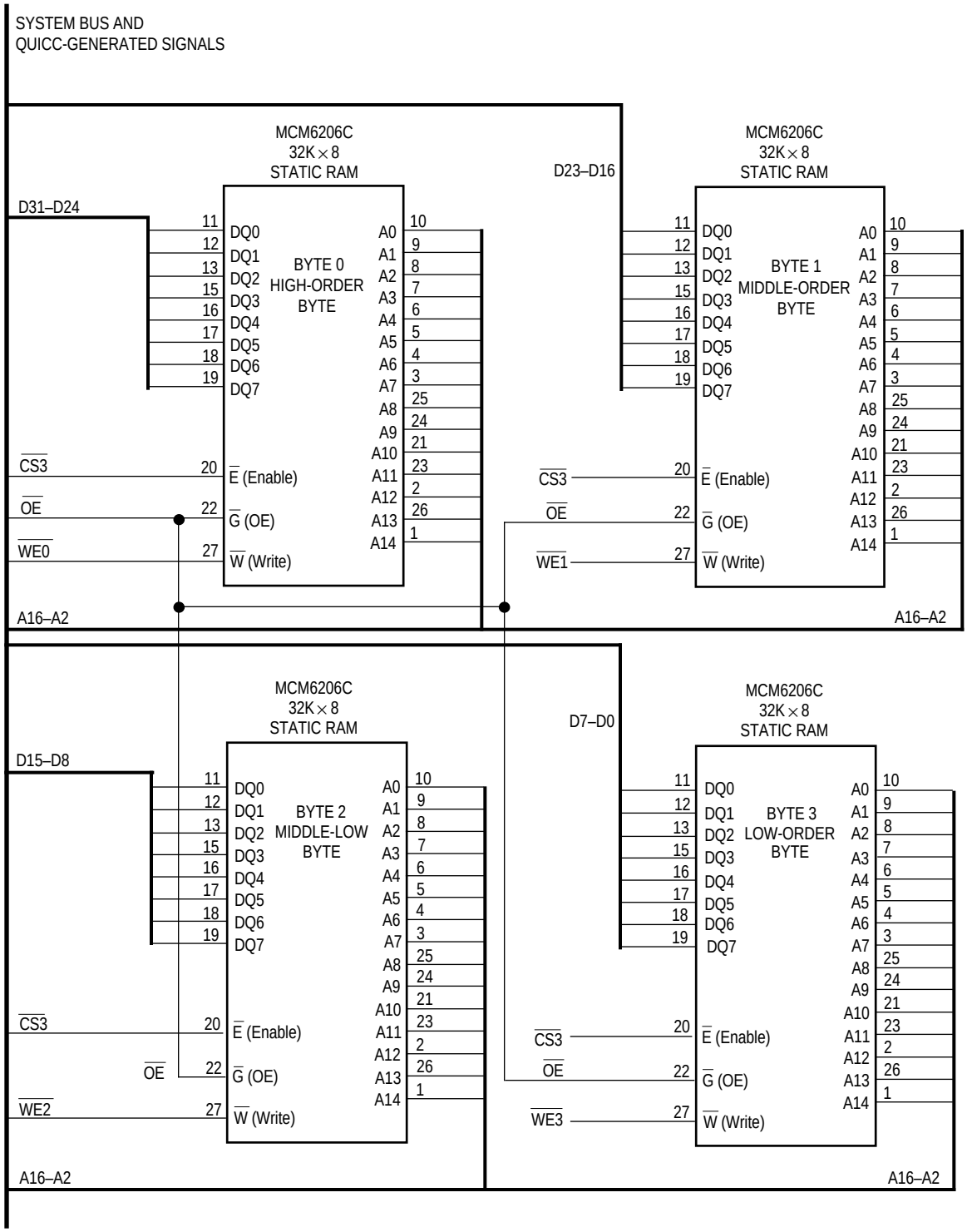


Figure 9-4. Glueless Interface to SRAM

9.1.2.5 EEPROM. Figure 9-5 shows the glueless interface to an EEPROM device to give a small amount of nonvolatile storage. In this case, a byte-wide EEPROM bank is defined to minimize cost. If the port size of the chip select is selected to be 8-bits, then each byte of the

EEPROM may be accessed in succession. The $\overline{CS4}$ pin should be programmed to respond to an 8-Kbyte area in this design.

Only one byte should be written at a time. After a write is made, software is responsible for waiting the appropriate time (e.g., 10 ms) or for performing data polling to see if the newly written data byte is correct.

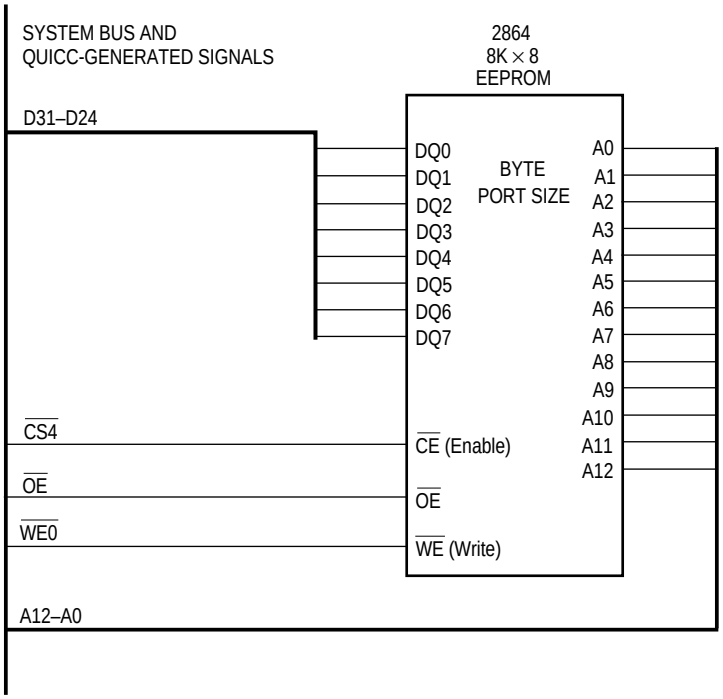


Figure 9-5. Glueless Interface to EEPROM

9.1.2.6 DRAM SIMM. Figure 9-6 shows the glueless interface to an MCM36100S DRAM single in-line memory module (SIMM). The $\overline{RAS1}$ line should be programmed to respond to a 4-Mbyte address space.

This particular SIMM also includes parity support, which is supported with the PRTY3–PRTY0 signals.

This design also uses the $\overline{RAS1}$ double-drive capability, whereby the $\overline{RAS1DD}$ signal is output by the QUICC to increase the effective drive capability of the $\overline{RAS1}$ signal.

After power-on reset, the software must wait the required time before accessing the DRAM. The required eight read cycles must be performed either in software or by waiting for the refresh controller to perform these accesses.

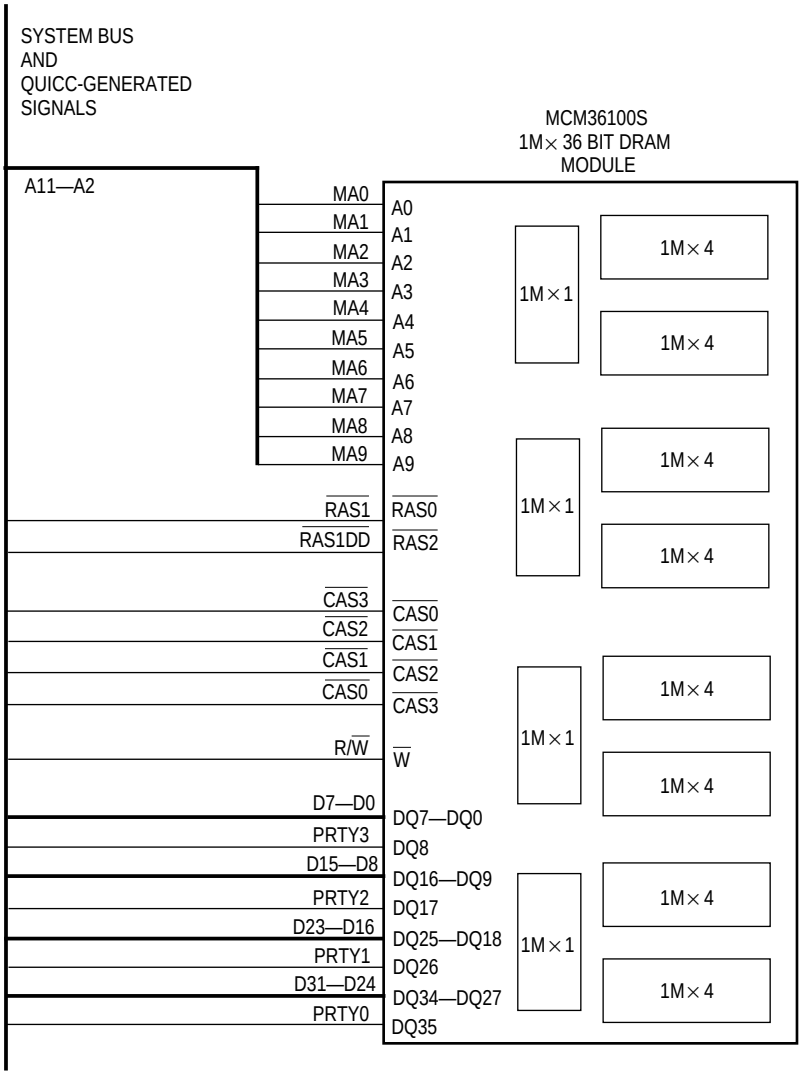


Figure 9-6. Glueless Interface to MCM36100S SRAM

9.1.2.7 DRAM DEVICES. Figure 9-7 shows the glueless interface to standalone DRAM devices. In this case, the MCM54260 256K × 16 DRAM device is chosen. This allows a full 32-bit-wide DRAM solution using only two DRAM devices, with byte writes still supported using the upper and lower $\overline{\text{CAS}}$ pins. The $\overline{\text{RAS1}}$ line should be programmed to respond to a 1-Mbyte address space.

The address multiplexing scheme shown is the same as that for the DRAM SIMM. No parity support is provided in this case. The $\overline{\text{RAS1DD}}$ signal is not used in this case since only two devices are supported.

After power-on reset, the software must wait the required time before accessing the DRAM. The required eight read cycles must be performed either in software or by waiting for the refresh controller to perform these accesses.

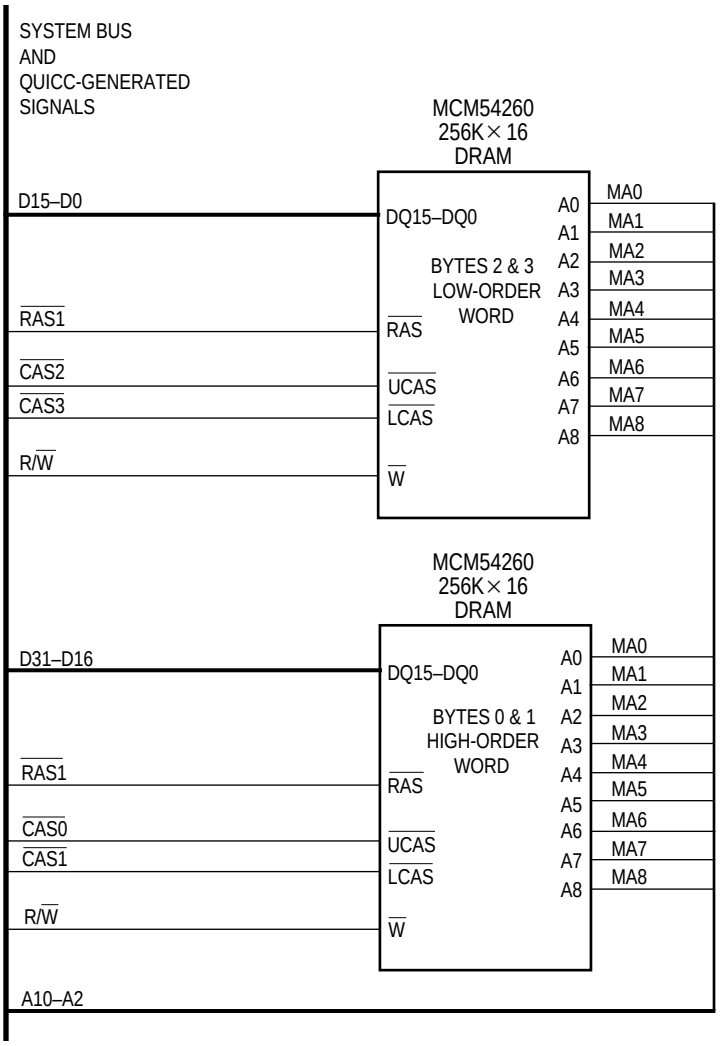


Figure 9-7. Glueless Interface to Standalone DRAM

9.1.3 Software Configuration

The following paragraphs discuss a number of key points for a software writer desiring to initialize the system. The items discussed are those that are required to enable the previous hardware configurations.

9.1.3.1 BASIC INITIALIZATION. The following paragraphs describe the basic initialization procedures for the QUICC.

The module base address register (MBAR) should be set as desired; however, it is unwise to allow the QUICC 8-kbyte block of address space to overlap any memory array.

The module base address register enable (MBARE) should not be accessed.

The module configuration register (MCR) should be set as desired. ASTM and BSTM may be left cleared since their functions are not used in this system. The user should program

BCLROID2–BCLROID0 to $3 and program the SDMA, IDMA1, and IDMA2 arbitration IDs in the SDMA configuration register and IDMA channel configuration registers to 4, 2, and 0, respectively, to allow the SDMA and DRAM refresh to preempt IDMA transfers. SHEN1–SHEN0 should be left cleared. SUPV should be cleared. BCLRIID2–BCLRIID0 are not used and can be programmed to $0. IARB3–IARB0 can be left programmed to $F to allow SIM interrupts to have priority over CPM interrupts that occur at the same level.

The software watchdog interrupt vector register (SWIV) should be set as desired.

In the system protection control register (SYPCR), DBFE and BME should normally be set. If the software watchdog is not used, the SWE bit should be cleared.

The periodic interrupt control register (PICR) may be set as desired.

The breakpoint address register (BKAR) and breakpoint control register (BKCR) should be set as desired.

The port E pin assignment register (PEPAR) should be set to $0180. This configures the $\overline{\text{RAS1DD}}$ pin, the $\overline{\text{WEx}}$ lines instead of A31–A28 lines, the four $\overline{\text{CASx}}$ lines, $\overline{\text{CS7}}$, and $\overline{\text{AVEC}}$.

9.1.3.2 CONFIGURING THE MEMORY CONTROLLER. The following paragraphs describe configuring the memory controller registers.

The global memory register (GMR) should be configured as follows:

The RFCNT bits may be set as desired. At 25 MHz, an RFCNT value of 24 (decimal) gives one refresh every 15.6 μs.

RFEN should be set.

RCYC depends on the DRAM speed. At 25 MHz (an 80-ns DRAM SIMM), RCYC should be 01.

PGS2–PGS0 should be set to 1M for the DRAM SIMM or to 256K for the 256K × 16 DRAM devices.

DPS should be set to 00.

WBT40 is not used and should be cleared.

WBTQ depends on timing; it should be set for 80-ns DRAM SIMMs.

DWQ depends on timing.

DW40 is not used and should be cleared.

EMWS is not used in this design since there is only one QUICC or MC68030-type bus master. It should be cleared.

SYNC is not used in this design since there is only one QUICC or MC68030-type bus master. It should be cleared.

OPAR may be chosen by the user.

PBEE should be set to cause parity errors to generate bus errors.

TSS40 is not used and should be cleared.

NCS should normally be cleared.

DWQ depends on timing. It must be set if page mode is used for the DRAMs.

DW40 is not used and should be cleared.

AMUX should be set.

The memory controller status register (MSTAT) is used for reporting parity errors and does not require initialization.

The eight base registers (BRs), one for each memory bank, should be configured as follows:

The BA31–BA11 bits may be set as desired. Different memory arrays should not overlap.

For simplicity, FC3–FC0 can be cleared.

TRLXQ should normally be cleared for memory interfaces.

BACK40 is not used and should be cleared.

CSNT40 is not used and should be cleared.

CSNTQ should normally be cleared.

PAREN should be set for memory banks that use parity.

WP should be set for EPROM, burst EPROM, and flash EPROM; otherwise, it should be cleared.

V should be set if the memory bank is used.

The eight option registers (ORs), one for each memory bank, should be configured as follows:

The TCYC bits should be set to determine the number of wait states required.

The AM27–AM11 bits should be programmed to determine the block size of the chip select or $\overline{\text{RASx}}$ line. This should be the total number of bytes in each memory array.

FCM3–FCM0 may be set to all zeros to allow the chip select or $\overline{\text{RASx}}$ line to assert on all function codes except CPU space (interrupt acknowledge). It is advisable to program FCM3–FCM0 to zeros, at least during the initial stages of debugging.

BCYC1–BCYC0 are not used and should be cleared.

PGME may be set as desired to enable page mode if this is a DRAM bank.

SPS1–SPS0 should be set to 10 for the byte-wide memory banks, such as EPROM, and cleared for the 32-bit-wide memory banks.

DSSEL should be set only if this is a DRAM bank.

9.1.3.3 USING THE QUICC IN 16-BIT DATA BUS MODE. For systems that do not require a full 32-bit data bus capability, the QUICC offers a 16-bit data bus mode. A system with 16-bit data bus mode is almost the same as the system shown in this section. Only a few changes are required.

- To configure the QUICC for 16-bit data bus mode, the system $\overline{\text{RESET}}$ signal should be connected to the PRTY3 pin. In 16-bit data bus mode, the PRTY2 and PRTY3 pins are not used.
- All port sizes should be configured to either 8 or 16 bits. The DRAM port size must be 16-bits if internal address multiplexing is used.
- Only the D31–D16 pins should be connected to the system bus. Only PRTY0 and PRTY1 should be used. The unused data and parity pins should be pulled high or low through a resistor to save power and to avoid floating inputs.
- Only the upper word of the static RAM array should be used. The $\overline{\text{CS3}}$ line should be programmed to respond to a 64-Kbyte address space.
- The DRAM SIMM should be replaced with a 16- or 18-bit-wide version. The $\overline{\text{RAS1}}$ line should be programmed to respond to a 2-Mbyte address space.
- Only the upper 256K $\times$ 16 bit DRAM device should be used. The $\overline{\text{RAS1}}$ line should be programmed to respond to a 512-Kbyte address space.

9.2 HOW TO TAKE A QUICC SOFTWARE TEST-DRIVE

At first glance, the QUICC has so many features that the proper initialization sequence may not be immediately obvious. The following paragraphs address the issue by showing the proper initialization sequence to use when configuring the QUICC in a system after a power-up reset.

Included is a step-by-step sequence that will allow the user to use and learn each feature of the QUICC in a methodical way. The sequence is applicable whether the QUICC is in normal mode (CPU32+ enabled) or slave mode (CPU32+ disabled). Although the sequence is designed as a first-time test-drive of the QUICC, it is also useful as a general pattern for an overall QUICC initialization.

Detailed information of a specific QUICC hardware configuration and the actual values that should be written to registers may be found in 9.1 Minimum System Configuration and 9.4 Using the QUICC MC68040 Companion Mode**.** Additionally, many examples of initializing certain peripherals are contained elsewhere in this manual where that peripheral is described.

Step 1: Decide on Reset Stack Pointer and Initial Program Counter

$\overline{\text{CS0}}$ will function after reset and allow code execution from the boot EPROM-type device. The stack pointer and the reset vector will be read from the first locations of the EPROM, and the program will execute from EPROM at the initial program counter address offset in the EPROM. Although the stack pointer is initialized, it is not ready to be used because it does not yet point to RAM.

Step 2: Stay in Supervisor Mode

After a reset, M68000 family CPUs are in supervisor mode. Program accesses are issued with the supervisor program function code, and data accesses are issued with the supervisor data function code. These function codes are required for some of the operations that

follow. (In embedded control applications, most users leave the CPU in supervisor mode permanently.)

Step 3: Write the VBR

Many users initialize the ROM to contain an initial exception vector table. If so, the user should write the CPU vector base register (VBR) to point to the starting location of the table. (The exception vector table defines where to find the routines that handle interrupts, bus errors, traps, etc.) Note, however, that exceptions will not be ready to be handled properly until the stack pointer points to addressable RAM.

Step 4: Write the MBAR

The module base address register (MBAR), which always exists at a fixed address, determines where the 8-Kbyte block of QUICC internal RAM and internal peripherals are to be mapped. To put the 8-Kbyte block at $700000, write $00700001 to the MBAR. The MBAR must be accessed before any other QUICC internal peripheral or RAM location. Remember that the MBAR must be written in a special way as described in the Section 6 System Integration Module (SIM60). (If multiple QUICCs exist in the system, it may be necessary to write the MBARE before writing MBAR.)

Step 5: Verify a Dual-Port RAM Location

First, verify that the MBAR address was programmed correctly. This can be done by testing one of the dual-port RAM locations. Write $33 and $CC to location $700000 and verify that these values can be correctly read. Location $700000 is the beginning of the QUICC internal RAM.

Step 6: Is This a Power-Up Reset?

Next, determine the cause of the reset from the reset status register (RSR). The RSR is normally cleared by the user after it is read (ones are written to RSR). If this is not a power-up reset, the user may wish to take actions other than the ones listed or notify the system of the cause of an unexpected reset to aid in debugging. Many of the following steps are only necessary after a power-on reset. See Section 4 Bus Operation for more details on the effects of the different types of resets.

Step 7: Deal with the Clock Synthesizer

The next step is to set up the clock synthesizer, which is located in the SIM60. The three registers that control the clock synthesizer are the CLKOCR, the PLLCR, and the CDVCR.

If a low-speed external crystal is used (such as 32 kHz or 4 MHz), multiply the QUICC clock frequency up to the desired speed (e.g., 25 MHz). If an external oscillator is used to provide the exact system frequency, then this step is not needed.

The clock synthesizer has other options, such as SyncCLK and BRGCLK dividers, as well as the ability to divide the general system clock frequency. These are used in low-power applications. At this time, leave all these options in their default conditions. These options should only be enabled when the rest of the application code is in a more stable state.

As a further protection, the user may set the MSB in the clock synthesizer registers, writing all other bits as zero. This will disable further writes to the clock synthesizer registers, protecting them from inadvertent accesses by the program during the debugging phases.

Step 8: Initialize System Protection

The next step is to protect the system with the bus monitor, double bus fault monitor, and software watchdog. The bus monitor and double bus fault monitor should be used during initial debugging. The software watchdog should be disabled at this time if it will not be used.

Step 9: Clear Entire Dual-Port RAM

After a power-on reset, the internal dual-port RAM comes up in a random state. The user can avoid many potential problems if the entire dual-port RAM is cleared at this time. This includes the 1762 bytes of internal system RAM as well as the 768 bytes of parameter RAM. By clearing the entire dual-port RAM, the user can be sure that no feature is accidentally invoked just because it was not initialized.

NOTE

It is critical that the user issue a CP reset command to the CR of the CPM after clearing the entire dual-port RAM. The reset is required to allow the RISC to reinitialize its internal state variables for all serial channels, since these variables are stored in the dual-port RAM and are not necessarily initialized to zeros.

Step 10: Write the PEPAR

The next important action is to configure the rest of the system bus pins by writing to the port E pin assignment register (PEPAR) in the SIM60. In this register, the user will make a number of choices about which functions are selected on the system bus (e.g., does the user want 32 address lines or 28 address lines and 4 write enable lines, etc.).

Step 11: Remap Chip Select 0

Up to this point, $\overline{CS0}$ has been running with many wait states and has been mapped to the entire address space. It is now time to configure $\overline{CS0}$ to the proper portion of memory controlled by the EPROM and to reduce the number of wait states. This is accomplished by writing the global memory register (GMR), option register 0 (OR0), and base register 0 (BR0) in the memory controller. Note that the valid bit in the BR0 should be set last.

Depending on the details of the application, most GMR bits do not affect $\overline{CS0}$, but it is a good idea to set up GMR before enabling any specific chip select on the QUICC. GMR programming is discussed more fully in 9.1 Minimum System Configuration and 9.4 Using the QUICC MC68040 Companion Mode.

Step 12: Initialize the System RAM

Ensure that $\overline{CS0}$ is programmed correctly—at least for these particular execution addresses.

The next step is to configure the system RAM. The address space for the RAM should include the address of the stack pointer initialized during reset. (One other option would be to point the stack pointer into the QUICC internal RAM; however, this internal RAM will probably be used later for serial channel buffer descriptors. Therefore, this idea is recommended only for the initial phases of debugging. (See step 3 discussed earlier.)

Once the system RAM is initialized, it should be tested by the user to see if it exists at all intended addresses.

Step 13: Copy the EVT to System RAM

The exception vector table (EVT) can be copied to system RAM at this time to allow greater flexibility in choosing and modifying the exception vector values. Most embedded operating systems require the EVT to be located in RAM. Once the table is copied, the VBR should be modified to point to the beginning of the exception vector table in system RAM, rather than pointing to ROM.

Step 14: Initialize All Other Memory and Peripherals

Initialize the remaining $\overline{\text{CSx}}$ and DRAM $\overline{\text{RASx}}$ pins in the memory controller. Then test access to each memory and peripheral in the system.

If DRAM is used, wait the appropriate time after power-up (usually around 100 μs) before accessing the DRAM. Before writing the DRAM, the user normally must make the required number of read accesses (usually eight) to the DRAM using the CPU or must wait for the DRAM refresh controller to complete that number of refreshes. Other initialization activity can proceed while waiting for the DRAM.

Step 15: Initialize the Rest of the SIM60

The next step is to finish initializing the SIM60 as desired. In particular, the module configuration register (MCR) has a number of "fine-tuning" choices that should be initialized.

Step 16: Generate a SIM60 Interrupt

The next step is to try an interrupt source. The simplest interrupt source on the QUICC to try is the periodic interrupt timer (PIT). It is completely controlled in software.

To prepare for this interrupt, initialize the PIT registers, but leave the timer disabled until everything else is ready.

First, write the PIRQL0–PIRQL2 bits in the PICR with the desired interrupt level. Suggestion: choose interrupt level 1. Do not use level 7 for this interrupt, at least during initial stages of debugging. Then write the PIV bits in the PICR to determine the vector. Example: write $40 (64 decimal) to choose the first user-defined vector in the exception vector table.

Next, using the VBR value written in a previous step, write the address of the interrupt handler into the appropriate vector offset of the exception vector table. For instance, for vector number $40 and the VBR pointing to $800000, the interrupt handler address is written to $80000100. (The vector number is multiplied by 4 to obtain the offset.)

NOTE

The exception vector table may have been located in ROM, in which case the PIT vector location should be initialized before power-on reset.

Also, before the interrupt can be processed, the CPU32+ status register (SR) must be programmed to change the interrupt mask bits from a value of $7 to a value of $0 (or at least a value lower than the interrupt level desired).

Next, write the PITR bits in the PITR to start the timer and wait for the interrupt.

Step 17: Test the CPM

After the SIM60 is programmed, it is time to begin testing the CPM. In order of increasing initialization complexity, the following sub-blocks may be tested. See the initialization examples included in this manual where the blocks are described.

Dual-Port RAM

Parallel I/O Ports A, B, and C

Baud Rate Generators

Four General-Purpose Timers

RISC Timer Tables

IDMA (without buffer chaining)

SPI (loopback mode test)

IDMA (with buffer chaining)

SMCs (loopback mode test)

SCCs (loopback mode without the time slot assigner)

SCCs (loopback mode with the time slot assigner)

The SPI, SMCs, and SCCs should be tested in loopback mode before attempting to send and receive data externally.

Step 18: Generate Interrupts with the CPM

When testing interrupts on the CPM, the user should implement this gradually. First, generate an interrupt with a timer or parallel I/O pin. Then proceed to more complicated interrupt structures like the serial channels.

Step 19: Enable External Interrupts

The next step is to allow external devices (if any) to interrupt the QUICC. These interrupts can enter the QUICC through the SIM60 ($\overline{\text{IRQx}}$ pins) or the CPM (parallel I/O pins with interrupt capability).

Step 20: Enable External Bus Masters

The final step in the initialization is to allow external bus masters (if any) to obtain ownership of the system bus. Some of the characteristics of the QUICC bus arbitration logic were chosen in the MCR of the SIM60. The user should review these settings before continuing with this step. Also, if an external bus master uses QUICC resources such as chip selects, the settings of the GMR, BRs, and ORs in the memory controller should be reviewed before continuing.

Step 21: Off to the Races

The user has now successfully examined all major features of the QUICC and can embark on a quick and successful product development cycle using the QUICC. For applications desiring to port code from the MC68302, see 9.3 Porting MC68302 IMP Code to the MC68360 QUICC in this manual.

9.3 PORTING MC68302 IMP CODE TO THE MC68360 QUICC

This subsection is designed as a guide for software engineers who desire to port software written for the MC68302 integrated multiprotocol processor (IMP) to the MC68360 QUICC. It discusses the MC68302 on a register-by-register basis and explains programming the same functions on the QUICC.

Although the QUICC contains a number of new features beyond the capabilities of the MC68302, the basic architectural approach is the same. Although it will be necessary to modify the MC68302 initialization code and, in some cases, the register and bit names, a great effort was made to preserve the basic code flow. An example would be the basic flow of an interrupt handler or an error handler. Thus, the knowledge that was obtained during the development of the MC68302 drivers need not be lost during the port to the QUICC.

9.3.1 CPU and Compilers

The QUICC contains a CPU32+ processor. This processor executes the M68000 code that was written on the MC68302; however, if such code was accessing MC68302 peripherals, it will require some modification.

Most M68000 compiler vendors also provide a CPU32 compiler. If the application code is recompiled using a CPU32 compiler, then an additional performance improvement will also be obtained due to the availability of additional CPU32 instructions over and above those of the M68000.

Also, using the QUICC 32-bit data bus mode provides even further performance improvement. This is especially true if the original M68000 code used 32-bit pointers and data fields. The best performance improvement will occur when the objects are long-word aligned.

9.3.2 Differences/Similarities

The 4-Kbyte RAM and peripheral area on the MC68302 is expanded to 8 Kbytes on the QUICC. Since the QUICC is an IMB device, the initialization of the system integration module (SIM60) is actually more similar to another IMB-based device, such as the MC68340,

than it is to the system integration block on the MC68302. The QUICC communications features, however, are very similar to those of the MC68302.

9.3.3 Notes About Porting

Although the following paragraphs show how to port MC68302 functions to the QUICC, it should not be assumed that the port operation will provide a complete initialization of the QUICC. The QUICC contains features not available on the MC68302 that may require additional initialization. For instance, the QUICC contains an on-chip PLL to generate a high-speed system clock frequency from a low-speed crystal, such as a 32-kHz crystal. Since this feature is not available on the MC68302, it is not mentioned in this discussion. Refer to 9.2 How to take A QUICC Software Test-Drive as a guide to complete QUICC initialization.

Although the QUICC was designed to allow a convenient upgrade path from the MC68302, this discussion does not guarantee that every MC68302 operation can be exactly and precisely duplicated on the QUICC. For instance, when using the serial channels, the time between setting the ready bit of a buffer descriptor and the assertion of the $\overline{\text{RTS}}$ pin may not be the same on the QUICC as the MC68302. Also, although the MC68302 hardware watchdog can be implemented in the QUICC bus monitor, the MC68302 maximum timeout is 16K clocks; whereas, the QUICC maximum timeout period is 1K clocks (before $\overline{\text{BERR}}$ is asserted on the system bus).

The QUICC offers features that simplify what would otherwise be a more code-intensive process. For specific examples, see the INIT TX AND RX PARAMETERS, the GRACEFUL STOP TRANSMIT, and the CLOSE BD commands. The user can do a direct port using the old commands and techniques; however, the new commands may be used in many instances to simplify the application process.

Additionally, the most direct port to the QUICC will not necessarily take advantage of a number of new QUICC features. For specific examples, see the dynamic allocation of SCC interrupt levels and the buffer descriptor capability of the independent DMA (IDMA) channels. Some comments will be made about these features where appropriate.

9.3.4 How To Port MC68302 Functions

The following paragraphs detail the different MC68302 functions and how/where to implement them on the QUICC. The MC68302 functions are listed in ascending order in the MC68302 memory map.

9.3.4.1 SYSTEM CONFIGURATION REGISTERS. The following paragraphs describe the MC68302 configuration registers.

9.3.4.1.1 Base Address Register (BAR). This register is most closely related to the MBAR on the QUICC:

The base address field is configured in the BA bits of the MBAR. Be sure to allow 8 Kbytes in the memory map of the QUICC system at this address. Also, the V-bit of MBAR must be set before the contents are valid.

The CFC bit and FC2–FC0 bits may be duplicated by setting all but one of the AS7–AS0 bits in the MBAR. Note that the MBAR offers many more options since multiple AS bits

may be simultaneously set. If CFC is zero, leave AS7–AS0 as zeros.

9.3.4.1.2 System Control Register (SCR). The functions of this register are found in various places in the QUICC.

LPCD, LPEN, LPP16, and LPREC bits implement low-power functions in the MC68302. The QUICC allows full operation at low speeds (due to its static design), as well as a number of enhanced power reduction options and a special low-power stop instruction (LPSTOP). The user should reference Section 6 System Integration Module (SIM60) for full details.

The HWDCN2–HWDCN0, HWDEN, and HWT bits are used to control and derive status from the MC68302 hardware watchdog. This function, called the bus monitor on the QUICC, is part of the SIM60. See the BME and BMT1–BMT0 bits in the system protection control register (SYPCR).

The SAM bit is used to control external masters that access the MC68302. This function is controlled in the SIM60 by the BSTM bit in the module configuration register (MCR).

The FRZ2–FRZ1 and FRZW bits control freeze operation. In the QUICC, more freeze options are available. See the freeze bits in the MCR of the SIM60, the SDMA configuration register (SDCR), timer global configuration register (TGCR), and the IDMA channel configuration register (ICCR) of the CPM.

The BCLM bit and IPA bits are used to allow the M68000 core to give all interrupts higher priority than external bus masters. In the QUICC, this function can be refined to just a range of interrupt levels. See the BCLRISM2–BCLRISM0 bits in the MCR.

The ADCE and ADC bits determine the result (such as a $\overline{\text{BERR}}$ assertion) of an overlap in the chip select ranges. This function is not available on the QUICC.

The EMWS bit adds an additional wait state for external masters that are accessing the MC68302 in a synchronous fashion and adds a wait state when external masters use MC68302 chip selects. This function is implemented in the DW40 and EMWS bits of the global memory register (GMR) in the QUICC memory controller.

The RMCST bit determines the M68000 bus functionality during the TAS instruction and gives the option of exact MC68000-bus compatibility on the MC68302. On the QUICC, this function is not required since an M68030-type bus is used.

The MC68302 WPVE and WPV bits are used to determine the result (such as a $\overline{\text{BERR}}$ assertion) of a write operation to a read-only chip select. See the WPER bit in the memory controller status register (MSTAT) for the QUICC definition.

The VGE bit allows the MC68302 to generate vectors for interrupt acknowledge cycles while in slave mode. With QUICC in slave mode, all internal interrupts generate vectors. For interrupts using the $\overline{\text{IRQx}}$ pins, this decision is based on the autovector register. Either a vector is placed on the bus, or the $\overline{\text{AVECO}}$ pin is asserted (which can be ignored).

The ERRE bit is used with the DRAM refresh controller on the MC68302. The QUICC contains a full DRAM controller that does not require the assistance of the RISC controller; therefore, this bit is not implemented in the QUICC.

The other status bits in this register were already discussed.

9.3.4.2 SYSTEM RAM. The QUICC contains 2560 bytes of dual-port RAM rather than 576 bytes. Both the MC68302 and the QUICC split the dual-port RAM into two parts: user data RAM and parameter RAM.

The user data RAM of the QUICC may be used in the same way it is used in the MC68302. However, on the QUICC, it has one major additional feature—it may be used to store buffer descriptors for serial data. (In fact, the QUICC has no special location for buffer descriptors in the dual-port RAM. They may be mapped anywhere.) Thus, the user should plan on the user data RAM being used for buffer descriptors, in addition to whatever the RAM was being used for in the MC68302 application.

The parameter RAM of the MC68302 is very similar to the parameter RAM on the QUICC, except that there is no special location for buffer descriptors on the QUICC. This gives the user the ability to choose the location and number of buffer descriptors for each serial channel.

The specific protocol parameter area of the MC68302 is available on the QUICC. On the QUICC there are four such areas; whereas, on the MC68302 there are only three. Like the MC68302, each specific protocol parameter area is comprised of two parts—a protocol independent part that is common to all SCCs, regardless of their protocol, and a protocol specific part that is unique to each SCC, based on the protocol.

On the MC68302, the SMCs and SCP were given special locations for buffer descriptors and parameter areas. On the QUICC, the SMCs and SPI are controlled like the SCCs, with buffer descriptors that may be located anywhere in the user RAM, and with a specific protocol parameter RAM. In fact, all the QUICC SCCs, SMCs, and SPI have a common subset of parameter RAM.

The following description will discuss the buffer descriptors, protocol-independent parameter RAM, and protocol-dependent parameter RAM for the MC68302 and the QUICC.

9.3.4.2.1 Buffer Descriptors. The buffer descriptors for the MC68302 and the QUICC are essentially the same.

The status and control field is a 16-bit field on both the MC68302 and the QUICC. All bits are in their original positions with the following exceptions: the X-bit is removed, since the determination of an internal or external address is made by the full address decoding on the QUICC. The user may still set this bit location without harm. In UART mode, the PM and A bits had to be moved. In BISYNC mode, the B-bit had to be moved. Additional status and control bits were added on the QUICC in previously unused bit positions.

The data length field is identical between the MC68302 and the QUICC.

The data buffer pointer is the same between parts; however, on the QUICC, all 32 bits are used. Additionally, if the receive FIFO is set to 32-bit-wide mode, then the data buffer pointer must be aligned to a long word.

9.3.4.2.2 Protocol-Independent Parameter RAM Values. These values are very similar on both devices.

The RBASE and TBASE values are new to the QUICC. They point to the start of the buffer descriptor tables, must be long-word aligned, and must point to the dual-port RAM area.

The RFCR and TFCR registers have one additional purpose and a different bit placement on the QUICC. If RFCR and TFCR are cleared on the MC68302 or not used with the MC68302, then the equivalent function on the QUICC can be implemented by writing $18 to the QUICC RFCR and TFCR. Note that on the QUICC there is an additional function code pin to signify DMA operation, and the suggestion of $18 uses this capability.

The MRBLR value has the same function on both devices. Additionally, if the receive FIFO is set to 32-bit-wide mode, the MRBLR value must be aligned to a long word.

The internal state parameter has the same function on both devices.

The MC68302 RBD# and TBD# have the same concept on the QUICC, except the parameters are called RBPTR and TBPTR and are now 16-bit values. They point to the current buffer descriptor; however, they are now offsets from the beginning location of the QUICC dual-port RAM. For example, the current transmit buffer descriptor location may be found by MBAR + TBPTR.

The internal data pointer parameter has the same function on both devices.

The internal byte count parameter has the same function on both devices.

9.3.4.2.3 Protocol-Dependent Parameter RAM Values. These values are very similar between devices, although new functions are often added on the QUICC. Where possible, any parameter for a given protocol that is the same for the MC68302 and the QUICC carries the same name on both devices.

The following points note changes between the MC68302 and the QUICC protocol-dependent parameter RAM for a given protocol:

In the UART mode, if the MAX_IDL entry is programmed to $0000 on the QUICC, the function will be disabled.

In the UART mode, the ability to send special control characters, such as XOFF, no longer requires the CHARACTER8 entry. Rather, a new entry, called the TOSEQ entry, has been created. The programming of the TOSEQ entry, however, is the same as the old MC68302 CHARACTER8 entry.

In UART mode, note the new parameters for a receive character mask (RCCM) and a function that times the length of a receive break (BRKLN).

The HDLC parameter RAM is the same, except for two new entries: RFTHR and RFCNT. They allow the user to reduce the number of received frame interrupts generated, and should be used in higher data rate applications.

The BISYNC parameter RAM is unchanged.

DDCMP is a microcode RAM product on the QUICC. The port of DDCMP from the MC68302 is not discussed.

V.110 is not supported on the QUICC.

For the transparent protocol, neither the MC68302 nor the QUICC requires parameter RAM for its implementation. The QUICC, however, offers the new feature of allowing CRCs to be generated and checked on transparent frames. In that case, the QUICC's transparent parameter RAM must be initialized.

9.3.4.3 INTERNAL REGISTERS (SYSTEM INTEGRATION BLOCK). Both the MC68302 and the QUICC have a number of internal registers. The following paragraphs detail the registers according to their ascending order in the MC68302 memory map. Note that the address offsets of these registers on the QUICC are different from the address offsets on the MC68302.

IDMA

The MC68302 CMR is closely related to the QUICC CMR.

The STR, RST, and BT bits are the same on both parts.

The DSIZE bits are compatible, but on the QUICC, an encoding exists for long-word size.

The DAPI, SAPI, and REQG bits are the same.

INTE and INTN are, in effect, moved to the CMAR, but more control is given to the user.

ECO is in bit 15 of the QUICC CMR.

The QUICC CMR contains new functions in the RCI, S/D, and SRM bits.

SAPR on the MC68302 is identical to SAPR on the QUICC.

DAPR on the MC68302 is identical to DAPR on the QUICC.

BCR on the MC68302 is increased to 32 bits on the QUICC.

CSR on the MC68302 is extended on the QUICC.

The DONE, BED, and BES bits are the same on both parts.

The DNS bit is not needed on the QUICC.

The QUICC additionally contains the OB and BRKP bits.

The QUICC contains a CMAR that is not available on the MC68302.

The FCR is the same on both parts, except that the QUICC has a provision for four function code lines; whereas, the MC68302 only has three.

INTERRUPTS

The GIMR on the MC68302 is most closely related to the QUICC CICR and AVR.

The V7–V5 bits become the VBA2–VBA0 bits.

The ET1, ET6, and ET7 bits control the edge/level sensitivity of the $\overline{\text{IRQ1}}$, $\overline{\text{IRQ6}}$, and $\overline{\text{IRQ7}}$ pins on the MC68302. On the QUICC, these pins are handled by the SIM60. On the SIM60, all $\overline{\text{IRQx}}$ pins except $\overline{\text{IRQ7}}$ are level sensitive. These bits do not exist on the QUICC.

The IV1, IV6, and IV7 bits are most closely related to the AV1, AV6, and AV7 bits of the AVR in the SIM60.

The MOD bit does not exist on the QUICC. On the QUICC, the dedicated mode is used.

On the QUICC, new functions are available in the CICR such as a choice of the CPM interrupt level, a choice of the way the SCC interrupts are grouped, and a choice of the highest priority CPM interrupt source. These functions were not available on the MC68302.

The MC68302 IPR is increased to 32 bits and is called the CIPR on the QUICC. On the QUICC, the SIM60 can generate interrupts without using the CIPR.

The MC68302 IMR is increased to 32 bits and is called the CIMR on the QUICC.

The MC68302 ISR is increased to 32 bits and is called the CISR on the QUICC.

PARALLEL I/O PORTS

The parallel I/O pin assignment is different on the QUICC, but the three basic register types are carried over intact. The QUICC has three I/O ports (A, B, and C); whereas, the MC68302 has two (A and B).

The PxCNT control register on the MC68302 is the PxPAR on the QUICC.

The PxDDR data direction register is the same on both devices.

The PxDAT data latch and input register is the same on both devices.

On the QUICC, some of the parallel ports have an open-drain register, PxODR, which allows a port pin to be configured as an open-drain pin.

CHIP SELECTS

The MC68302 contains four base registers (BRx) and four option registers (ORx) to control the chip selects. The QUICC contains a global memory register (GMR), a memory status register (MSTAT), eight base registers (BRx), and eight option registers (ORx). The GMR would normally be initialized first on the QUICC. Also, the QUICC chip selects contain many enhancements over the MC68302 chip selects not discussed here.

The MC68302 BRx register most closely corresponds to the QUICC BRx.

The EN bit becomes the V-bit in the QUICC BRx.

The RW bit in the BRx and the MRW bit in the ORx of the MC68302 become the WP bit on the QUICC BRx. On the QUICC, the choice exists for asserting the chip select for reads and writes (WP = 0) or just reads (WP = 1).

The MC68302 base address bits A23–A13 are found in BA31–BA11 of the BRx on the QUICC. Note that the QUICC provides support for 32-bit address recognition (BA31–BA24 are new) as well as support for smaller starting address sizes (BA12–BA11 are new), giving the ability to begin a block on a 2K address boundary, rather than 8K on the MC68302. To transfer a starting address from the MC68302 to the QUICC, set any bit in

BA23–BA13 that was set in A23–A13, and clear BA31–BA24 and BA12–BA11.

The three function code (FC2–FC0) bits become the four function code (FC3–FC0) bits of the QUICC ORx.

The MC68302 ORx register most closely corresponds to the QUICC ORx.

The CFC bit is implemented as the FCM3–FCM0 bits on the QUICC ORx. This gives more flexibility in determining the function codes that cause the chip select to activate. If the MC68302 CFC bit was cleared, then clear FCM3–FCM0 on the QUICC. Also, to match MC68302 behavior, clear the NCS bit in the QUICC GMR.

The MRW bit in the ORx and the RW bit in the BRx of the MC68302 simply become the WP bit on the QUICC BRx. On the QUICC, the choice exists for asserting the chip select for reads and writes (WP = 0) or just reads (WP = 1).

The MC68302 base address mask bits A23–A13 become the AM27–AM11 bits in the QUICC ORx. Note that this allows both larger and smaller block sizes than what the MC68302 provides. To transfer a block range, take bits A23–A13 of the MC68302 and write them to AM23–AM13. Then set AM32–AM24 and clear AM12–AM11.

The three MC68302 DTACK bits become the four TCYC3–TCYC0 bits of the QUICC ORx. Note that the maximum number of wait states is increased from 6 to 15 on the QUICC.

TIMERS

The MC68302 contains two general-purpose timers. The QUICC contains four general-purpose timers, which are the same as the MC68302 timers, but slightly enhanced. The QUICC also contains a timer global configuration register (TGCR), which allows all four timers to be enabled simultaneously and allows the timers to be internally cascaded into 32-bit timers.

The MC68302 TMRx register most closely corresponds to the QUICC TMRx.

The RST bit is now located in the QUICC TGCR.

The ICLK bits are still in the same location of the QUICC TMR. The bit encodings are the same except for 00 combination, which is now implemented by the EN bit in the QUICC TGCR.

The FRR bit is still in the same location of the QUICC TMR.

The ORI bit is still in the same location of the QUICC TMR.

The OM bit is still in the same location of the QUICC TMR, but the meaning of OM = 0 mode can be different. The active-low pulse can be longer than on the MC68302, depending on the frequency of the input clock source.

The CE bits are still in the same location of the QUICC TMR.

The PS bits are still in the same location of the QUICC TMR.

The MC68302 TRRx register is the same as the QUICC TRRx.

The MC68302 TCRx register is the same as the QUICC TCRx.

The MC68302 TCNx register is the same as the QUICC TCNx.

The MC68302 TERx register is the same as the QUICC TERx.

WATCHDOG TIMER

The MC68302 timer 3 is called the watchdog timer. This software watchdog timer must be serviced periodically or can be used to generate a system reset. The software watchdog timer on the QUICC exists in the SIM60. It corresponds to the software watchdog available on other members of the MC68300 family.

The WRR and WCN on the MC68302 do not have direct counterparts on the QUICC. To program the software watchdog on the QUICC, the user should initially program the SYPCR, the SWIV (optional), and service the software watchdog using the SWSR. Additionally, an indication of a software watchdog reset may be found in the RSR.

9.3.4.4 INTERNAL REGISTERS (COMMUNICATION PROCESSOR). The following paragraphs detail the registers associated with the communications processor, according to their ascending order in the MC68302 memory map. Note that the address offsets of the QUICC registers are different than the offsets on the MC68302 registers.

The 8-bit command register (CR) also exists on the QUICC, but it is expanded to 16 bits.

The FLG bit still exists in the same location of the QUICC CR.

The two CH NUM bits have been shifted to bit locations 7 and 6 and expanded to four bits rather than two (bits 6 and 5 also), since more peripherals may now receive commands. If bits 6 and 5 are cleared, the SCC channel number encodings are compatible from the MC68302 to the QUICC.

The two OPCODE bits have been increased to four bits, shifted left in the register, and redefined to include many more commands. However, the original commands available on the MC68302 are still available.

The RST bit is shifted to become bit 15 of the QUICC CR. To reset the MC68302, the value $81 was issued to the CR. To reset the QUICC, the value $8001 is issued to the CR.

SCCs

The MC68302 SCC registers are mapped to the QUICC as follows.

The SCONx register controls the clocking scheme and baud rates of the MC68302 SCCs. On the QUICC, four independent baud rate generators that are not associated with specific SCCs are available. Therefore, this register is implemented in the QUICC baud rate generator and is called the BRGC. Note that the BRGC contains new bits, including a reset bit and an enable bit.

The DIV4 bit is now a divide-by-16 option and is implemented in the DIV16 bit of the QUICC BRGC.

The CD10–CD0 bits become CD11–CD0 and are located in the QUICC BRGC.

The RCS bit is implemented in the bank of clocks control in the three RxCS bits of the

QUICC serial interface clock route register (SICR). Note that many more options are now available.

The TCS bit is implemented in the bank of clocks control in the three TxCS bits of the QUICC SICR. Note that many more options are now available.

The EXTC bit becomes the EXTC1–EXTC0 bits in the BRGC. Note that more options are now available. For compatibility, if the EXTC bit was cleared, then the EXTC1–EXTC0 bits should also be cleared.

The WOMS bit gives the MC68302 wired-OR capability on the serial transmit data pin. This can now be implemented (as well as other wired-OR capability) in the QUICC PxODR registers, which can configure a pin to operate as wired-OR, regardless of whether it is connected to the SCC.

The SCM register is implemented using the general SCC mode register (GSMR) and the protocol-specific mode register (PSMR). One GSMR and one PSMR exist for each SCC. The definition of the PSMR differs based on the protocol used.

The two MODE bits have been expanded to four MODE bits in the GSMR to include the new protocols that the QUICC provides.

The ENT bit still exists, but is shifted two bit positions left in the GSMR.

The ENR bit still exists, but is shifted two bit positions left in the GSMR.

The DIAG bits still exist on the QUICC but are shifted left two positions. Normal operation = 00, loopback = 01, and automatic echo = 10 still exist on the QUICC. Software operation = 11 is now implemented by programming the PCPAR, PCDIR, PCODR, and PCINT in the port C parallel I/O section. The new structure for software operation is much more flexible than on the MC68302 and provides hardware-updated, accurate, real-time status. Since the software operation combination is not needed in the QUICC DIAG bits in the GSMR, this combination is used to support a new feature, simultaneous loopback and echo.

The rest of the SCM bits differ based on protocol and are discussed in the following paragraphs by protocol.

The UART mode register is now implemented using the GSMR and the PSMR. One GSMR and one PSMR exist for each SCC. The definition of the PSMR differs based on the protocol used.

The SL bit is located in the PSMR.

The RTSM bit is located in the GSMR.

The CL bit becomes two bits in the PSMR for more character length options.

The FRZ bit is located in the PSMR.

The UM bits are located in the PSMR. Note that the 10 combination for asynchronous DDCMP is now reserved.

The PEN bit is located in the PSMR.

The one RPM bit becomes two RPM bits in the PSMR for more receive parity options.

The TPM bits remain the same and are located in the PSMR.

The HDLC mode register is implemented using the GSMR and the PSMR. One GSMR and one PSMR exist for each SCC. The definition of the PSMR differs based on the protocol used.

The ENC bit becomes the TENC and RENC bits in the GSMR. Note that the QUICC contains many more data encoding options.

The FLG bit becomes the RTSM bit in the GSMR.

The RTE bit is located in the PSMR.

The FSE bit is located in the PSMR.

The C32 bit becomes the two CRC bits in the PSMR. If C32 = 0, then initialize CRC to 00. If C32 = 1, then initialize CRC to 10.

The NOF bits are located in the PSMR.

The BISYNC mode register is implemented using the GSMR and the PSMR. One GSMR and one PSMR exist for each SCC. The definition of the PSMR differs based on the protocol used.

The ENC bit becomes the TENC and RENC bits in the GSMR. Note that the QUICC contains many more data encoding options.

The SYNF bit becomes the RTSM bit in the GSMR.

The RBCS bit is located in the PSMR.

The RTR bit is located in the PSMR.

The BCS bit becomes the two CRC bits in the PSMR. If BCS = 0, initialize CRC to 11. If BCS = 1, initialize CRC to 01.

The REVD bit becomes the RVD bit in the PSMR. Note that the RVD bit in the PSMR is not the same as the REVD bit in the GSMR.

The NTSYN bit should not have been set in BISYNC mode, and therefore is not ported.

If the EXSYN bit was set, the CDP and CTSP bit in the GSMR should be set for compatibility. The user may also wish to leave the CDS and CTSS bits cleared in the GSMR.

The PM bit becomes the TPM and RPM bits in the PSMR. Note that more parity options are now available.

NOTE

The SYNL bits in the GSMR were added to offer more synchronization options than are available on the MC68302.

The DDCMP mode register is a microcode RAM product on the QUICC. The port of DDCMP from the MC68302 is not discussed in this section.

The V.110 mode register is not supported on the QUICC.

The transparent mode register is implemented using the GSMR and the PSMR. One GSMR and one PSMR exist for each SCC. The definition of the PSMR differs based on the protocol used.

The REVD bit is located in the GSMR.

The NTSYN bit becomes the TTX and TRX bits in the GSMR. On the QUICC the user can independently enable the totally transparent protocol to work on the receiver or transmitter while another protocol (such as HDLC) runs on the transmitter or receiver.

If the EXSYN bit was set, the CDP and CTSP bit in the GSMR should be set for compatibility. The user may also wish to leave the CDS and CTSS bits cleared in the GSMR. The user may wish to set the RSYN bit in the GSMR to remain compatible with the MC68302.

NOTE

The SYNL bits in the GSMR were added to offer more synchronization options than are available on the MC68302. Additionally, a CRC may be generated in transparent mode using the TCRC bits.

The data synchronization register (DSR) is also available on the QUICC. When used in UART mode to generate fractional stop bits, the user should note that the encodings have changed slightly.

The SCC event register (SCCE) is also available on the QUICC, but the register now has 10 bit positions to include new bit functions. The configuration of this register depends on the protocol mode.

In UART mode, the RX, TX, BSY, CCR, BRK, and IDL bits exist unchanged. The CD and CTS bits are no longer needed because the interrupt controller supports a separate vector for each event.

In HDLC mode, the RXB, TXB, BSY, RXF, TXE, and IDL bits exist unchanged. The CD and CTS bits are no longer needed because the interrupt controller supports a separate vector for each event.

In BISYNC mode, the RX, TX, BSY, RCH, and TXE bits exist unchanged. The CD and CTS bits are no longer needed because the interrupt controller supports a separate vector for each event.

DDCMP is a microcode RAM product on the QUICC. The port of DDCMP from the MC68302 is not discussed in this section.

V.110 is not supported on the QUICC.

In totally transparent mode, the RX, TX, BSY, RCH, and TXE bits exist unchanged. The CD and CTS bits are no longer needed because the interrupt controller supports a separate vector for each event.

The SCC mask register (SCCM) is also available on the QUICC, but the register now has 10 bit positions to include new bit functions. The configuration of this register depends on the protocol mode. Since the SCCM has the exact format as the SCCE, see the preceding SCCE description for details.

The SCC status register (SCCS) is also available on the QUICC. The configuration of this register depends on the protocol mode. In all cases, the $\overline{CTS}$ and $\overline{CD}$ bits are available in the PCDAT register, which allows the user to read the current value of these signals directly from the pins. Additionally, the ID bit is available in the QUICC SCCS register in UART and HDLC modes.

NOTE

New status functions are available in the QUICC SCCS, such as whether the HDLC controller is currently receiving flags and the DPLL carrier sense status.

The 16-bit SPMODE register consists of two halves: the 8-bit SMC mode register and the 8-bit SCP mode register.

The SMCs on the QUICC are much more similar to the SCCs on the MC68302. To port an ISDN (GCI or IOM2) application to the QUICC that uses the MC68302 SCCs, the user should use the features provided in the serial interface and SMC.

The QUICC does not contain special support for IDL in the SMCs because IDL no longer uses the A or M bits, but rather uses an out-of-band control bus called the SCP. On the QUICC, the SCP function may be implemented with the SPI.

NOTE

On the QUICC, the two SMCs can also provide a UART or totally transparent control function.

The SCP on the QUICC is enhanced to full serial peripheral interface (SPI) functionality. The SPI is found on many Motorola devices, such as the MC68HC11. The 8-bit SCP mode register is most closely related to the 16-bit SPI mode register (SPMODE) on the QUICC.

The EN bit is in the SPMODE register.

The PM3–PM0 bits are located in the SPMODE register.

The CI bit is located in the SPMODE register.

The LOOP bit is located in the SPMODE register.

The STR bit becomes the R-bit of the SPI transmit buffer descriptor. The buffer descriptor structure of the SPI is like that of the SCCs.

The SIMASK register on the MC68302 is used in GCI and IDL modes to select individual bits of the B-channels to be routed to particular SCCs. On the QUICC, the serial interface has a time slot assigner that is much more flexible and can be used in any time division multiplexed (TDM) interface, not just GCI and IDL. The time slot assigner on the QUICC is programmed using a feature called the serial interface RAM. See 7.8.4 SI RAM.

The SIMODE register on the MC68302 controls the serial interface. On the QUICC, the serial interface control is greatly enhanced and is comprised of the serial interface RAM, the serial interface global mode register (SIGMR), the serial interface mode register (SIMODE),

the serial interface clock route register (SICR), and the serial interface command register (SICMR).

The MS bits select the serial interface mode on the MC68302: NMSI, IDL, GCI, and PCM highway. On the QUICC, the choice is simply to enable or disable the time slot assigner. Once the time slot assigner is enabled, it may be programmed in the SI RAM to implement IDL, GCI, PCM highway, and many other options. Additionally, on the QUICC, two TDM interfaces are simultaneously available, not just one. Thus, on the QUICC, the MS bits effectively become the two enable bits, ENa and ENb, in the SIGMR.

The MSC2 and MSC3 bits are generalized on the QUICC to the SC bits in the SICR. On the QUICC, one SC bit exists for each SCC.

The DRB and DRA bit functions are implemented by programming the QUICC SI RAM.

The B1RB and B1RA bit functions are implemented by programming the QUICC SI RAM.

The B2RB and B2RA bit functions are implemented by programming the QUICC SI RAM.

The SDC1 and SDC2 bit functions as well as many other options are implemented by programming the QUICC SI RAM. Note that four strobe outputs are available on the QUICC, not just two.

The two SDIAG bits are located in the QUICC SIMODE register and are renamed the SDM bits. Furthermore, on the QUICC, two sets of SDM bits exist to support TDMa and TDMb.

The SYNC bit function is implemented by programming the QUICC SI RAM.

The SCIT bit function is called the GM bit in the QUICC SIMODE register, and is available on both TDMa and TDMb.

The SETZ bit function is called the STZ bit in the QUICC SIMODE register, and is available on both TDMa and TDMb.

NOTE

The QUICC does not require the use of the L1SY0 and L1SY1 pins to synchronize the SCC to a given time slot. This is now handled by the time slot assigner using an external frame sync. Thus, the QUICC contains other programming options to specify the desired behavior of the frame sync and the associated external clocks. Examples are the FE, CE, CRT, DSC, and FSD bits in SIMODE.

9.4 USING THE QUICC MC68040 COMPANION MODE

The following paragraphs describe how to increase the performance of a QUICC-based system by using an MC68EC040 to replace the CPU32+ core on the QUICC. When the CPU32+ core on the QUICC is disabled, the QUICC is in slave mode. In slave mode, another external processor may be used instead of the on-chip CPU32+ core. The QUICC, however, has special features for providing a glueless interface to an external MC68EC040 (or other M68040 family member). This mode is called MC68040 companion mode.

The following paragraphs discuss both the hardware and software issues concerning this solution in a system that contains one MC68EC040 and one QUICC. It is also possible to interface more than one QUICC to an MC68EC040. This topic is discussed briefly in paragraph 9.4.4 Interfacing Multiple QUICCs to an MC68EC040**.**

9.4.1 MC68EC040 to QUICC Interface

The following paragraphs discuss the hardware and software issues relating to the connection between the MC68EC040 and the QUICC. The features of the QUICC that may be used to assist the MC68EC040 are also detailed. Reference Figure 9-8 during this discussion.

9.4.1.1 MC68EC040 READS AND WRITES TO QUICC. The basic connection is made through the data and address bus. All 32 data lines are routed between devices, which is required for the connection. In slave mode, the QUICC is not allowed to use its 16-bit data bus mode.

Twenty-eight address lines are routed between devices, giving a 256-Mbyte shared address capability. It is possible to share all 32 address lines between devices, but the QUICC would then lose its write enable lines ($\overline{WE3}$–$\overline{WE0}$). These lines are very useful in memory interfaces, and are used in this application.

In the MC68040 companion mode, the QUICC provides a number of signal changes to accommodate the MC68EC040. These changes allow direct connection between the MC68EC040 and the QUICC. These QUICC signals carry the same names as the comparable MC68EC040 signal: transfer start ($\overline{TS}$), transfer acknowledge ($\overline{TA}$), transfer exception acknowledge ($\overline{TEA}$), transfer burst inhibit ($\overline{TBI}$), transfer type (TT1–TT0), and transfer mode (TM2–TM0).

NOTE

The QUICC monitors the address bus in slave mode to perform memory controller functions. Therefore, the user should never gate 040 bus control signals (i.e., $\overline{TA}$, $\overline{TS}$ signals) between 040 and QUICC, even if the address output by the 040 is not controlled or used by the QUICC.

In addition, the QUICC R/$\overline{W}$ and SIZ1–SIZ0 signals that are used in all configurations are directly connected to the MC68EC040. When the MC68EC040 begins an access, the QUICC interprets the SIZx pins, using the MC68EC040 encoding rather than the QUICC encoding.

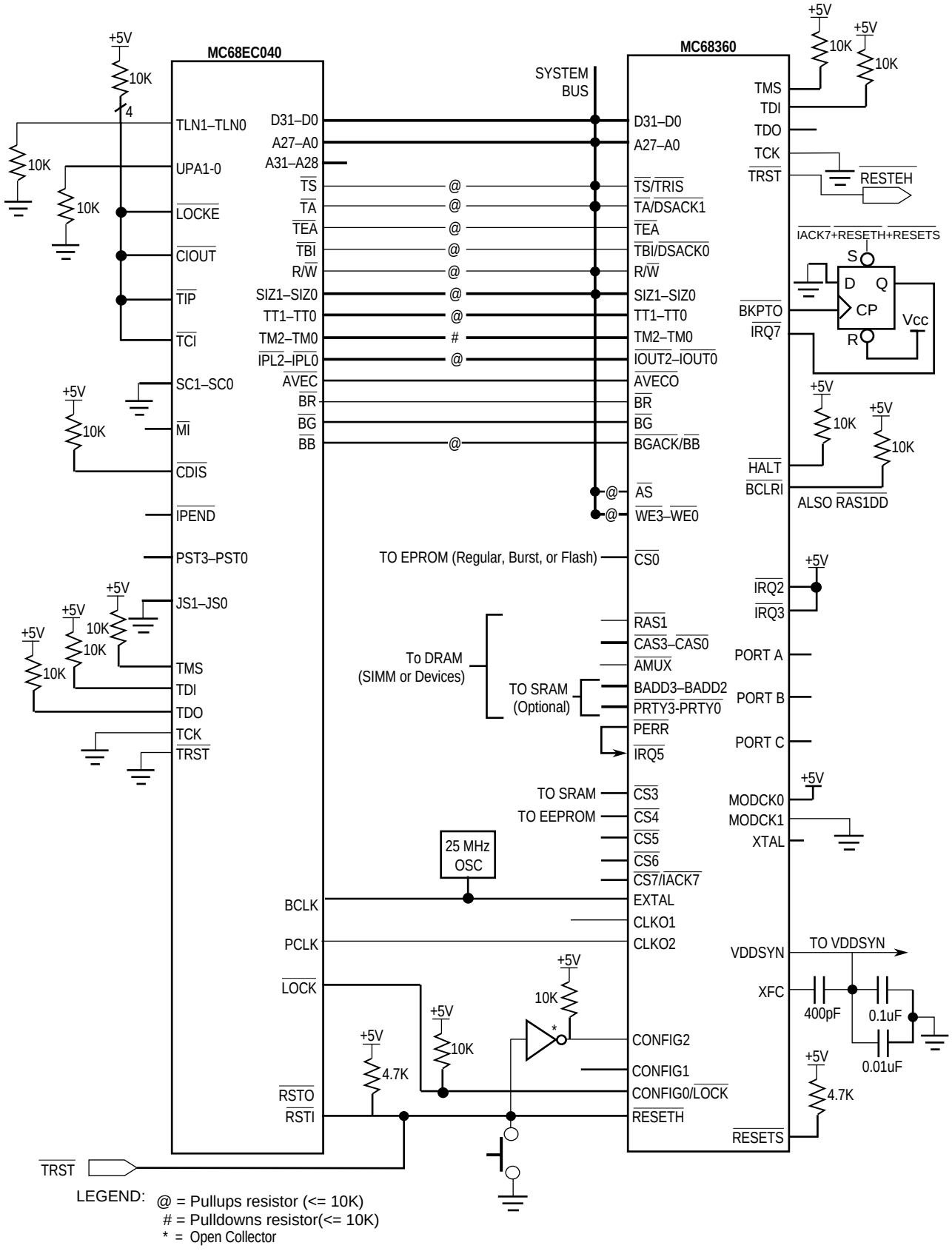


Figure 9-8. MC68EC040 to QUICC Interface

9.4.1.2 CLOCKING STRATEGY. In this application, the system clock is generated from a 25-MHz external oscillator into the QUICC. The MODCKx pins are configured for this. The QUICC internal PLL then multiplies the frequency by 2, and outputs 25 MHz on CLKO1 (not used) and 50 MHz on CLKO2 (with minimal skew between EXTAL, CLKO1, and CLKO2). Minimal skew is very important in this application.

The oscillator is connected to the MC68EC040 BCLK, and the QUICC EXTAL and CLKO2 are connected to the MC68EC040 PCLK. Why not connect the oscillator to the QUICC EXTAL pin and connect the QUICC CLKO1 to the MC68EC040 BCLK? The answer is that the CLKO1 signal does not oscillate while the PLL is locking during power-on reset, and the MC68EC040 needs continuous clocks, even during reset. (CLKO2, however, does oscillate at the EXTAL frequency during power-on reset.) Thus, this solution provides continuous clocks to the MC68EC040 at all times.

An external low-frequency crystal cannot be used in this design because the CLKO1 and CLKO2 pins will stop oscillating while the PLL relocks, which would occur when the software writes to the PLL to multiply the system frequency up to 25 MHz. This would violate the MC68EC040 minimum operating frequency requirement of 16 MHz.

The QUICC clocking section allows for the clock oscillator to be kept running through the VDDSYN pin in a power-down situation. Low-power modes should not be used in this design due to the MC68EC040 requirement of a minimum operating frequency of 16 MHz.

9.4.1.3 RESET STRATEGY. If a QUICC is configured to provide the global chip select, it will also provide an internal power-on reset generation. Thus, the $\overline{\text{RESET}}$ pin of the QUICC just needs to be connected to the $\overline{\text{RSTI}}$ pin on the MC68EC040. The $\overline{\text{RSTO}}$ pin on the MC68EC040 is not used in this design; thus, the MC68040 RESET instruction will have no effect. (It could be added by using an external reset circuit to reset the MC68EC040 and to connect the $\overline{\text{RSTO}}$ to the $\overline{\text{RESET}}$ pin on the QUICC through an open-drain gate.) If a push-button switch is needed, it can be connected by an open-drain buffer to the hard reset ($\overline{\text{RESETH}}$) line, once debounced. The soft reset ($\overline{\text{RESETS}}$) line is not used in this design. It could be used to reset the QUICC without resetting the parallel I/O pins, chip selects, etc.

9.4.1.4 INTERRUPTS. External interrupts may be brought into the QUICC through either the $\overline{\text{IRQx}}$ pins or parallel I/O pins. The QUICC prioritizes these interrupts with its own internally generated interrupts (e.g., timers) to obtain the current highest pending request. In slave mode, the QUICC can output this request to another processor in the system.

NOTE

When the QUICC is in slave mode, the user should *not* use an $\overline{\text{IRQx}}$ pin that is on an interrupt level occupied by either the CPM, periodic interrupt timer, or sofware watchdog.

The request can take the form of a single request pin ($\overline{\text{RQOUT}}$) or three request pins ($\overline{\text{IOUT2}}$–$\overline{\text{IOUT0}}$) that encode the priority of the request. Since the MC68EC040 uses encoded inputs ($\overline{\text{IPL2}}$–$\overline{\text{IPL0}}$), the $\overline{\text{IOUT2}}$–$\overline{\text{IOUT0}}$ pins are chosen.

NOTE

If the MC68040 were used rather than the MC68EC040, the IPL2–IPL0 pins take on meanings during reset to determine the buffer strength. Thus, additional logic may be required during reset to select the desired buffer strength of the MC68040. This logic does not exist on the MC68EC040 (except on very early versions of that device).

In addition, the QUICC allows the $\overline{IOUT2}$–$\overline{IOUT0}$ pins to be generated in two different ways: at the expense of parity pins or at the expense of some interrupt requests. In this application, parity is used with some of the external memories; thus, three of the interrupt inputs are sacrificed.

Once the MC68EC040 recognizes the interrupt, it responds with an interrupt acknowledge cycle. The QUICC recognizes the MC68EC040 interrupt acknowledge cycle using the address pins, TT1–TT0 pins (they are both high), and the TM2–TM0 pins (which output the inverse of the value on $\overline{IPL2}$–$\overline{IPL0}$). If the source of the interrupt is the CPM, the periodic interrupt timer, or the software watchdog, the QUICC responds by placing the vector on the bus. If the source of the interrupt is an $\overline{IRQx}$ pin, the QUICC then responds by outputting the $\overline{AVECO}$ signal to the MC68EC040, depending on what was programmed into the autovector register in the SIM60.

When the QUICC is in slave mode, what is normally the $\overline{AVEC}$ pin (an input) becomes the $\overline{AVECO}$ pin (an output) for use with external processors.

9.4.1.5 BUS ARBITRATION. When the QUICC is configured to MC68040 companion mode, the QUICC bus arbitration pins are configured to directly interface to those of the MC68EC040. The QUICC bus request ($\overline{BR}$) is an input from the MC68EC040, and the QUICC bus grant ($\overline{BG}$) is an output to the MC68EC040. The QUICC also has a bus busy ($\overline{BB}$) function to match that of the MC68EC040.

The MC68EC040 $\overline{LOCK}$ pin can be connected to the QUICC to allow MC68EC040 locked cycles to continue without interruption from the QUICC. The CONFIG0 pin on the QUICC becomes the $\overline{LOCK}$ input to the QUICC after system reset when the QUICC is in MC68040 companion mode. Since the MC68EC040 $\overline{LOCK}$ pin is inactive high, it may be connected directly to the QUICC $\overline{LOCK}$ pin with a pullup resistor.

The $\overline{BR}$ input from the MC68EC040 is not the highest priority in this system. The SDMA channels have a higher priority. Thus, the $\overline{BCLRO}$ function of the QUICC is not used in this design because it is not needed.

The QUICC also has a $\overline{BCLRI}$ signal that allows internal masters to be cleared off the bus. This pin can be used with the MC68EC040 $\overline{IPEND}$ pin to give the MC68EC040 interrupts priority over the QUICC internal masters. This function is not implemented in the design for the sake of simplicity and use of the alternate function of the pin, the $\overline{RAS1DD}$ function. However, if it were implemented, the $\overline{IPEND}$ signal from the MC68EC040 would have to be latched and kept low by the MC68EC040 until completion of the interrupt routine. (If it was not latched, the QUICC could take the bus back from the MC68EC040 as soon as the inter-

rupt acknowledge cycle was complete, which defeats the purpose of connecting the $\overline{\text{IPEND}}$ pin to the $\overline{\text{BCLRI}}$ pin.)

When the QUICC does not need the bus, it will assert the $\overline{\text{BG}}$ pin to the MC68EC040 continually, improving the MC68EC040 memory access performance.

9.4.1.6 BREAKPOINT GENERATION. In MC68EC040 companion mode, the QUICC can be used to generate a breakpoint signal using its breakpoint address register. This register will respond to QUICC or MC68EC040 accesses, but only the first access of an MC68EC040 burst access.

The result of a breakpoint is the assertion of the $\overline{\text{BKPTO}}$ pin on the QUICC. In this application, it was decided to route this output back to an interrupt input on the QUICC and generate a nonmaskable interrupt to the MC68EC040. If the MC68EC040 encounters the breakpoint trap, it will respond with a breakpoint acknowledge cycle. The QUICC does not recognize this cycle and takes no action.

9.4.1.7 BUS MONITOR FUNCTION. In MC68EC040 companion mode, the QUICC will monitor the bus for bus cycles that are not properly terminated. The cycles can originate from the QUICC or the MC68EC040. If the MC68EC040 originates such a cycle and that cycle times out without a $\overline{\text{TA}}$ or $\overline{\text{TEA}}$ occurring, the QUICC will assert $\overline{\text{TEA}}$ back to the MC68EC040 to end the cycle.

9.4.1.8 SPURIOUS INTERRUPT MONITOR. In MC68EC040 companion mode, the QUICC will watch for spurious interrupt cycles generated by the MC68EC040, but only on the interrupt levels that the QUICC supports internally (e.g., the levels used by the SIM60 and the level of the CPM). If such a condition occurs, $\overline{\text{TEA}}$ will be asserted by the QUICC.

9.4.1.9 SOFTWARE WATCHDOG. If desired, the MC68EC040 can program the QUICC software watchdog to generate a level 7 interrupt or a system reset. In this application, the software watchdog is configured in software to generate a reset so that the breakpoint logic can use level 7 interrupts. No additional hardware is required because the connection between the reset pins of the QUICC and the MC68EC040 is already made.

9.4.1.10 PERIODIC INTERVAL TIMER. If desired, the MC68EC040 can use the periodic interval timer on the QUICC to generate a system interrupt, such as for a real-time kernel. No additional hardware is required for this function.

9.4.1.11 MC68EC040 CACHING CONFIGURATION. The MC68EC040 can cache or not cache data and can program memory as desired. However, it is strongly advisable not to cache the data that is accessed by the QUICC serial channels because of the overhead incurred every time a cached data area is written.

The MC68EC040 can decide to cache or not cache 16-byte portions of a bank of memory on a dynamic basis by using a software technique summarized in 9.5 Selecting Cache Modes on the MC68EC040.

9.4.1.12 MC68EC040 SNOOPING. The snooping function is not supported since only one active processor exists in the system and the QUICC data buffers are not cached.

9.4.1.13 DOUBLE BUS FAULT. In MC68040 companion mode, the QUICC double bus fault monitor is not operational.

9.4.1.14 JTAG AND THREE-STATE. Both the MC68EC040 and the QUICC provide JTAG test access ports, commonly known as JTAG. This interface uses five pins: TMS, TDI, TDO, TCK, and $\overline{\text{TRST}}$. TMS and TDI are left unconnected because they have internal pullups. The JTAG ports of both parts are disabled in this application; however, the capability could be easily added.

When the QUICC is in master mode, it provides a $\overline{\text{TRIS}}$ pin that allows all outputs on the device to be three-stated. In slave mode, this feature is not available since the QUICC is a peripheral of the system. Thus, the transfer start ($\overline{\text{TS}}$) pin is available instead of the $\overline{\text{TRIS}}$ pin and does not conflict with it.

9.4.1.15 QUICC SERIAL PORTS. The functions on QUICC parallel I/O ports A, B, and C may be used as desired in this application and have no bearing on the MC68EC040 interface. However, any unused parallel I/O pins should be configured as outputs so they are not left floating.

9.4.2 Memory Interfaces

In this application, a number of memory arrays have been developed for EPROM, burst EPROM, flash EPROM, EEPROM, SRAM, burst SRAM, and DRAM. Each memory interface can be attached to the system bus as desired.

One issue not discussed is the decision of whether external buffers are needed on the system bus. This issue depends on the number of memory arrays used in the design and possibly the layout (i.e., capacitance) of the system bus.

Another issue left to the user is the number of wait states used with each memory system. This depends on the memory speed, whether external buffers are used, and the loading on the system bus pins. (The QUICC provides capacitance de-rating figures to calculate the effect of more or less capacitance on the AC Timing Specifications.)

9.4.2.1 QUICC MEMORY INTERFACE PINS. In this design, a number of QUICC pins are available to the memory arrays (see Figure 9-8). These pins are active, regardless of whether the bus cycle was originated by the MC68EC040 or by one of the QUICC DMA cycles. The QUICC detects the MC68EC040 bus cycle by the $\overline{\text{TS}}$ pin. If the QUICC generates the bus cycle, the QUICC asserts the $\overline{\text{AS}}$ pin.

Eight $\overline{\text{CSx}}$ or $\overline{\text{RASx}}$ pins are available in the system. In this design, $\overline{\text{CS0}}$ is used for any of the EPROM arrays since it is the global (boot) chip select. $\overline{\text{RAS1}}$ is used for the DRAM arrays because of its double-drive capability. $\overline{\text{CS2}}$/$\overline{\text{RAS2}}$ is not used in the design and is available for other purposes, such as a second DRAM bank. $\overline{\text{CS3}}$ is for SRAM arrays; $\overline{\text{CS4}}$ is for EEPROM. $\overline{\text{CS5}}$, $\overline{\text{CS6}}$, and $\overline{\text{CS7}}$ are unused.

In this design, it is assumed that the full 32-bit capability of the MC68EC040 is used; thus, all memory arrays are 32 bits wide. (The only exception to this is the EEPROM, which is han-

dled differently. See 9.4.2 Memory Interfaces.) To provide dynamic bus sizing, the MC68150 device may be added to the design shown here.

Parity is supported for both SRAM and DRAM arrays using the four-byte parity lines PRTY3–PRTY0. When a parity error occurs, the error indication on the $\overline{\text{PERR}}$ pin causes the QUICC to generate a level 5 interrupt to the MC68EC040. (Level 7 has already been used for the breakpoint generation interrupt.) The parity error timing is not fast enough to allow an MC68EC040 bus error to be generated on the bus cycle that generated a parity error.

The QUICC supports MC68EC040 bursting using the BADD3–BADD2 pins. These pins normally reflect the values on A3–A2, but, in the case of a burst, are used to increment the address to the memory array. If the memory devices already support MC68EC040 bursting internally, the BADD3–BADD2 pins are not required.

The DRAM arrays require the four $\overline{\text{CAS3}}$–$\overline{\text{CAS0}}$ pins. Also, since an external address multiplexer is used, the AMUX pin is required to select between rows and columns. If, however, the user's configuration does not require DRAM, the AMUX pin can be used as an $\overline{\text{OE}}$ pin instead. This would save an inverter in a number of memory arrays.

NOTE

Many memory arrays show an inverter on the R/$\overline{\text{W}}$ pin to create the $\overline{\text{OE}}$ signal. When using multiple memory arrays, it is possible to share one inverter between multiple memory arrays; however, this configuration is not shown.

The QUICC also provides four write enable ($\overline{\text{WEx}}$) pins to select the correct byte during write operations.

9.4.2.2 REGULAR EPROM. Figure 9-9 shows the glueless interface to standard EPROM in the system. The assumption is made that only the MC68EC040 will access this array. No bursting capability is used. The CONFIG2–CONFIG0 pins are configured to initialize the system to slave mode, $\overline{\text{CS0}}$ operating on a 32-bit port at reset, the MBAR at its normal location, and MC68040 companion mode.

It would have been possible to use 16-bit-wide EPROM to reduce the chip count, if desired. (See 9.4.2.3 Burst EPROM. for an example.)

9.4.2.3 BURST EPROM. Figure 9-10 shows the glueless interface to two burst EPROMs available from National Semiconductor. These devices support a glueless interface to the MC68040. In this design, the assumption is made that only the MC68EC040 will access this array.

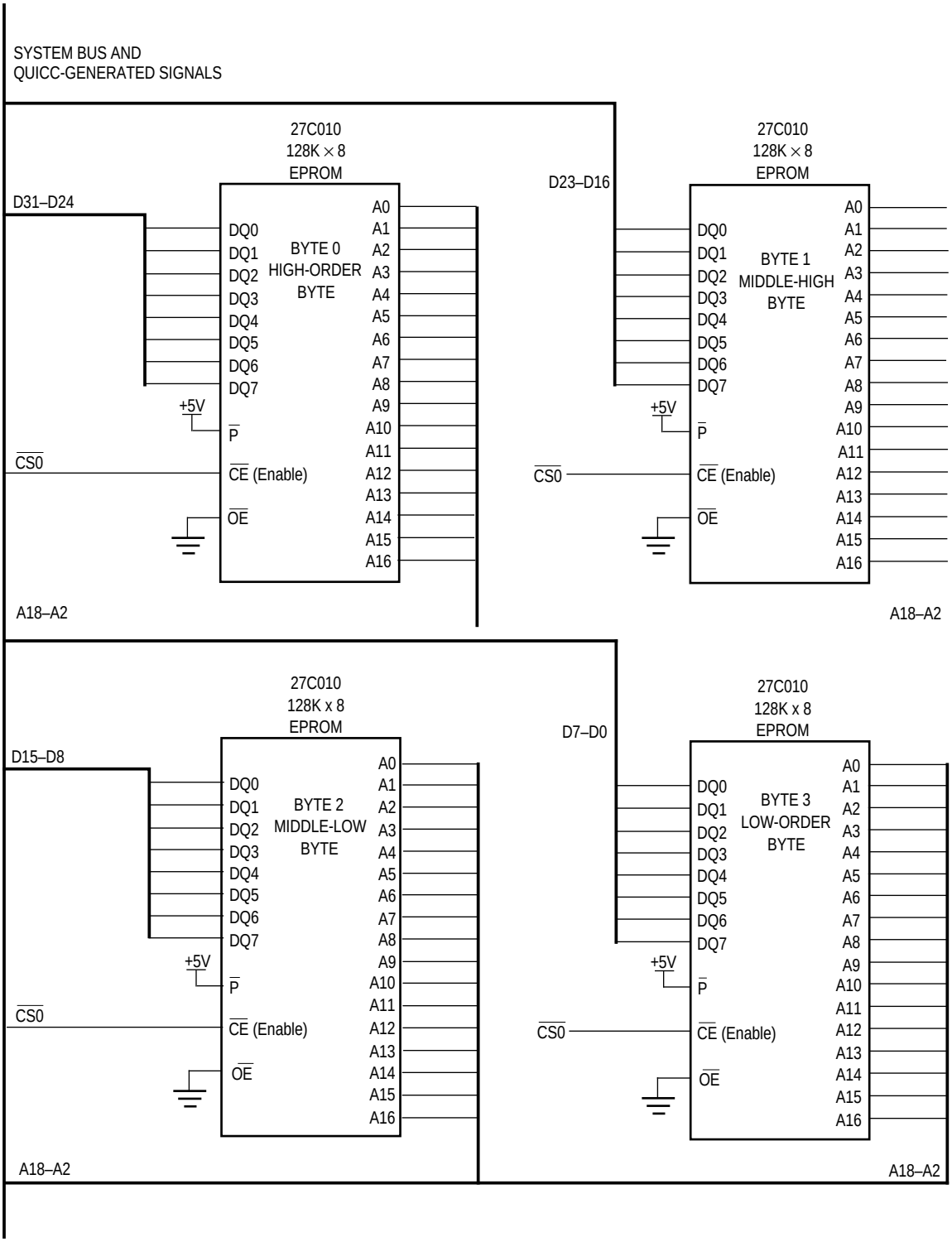


Figure 9-9. 512-Kbyte EPROM Bank—32 Bits Wide

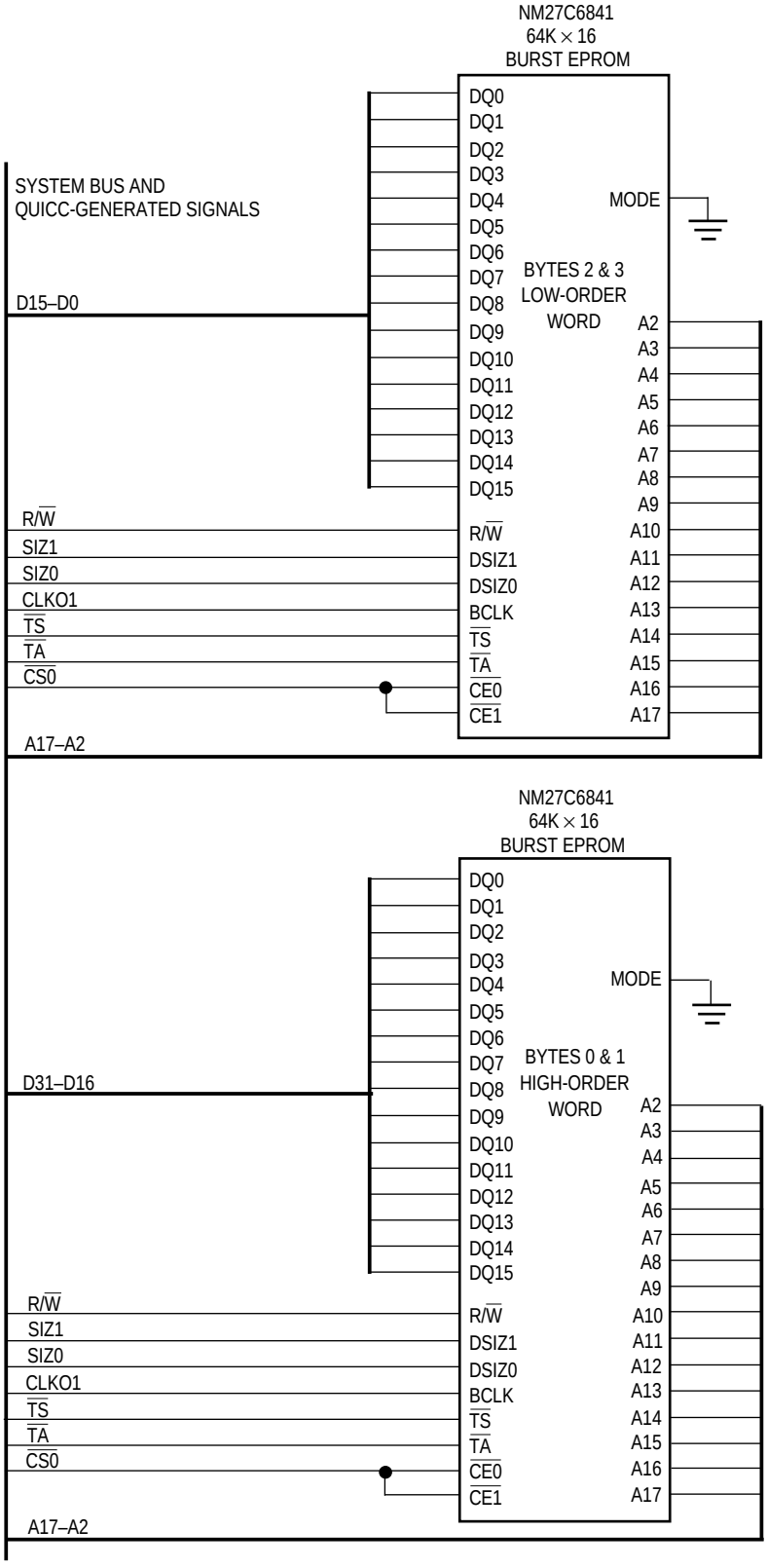


Figure 9-10. 256-Kbyte Burst EPROM Bank—32 Bits Wide

9.4.2.4 FLASH EPROM. Figure 9-11 shows the interface to flash EPROM devices. In this design, the assumption is made that only the MC68EC040 will access this array. The inverter is only required if DRAM is used elsewhere in the system, since the $\overline{OE}$ function is lost when the AMUX pin is used. This design assumes that the write operations are $\overline{CE}$ controlled, rather than $\overline{WE}$ controlled. Most flash EPROM manufacturers now support this alternative timing method.

9.4.2.5 REGULAR SRAM. Figure 9-12 shows the interface to SRAM. In this design, both the MC68EC040 and the QUICC may access the SRAM array. The inverter is only required if DRAM is used elsewhere in the system, since the $\overline{OE}$ function is lost when the AMUX pin is used. This design also allows the QUICC to support bursting by the MC68EC040 using the BADD3–BADD2 signals. The QUICC also uses these signals as A3–A2 address lines during its normal SRAM accesses. If MC68EC040 bursting is not supported, the BADD3–BADD2 signals can be replaced with the A3–A2 signals.

9.4.2.6 BURST SRAM. Figure 9-13 shows the interface to Motorola's 32K $\times$ 9 MCM62940A bursting SRAM. This can provide better memory throughput speeds than the regular SRAM array. Figure 9-13 shows the solution for an array that is accessed by the MC68EC040 and the QUICC.

To speed access times, the chip select pin (S0) of the burst SRAM is connected directly to the A27 pin of the system bus. This gives half of the usable address space to this array, although this should not be a problem in most systems considering the 28 address pins available. No chip select pin from the QUICC is used.

The inverter on the R/$\overline{W}$ signal is only required if DRAM is used elsewhere in the system, since the $\overline{OE}$ function is lost when the AMUX pin is used. The CLKO1 signal is derived directly from the QUICC CLKO1 pin. Parity is also supported in the array, using the PRTY3–PRTY0 signals on the system bus.

Although not used in this design, Motorola also offers a larger version of the MCM62940A, called the MCM67M618, that is organized into a 64K $\times$ 18 array.

In Figure 9-13, the $\overline{TSC}$ pin is connected to the $\overline{AS}$ pin to handle the QUICC accesses. The QUICC accesses do not use the bursting feature of the MCM62940A. $\overline{BAA}$, which could be asserted during the QUICC accesses due to the QUICC $\overline{DSACK1}$ being multiplexed with the MC68EC040 $\overline{TA}$, is a don't care as long as $\overline{TSC}$ remains low. After $\overline{TSC}$ negated, the $\overline{OE}$ and $\overline{WE}$ signals are negated, disabling the MC62940A burst

.

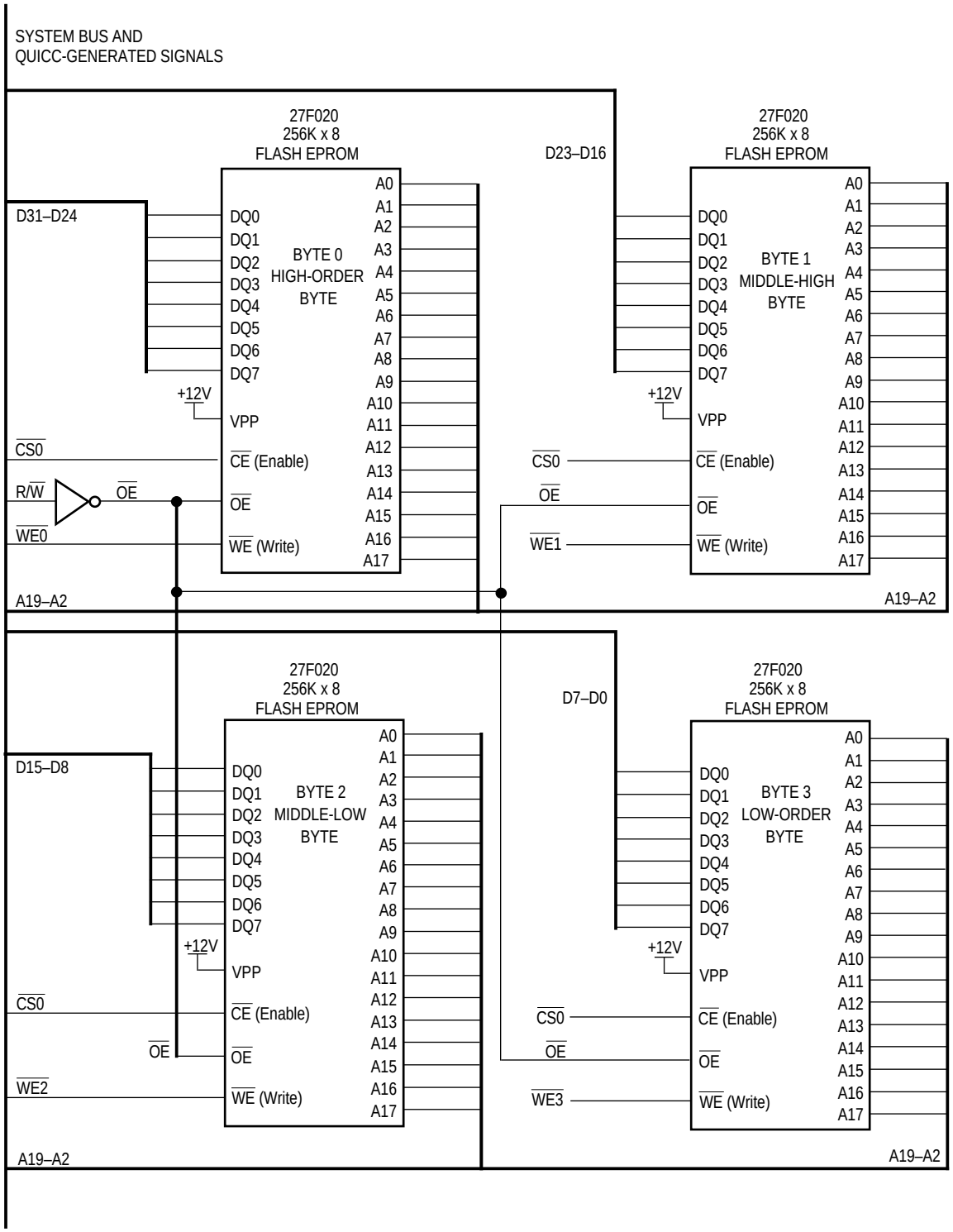


Figure 9-11. 1-Mbyte Flash EPROM Bank—32 Bits Wide

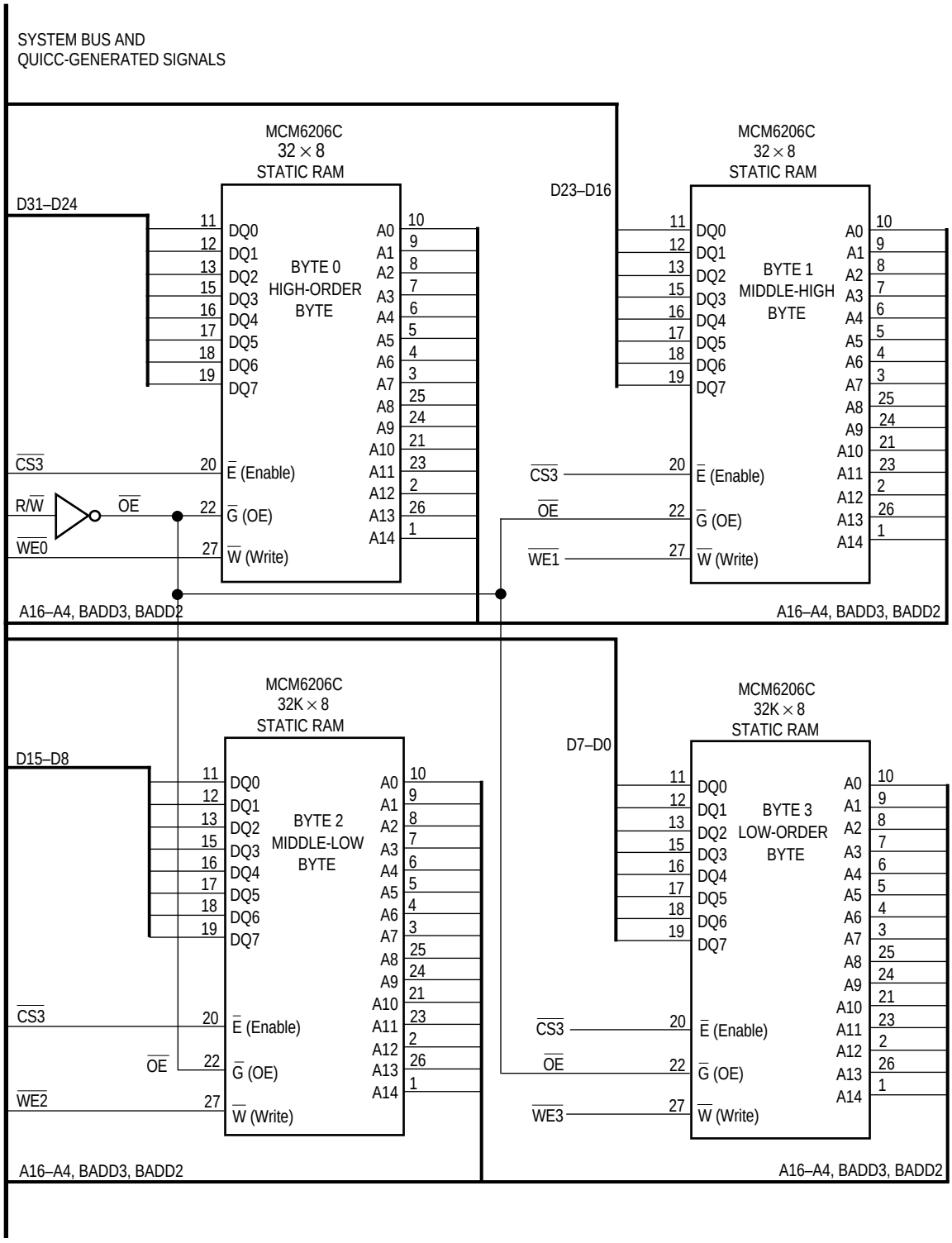


Figure 9-12. 128-Kbyte Static RAM Bank—32 Bits Wide

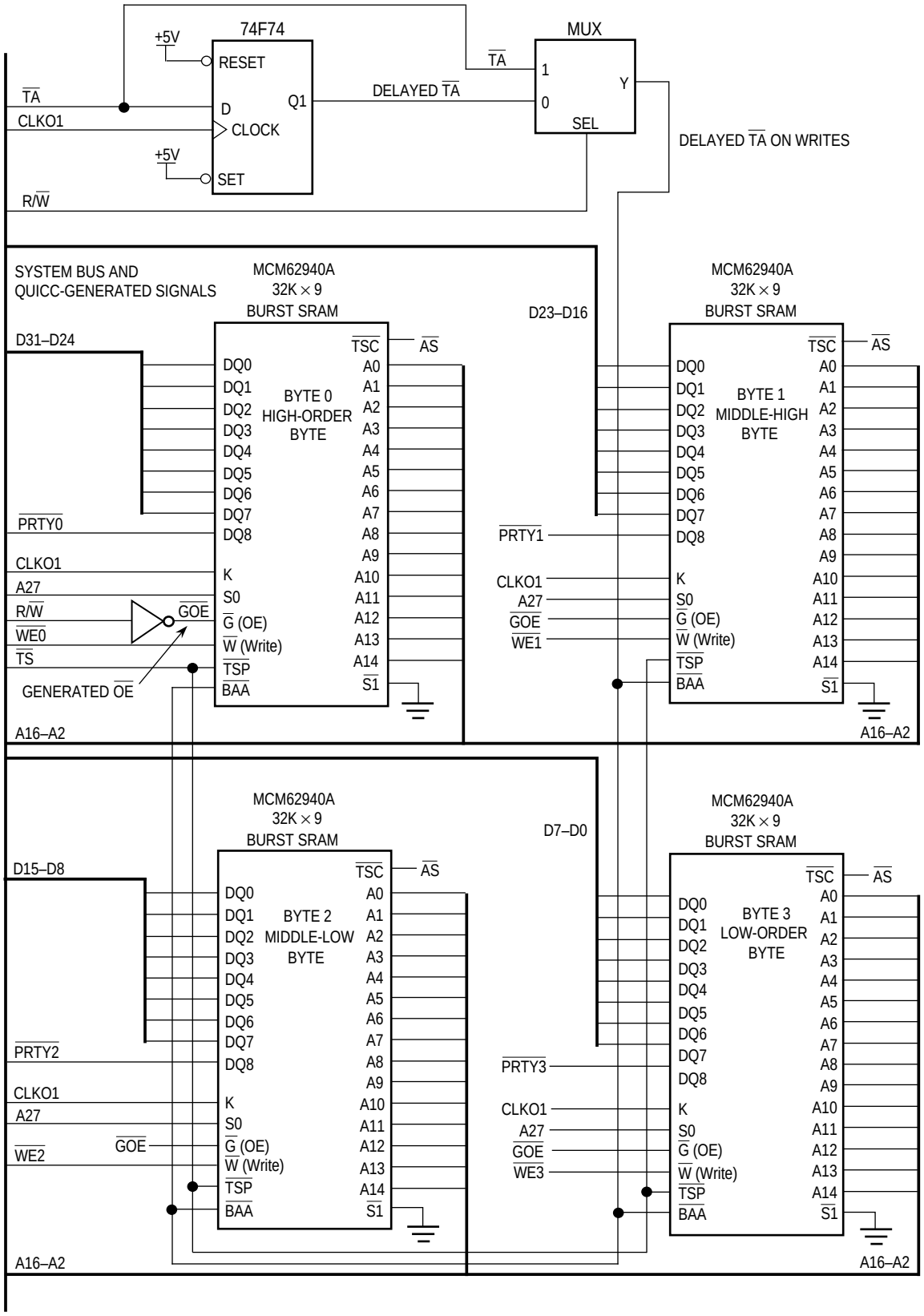


Figure 9-13. 128-Kbyte Burst SRAM Bank—32 Bits Wide

9.4.2.7 EEPROM. Figure 9-14 shows the interface to an EEPROM device to give a small amount of nonvolatile storage. Although both the MC68EC040 and the QUICC may access the EEPROM, it is most likely that only the MC68EC040 will access the EEPROM. The inverter is only required if DRAM is used elsewhere in the system, since the $\overline{OE}$ function is lost when the AMUX pin is used.

The EEPROM device sits on just the D7–D0 data pins. Thus, when the MC68EC040 accesses the EEPROM, the only valid data that is read or written is on the D7–D0 pins (i.e., byte 3). Thus, reads or writes to successive locations of the EEPROM will require the address to be incremented by 4 each time. Additionally, the $\overline{CS4}$ pin should be programmed to respond to a full 32-Kbyte area, even though only 8 Kbytes are present.

After a write is made, software is responsible for waiting the appropriate time (e.g., 10 ms) or for performing data polling to see if the newly written data byte is correct.

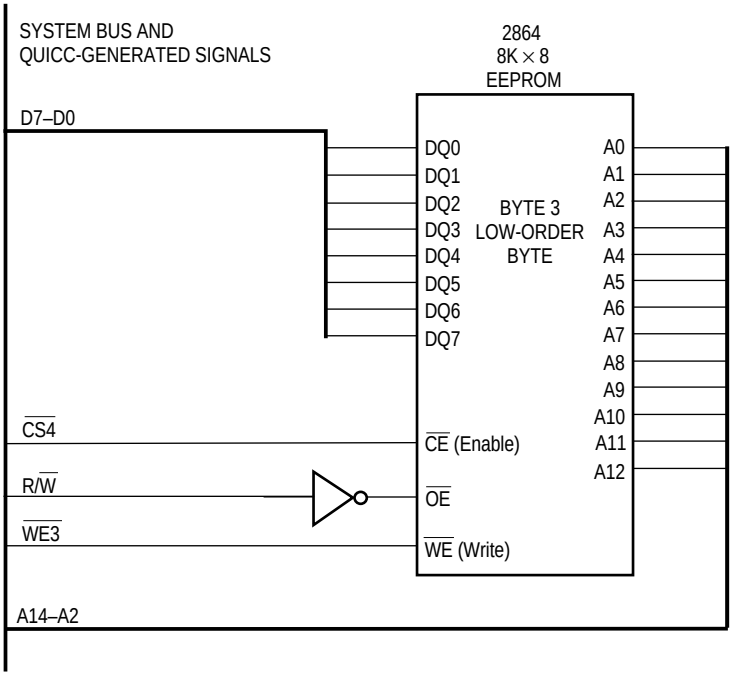


Figure 9-14. 8-Kbyte EEPROM Bank—8 Bits Wide

9.4.2.8 DRAM SIMM. Figure 9-15 shows the interface to an MCM36100S DRAM single in-line memory module (SIMM). Both the MC68EC040 and the QUICC can access the DRAM.

When the QUICC is a slave to an external MC68EC040, the address multiplexing for the DRAM must be done externally to the QUICC, which is accomplished in the three F157 multiplexers. The external address multiplexing scheme was chosen to allow the QUICC to provide burst accesses by the MC68EC040, using the BADD3–BADD2 pins. The QUICC can also use these pins as its A3–A2 address lines during its own accesses to the DRAM. In addition, two of the multiplexer outputs are unused in this design, allowing expansion to

future 16M × 36 DRAM SIMMs. In fact, this multiplexing scheme allows SIMMs of many different sizes to be used on the board without hardware modification.

This particular SIMM also includes parity support, supported with the PRTY3–PRTY0 signals.

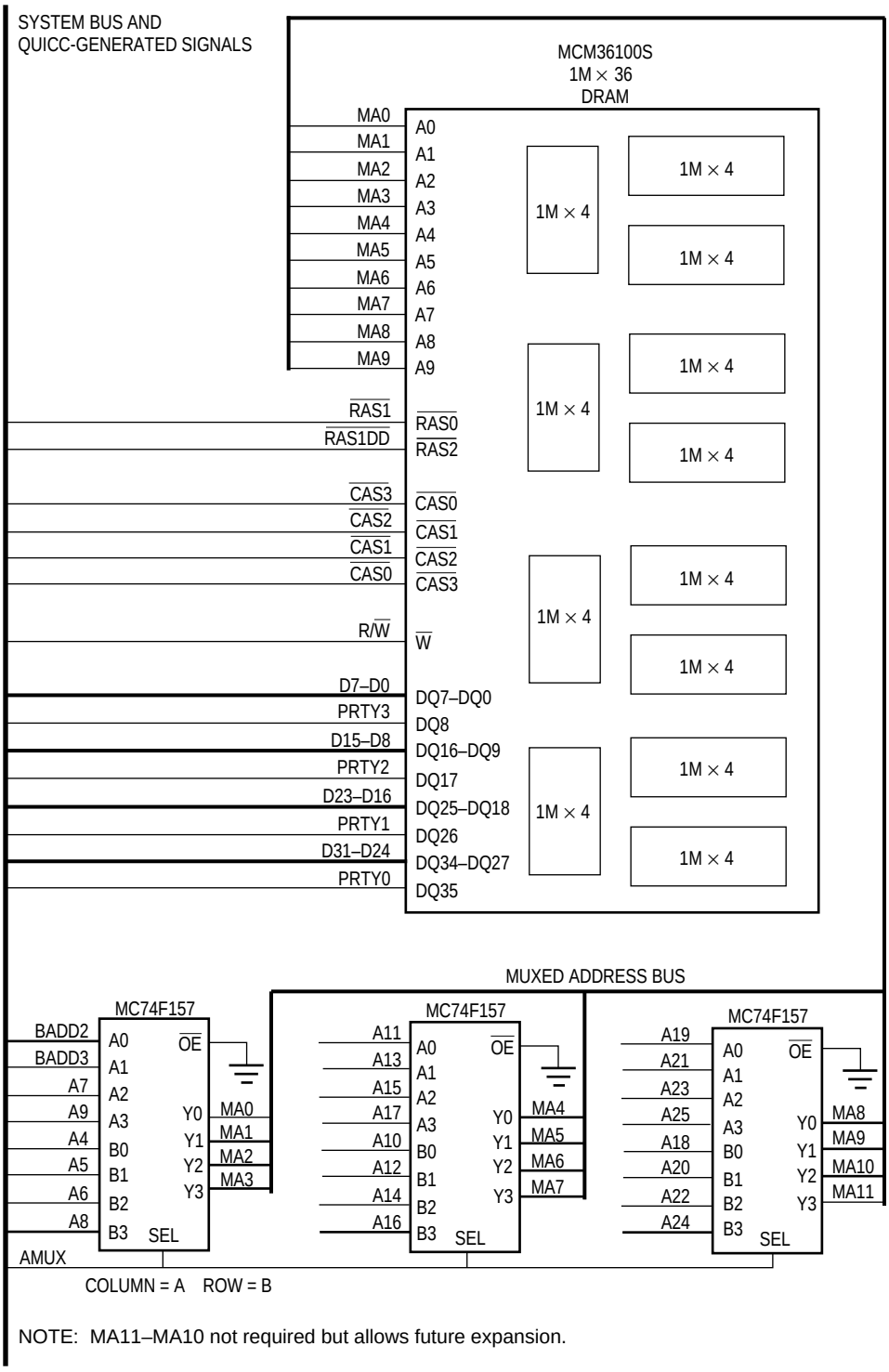
This design also uses the $\overline{\text{RAS1}}$ double-drive capability, whereby the $\overline{\text{RAS1DD}}$ signal is output by the QUICC to increase the effective drive capability of the $\overline{\text{RAS1}}$ signal. The $\overline{\text{RAS1}}$ line should be programmed to respond to a 4-Mbyte address space.

After power-on reset, the software must wait the required time before accessing the DRAM. The required eight read cycles must then be performed either in software or by waiting for the refresh controller to perform these accesses.

9.4.2.9 DRAM DEVICES. Figure 9-16 shows the interface to a standalone DRAM device. In this case, the MCM54260 256K × 16 DRAM device is chosen. This allows a full 32-bit-wide DRAM solution using only two DRAM devices, with byte writes still supported using the upper and lower $\overline{\text{CASx}}$ pins. Both the MC68EC040 and the QUICC can access the DRAM array. The $\overline{\text{RAS1}}$ line should be programmed to respond to a 1-Mbyte address space.

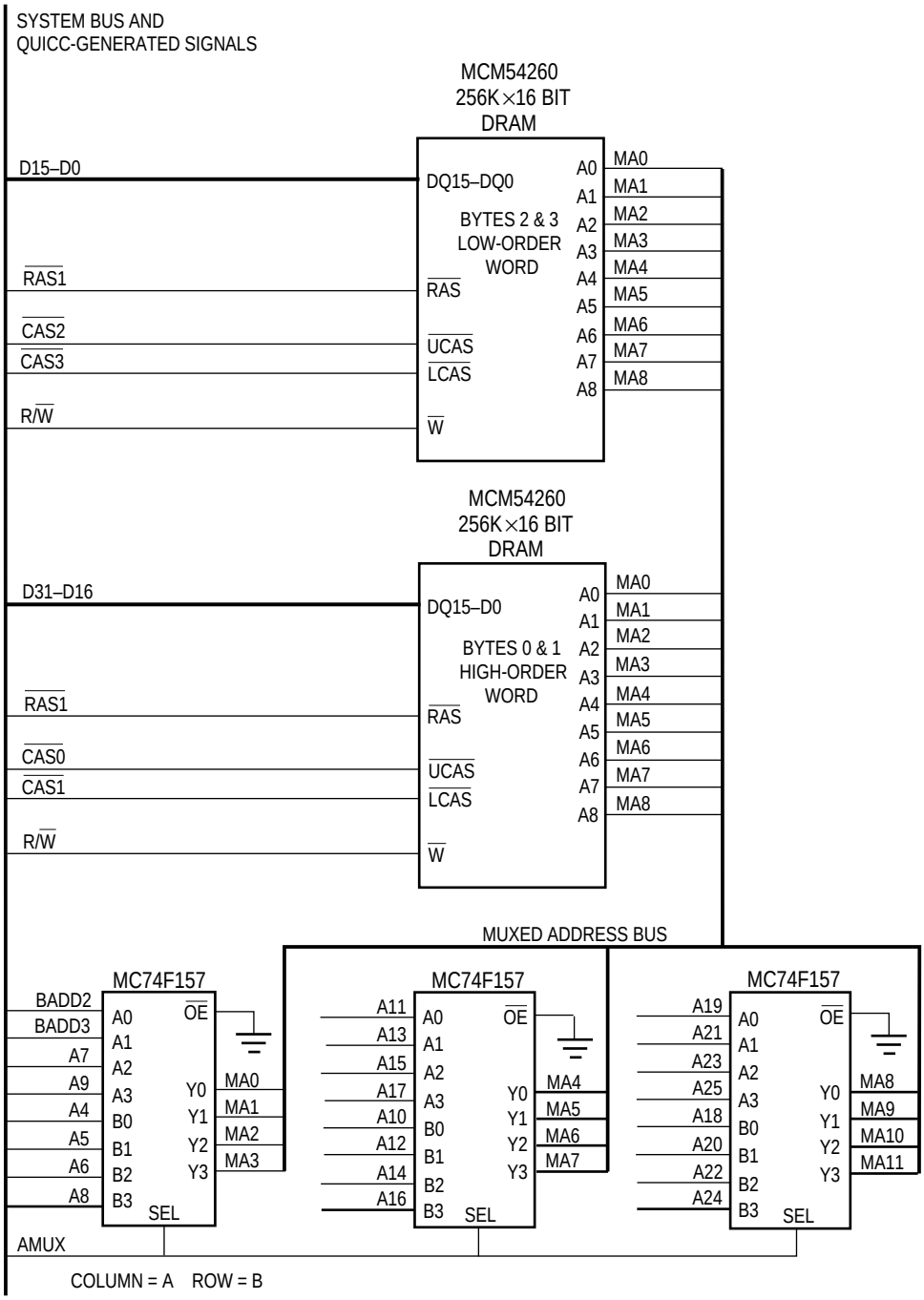
The address multiplexing scheme is the same as that for the DRAM SIMM. No parity support is provided in this case. The $\overline{\text{RAS1DD}}$ signal is not used in this case since only two devices are supported.

After power-on reset, the software must wait the required time before accessing the DRAM (approximately 100 μs). The required eight read cycles must then be performed either in software or by waiting for the QUICC refresh controller to perform these accesses.



NOTE: MA11–MA10 not required but allows future expansion.

Figure 9-15. 4-Mbyte DRAM Bank—36 Bits Wide



NOTE: MA11–MA9 not required but allows future expansion.

Figure 9-16. 1-Mbyte DRAM Bank—32 Bits Wide

9.4.3 Software Configuration

The following paragraphs discuss a number of key points for to a software writer desiring to initialize the system. The points discussed are those that are required to enable the previously mentioned hardware configurations.

9.4.3.1 BASIC INITIALIZATION. The following paragraphs describe basic software initialization.

The module base address register (MBAR) should be set as desired. However, the QUICC 8-Kbyte block of address space should not overlap any memory array.

The module base address register enable (MBARE) should not be accessed.

In the module configuration register (MCR), ASTM and BSTM should be set. The user should program BCLROID2–BCLROID0 to $3 and program the SDMA, IDMA1, and IDMA2 arbitration IDs in the SDMA configuration register and IDMA channel configuration registers to 4, 2, and 0, respectively, to allow the SDMA and DRAM refresh to preempt IDMA transfers. SHEN1–SHEN0 should be left cleared. SUPV should be cleared. BCLRIID2–BCLRIID0 are not used and can be programmed to $0. IARB3–IARB0 can be left programmed to $F to allow SIM interrupts to have priority over CPM interrupts that occur at the same level.

In the system protection control register (SYPCR), DBFE should be cleared. BME should be set. If the software watchdog is not used, the SWE bit should be cleared. If the software watchdog is used, the SWRI bit should be left set to cause a system reset, rather than an interrupt.

The periodic interrupt control register (PICR) may be set as desired.

The port E pin assignment register (PEPAR) should be set to $31C0, which configures three $\overline{\text{IOUTx}}$ lines to go out on some of the $\overline{\text{IRQx}}$ pins, the $\overline{\text{RAS1DD}}$ pin, and the use of the $\overline{\text{WE}}$ lines instead of the A31–A28 lines. It also configures the AMUX pin (assuming DRAM is used in the system; otherwise, the $\overline{\text{OE}}$ function should be programmed), four $\overline{\text{CASx}}$ lines, $\overline{\text{CS7}}$, and $\overline{\text{AVECO}}$.

9.4.3.2 CONFIGURING THE MEMORY CONTROLLER. The following paragraphs describe configuring the memory controller registers.

The global memory register (GMR) should be configured as follows:

The RFCNT bits may be set as desired. At 25 MHz, an RFCNT value of 24 (decimal) gives one refresh every 15.6 μs.

RFEN should be set.

RCYC depends on the DRAM speed. At 25 MHz (an 80-ns DRAM SIMM), RCYC should be 00.

PGS2–PGS0 is not relevant since page mode and internal address multiplexing is not used.

DPS should be set to 00.

WBT40 depends on timing; it is usually cleared for 80-ns DRAM SIMMs.

WBTQ depends on timing; it is usually set for 80-ns DRAM SIMMs.

EMWS is not used in this design since there is only one QUICC. It should be cleared.

SYNC is not used in this design since there is only one QUICC. It should be cleared.

OPAR may be chosen by the user.

PBEE should be cleared because parity errors should not generate bus errors in MC68040 companion mode.

TSS40 should be set to meet MC68EC040 electrical specification 11. However, it may be cleared for faster operation if spec 11 is reduced from 30 ns to 25 ns because of a lightly loaded MC68EC040 bus.

NCS should normally be set.

DWQ should be cleared since page mode is not allowed in MC68040 companion mode.

DW40 depends on the timing analysis.

AMUX should be cleared.

The memory controller status register (MSTAT) is used for reporting write protect and parity errors and does not require initialization.

The eight base registers (BRs), one for each memory bank, should be configured as follows:

The BA27–BA11 bits may be set as desired. Different memory arrays should not overlap. BA31–BA28 should be cleared since the byte write lines are used with an external master in the system.

For simplicity, FC3–FC0 can be cleared.

TRLXQ should normally be cleared for memory interfaces.

BACK40 should be set if the QUICC provides bursting for the MC68EC040 accesses to standard SRAM and DRAM.

CSNT40 should normally be set.

CSNTQ should normally be cleared.

PAREN should be set for memory banks that use parity.

WP should be set for EPROM, burst EPROM, and flash EPROM; otherwise, it should be cleared.

V should be set if the memory bank is used.

The eight option registers (ORs), one for each memory bank, should be configured as follows:

The TCYC bits should be set to determine the number of wait states required.

The AM27–AM11 bits should be set to determine the block size of the chip select or $\overline{\text{RASx}}$ line. This should be the total number of bytes in each memory array except the EEPROM, which should be 32 Kbytes rather than 8 Kbytes.

FCM3–FCM0 may be set to all zeros to allow the chip select or $\overline{\text{RASx}}$ line to assert on all function codes except CPU space (interrupt acknowledge). It is advisable to program FCM3–FCM0 to zeros, at least during the initial stages of debugging.

BCYC1–BCYC0 may be set to zeros (no wait states) if the QUICC is controlling the bursting for the MC68EC040 and the timing supports one-clock MC68EC040 bursting. However, with 60- or 70-ns DRAMs at 25 MHz, BCYC1–BCYC0 should be set to 01 for two-clock MC68EC040 bursting.

PGME should be cleared.

SPS1–SPS0 should be cleared.

DSSEL should be set only if this is a DRAM bank.

9.4.4 Interfacing Multiple QUICCs to an MC68EC040

It is possible to interface multiple QUICCs to an MC68EC040. The first QUICC can be configured as previously shown in this subsection. Additional QUICCs should be configured as noted in the following list:

- The additional QUICCs should have their CONFIG2–CONFIG0 pins configured for slave mode, global chip select *disabled*, and MBAR at $003FF04. These QUICCs still recognize and respond to MC68040 cycles via the $\overline{\text{TS}}$ pin, even though their CONFIG2–CONFIG0 pins are not configured for MC68040 companion mode.
- The MBAR of the additional QUICCs should be programmed using the $\overline{\text{MBARE}}$ pin and MBARE register as described in Section 6 System Integration Module (SIM60).
- An external bus arbiter is required to take the bus request of the additional QUICC (which is an output because of the CONFIG2–CONFIG0 pins) and prioritize it with the MC68EC040, present it to the original QUICC, and issue a bus grant to the appropriate device.
- An external interrupt prioritizer is required to determine which QUICC $\overline{\text{IOUT2}}$–$\overline{\text{IOUT0}}$ pins are currently routed to the MC68EC040. Alternatively, the additional QUICC should have its interrupts brought out on a single $\overline{\text{RQOUT}}$ pin, which is routed to one of the original QUICC interrupt inputs. This would eliminate the external logic.
- The additional QUICCs should not be configured to perform any memory controller support functions for the MC68EC040. Only the original QUICC should be used for this purpose.

9.5 SELECTING CACHE MODES ON THE MC68EC040

When the QUICC is used in its MC68040 companion mode with the MC68040, it is recommended that the QUICC serial data buffers be cache inhibited by the MC68040. This avoids the overhead that would result from the cache coherency algorithms of the MC68040 following the frequent write accesses by the QUICC to one of its serial data buffers located in external memory. When using the MC68040, the MC68040 memory management unit (MMU) may be used to cache inhibit the data buffers. However, the lower cost MC68EC040 does not have an MMU. Therefore, what technique can be used by the MC68EC040 to cache inhibit serial data buffers? The following paragraphs discuss a method for selecting caching modes on 16-byte boundaries for an MC68EC040.

The MC68EC040 delivers high performance at a low system cost for embedded control by providing the same high integer performance and large 4K instruction/data caches as the MC68040 without the floating-point unit and MMU. The MC68040 MMU makes the caching

mode selectable for each 4K or 8K page within memory. The use of several caching modes within the same address range is made more difficult without the MMU. The access control unit of the MC68EC040 provides two access control registers each for data and instructions. Each access control register allows caching modes to be defined in 16-Mbyte to 4-Gbyte sections. This coarse division of the memory map is not ideal for all embedded applications. The following paragraphs explain an addressing scheme that allows all parts of the memory map to be independently available using any of the MC68EC040 caching modes on line boundaries (a line equals 16 bytes). Since any part of the memory map is accessible as cache-inhibited, copyback, or write-through, there is no requirement to split caching modes on 16-Mbyte boundaries. Furthermore, since the cache mode can be selected down to the line boundary, different areas within the same address region (e.g., DRAM) are accessible in any or all of the three caching modes.

9.5.1 The Algorithm

Two address bits are used to divide the MC68EC040 4-Gbyte addressing range into four 1-Gbyte sections. The only difference between the sections is the caching modes. The caching mode of one section is made cache-inhibited, serialized (address bits = 00); the next section is made cachable, copyback (01); and the last two sections are made cachable, write-through (1x). The address bit ordering (00, 01, 1x) allows the 1-Gbyte sections to be nested. Address bits 00 are at the bottom of the address map, address bits 01 are in the middle, and address bits 1x occupy the top half of the address map. Each 1-Gbyte section is mirrored onto every other section to provide a single 1-Gbyte addressing range. The address mirroring is done by externally ignoring the two address bits used to select the caching mode. The regions are mapped into memory using the remaining 30 address bits. The caching mode of any part of the 1-Gbyte address range is now selected by software. If the address bits used for cache mode selection are 00, the access is cache-inhibited. If the address bits are 01, the access is a copyback. If the address bits are 10 or 11, the access is a write-through. The address bits can be any of the top eight address bits (A31–A24) and do not need to be contiguous.

9.5.2 Protection

This cache addressing method does not provide any internal protection from incorrectly accessing an address with the wrong caching scheme. The answer is to rely on the software to correctly access with the correct caching mode or to externally qualify the accesses with the caching mode address bits. Special care is required to avoid mixing caching modes within the same memory line. An example of the problem is a cachable and noncachable access to the same line. A copyback access to one long word of a line will cause all four long words of the line to be read into memory. A cache-inhibited access to another long word of the same line would not hit in the cache, but rather hit in the external memory. A cache line push of the copyback line can now overwrite the cache-inhibited long word in external memory. A subsequent memory access to the cache-inhibited long word of the same line will now differ based on whether the cache push of the line has occurred. The solution is to use line boundaries when choosing caching modes. To change the cache mode of a line from cachable to cache inhibited, do a CPUSH or CINVAL of the line when making the change to ensure that the cache does not contain a copy of a cache-inhibited line. To switch from cache inhibited to cachable, the internal caches do not require any change.

9.5.3 MC68EC040 Cache Behavior

To better understand the cache operation of the MC68EC040, the following brief explanation describes how the MC68EC040 caches and access control unit work. When the MC68EC040 comes out of reset, the caches and access control units are disabled, and all accesses are in cache-inhibited, nonserialized mode. When the caches are first enabled, the information in the cache is unknown; therefore, it is important to invalidate the caches before enabling them. The caches are enabled by accessing the cache access control register (CACR). The data access control unit is enabled by enabling one or both of the data access control registers (DAC0 and DAC1). When a data access is made, the MC68EC040 compares the upper eight bits of the address to the base address and address mask of the enabled data access control registers. An address match with the data access control register occurs if the upper eight bits of the access address matches the base address or is masked off by the data access control register address mask. If the address does not match either data access control register, the caching mode is the default (cache-inhibited, nonserialized if the cache is disabled or write-through if the cache is enabled). If the address does match one of the data access control registers, the cache mode bits of the matching data access control register select the caching mode. If both data access control registers match, DAC0 takes priority over DAC1. The instruction accesses work the same, except the instruction access control unit is used.

9.5.4 Enabling the Caching Modes

To enable the multiple caching modes, enable DAC0 and IAC0 for cache-inhibited, serialized (cache mode bits = 10), mask out (set to ones) all but two address bits, and set the remaining two address bits to the cache-inhibited, serialized region (e.g., address bits = 00). Enable DAC1 and IAC1 for cachable, copyback (cache mode bits = 01), mask out (set to ones) all but two address bits, and set the remaining two address bits to the cache-inhibited, copyback region (e.g., address bits = 01). The write protect bit, user page attribute bits, and function code bits are set as the user requires for both DAC0 and DAC1. When the caches are enabled, all accesses in which the nonmasked address bits are 00 use DAC0 or IAC0 and are cache-inhibited, serialized. All accesses in which the nonmasked address bits are 01, use DAC1 or IAC1 and are cachable, copyback. All accesses in which the nonmasked address bits do not match either access control register are not affected by the access control units and default to cachable, write-through.

The following MC68EC040 code is used for enabling the access control unit for this caching scheme:

```
MOVE.LD0, –(A7)
CINVABC
MOVE.L#$80008000,D0
MOVECD0,CACR
MOVE.L#$003FC020,D0
MOVECD0, DAC0
MOVECD0, IAC0
MOVE.L#$403FC010,D0
MOVECD0, DAC1
MOVECD0, IAC1
MOVE.L(A7)+,D0
```

NOTE

Depending on the assembler used, the acronyms DTT and ITT may have to be used instead of DAC and IAC.

9.6 INTERFACING THE QUICC TO THE 53C90 SCSI CONTROLLER

In the late 1970's, Schugart and Associates introduced a parallel bus called Schugart Associates system interface (SASI). Because of SASI's generic nature and ability to function as a device-independent peripheral or system bus, other manufacturers quickly adopted it. In 1982, ANSI standardized an enhanced version of SASI renaming it the small computer system interface (SCSI). Since its standardization as a bus and due to its diverse potential, SCSI has enjoyed great popularity as an alternative means to network-dissimilar high-performance hosts.

The following paragraphs give a general description of the SCSI bus, including its major signals and functions. The hardware and software interface between the QUICC and the 53C90 SCSI controller is also discussed. This subsection highlights an example of a QUICC IDMA channel and the memory controller features that allow the QUICC to interface to slower peripherals.

9.6.1 SCSI General Overview

SCSI is an 8-bit, parallel I/O bus that provides a host computer with the capability of adding different disk drives, tape drives, printers, and even communication devices without major modifications to the system hardware or software. It uses logical rather than physical addressing for all data blocks.

A maximum of eight devices can be attached to the SCSI bus. Of these eight, only one pair of devices can communicate at one time. Each SCSI device has an ID bit assigned to it that is the bit-significant representation of the SCSI address referring to one of the signal lines DB7–DB0. DB7 has the highest priority. When two devices communicate over the SCSI bus, one acts as an initiator (host), and the other acts as a target (controller). The initiator originates the operation, and the target performs the operation.

9.6.2 Physical Interface

The physical bus interface is composed of a group of characteristics: speed, bus signals, and device count.

Speed, the ability to transfer data, uses two handshaking protocols: synchronous and asynchronous. Synchronous transfers a series of bytes before the handshake occurs; whereas, asynchronous requires a handshake for every byte transferred. Rates up to 6 Mbytes/sec can be accomplished asynchronously; 10 Mbytes/sec is possible on the synchronous protocol.

SCSI devices are daisy-chained using a common cable. This 50-conductor cable is used by the bus signals to interchange data, commands, status, and message information. Table 9-1 and Figure 9-17 describe the SCSI bus signals.

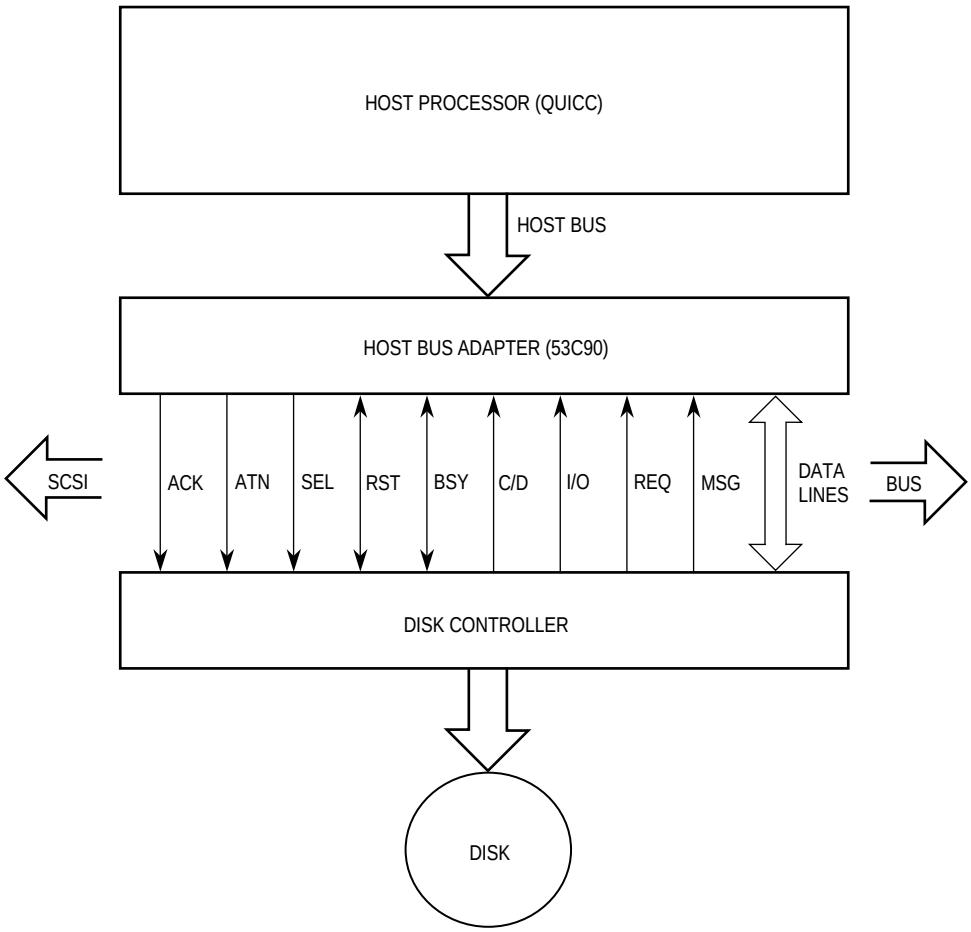


Figure 9-17. SCSI Bus Signals

Table 9-1. SCSI Bus Signals

Signal	Driven By	Signal Explanation
DB0–DB7	Initiator/Target	8-Bit Bidirectional Data Bus.
DBP	Initiator/Target	Data-Bus Parity Line. Optional.
ATN	Initiator	Attention. Used to send a message to the target when it controls the bus.
BSY	Initiator/Target	Busy. Indicates that the bus is unavailable for use.
ACK	Initiator	Acknowledge. Used by the initiator for handshaking.
RST	Any Device	Reset. Used to initiate a bus-free phase.
MSG	Target	Driven by the target to indicate that the current transfer is a message.
SEL	Initiator	Select. Used by the initiator to select a target before command execution. Also used by the target to reconnect when the reselection phase is implemented.
C/D	Target	Control/Data. Used during the information transfer phases to transfer commands, status, data or messages over the bus.
REQ	Target	Request. Used by the target during information transfer phases.
I/O	Target	Input/Output. Determines the direction of the transfer.

The SCSI bus has a total of 18 signals—9 are used for control, and 9 are used for data (one parity bit). The bidirectional data lines transfer data, commands, status, and message infor-

mation. The control signals and the bus phases determine when and what direction data is transferred. A description of the phases is given in 9.6.3 Logical Interface.

Device count refers to the maximum number of devices on the bus. The SCSI bus supports up to eight devices, including the host. It also provides two electrical interfaces, single-ended or differential. The single-ended driver and receiver configuration uses TTL logic levels and is primarily intended for applications within a cabinet where the cable length will not exceed 6 meters (20 feet). Differential, on the other hand, uses EIA RS-485 signals and can drive cable lengths up to 25 meters (82 feet). Figure 9-18 and Figure 9-19 show the single-ended and differential configurations.

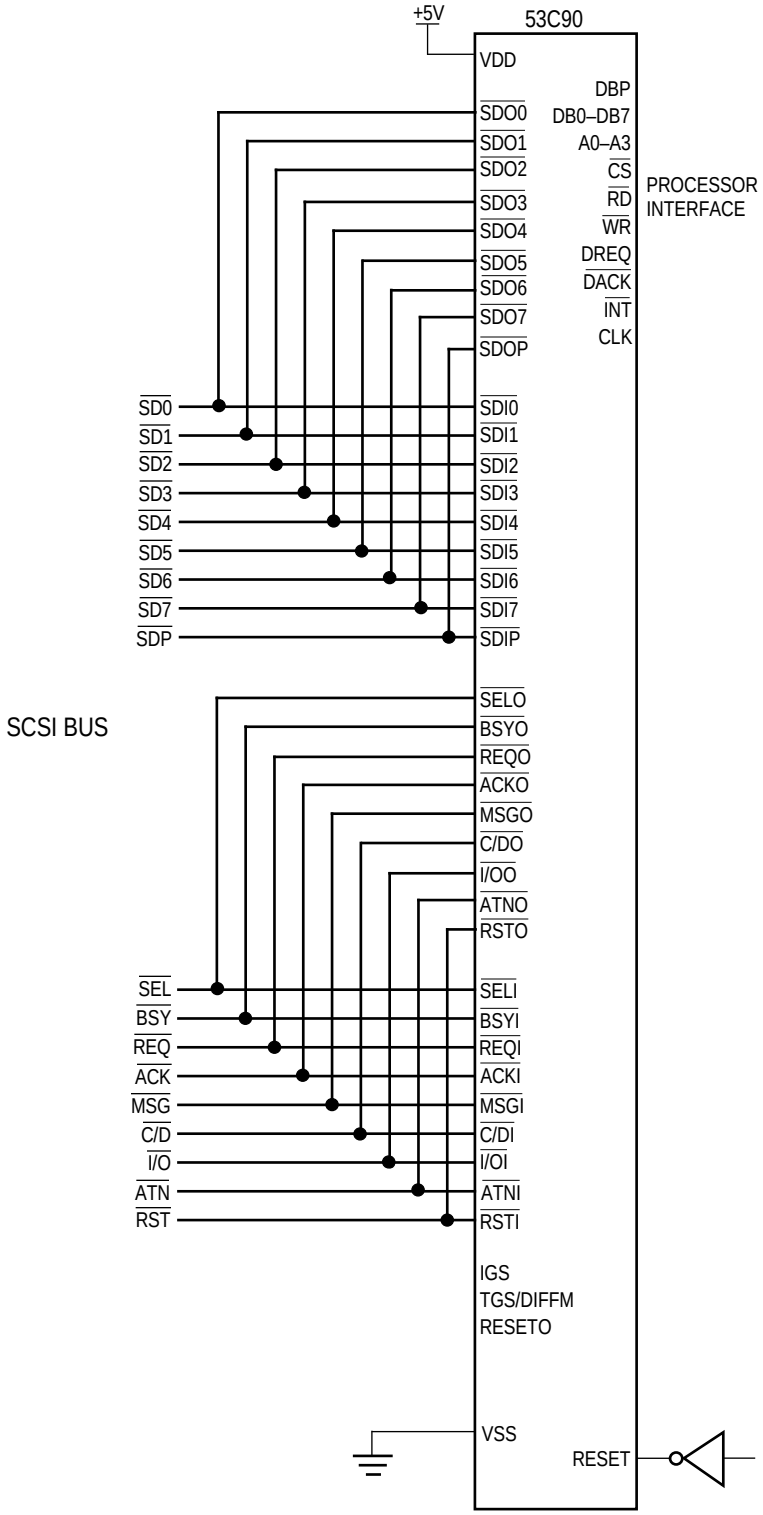
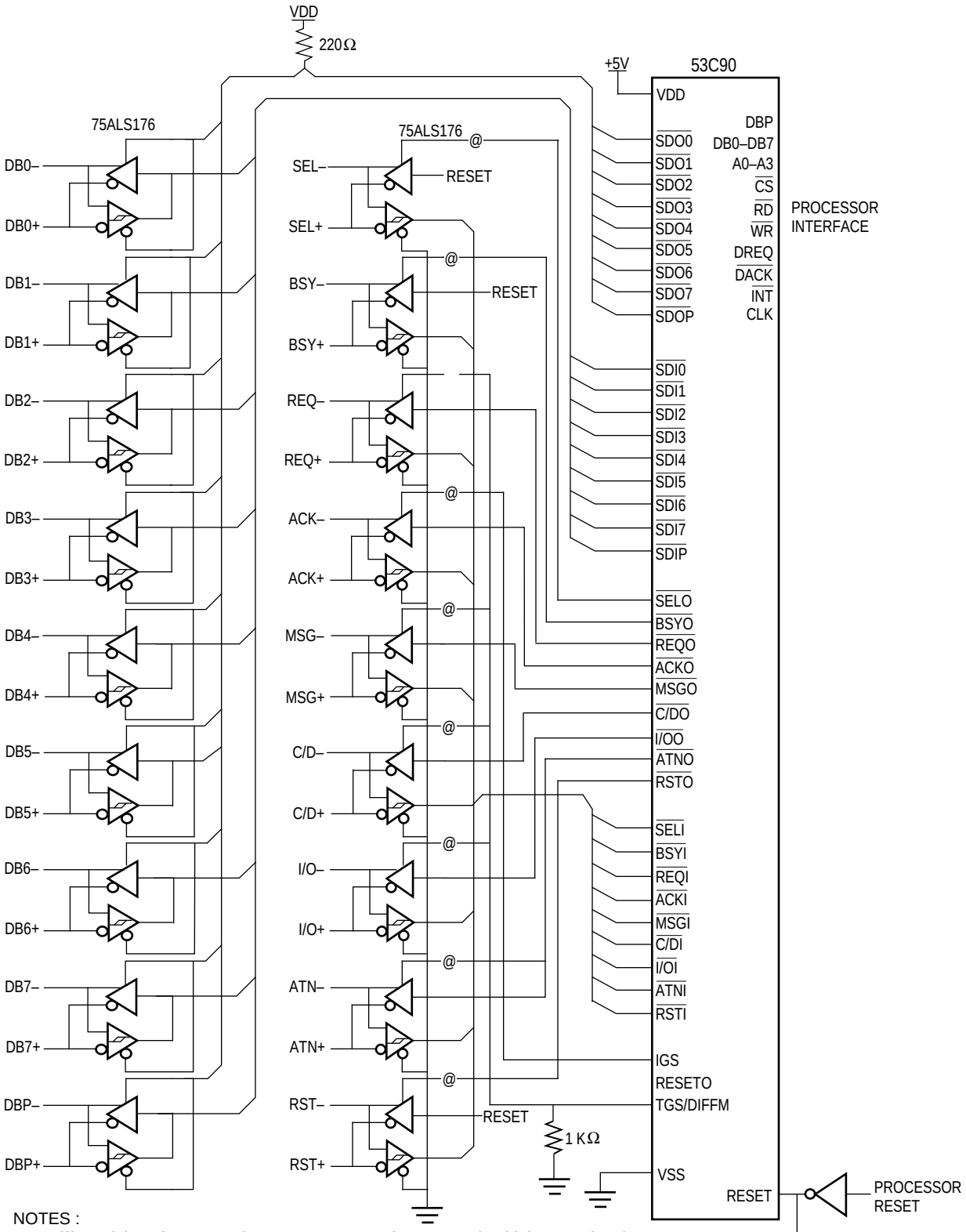


Figure 9-18. Single-Ended SCSI Bus Interface



NOTES :
1. Differential mode causes the SDOx outputs to become active-high SCSI data bus buffer enables and the SDIx pins to become a bidirectional SCSI data bus.
2. Differential mode causes the SELO, BSYO, and RSTO pins to also become active-high buffer enables. Observe special use of the RESET on the SEL, BSY, and RST buffers.

LEGEND:
@= 220-Ω pullup required.

Figure 9-19. Differential SCSI Bus Interface

9.6.3 Logical Interface

Communicating across the SCSI bus occurs through a series of phases. The protocol is made up of communication cycles, where each cycle is a sequence of states and the bus can never be in more than one state at a time. The protocol has eight phases, and each phase performs a specific function, such as idling, arbitrating, selecting, reselecting, and sending commands, data, messages, and status. From a hardware perspective, each phase is determined by the SCSI bus control lines. Figure 9-20, a phase sequence diagram, shows how the phases fit together to form the communication cycle. Some phases, such as arbitration, reselection, and message out, are optional. They are used when the bus has multiple initiators. The following list describes the different phases:

BUS-FREE—the idling state of the SCSI bus. When in this state, no SCSI device is actively using the bus, and the bus is available for subsequent users.

ARBITRATION—used to determine which device gains control of the SCSI bus. Arbitration is necessary when two or more devices are simultaneously competing to use the bus.

SELECTION—used to establish a communications link between an initiator and a target device to perform a SCSI command. Typically, the command is a read or write function to the target.

RESELECTION—optional phase controlled by the target. This phase allows the target to reconnect to an initiator so that an operation that was previously started by the initiator, but suspended by the target, can be finished.

INFORMATION-TRANSFER—includes command, data, status, and message phases. They are entered by asserting the BSY line and negating the SEL line. The C/D, I/O, and MSG signals are used to differentiate between the phases.

Command—allows the target to receive a command from the initiator in the form of a command-descriptor block (CDB). It is used to tell the target which type of operation to perform (e.g., read/write a sector from a disk). CDBs are usually 6, 10, or 12 bytes in length.

Data—two phases: data in and data out (direction is determined by the originator). The data phase is entered when the target negates the C/D and MSG lines and asserts (input) or negates (output) the I/O line.

Message—similar to the data phase because it can be message in or message out. It conveys information on the status of a command and is entered when the target asserts the C/D and MSG lines during REQ and ACK handshakes. I/O has the same function as in the data phase.

Status—used to send one status byte from the target to the initiator. This phase falls between the data and message phases and consists of sending a good status byte if the data transmitted was received correctly. The status phase is entered when the target asserts the C/D and I/O lines while MSG is negated.

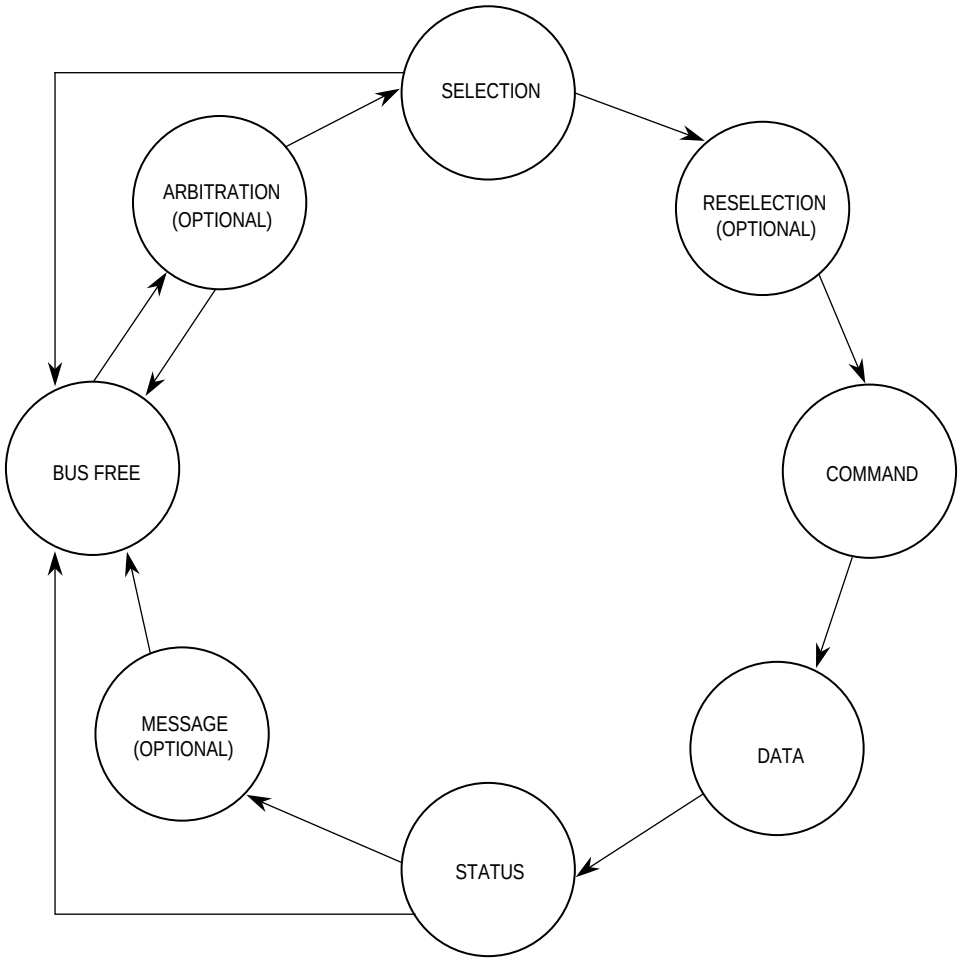


Figure 9-20. Phase Sequences of the SCSI Bus

Table 9-1 is a truth table summary of the preceding description (1 = signal asserted and 0 = signal negated). I

Table 9-2. Information Transfer Phases

Signal					Direction	Phase
SEL	BSY	MSG	C/D	I/O		
0	1	0	0	0	To Target	Data Out
0	1	0	0	1	From Target	Data In
0	1	0	1	0	To Target	Command
0	1	0	1	1	From Target	Status
0	1	1	0	0	—	Reserved
0	1	1	0	1	—	Reserved
0	1	1	1	0	To Target	Message Out
0	1	1	1	1	From Target	Message In

9.6.4 Functional Description

The 53C90 SCSI controller consists of three major sections: the processor interface, the data path, and the logic block. The processor interface includes the 8-bit data bus, parity bit, chip select, read/write strobes, A0–A3 address lines, interrupt request, and DMA signals. The data path consists of a 16-byte FIFO, parity generation, DMA interface, and SCSI data and control bus inputs/outputs.

The logic block consists of a hierarchy of sequencers that direct the SCSI bus control signal timing. The 53C90 has a set of on-chip state machines that directly perform many SCSI sequences. The instruction sequencer and master sequencer provide control of these state machines. Since the 53C90 has no real decision-making capability, it relies on the CPU32+ for supervisory control and for integrating the performed SCSI sequences into complete operations. Processor service is requested through a standard interrupt structure, and the 53C90 reports the status through its register set.

Overall control of the 53C90 is done through the processor interface. The processor data bus and associated control signals provide the means to initialize and set the operating mode of the SCSI as well as provide the data path for information swapping. The processor can write to a set of 12 registers, instructing the 53C90 what function to perform, then read another set of registers to determine the status.

Once the 53C90 starts executing, all data transfers are handled by the DMA. Since the 53C90 does not have an onboard DMA controller, it relies on one of the two independent DMA (IDMA) channels of the QUICC. In this case, IDMA1 is arbitrarily chosen. IDMA1 of the QUICC moves the data to and from the 53C90 FIFO. The FIFO provides a 9-bit-wide by 16-byte-deep buffer. It can be accessed by either the IDMA or the processor at register address $02 and can be read or written as a register. The bottom of the FIFO is read and unloaded while the top is written to and loaded.

Therefore, for an SCSI transfer to occur, the processor has to initialize the 53C90 by writing to its registers, initialize IDMA1 by setting up its registers, and then monitor the status of the 53C90.

Table 9-2 lists the read and write registers in the 53C90.

Table 9-3. 53C90 Read and Write Registers

Address (Hex)	Read Register	Write Register
$00	Transfer Counter LSB	Transfer Counter LSB
$01	Transfer Counter MSB	Transfer Counter MSB
$02	FIFO	FIFO
$03	Instruction Executing	Instruction Holding
$04	Status	Destination ID
$05	Interrupt	Select/Reselect Timeout
$06	Sequence Step	Synchronous Period
$07	FIFO Flags/Seq. Step	Synchronous Offset
$08	Configuration 1	Configuration 1
$09	Reserved	Clock Conversion (Presel)
$0A	Reserved	Test Mode
$0B	Configuration 2	Configuration 2
$0C-$0F	Reserved	Reserved

9.6.5 Hardware Configuration

The following paragraphs describe the hardware configuration of the SCSI controller interface. Figure 9-21 shows the SCSI bus interface.

9.6.5.1 CLOCKING STRATEGY. In this application, the system clock is generated from a 32.768-kHz crystal into the QUICC. The QUICC internal PLL multiplies the frequency up to 25 MHz and outputs 25 MHz on CLKO1, which is fed into the 53C90 CLK input. CLKO2, which has 50 MHz available, might be disabled in software to save power. It is also possible to run the QUICC at 20 MHz and feed the 53C90 40 MHz from CLKO2 to achieve fast SCSI synchronous mode. In that mode, make sure the prescale bit in the clock conversion register ($09) of the 53C90 is set. Timing issues also need to be considered.

The use of the 32.768-kHz crystal is not a requirement but is a low-cost solution. A 25-MHz external oscillator can also be used. The QUICC clocking section allows for the clock oscillator to be kept running through the VDDSYN pin in a power-down situation, if desired.

9.6.5.2 RESET STRATEGY. If a pushbutton switch is needed, it can be connected by an open-drain buffer to the QUICC $\overline{\text{RESETH}}$ line, once debounced. The QUICC reset control logic asserts $\overline{\text{RESETH}}$ for a minimum of 512 cycles (20.48 μs at 25 MHz); this is inverted and routed to the 53C90 reset input since it is an active-high reset.

9.6.5.3 READ/WRITE TIMING. The CPU32+ is running three clock cycles on a normal zero-wait-state bus cycle, implying 120 ns for the read or write bus cycle (f = 25 MHz, clock period is 40 ns). Looking at the QUICC read timing, the read strobe ($\overline{\text{OE}}$) is asserted after the falling edge of S0. Electrical specification 9 can be as long as 20 ns. From the time the read strobe is asserted until data is valid from the 53C90 can be as long as 70 ns, giving a total time of 90 ns, which would occur after the falling edge of S4 where the data should be read. Therefore, one wait state should be inserted on the read cycle. Cycle time is also important. The 53C90 needs a minimum of 40 ns before the next read; whereas, the QUICC guarantees a minimum of 30 ns. TRLXQ and CSNTQ bits in the base register of the QUICC chip select should be set to ones to relax the timing and allow successive reads.

Using $\overline{\text{WE0}}$ as the strobe for the write cycle, the CPU32+ will guarantee a minimum hold time of 35 ns; the 53C90 needs 2 ns. The setup time is a 35-ns minimum for the QUICC; the 53C90 needs 11 ns. Therefore, no wait states are needed for the write cycle, but cycle time is an issue. The 53C90 needs a minimum of 60 ns before the next write; whereas, the CPU32+ guarantees 35 ns. Again, set the TRLXQ and CSNTQ bits to one to relax the timing and allow successive writes

9.6.5.4 INTERRUPT HANDLING. There are multiple SCSI instructions that would generate an interrupt to the QUICC. Since the 53C90 does not put a vector on the bus, it will use $\overline{\text{AVEC}}$ to terminate the bus cycle and generate an automatic vector number. This interrupt is coming in at $\overline{\text{IRQ2}}$, leading to a vector number of 24 + 2 = 26 ($1A). The autovector register should be initialized to handle an autovector on $\overline{\text{IRQ2}}$. When the QUICC is interrupted, the interrupt service routine should read the status register and sequence step register of the 53C90 before reading the interrupt register. Reading the interrupt register resets the interrupt pin ($\overline{\text{INT}}$), the contents of the sequence step register, bits 7–3 of the status register, and the contents of the interrupt register itself.

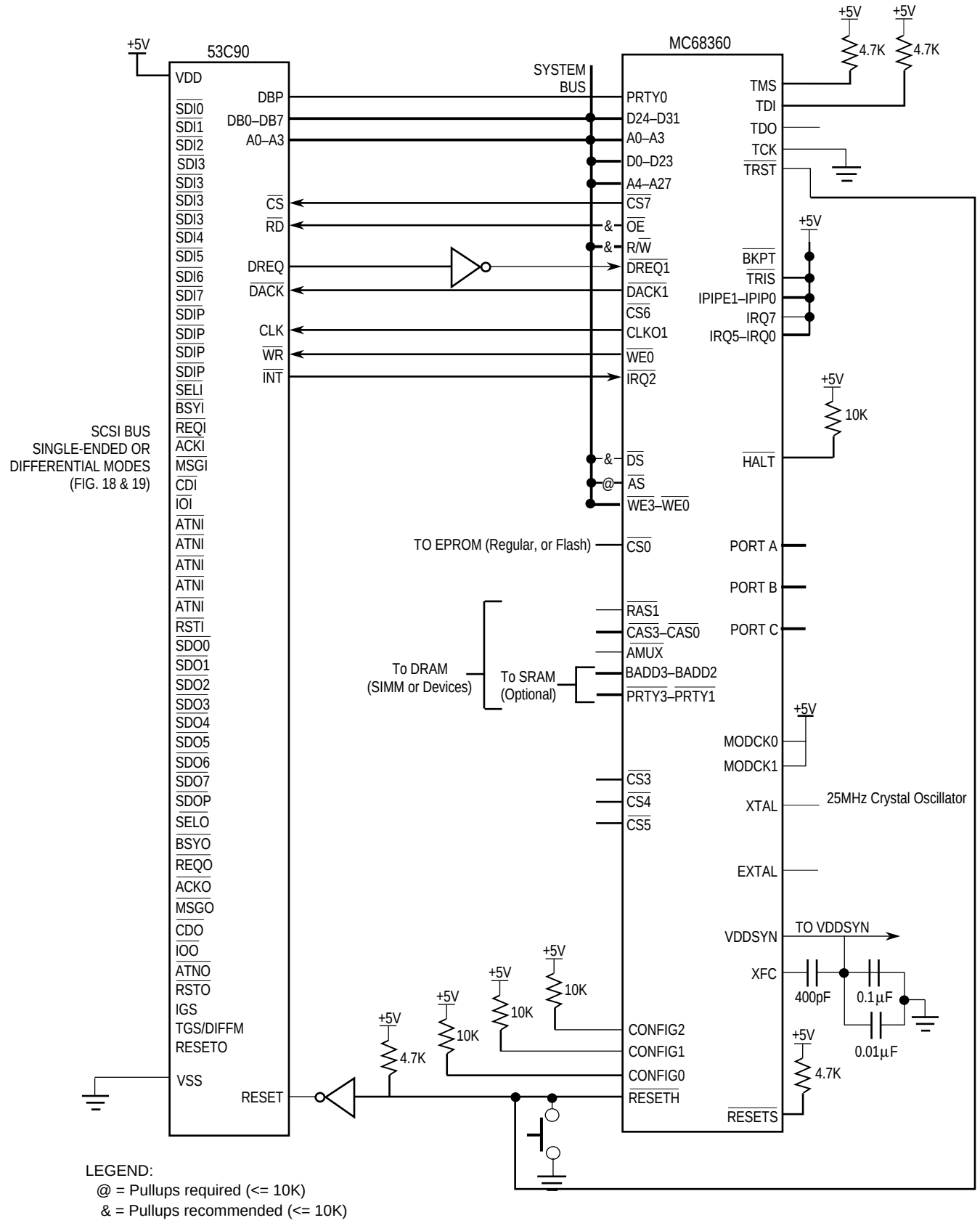


Figure 9-21. QUICC to SCSI Bus Interface

After determining what caused the interrupt, the service routine would branch to the appropriate portion of the program to execute. $\overline{IRQ2}$ was chosen to cause the 53C90 interrupts to have a lower priority than interrupts from other high-speed serial activity. The priority can be moved up if the system requires it.

9.6.5.5 IDMA1 SETUP AND TIMING. After the processor initializes the 53C90 and all its registers, IDMA1 will take care of all the data movement to and from memory. The 53C90 has a DMA interface, and special care should be taken in the hardware connections. Chip select and $\overline{DACK}$ cannot be asserted at the same time on the 53C90. When IDMA1 is requested, it will put out the address of the 53C90 and assert the chip select. To overcome this, another chip select is used with a different address range. The 53C90 needs the minimum memory range available (2K block). Assuming all peripherals are at address $04000000 and the 53C90 is at address $04001000, then $\overline{CS7}$ will decode $04001000 on the read/write cycles, and $\overline{CS6}$ will decode $04001800 on the DMA cycles. This way the 53C90 will take a 4K address block. $\overline{CS6}$ is used because it will provide $\overline{DSACK1}$ to terminate the DMA cycles.

The 53C90 requires $\overline{DACK}$ to be cycled once for each DMA access. This is available on the QUICC for both single and dual address transfers. In this case, however, packing of data from the 8-bit SCSI port to a 32-bit memory is desired; therefore, dual address transfers *must* be used. Thus, the SCSI will be accessed four times followed by a 32-bit access to memory. The maximum transfer rate in this case is 7.1 Mbyte/sec:

(4 byte $\times$ 25 Mclocks/sec) / ((4×3) + 2 clocks per transfer) = 7.1 Mbyte/sec

Accesses to the 53C90 are three clocks; whereas, memory may be accessed as fast as two clocks.

IDMA1 has eight registers that define its specific operation:

32-Bit Source Address Pointer Register (SAPR)
32-Bit Destination Address Pointer Register (DAPR)
16-Bit Channel Configuration Register (ICCR)
8-Bit Channel Mask Register (CMAR)
8-Bit Function Code Register (FCR)
32-Bit Byte Count Register (BCR)
8-Bit Channel Status Register (CSR)
32-Bit Channel Mode Register (CMR)

These registers provide the addresses, transfer count, and configuration information necessary to set up a transfer. They also provide a means of controlling the IDMA channel and monitoring its status. All registers can be modified by the CPU32+ core. The data holding register is another 32-bit register in the IDMA, but it is not accessible by the CPU32+ core. It is used for temporary data storage.

Every IDMA operation involves the following steps: channel initialization, data transfer and block termination. In the initialization phase, the CPU32+ loads the registers with control information, then starts the channel. In the transfer phase, the IDMA accepts requests for operand transfers and provides addressing and bus control for the transfers. The termination phase occurs when the transfer is complete and the IDMA interrupts the CPU32+.

9.6.5.6 QUICC I/O PORTS. The functions on QUICC parallel I/O ports A, B, and C may be used as desired in this application. However, any unused parallel I/O pins should be configured as outputs so they are not left floating. Do not forget to enable the $\overline{\text{DREQ1}}$ and $\overline{\text{DACK1}}$ pins on port B.

9.6.6 Active SCSI Terminations

There are multiple devices that provide switchable precision SCSI bus terminations. They are available in surface mount packages for different bus sizes. By utilizing flexible system design techniques, enabling or disabling terminations can be accomplished in software or hardware. Motorola's family of SCSI terminators currently includes the following devices:

- MCCS142233 (passive, 9 resistor pairs, 220Ω/330Ω)
- MCCS142234 (active, 9 bits, 110Ω)
- MCCS142235/36/37 (active, 18 bits, 110Ω, different voltages, regulators, and enables)
- MC34268 (SCSI-2 active terminator).

For more information, refer to the individual data sheets on each part.

9.6.7 Software Configuration

The following paragraphs discuss a number of key points for a software engineer desiring to initialize the system. The only items discussed are those that are required to enable the previously discussed hardware configuration. See 9.1 Minimum System Configuration for additional information.

9.6.7.1 CONFIGURING IDMA1. In this example, it is assumed that data is transferred from the 53C90 to 32-bit-wide memory. Therefore, the source size is one byte and the destination size is long word. The source address should be outside the 53C90 memory space but should still access the FIFO at $02 ($04001802), and the destination address should be $A0000000. The number of bytes to be transferred is 4 Mbytes.

ICCR = $0720. The IDMA ignores the FREEZE signal, IDMA1 has priority over IDMA2 and all interrupt handlers, IDMA1 arbitration ID is 2 while IDMA2 is 0, and the system clock operates normally within the IDMA.

FCR1 = $89. Source function code is 1000; destination function code is 1001.

SAPR1 = $04001802. Source address.

DAPR1 = $A0000000. Destination address.

BCR1 = $00400000. Byte transfer count.

CSR1 = $FF. Clear any CSR bits that are set.

CMAR1 = $FF. Enable all interrupts.

CMR1 = $8941. The 53C90 uses the control signals during the read portion of the transfer. The IDMA1 uses asynchronous request mode, dual address transfer, single buffer mode, and external request burst transfer mode. The SAPR and DAPR are incremented according to source size (byte) and destination size (long word). Setting STR starts the channel.

9.6.7.2 CONFIGURING THE MEMORY CONTROLLER. The following paragraphs describe configuring the memory controller.

For information on configuring the global memory register (GMR), refer to 9.1 Minimum System Configuration.

The memory controller status register (MSTAT) is used for reporting parity errors and does not require initialization.

Eight base registers (BRs) exist, one for each memory bank. BR6 and BR7 for $\overline{CS6}$ and $\overline{CS7}$ will be specified with the 53C90 address being $04001000. Please refer to 9.1 Minimum System Configuration for DRAM and other memory configurations.

BR7 = $0400104D. Address decoded is $04001xxx, function codes are xxxx (don't care, will be masked in OR7), TRLXQ and CSNTQ are set, parity is enabled, read and write accesses are allowed, and this base register is valid.

BR6 = $04001805. Same as BR7, except TRLXQ and CSNTQ are not set and the next consecutive 2K memory block is selected.

Eight option registers (ORs) exist, one for each memory bank. The following information is valid for registers OR6 and OR7:

DSSEL should be 0.

SPS1–SPS0 should be 10 (indicating port size is 8 bits).

PGME should be 0 since this is not DRAM.

BCYC1–BCYC0 are not used and should be cleared.

FCM3–FCM0 may be cleared to zeros to allow the chip select or $\overline{RAS}$ line to assert on all function codes, except CPU space (interrupt acknowledge). It is advisable to program FCM3–FCM0 to zeros, at least during the initial stages of debugging.

The AM27–AM11 bits will mask the address if they are cleared. In this application, they are all set to allow decoding.

The TCYC bits should be set to determine the number of wait states required—one wait state on $\overline{CS7}$ (0010) and no wait states on $\overline{CS6}$ (0001).

Therefore, OR7 = $2FFFF804 and OR6 = $1FFFF804.

9.7 USING THE QUICC AS A TAP CONTROLLER FOR BOARD SELF-TEST

An assembled board is often tested with complex test equipment using a unique test port or a bed-of-nails fixture. This procedure becomes more difficult as device packages and features become smaller. The objective of the JTAG standard is to define a boundary scan architecture that can be adopted as a part of an integrated circuit to perform both an in-circuit test and a verification of the interconnection between different devices.

The JTAG standard defines test logic that can be integrated into a device to perform:

1. Testing of the interconnection between devices once they have been mounted on a printed circuit board or any other substrate.

2. Testing of the device itself.
3. Observation or modification of the activity on the board.

The test logic is comprised of features that include the boundary scan register and is accessed through the test access port (TAP).

9.7.1 Board Layout

The test equipment interfaces to the board through a test bus, with the tester acting as a test bus master. This bus is comprised of the boundary scan signals and consists of one or several parallel signal paths. Although different architectures are described in the JTAG standard document, this application shows only one serial connection. As shown in Figure 9-22, one data signal loop is created with relevant device. The test data in (TDI) signal enters the device and exits the device as test data out (TDO) at the other end of the shift register. In addition, a clock signal (TCK) and a mode signal (TMS) are distributed to all devices in parallel.

An example of a board designed with the boundary scan path architecture is shown in Figure 9-22. Most devices, like device 2 through device 5, are included in the loop. Other devices such as memories can be tested directly from a microprocessor. Usually, the board will also contain small logic functions or analog ICs that do not contain boundary scan logic.

Once the board is tested with the test equipment, it is typically stored until it is installed in the final system. This storage time is comprised of warehouse and shipping time. How then can the board be retested before it begins to interact with a larger system in the end application? This is of particular importance in telecommunication systems with distributed intelligence, since an error can propagate throughout the entire system if one node is malfunctioning.

Today, almost every board design includes some kind of processor or controller. This controller can be used as the test bus master to perform a board test using an existing boundary scan path. The following paragraphs describe how this concept can be achieved using the QUICC as the board controller.

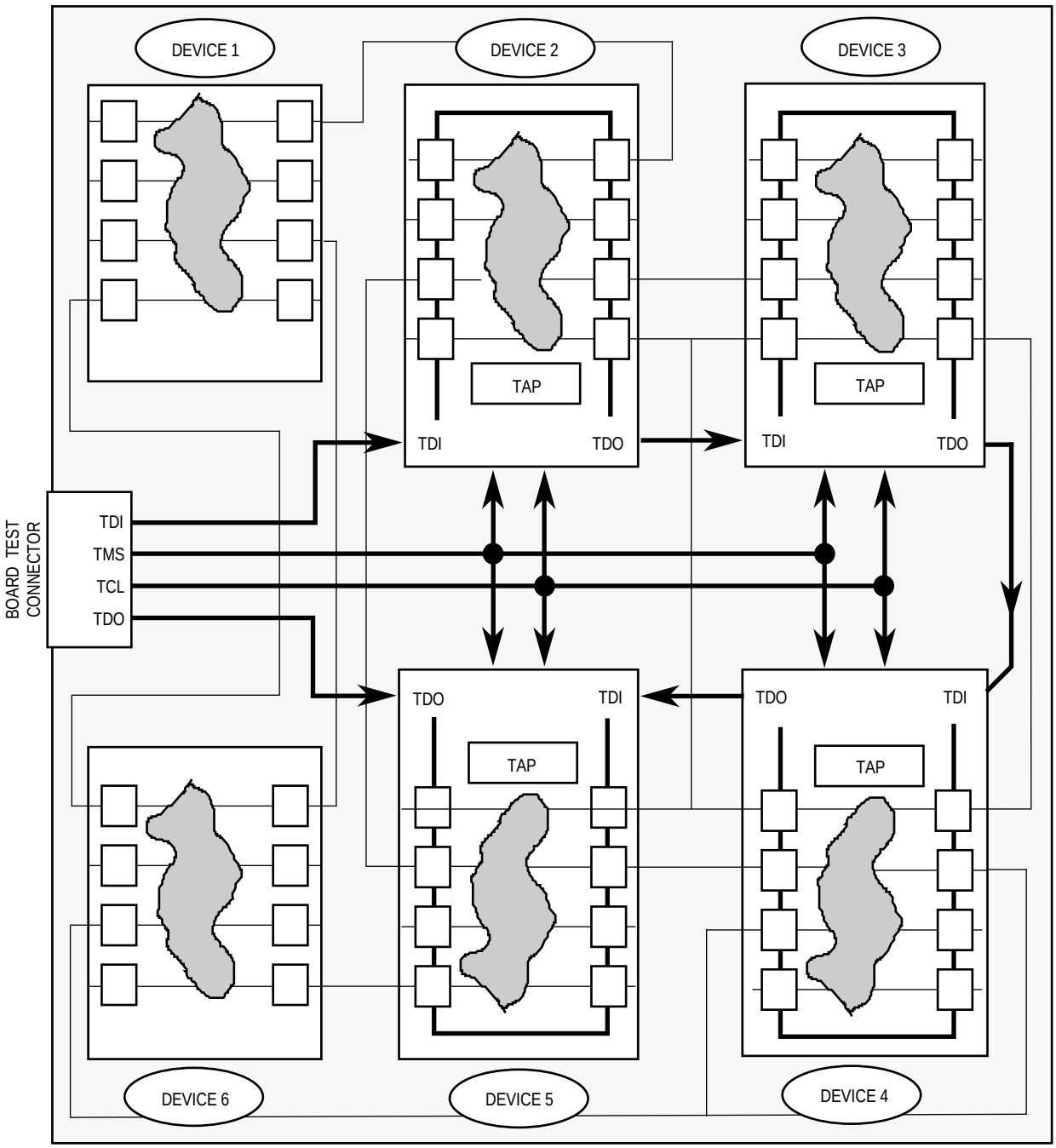


Figure 9-22. JTAG Scan Path

9.7.2 Board Testing

The patterns used for factory testing can be modified for self-testing of an installed board. The self-test can be used to check direct interconnections as well as board functions. The testing of memories on the board can be accomplished under direct program control of the QUICC; however, the boundary scan vectors under control of the onboard controller could be used to test other parts of the board.

Figure 9-23 shows an enlarged version of a device shown in Figure 9-22. The TAP is the interface between the external bus and all internal boundary scan logic. The interface is comprised of four mandatory signals, TDI, TDO, TCK, TMS, and the optional test reset (TRST).

The TAP controller is a sequential state machine. The JTAG standard defines operations that must be performed, others that should be performed, and operations that are optional and may vary between different devices and vendors. The TAP state machine is stepped by different combinations of the TMS and TCK signals. Data is shifted in through the TDI pin to the instruction register, bypass register, or boundary scan register, depending on the state machine action. The state machine also controls what internal signal path is routed to the TDO pin.

Although the TAP details are application-specific, the discussion of how to access the TAP controllers using the QUICC is common to all boards. This access is described in more detail.

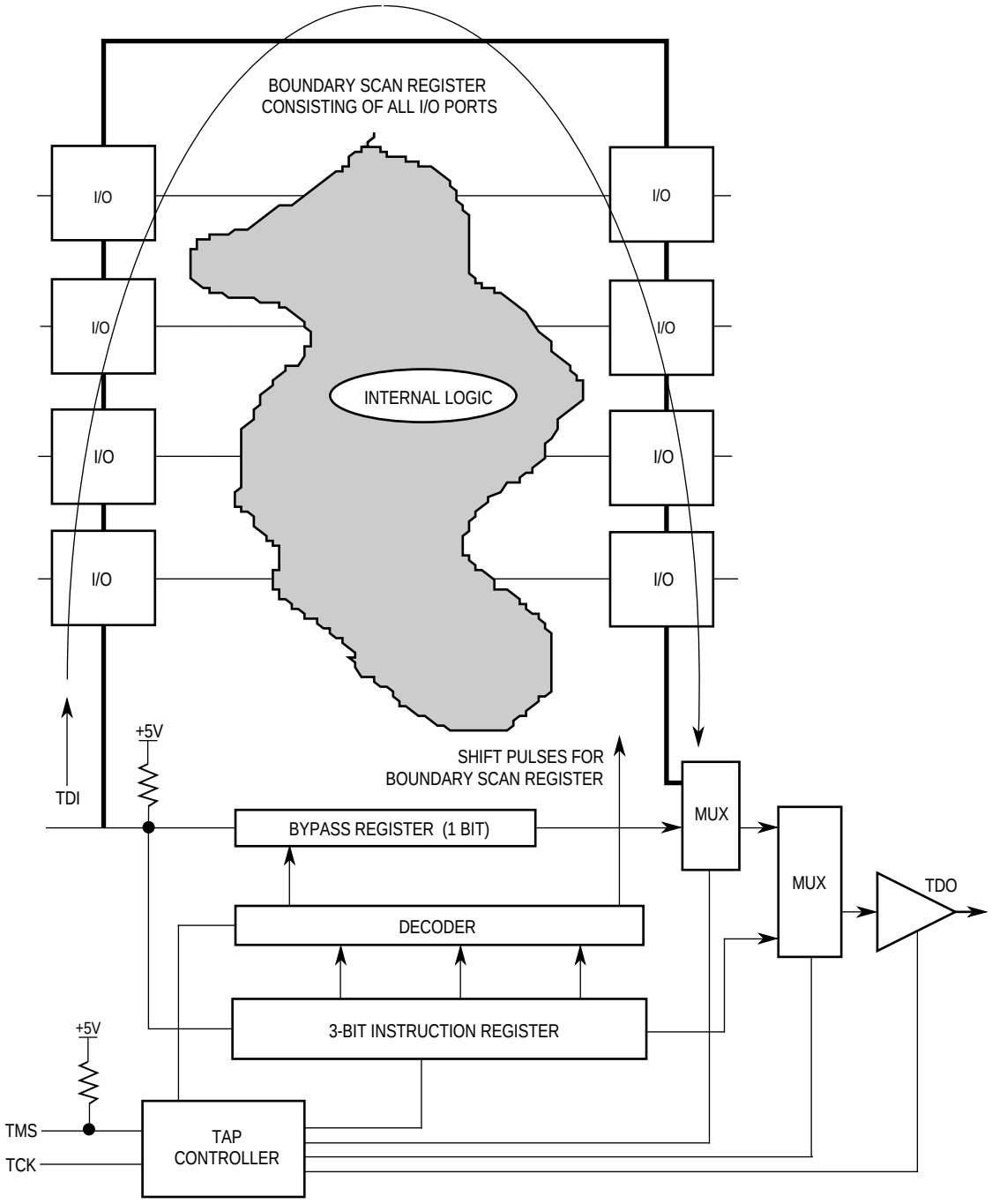


Figure 9-23. TAP Controller and Registers

9.7.3 Microcontroller Interface

Figure 9-24 and Figure 9-25 show two ways of routing the signals on a board. Figure 9-24 shows the QUICC TAP port being used in the board test, as well as the ability for the QUICC to become the test bus master using its I/O pins. Figure 9-25 shows the QUICC used simply as a test bus master.

To control the signal flow, one I/O pin is dedicated as test master mode (MM) control. The latch is used to allow the master mode test bus signals to become I/O in the normal board application, if required. Otherwise, the latch may be removed. The arrows indicate the signal flow direction for TDI and TDO. In each figure, TDI is the input data signal from the external tester, and MTDI is the input data generated by the QUICC.

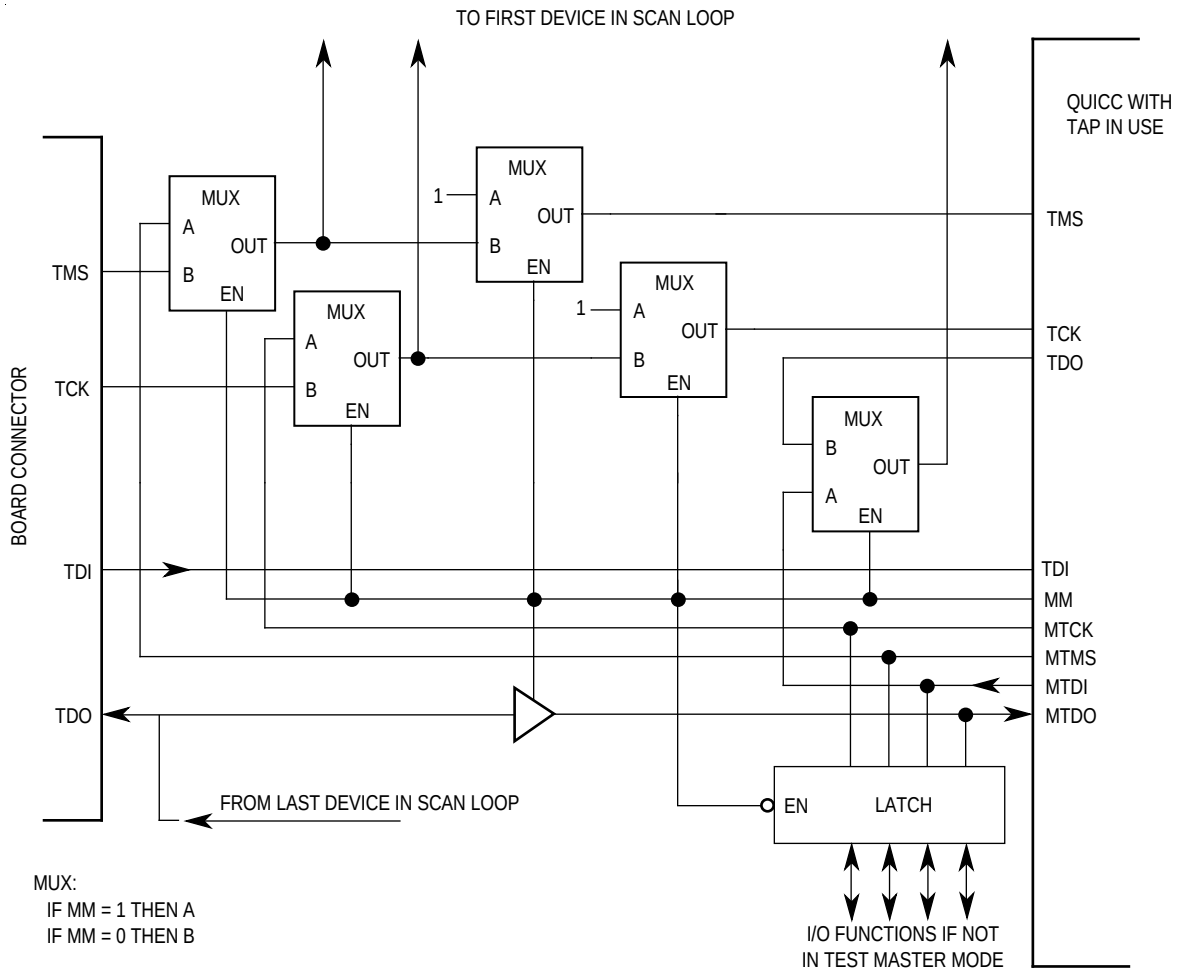


Figure 9-24. Signal Routing with TAP Port Used in Board Test

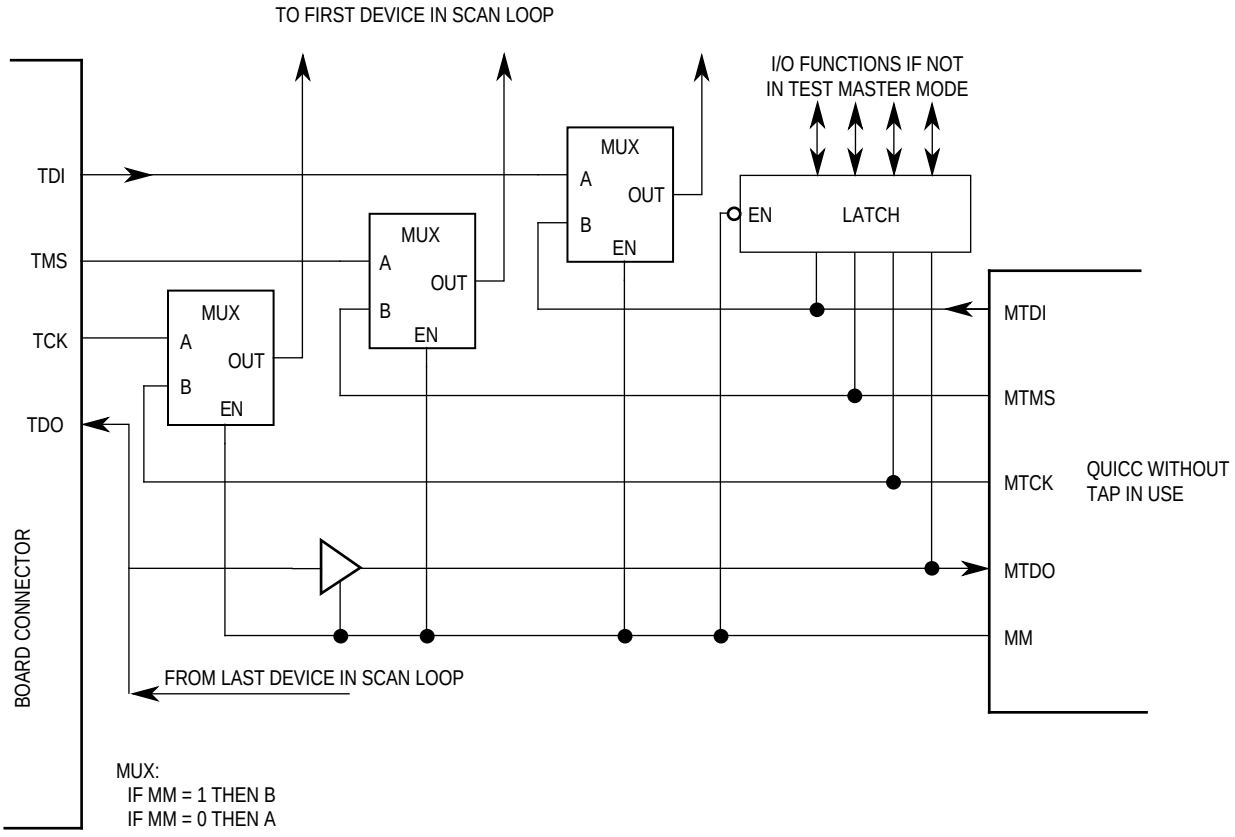


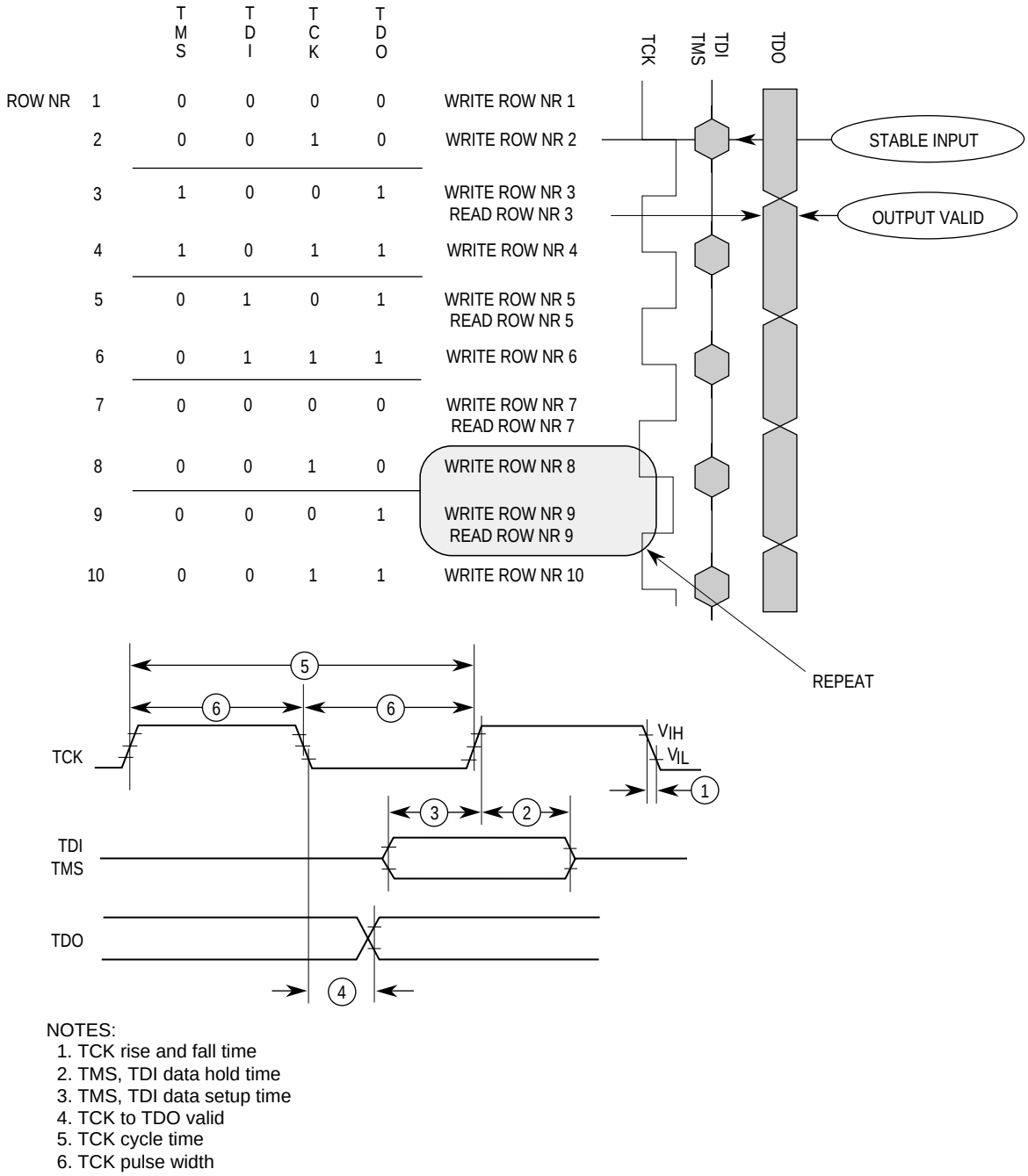
Figure 9-25. Signal Routing for Test Bus Master

9.7.4 Test Pattern Generation

The easiest way to generate the test pattern is to use the QUICC CPU32+ to transfer the test pattern over its I/O pins using a bit-banging technique.

The test pattern output data is written to the I/O ports, and therefore to the pins, by writing to the digital output ports (MTCK, MTMS, and MTDI); the result is read back by simply reading the MTDO pin. A data area is created in memory to hold the test pattern. The CPU32+ compares the result on the MTDO pin to the respective expected result in memory.

Figure 9-26 shows an example of the pattern that would be stored in the memory array. The three leftmost columns are output signals written to the I/O port, and the right column is the result expected to be read back. Since the clock signal is part of the pattern and not separately generated, both clock phases are represented, and thus two entries comprise each TCK clock cycle.



NOTES:
1. TCK rise and fall time
2. TMS, TDI data hold time
3. TMS, TDI data setup time
4. TCK to TDO valid
5. TCK cycle time
6. TCK pulse width

Figure 9-26. Bit Banging of Boundary Scan Pattern

The bit-banging pattern is created by writing and reading the array rows to the port as fast as the CPU allows. The instructions to implement the shaded part of Figure 9-26 (indicated by "repeat") are listed below. They are needed to perform one test clock cycle along with the data signals. Assuming the output array address is located in address register A0, the input array address is in address register A2, with the I/O port located at address A1 and the length of the pattern in D0, one repeat cycle is as follows:

```
LOOPMOVE.B(A0)+, (A1)WRITE ROW 8
 MOVE.B(A0)+,(A1)WRITE ROW 9
 MOVE.B(A1),(A2)+READ ROW 9
```

(Next, compare TDO read value to TDO of pattern and take desired action if the results do not match)

```
DBcc  D0,LOOP
```

The loop is repeated as long as the test condition is false and the counter D0 is more than –1. The test condition can be set to always produce a false prior to entering the loop since the MOVE instructions do not set any flags. Assuming a 40-ns clock, this code loop has approximately a 500-kHz test clock frequency.

The state machine is static, and all logic can be clocked between DC and the maximum device frequency. Additionally, there are no constraints on the clock duty cycle.

All inputs are sampled on the rising clock edge. Data must be valid during a setup time before the transition and a hold time after transition. Different input pins have different requirements, depending on whether they belong to the TAP or are normal I/O pins. Output pins change state on the falling clock edge and can be either high, low, or three-state. The exact I/O pin timings can be found in the AC Timing Specifications of this manual.

The repeat pattern in Figure 9-26 is created to meet these requirements. In row 8, the data write pattern is stable, and only the TCK pin changes state from 0 to 1. The data write pattern in row 9 sets up the data pattern for the next 0 to 1 clock transition and changes the current clock state back to 0, thus enabling the TDO signal out of the external device. The data read pattern in row 9 reads the TDO into the microprocessor memory array, where it can be compared against the expected value.

9.8 INTERFACING AN MC68EC030 MASTER TO THE QUICC IN SLAVE MODE

The following paragraphs describe the interface to the QUICC using an MC68EC030 to replace the CPU32+ core on the QUICC. When the CPU32+ core on the QUICC is disabled, the QUICC is in slave mode. In slave mode, another external processor may be used instead of the on-chip CPU32+ core. The QUICC, however, has special features for providing a glueless interface to an external MC68EC030 (as well as other M68030 and M68040 family members).

The following paragraphs discuss both the hardware and software issues concerning this solution in a system that contains one MC68EC030 and one QUICC. It is also possible to interface more than one QUICC to an MC68EC030.

9.8.1 MC68EC030 to QUICC Interface

The following paragraphs discuss the hardware and software issues relating to the connection between the MC68EC030 and the QUICC. The features of the QUICC that may be used to assist the MC68EC030 are also detailed. Reference Figure 9-27 during this discussion.

9.8.1.1 MC68EC030 READS AND WRITES TO QUICC. The basic connection is made through the data and address bus. All 32 data lines are routed between devices, which is required for the connection. In slave mode, the QUICC is not allowed to use its 16-bit data bus mode.(Assertion of $\overline{16BM}$ pin during reset)

Twenty-eight address lines are routed between devices, giving a 256-Mbyte shared address capability. It is possible to share all 32 address lines between devices, but the QUICC would then lose its write enable lines ($\overline{WE3}$–$\overline{WE0}$). Since these lines are very useful in memory interfaces, they are used in this application.

When running in normal slave mode with an MC68EC030 master, the QUICC provides a few signal changes to support the MC68EC030. These signal changes allow the QUICC to monitor and control the system buses in a glueless manner. The changed bus signals are bus request ($\overline{BR}$), bus grant ($\overline{BG}$), and bus grant acknowledge ($\overline{BGACK}$). When operating in normal slave mode, the direction of these signals is reversed. Therefore, $\overline{BR}$ is an output; $\overline{BG}$ is an input. In addition, $\overline{BGACK}$ becomes an I/O signal, rather than just an input.

9.8.1.2 CLOCKING STRATEGY. In this application, a single 25-MHz external oscillator is used to drive the QUICC and the MC68EC030, which allows the synchronous mode of the QUICC memory controller to be used. When considering buffering of outputs and board lay-out, designers need to consider the synchronous timing requirements of the QUICC. Designers considering the possibility of running asynchronously or with faster MC68EC030 clock speeds should reference paragraph 9.8.5 Using a Higher Speed MC68EC030 Master with the QUICC.

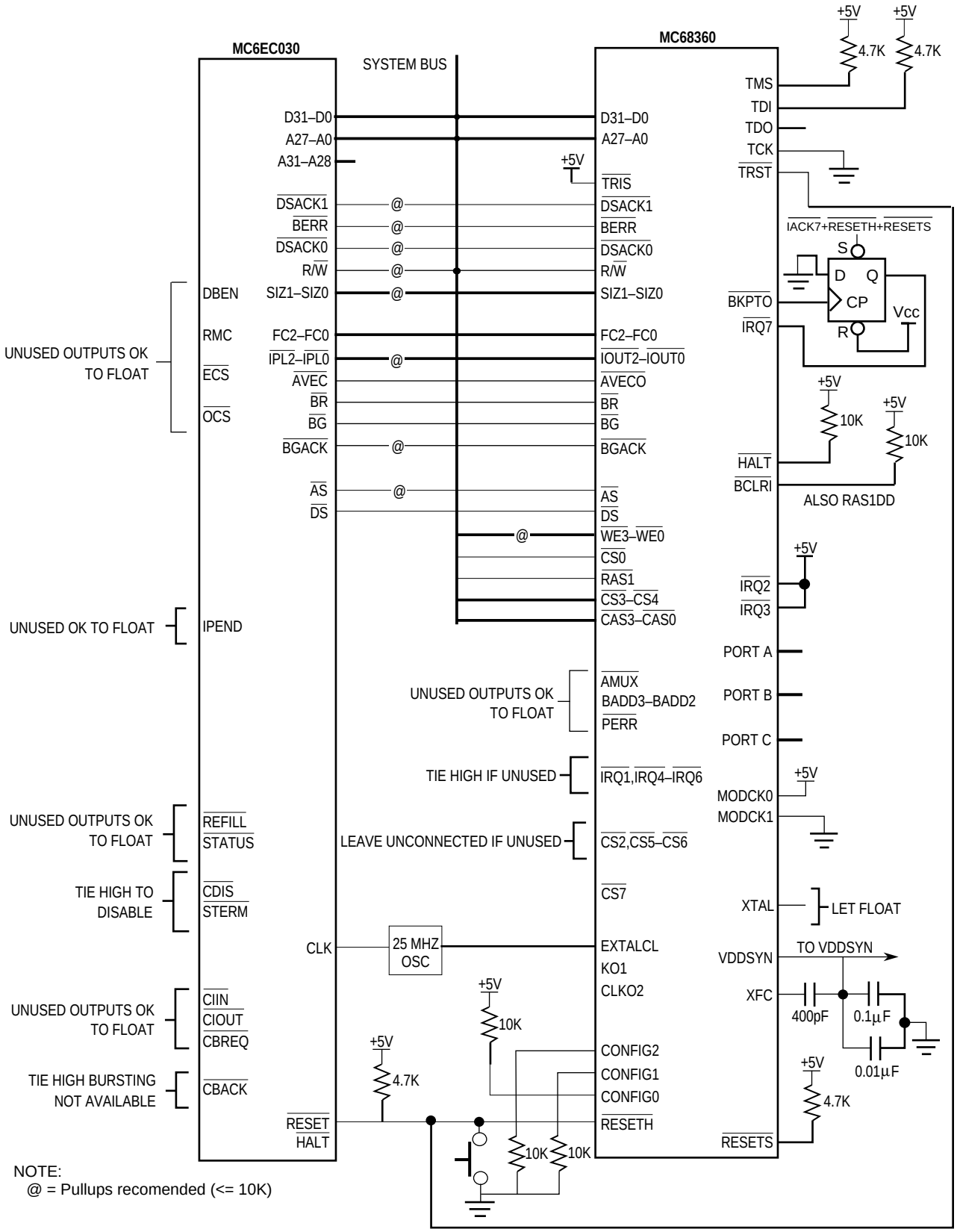


Figure 9-27. MC68EC030 to QUICC Interface

It is important for the designer to distinguish the difference between an MC68EC030 accessing memory synchronously and the QUICC memory controller operating in synchronous mode. The MC68EC030 has the ability to access memory in standard asynchronous bus cycles (i.e., three clocks or longer) and synchronous bus cycles (two clocks). For MC68EC030 asynchronous accesses, the QUICC memory controller can generate chip selects, $\overline{\text{DSACKx}}$, etc. However, the QUICC does not support MC68EC030 synchronous bus cycles (nor does it support MC68EC030 bursting). This does not mean that the MC68EC030 cannot perform two-clock accesses in a QUICC system—only that the QUICC will not assist (i.e., generate chip selects, etc.) during these accesses.

The QUICC memory controller can operate asynchronously or synchronously (defined by the BSTM bit in the MCR and the SYNC bit in the GMR). When the QUICC memory controller is operating synchronously, the external signals being monitored are not synchronized internally by the QUICC, and must be provided with the proper setup and hold times. When the synchronous function is not enabled, externally generated bus signals are latched on the negative edge of the QUICC clock before being recognized, permitting external signals to be completely asynchronous to the QUICC clock. The QUICC memory controller is normally used in synchronous mode with the MC68EC030, even if the MC68EC030 is generating only asynchronous bus cycles.

The QUICC clocking section allows for the clock oscillator to be kept running through the VDDSYN pin in a power-down situation, if desired. Low-power issues are not addressed.

9.8.1.3 RESET STRATEGY. If a QUICC is configured to provide the global chip select, it will also provide an internal power-on reset generation. Thus, the $\overline{\text{RESETH}}$ pin of the QUICC just needs to be connected to the $\overline{\text{RESET}}$ pin on the MC68EC030. If a pushbutton switch is needed, it can be connected by an open-drain buffer to the $\overline{\text{RESET}}$ line, once debounced.

9.8.1.4 INTERRUPTS. External interrupts may be brought into the QUICC through either the $\overline{\text{IRQx}}$ pins or parallel I/O pins. The QUICC prioritizes these interrupts with its own internally generated interrupts (e.g., timers) to obtain the current highest pending request. In slave mode, the QUICC can output this request to another processor in the system.

The request can take the form of a single request pin or three request pins ($\overline{\text{IOUT2}}$–$\overline{\text{IOUT0}}$) that encode the priority of the request. Since the MC68EC030 uses encoded inputs ($\overline{\text{IPL2}}$–$\overline{\text{IPL0}}$), the $\overline{\text{IOUT2}}$–$\overline{\text{IOUT0}}$ pins are chosen.

In addition, the QUICC allows the $\overline{\text{IOUT2}}$–$\overline{\text{IOUT0}}$ pins to be generated in two different ways: at the expense of parity pins or at the expense of some interrupt requests. In this application, parity is not used in the system; thus, the parity pins are chosen for this function, leaving more interrupt pins available.

Once the MC68EC030 recognizes the interrupt, it responds with an interrupt acknowledge cycle. The QUICC recognizes the MC68EC030 interrupt acknowledge cycle using the address and function code pins (FC3–FC0). The QUICC then responds by placing the vector on the bus or by outputting the $\overline{\text{AVECO}}$ signal to the MC68EC030, depending on what was programmed into the autovector register in the SIM60.

When the QUICC is in slave mode, what is normally the AVEC pin (an input) becomes the AVECO pin (an output) for use with external processors.

9.8.1.5 BUS ARBITRATION. When the QUICC is operating in slave mode with an MC68EC030 master, the QUICC bus arbitration pins match those of the MC68EC030.

NOTE

If an MC68040 were the master, there is a special companion mode on the QUICC which would configure the QUICC bus arbitration pins to match those of the MC68040. The QUICC BR is an output to the MC68EC030, and the QUICC BG is an input from the MC68EC030.

In slave mode, the QUICC does not support the read-modify-write cycle (RMC). If support of read-modify-write is necessary, it would be necessary for the designer to implement the functionality in external logic.

The BR output from the QUICC is sent directly to the MC68EC030 in this system. If other bus masters were present, external hardware could determine their relative bus priority. The BCLRO function of the QUICC is not used in this design because the BR pin is an output; therefore, BCLRO is not needed.

The QUICC also has a bus clear in (BCLRI) signal that allows internal masters to be cleared off the bus. This pin can be used with the MC68EC030 IPEND pin to give the MC68EC030 interrupts priority over the QUICC internal masters. This function is not implemented in the design for the sake of using the alternate function of the pin, the RAS1DD function. However, if it were implemented, the IPEND signal from the MC68EC030 would have to be latched and kept low by the MC68EC030 until completion of the interrupt routine. (If it was not latched, the QUICC would take the bus back from the MC68EC030 as soon as the interrupt acknowledge cycle was complete, which defeats the purpose of connecting the IPEND pin to the BCLRI pin.)

9.8.1.6 BREAKPOINT GENERATION. In slave mode, the QUICC can be used to generate a breakpoint signal using its breakpoint address register. This register will respond to QUICC external master accesses.

The result of a breakpoint is the assertion of the BKPTO pin on the QUICC. In this application, it was decided to route this output back to an interrupt input on the QUICC and generate a nonmaskable interrupt to the MC68EC030.

9.8.1.7 BUS MONITOR FUNCTION. In slave mode, the QUICC will monitor the bus for bus cycles that are not properly terminated. The cycles can originate from the QUICC or the MC68EC030. If the MC68EC030 originates such a cycle and that cycle times out without an AS, DSACKx, BERR, or HALT occurring, the QUICC will assert BERR back to the MC68EC030 to end the cycle.

9.8.1.8 SPURIOUS INTERRUPT MONITOR. In slave mode, the QUICC will watch for spurious interrupt cycles generated by the MC68EC030, but only on the interrupt levels that the

QUICC supports internally (e.g., the level of the SIM60 and the level of the CPM). If such a condition occurs, $\overline{\text{BERR}}$ will be asserted by the QUICC.

9.8.1.9 SOFTWARE WATCHDOG. If desired, the MC68EC030 can program the QUICC software watchdog to generate a level 7 interrupt or a system reset. In this application, the software watchdog is configured in software to generate a reset so that the breakpoint logic can use level 7 interrupts. No additional hardware is required because the connection between the reset pins of the QUICC and the MC68EC030 is already made.

9.8.1.10 PERIODIC INTERVAL TIMER. If desired, the MC68EC030 can use the periodic interval timer on the QUICC to generate a system interrupt, such as for a real-time kernel. No additional hardware is required for this function.

9.8.1.11 MC68EC030 CACHING CONFIGURATION. The MC68EC030 can cache or not cache data and program memory as desired. However, it is strongly advisable not to cache the data that is accessed by the QUICC serial channels because of the overhead incurred every time the cached data area is written.

9.8.1.12 DOUBLE BUS FAULT. In slave mode, the QUICC double bus fault monitor is not operational.

9.8.1.13 JTAG AND THREE-STATE. The QUICC provides JTAG ports, commonly known as JTAG. This interface uses five pins: TMS, TDI, TDO, TCK, and $\overline{\text{TRST}}$. TMS and TDI are left unconnected because they have internal pullups. The JTAG ports of both parts are disabled in this application; however, the capability could be easily added.

When the QUICC is in master mode, it provides a $\overline{\text{TRIS}}$ pin that allows all outputs on the device to be three-stated. In slave mode, this feature is not available since the QUICC is a peripheral of the system.

9.8.1.14 QUICC SERIAL PORTS. The functions on QUICC parallel I/O ports A, B, and C may be used as desired in this application and have no bearing on the MC68EC030 interface. However, any unused parallel I/O pins should be configured as outputs, so they are not left floating.

9.8.2 Memory Interfaces

In this application, a number of memory arrays have been developed for EPROM, flash EPROM, EEPROM, SRAM, and DRAM. Each memory interface can be attached to the system bus as desired.

One issue not discussed is the decision of whether external buffers are needed on the system bus. This issue depends on the number of memory arrays used in the design and possibly the layout (i.e., capacitance) of the system bus.

Another issue left to the user is the number of wait states used with each memory system. This depends on the memory speed, whether external buffers are used, and the loading on the system bus pins. (The QUICC provides capacitance de-rating figures to calculate the effect of more or less capacitance on the AC Timing Specifications.)

9.8.2.1 QUICC MEMORY INTERFACE PINS. In this design, a number of QUICC pins are available to the memory arrays (see Figure 9-27). These pins are active, regardless of whether the bus cycle was originated by the MC68EC030 or by one of the QUICC DMA cycles. The QUICC detects the MC68EC030 bus cycle by the $\overline{AS}$ pin. If the QUICC generates the bus cycle, the QUICC asserts the $\overline{AS}$ pin.

Eight $\overline{CSx}$ or $\overline{RASx}$ pins are available in the system. In this design, $\overline{CS0}$ is used for any of the EPROM arrays since this is the global (boot) chip select. $\overline{RAS1}$ is used for the DRAM arrays because of its double-drive capability. $\overline{CS2}$/$\overline{RAS2}$ is not used in the design and is available for other purposes, such as a second DRAM bank. $\overline{CS3}$ is for SRAM arrays; $\overline{CS4}$ is for EEPROM. $\overline{CS5}$, $\overline{CS6}$, and $\overline{CS7}$ are unused.

In this design, it is assumed that the full 32-bit capability of the MC68EC030 is used; thus, all memory arrays are 32 bits wide. (The only exception to this is the EEPROM, which is handled differently. See 9.8.2.4 EEPROM.)

This application note does not use the parity support provided by the QUICC. Therefore, the PRTY3–PRTY0 lines are available for their alternate functions. Parity support is available only if the SYNC bit in the GMR is set.

The QUICC does not internally support MC68EC030 external master bursting. If the user wishes to implement bursting on a particular memory array, the bursting support must be generated externally.

The DRAM arrays require the four $\overline{CAS3}$–$\overline{CAS0}$ pins. Also, since an external address multiplexer is used, the AMUX pin is required to select between rows and columns. If, however, the user's configuration does not require DRAM, the AMUX pin can be used as an $\overline{OE}$ pin instead. This would save an inverter in a number of memory arrays, making the memory interface completely glueless.

NOTE

Many memory arrays show an inverter on the R/$\overline{W}$ pin to create the $\overline{OE}$ signal. When using multiple memory arrays, it is possible to share one inverter between multiple memory arrays; however, this configuration is not shown.

The QUICC also provides four write enable ($\overline{WEx}$) pins to select the correct byte during write operations.

9.8.2.2 REGULAR EPROM OR FLASH EPROM. Figure 9-28 shows the glueless interface to standard boot EPROM in the system. The assumption is made that only the MC68EC030 will access this array. The MC68EC030 offers dynamic bus sizing on-chip; thus, only an 8-bit EPROM is required. The CONFIG2–CONFIG0 pins on the QUICC can be pulled low through resistors to select slave mode with an 8-bit port size for the global chip select. The QUICC will support $\overline{DSACKx}$ generation to the MC68030 according to an 8-bit port size.

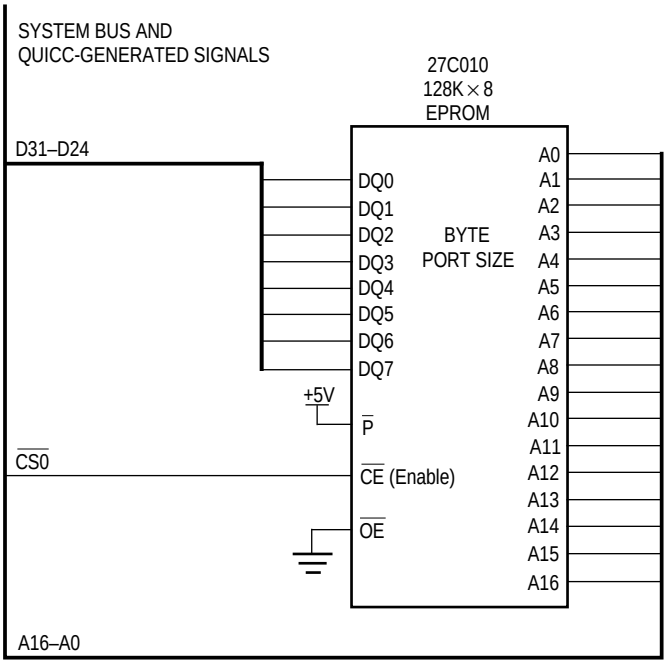


Figure 9-28. 128-Kbyte EPROM Bank—8 Bits Wide

Figure 9-29 shows the interface to flash EPROM devices. In this design, the assumption is made that only the MC68EC030 will access this array. The inverter is only required if DRAM is used elsewhere in the system. This design assumes that the write operations are $\overline{CE}$ controlled, rather than $\overline{WE}$ controlled. Most flash EPROM manufacturers now support this alternative timing method.

NOTE

The QUICC CONFIG2–CONFIG0 pins are shown selecting slave mode and a 32-bit port size for $\overline{CS0}$. Thus, the CONFIG2–CONFIG0 pins as shown are appropriate for the flash EPROM, not the regular EPROM.

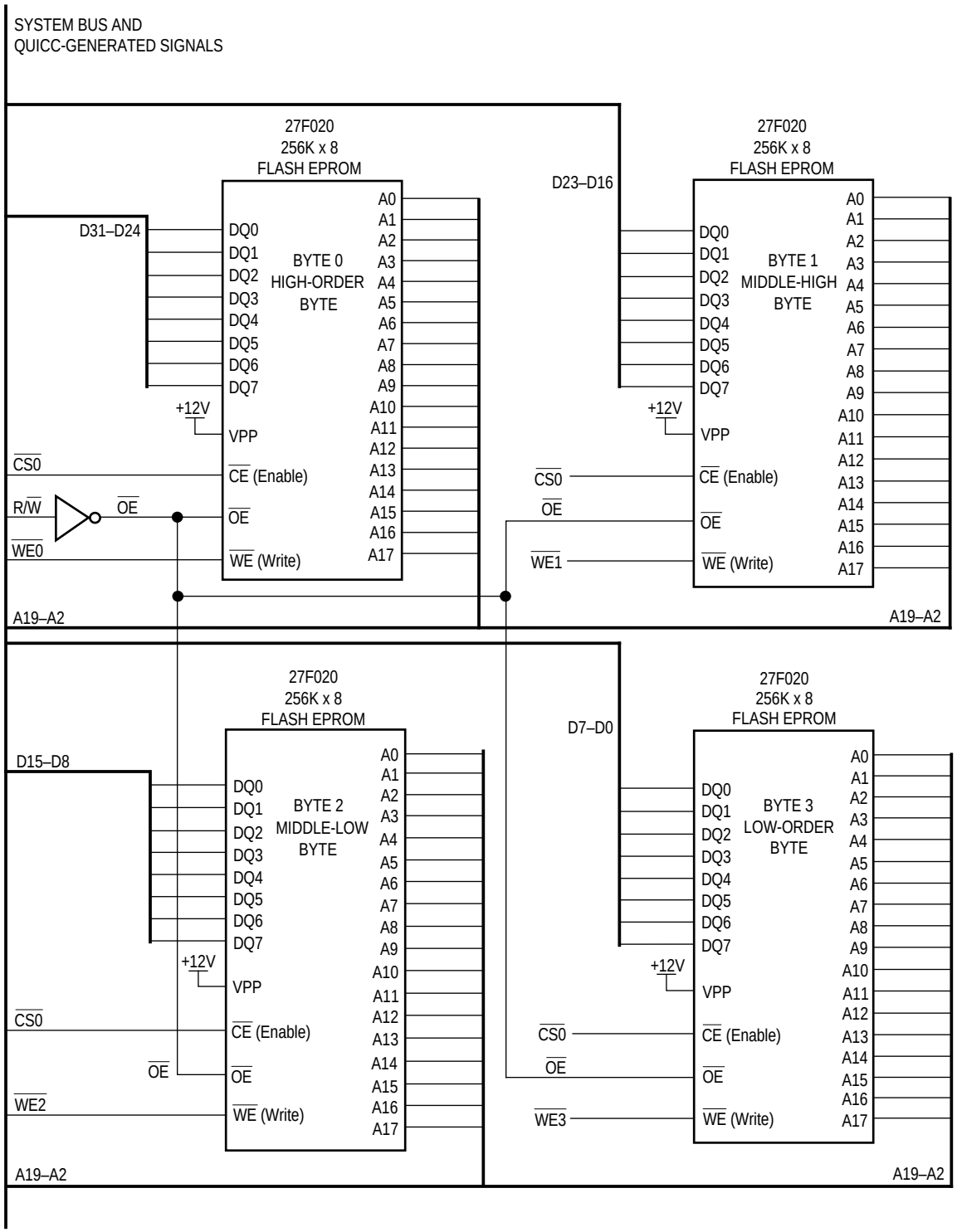


Figure 9-29. F1-Mbyte Flash EPROM Bank—32 Bits Wide

9.8.2.3 REGULAR SRAM. Figure 9-30 shows the interface to SRAM. In this design, both the MC68EC030 and the QUICC may access the SRAM array. The inverter is only required

if DRAM is used elsewhere in the system. The QUICC does not support bursting by the MC68EC030.

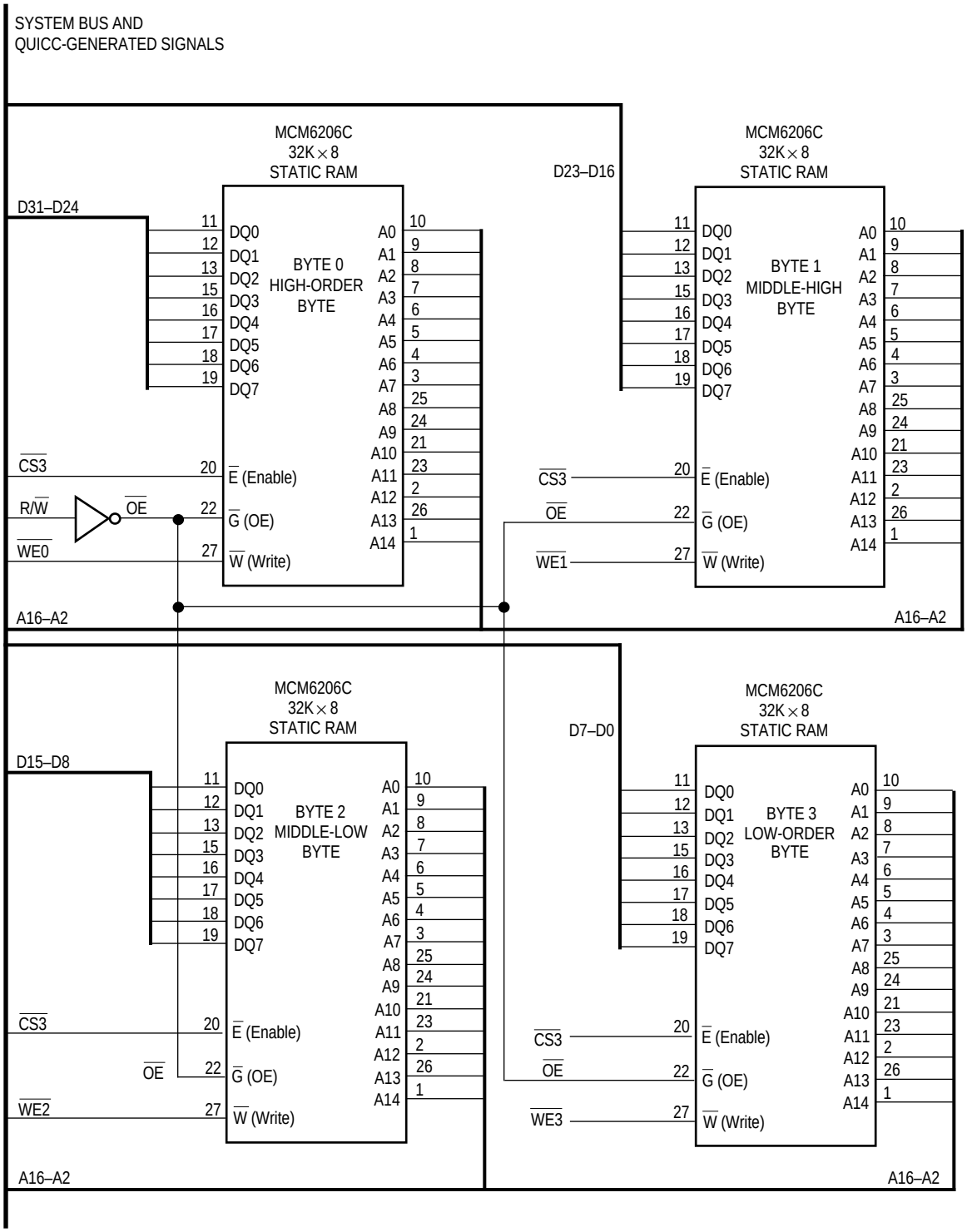


Figure 9-30. 128-Kbyte Static RAM Bank—32 Bits Wide

9.8.2.4 EEPROM. Figure 9-31 shows the interface to an EEPROM device to give a small amount of nonvolatile storage. In this case a byte-wide EEPROM bank is defined to minimize cost. If the port size of the chip select is selected to be 8-bits, then each byte of the EEPROM may be accessed in succession. The $\overline{CS4}$ pin should be programmed to respond to an 8K byte area in this design.

Only one byte should be written at a time. After a write is made, software is responsible for waiting the appropriate time (e.g. 10 ms) or for doing data polling to see if the newly written data byte is correct.

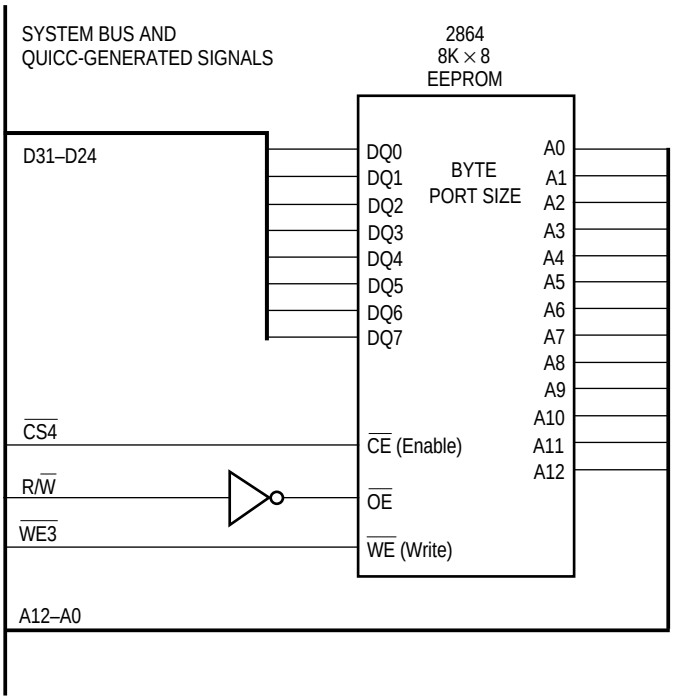


Figure 9-31. 8-Kbyte EEPROM Bank—8 Bits Wide

9.8.2.5 DRAM SIMM. Figure 9-32 shows the interface to an MCM32100S DRAM single inline memory module (SIMM). Both the MC68EC030 and the QUICC can access the DRAM.

When the QUICC is a slave to an external MC68EC030, the address multiplexing for the DRAM must be done externally to the QUICC, which is accomplished in the three F157 multiplexers. The external address multiplexing scheme is very simple and allows page mode operation to be provided for the MC68EC030, if desired. This multiplexing scheme externally provides, the same multiplexing method that the QUICC implements internally.

NOTE

This multiplexing scheme allows the use of page mode, but requires hardware modification if larger SIMMs are to be used on the board. If the user is interested in the latter, rather than the

former, see the DRAM multiplexing scheme in 9.4 Using the QUICC MC68040 Companion Mode.

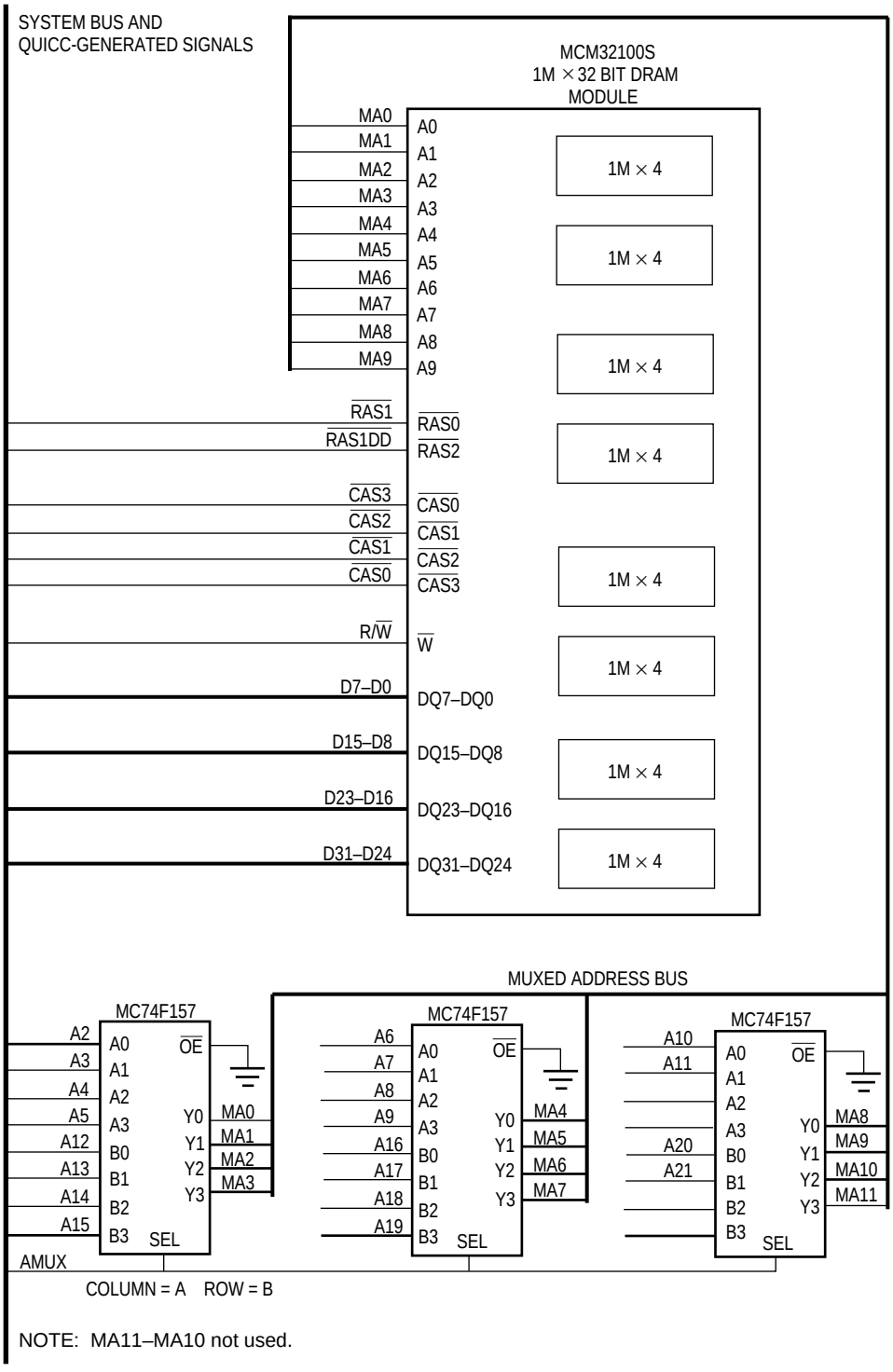


Figure 9-32. 4-Mbyte DRAM Bank—32 Bits Wide

This design also uses the $\overline{\text{RAS1}}$ double-drive capability, whereby the $\overline{\text{RAS1DD}}$ signal is output by the QUICC on the $\overline{\text{BCLRI}}$ pin to increase the effective drive capability of the $\overline{\text{RAS1}}$ signal. The $\overline{\text{RAS1}}$ line should be programmed to respond to a 4-Mbyte address space.

After power-on reset, the software must wait the required time before accessing the DRAM. The required eight read cycles must then be performed either in software or by waiting for the refresh controller to perform these accesses.

9.8.2.6 DRAM DEVICES. Figure 9-33 shows the interface to a standalone DRAM device. In this case the MCM54260 256K $\times$ 16 DRAM device is chosen. This allows a full 32-bit wide DRAM solution using only two DRAM devices, with byte writes still supported using the upper and lower $\overline{\text{CAS}}$ pins. Both the MC68EC030 and the QUICC can access the DRAM array. The $\overline{\text{RAS1}}$ line should be programmed to respond to a 1-Mbyte address space.

The address multiplexing scheme shown is the same as that for the DRAM SIMM. No parity support is provided in this case. The $\overline{\text{RAS1DD}}$ signal is not used in this case, since only two devices are supported.

After power-on reset, the software must wait the required time before accessing the DRAM, and then perform the required eight read cycles, either in software or by waiting for the refresh controller to perform these accesses.

9.8.3 Software Configuration

The following paragraphs discuss a number of key points for a software engineer desiring to initialize the system. The only items discussed are those that are required to allow the previously discussed hardware configuration.

9.8.3.1 BASIC INITIALIZATION. The following register initializations are basic to all types of applications.

The module base address register (MBAR) should be set as desired. However, the QUICC 8-Kbyte block should not overlap any memory array.

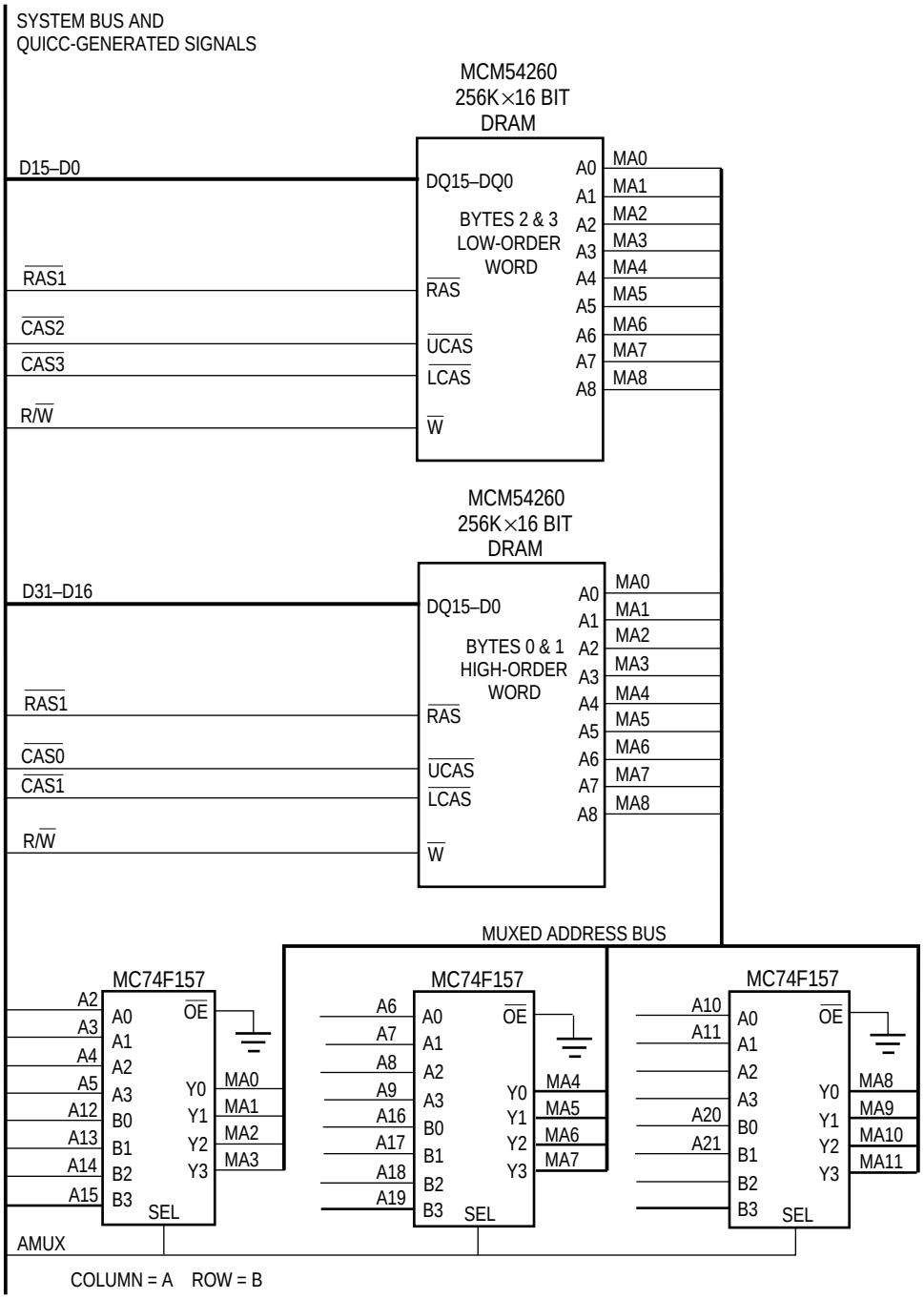
The module base address register enable (MBARE) should not be accessed.

In the module configuration register (MCR), ASTM and BSTM should be set to indicate synchronous operation. SHEN1–SHEN0 should be cleared.

In the system protection control register (SYPCR), DBFE should be cleared. BME should be set. If the software watchdog is used, the SWRI bit should be set.

The periodic interrupt control register (PICR) may be set as desired.

The port E pin assignment register (PEPAR) should be set to $51C0. This configures three $\overline{\text{IOUTx}}$ lines to go out on the unused parity pins, the $\overline{\text{RAS1DD}}$ pin, $\overline{\text{WE}}$ lines instead of the A31–A28 lines, the AMUX pin (assuming DRAM is used in the system; otherwise, the $\overline{\text{OE}}$ function should be programmed), four $\overline{\text{CASx}}$ lines, $\overline{\text{CS7}}$, and $\overline{\text{AVECO}}$.



NOTE: MA11–MA9 not used.

Figure 9-33. 1-Mbyte DRAM Bank—32 Bits Wide

9.8.3.2 CONFIGURING THE MEMORY CONTROLLER. The following register initializations are for the memory controller.

The global memory register (GMR) should be configured as follows:

The RFCNT bits may be set as desired. At 25 MHz, an RFCNT value of 24 (decimal) gives

one refresh every 15.6 μs.

RFEN should be set.

RCYC depends on the DRAM speed. At 25 MHz (an 80-ns DRAM SIMM), RCYC should be 00.

PGS2–PGS0 should be set to 011 for the 1M × 32 DRAM SIMM.

DPS should be set to 00 (32-bit DRAM port size).

WBT40 does not apply to this application.

WBTQ depends on timing; it should be set for 80-ns MCM32100 SIMMs.

DWQ should be set if page mode enabled (PGME = 1).

DW40 does not apply to this application.

EMWS is not used in synchronous mode. (SYNC = 1)

SYNC should be set for a synchronous operation of the memory controller.

OPAR does not apply to this application.

PBEE does not apply to this application.

TSS40 does not apply to this application.

NCS should normally be cleared.

GAMX should be cleared for an external master system.

The memory controller status register (MSTAT) is used for reporting write protect and parity errors and does not require initialization.

The eight base registers (BRs), one for each memory bank, should be configured as follows:

The BA27–BA11 bits may be set as desired. Different memory arrays should not overlap. BA31–BA28 should be cleared since the byte write lines are used with an external master in the system.

For simplicity, FC3–FC0 can be cleared.

TRLXQ depends on timing of memory/peripheral.

BACK40 does not apply to this application.

CSNT40 does not apply to this application.

CSNTQ should normally be cleared.

PAREN should be cleared since parity is not used in this application.

WP should be set for EPROM and flash EPROM; otherwise, it should be cleared.

V should be set if the memory bank is used.

The eight option registers (ORs), one for each memory bank, should be configured as follows:

The TCYC bits should be set to determine the number of wait states required.

The AM27–AM11 bits should be programmed to determine the block size of the chip select or $\overline{\text{RASx}}$ line. This should be the total number of bytes in each memory array except for the EEPROM, which should be 32 Kbytes, rather than 8 Kbytes.

FCM3–FCM0 may be set to all ones to allow the chip select or $\overline{\text{RASx}}$ line to assert on all function codes except CPU space (interrupt acknowledge). It is advisable to program FCM3–FCM0 to ones, at least during the initial stages of debugging.

BCYC1–BCYC0 is not applicable.

PGME should be set to enable page mode and cleared otherwise.

SPS1–SPS0 should be cleared (32-bit SRAM port).

DSSEL should be set only if this is a DRAM bank.

9.8.4 Interfacing Multiple QUICCs to an MC68EC030

It is possible to interface multiple QUICCs to an MC68EC030. The first QUICC can be configured as previously shown in this subsection. Additional QUICCs should be configured as noted in the following list:

- The additional QUICCs should have their CONFIG2–CONFIG0 pins configured for slave mode, global chip select *disabled*, and MBAR at $003FF04.
- The MBAR of the additional QUICCs should be programmed using the $\overline{\text{MBARE}}$ pin and MBARE register as described in the Section 6 System Integration Module (SIM60).
- An external bus arbiter is required to take the bus request of the additional QUICC (which is an output because of the CONFIG2–CONFIG0 pins) and prioritize it with the other QUICCs, present it to the MC68EC030, and issue a bus grant to the appropriate QUICC.
- An external interrupt prioritizer is required to determine which QUICC $\overline{\text{IOUT2}}$–$\overline{\text{IOUT0}}$ pins are currently routed to the MC68EC030. Alternatively, the additional QUICC should have its interrupts brought out on a single $\overline{\text{RQOUT}}$ pin, which is routed to one of the original QUICC interrupt inputs. This would eliminate the external logic.

9.8.5 Using a Higher Speed MC68EC030 Master with the QUICC

It is possible to interface an MC68EC030 and QUICC through an asynchronous bus. This should allow an external master to operate at higher frequencies than those of the QUICC with minimal effort. As of this writing, the QUICC top frequency is 25 MHz; whereas, MC68EC030s are available up to 40 MHz. One potentially attractive option for a designer would be to consider disabling the CPU32+ core and increasing system performance by adding a 40-MHz MC68EC030 asynchronously. While this option is available, it is important for the designer to consider what effects a higher speed MC68EC030 would ultimately have on system cost and performance over using the QUICC CPU32+ at a lower frequency.

For the designer to take full advantage of a high-speed MC68EC030, it will be necessary to add additional glue to that shown in Figure 9-27. The additional circuitry takes the form of a DRAM controller, which is used instead of using the QUICC memory controller. The need for the additional logic is twofold. First, if the QUICC memory controller capabilities are used, all memory accesses would be at the clock rate of 25 MHz. In addition, since the

MC68EC030 signals must be synchronized internally with the QUICC, additional wait states will be introduced. For applications with any significant level of external memory accesses, this would limit the additional cost/performance benefit of a 40-MHz MC68EC030. The second reason for needing an external DRAM controller is found by taking a closer look at how the QUICC memory controller would interface with a high-speed MC68EC030.

The signal interaction that would occur if two consecutive writes were performed is shown in Figure 9-27. After the assertion of $\overline{\text{DSACKx}}$ by the QUICC on a 25-MHz edge, the 40-MHz MC68EC030 would recognize $\overline{\text{DSACKx}}$ on the next falling 40-MHz edge, and proceed to negate $\overline{\text{AS}}$ one cycle after that. The problem is that the MC68EC030 begins another write cycle one-half MC68EC030 clock cycle later, and one-half clock cycle into that write cycle, $\overline{\text{AS}}$ is asserted. In asynchronous mode, since external signals are synchronized to the QUICC on QUICC falling clock edges, at 25 MHz there is no guarantee that the QUICC will see the negated $\overline{\text{AS}}$, which is present for only one MC68EC030 cycle (25 ns).

To take advantage of the QUICC memory controller capabilities, it would be necessary for designers to address issues like this. This task would need to be done with external hardware, such as a state machine, to ensure conformity to the QUICC electrical specifications. A second option would be to use an external memory controller and use the QUICC to generate only the chip selects. Either option will require additional hardware.

Designers considering an asynchronous interface also need to address timing issues associated with using the QUICC memory controller capabilities. When the QUICC provides the bus control for a system, it is offered at the speed of the QUICC. Therefore, a 40-MHz MC68EC030 will access the bus at 25 MHz. In addition, since the MC68EC030 signals must be synchronized with the QUICC, additional wait states will be introduced. Since the exact timing will depend on the type and speed of memory access, no exhaustive analysis is provided. The primary advantage of an MC68EC030 over the CPU32+ would be the onboard cache and the new instructions. Whether this advantage would outweigh the potential timing disadvantages will vary depending on system hardware and software considerations.

In any case, the QUICC makes a valuable peripheral chip in an MC68030-based design because of the wealth of serial functions it offers, even if the memory controller on the QUICC is not extensively used in such a design.

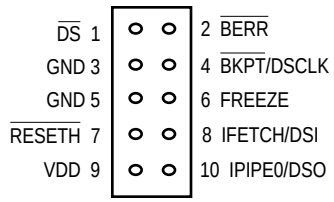
9.9 PUTTING A BACKGROUND DEBUG MODE CONNECTOR ON A TARGET BOARD

The QUICC, as well as other members of the M68300 family of integrated processors, contains a background debug mode (BDM). The BDM is essentially a debugger that is built into the CPU32+ on the QUICC. The user can communicate with the CPU32+ through the BDM port on the QUICC.

The BDM pins on the QUICC can be used in a number of ways. First, emulator manufacturers use these pins in the development of their QUICC emulator products. This use of the BDM pins is usually unseen and completely transparent to the user. Second, vendors of debug monitors use the BDM pins for offering low-cost debug monitor products. In this way, the target board can be monitored and debugged using a debugger resident on a PC. Third, some users develop their own BDM interface software to solve special debugging and test-

ing needs (such as in-field debugging). In the second and third cases, the target board needs to provide the BDM pins exposed so that an external monitor/debugger can connect to the target board. Figure 9-34 simply shows a standard connector for use with BDM equipment.

The standard connector originally an 8-pin connector, has been expanded to 10 pins as shown in Figure 9-34. This connector is sometimes referred to as the Berg connector. It has the standard 0.1-inch spacing between pins. The original 8 pins on the connector are the lower 8 pins (pins 3–10). If a debug monitor provides an 8-pin BDM connector, it should be plugged onto the original 8 pins. The additional 2 pins were added to allow hardware breakpoints to be implemented using the BDM connector. Since the QUICC contains an on-chip breakpoint address register with masking, this function can be implemented on-chip. To do this, the user can load the breakpoint address register in the SIM60 using the debugger commands (or under normal program control). Once execution of the program resumes, an address breakpoint then causes the QUICC to reenter BDM.



NOTES:
1. On other M68300 family devices RESETH is simply RESET, and IPIPE0 is simply IPIPE.
2. The original 8-pin connector consists of pins 3 through 10.

Figure 9-34. BDM Connector

SECTION 10
ELECTRICAL CHARACTERISTICS

This section contains detailed information on power considerations, DC/AC electrical characteristics, and AC timing specifications of the MC68360. Refer to Section 11 Ordering Information and Mechanical Data for specific part numbers corresponding to voltage, frequency, and temperature ratings.

10.1 MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage[1, 2]	V_{CC}	–0.3 to +6.5	V
Input Voltage[1, 2]	V_{in}	–0.3 to +6.5	V
Operating Temperature Range (CQFP and PGA for normal and extended temperature part)	T_A	0 to 70 or –40 to +85	°C
Maximum operating Junction Temperature (BGA only)	T_J	115	°C
Minimum operation Ambient Temperature (BGA only)	T_A	0	°C
Storage Temperature Range	T_{stg}	–55 to +150	°C

This device contains protective circuitry against damage due to high static voltages or electrical fields; however, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or V_{DD}).

NOTES:
1. Permanent damage can occur if maximum ratings are exceeded. Exposure to voltages or currents in excess of recommended values affects device reliability. Device modules may not operate normally while being exposed to electrical extremes.
2. Although sections of the device contain circuitry to protect against damage from high static voltages or electrical fields, take normal precautions to avoid exposure to voltages higher than maximum-rated voltages.

The following ratings define a range of conditions in which the device will operate without being damaged. However, sections of the device may not operate normally while being exposed to the electrical extremes.

10.2 THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance—Junction to Case 240-Pin QFP 241-Pin PGA 357-Pin BGA	θ_{JC}	 2 7 8	°C/W
Thermal Resistance—Junction to Ambient 240-Pin QFP 241-Pin PGA	θ_{JA}	 27.4 22.8	°C/W
Thermal Resistance—Junction to Ambient 357-pin BGA	θ_{JA}	47[1] 30[2] 20[3]	°C/W

Notes:
1. Assumes natural convection and a single layer board (no thermal vias).
2. Assumes natural convection, a multi layer board with thermal vias[4], 1.5 watt 68360 dissipation, and a board temperature rise of 30°C above ambient.
3. Assumes natural convection, a multi layer board with thermal vias[4],1.5 watt 68360 dissipation, and a board temperature rise of 15°C above ambient.
4. For more information on the design of thermal vias on multilayer boards and BGA layout considerations in general, refer to AN-1231/D, “Plastic Ball Grid Array Application Note” available from your local Motorola sales office.

$T_J = T_A + (P_D \bullet \theta_{JA})$
$P_D = (V_{DD} \bullet I_{DD}) + P_{I/O}$
where:
$P_{I/O}$ is the power dissipation on pins.

10.3 POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \theta_{JA}) \quad (1)$$

where:
T_A = Ambient Temperature, °C
θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W
P_D = $P_{INT} + P_{I/O}$
P_{INT} = $I_{CC} \times V_{CC}$, Watts—Chip Internal Power
$P_{I/O}$ = Power Dissipation on Input and Output Pins—User Determined

For most applications, $P_{I/O} < 0.3 * P_{INT}$ and can be neglected.

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving Equations (1) and (2) for K gives:

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2 \quad (3)$$

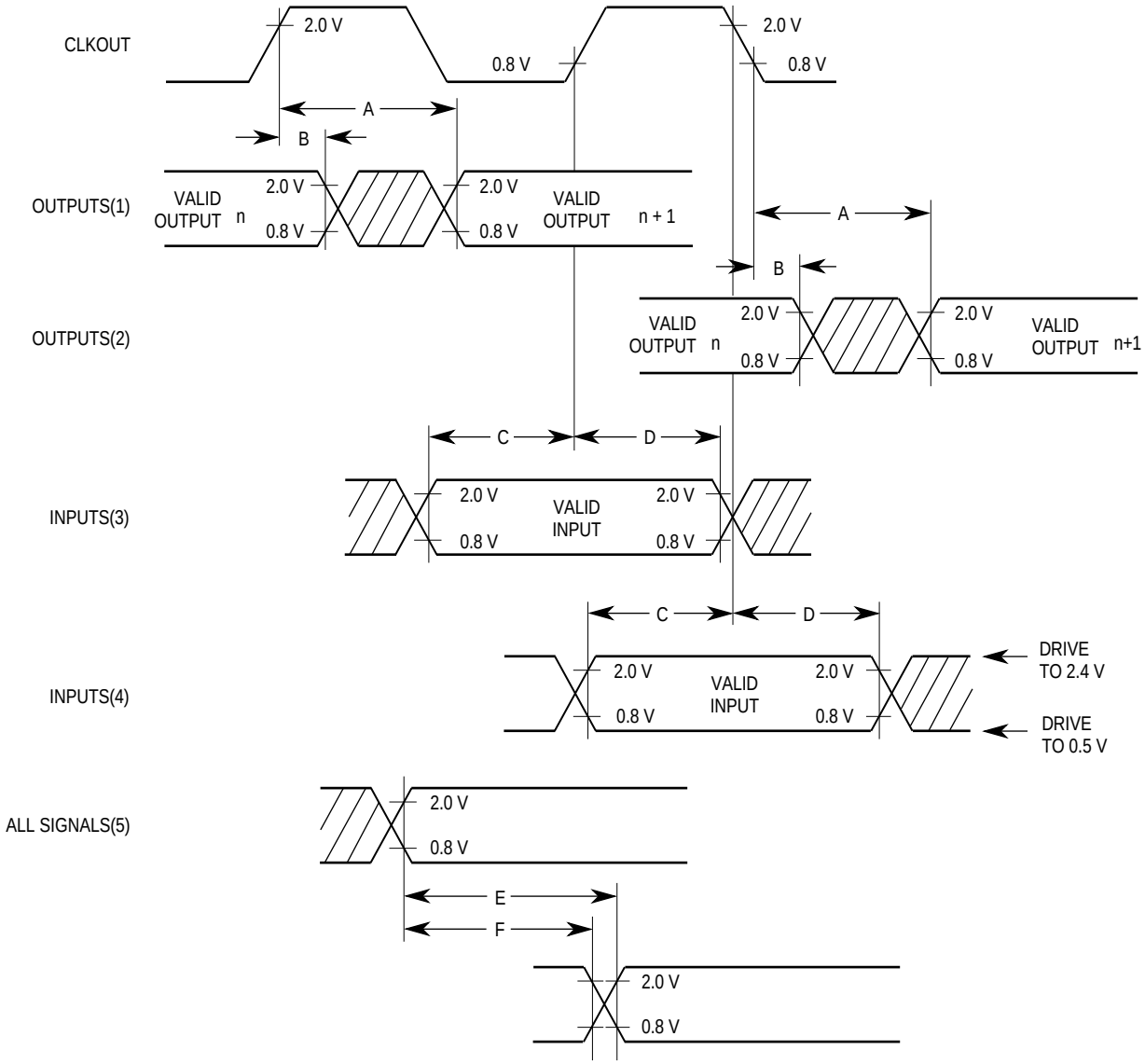
where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at thermal equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving Equations (1) and (2) iteratively for any value of T_A.

10.4 AC ELECTRICAL SPECIFICATION DEFINITIONS

The AC specifications presented consist of output delays, input setup and hold times, and signal skew times. All signals are specified relative to an appropriate edge of the clock and possibly to one or more other signals.

The measurement of the AC specifications is defined by the waveforms shown in Figure 10-1. To test the parameters guaranteed by Motorola, inputs must be driven to the voltage levels specified in the figure. Outputs are specified with minimum and/or maximum limits, as appropriate, and are measured as shown. Inputs are specified with minimum setup and hold times and are measured as shown. Finally, the measurement for signal-to-signal specifications are shown.

Note that the testing levels used to verify conformance to the AC specifications do not affect the guaranteed DC operation of the device as specified in the DC electrical characteristics.



NOTES:
1. This output timing is applicable to all parameters specified relative to the rising edge of the clock.
2. This output timing is applicable to all parameters specified relative to the falling edge of the clock.
3. This input timing is applicable to all parameters specified relative to the rising edge of the clock.
4. This input timing is applicable to all parameters specified relative to the falling edge of the clock.
5. This timing is applicable to all parameters specified relative to the assertion/negation of another signal.

LEGEND:
A. Maximum output delay specification.
B. Minimum output hold time.
C. Minimum input setup time specification.
D. Minimum input hold time specification.
E. Signal valid to signal valid specification (maximum or minimum).
F. Signal valid to signal invalid specification (maximum or minimum).

Figure 10-1. Drive Levels and Test Points for AC Specifications

10.5 DC Electrical Specifications

(GND = 0 Vdc, TA = 0 to 70°C; The electrical specifications in this document are preliminary; see numbered notes)

Characteristic	Symbol	Min	Max	Unit
Input High Voltage (except EXTAL)	V_{IH}	2.0	V_{CC}	V
Input Low Voltage	V_{IL}	GND	0.8	V
Input Low Voltage (3.3V Part Only; PA8-15, PB1, PC5, PC7 TCK)	V_{IL}	GND	0.5	V
Input Low Voltage (3.3V Part Only; All Other Pins)	V_{IL}	GND	0.8	V
EXTAL Input High Voltage	V_{IHC}	0.8*(V_{CC})	V_{CC}+0.3	V
Undershoot	—	—	–0.8	V
Input Leakage Current (All Input Only Pins except for TMS, TDI and TRST) V_{in} = V_{CC} or GND	I_{in}	–2.5	2.5	µA
Hi-Z (Off-State) Leakage Current (All Noncrystal Outputs and I/O Pins) V_{in} = 0.5/2.4 V	I_{OZ}	–20	20	µA
Signal Low Input Current V_{IL} = 0.8 V (TMS, TDI and TRST Pins Only) Signal High Input CurrentV_{IH} = 2.0 V(TMS, TDI and TRST Pins Only)	I_L IH	–0.5 –0.5	0.5 0.5	mA mA
Output High Voltage I_{OH} = –0.8 mA, V_{CC} = 4.75 V All Noncrystal Outputs Except Open Drain Pins	V_{OH}	2.4	—	V
Output Low Voltage (For 5 Volt Part) I_{OL} = 2.0 mACLKO1–2 ,FREEZE, IPIPE0–1, IFETCH, BKPTO I_{OL} = 3.2 mAA31–A0, D31–D0, FC3–0, SIZ0–1, PA0, 2, 4, 6, 8–15, PB0–5, PB8–17, PC0–11, TDO, PERR, PRTY0–3, IOUT0–2, AVECO, AS, CAS3–0, BLCRO, RAS0–7 I_{OL} = 5.3 mA, DSACK0–1, R/W, DS, OE, RMC, BG, BGACK, BERR I_{OL} = 7 mATXD1–4 I_{OL} =8.9 mAPB6, PB7, HALT, RESET, BR (Output)	V_{OL}	—	0.5 0.5 0.5 0.5 0.5	V
Output Low Voltage (For 3.3 volt part) I_{OL} = 2.0 mAA31–A0, FC3–0, SIZ0–1, D31–D0, PRTY0–3,CLKO1–2 ,FREEZE, IPIPE0–1, IFETCH, BKPTO I_{OL} = 3.2 mA PA0, 2, 4, 6, 8–15, PB0–5, PB8–17, PC0–11, TDO, PERR, PRTY0–3, IOUT0–2, AVECO, AS, CAS3–0, BLCRO, RAS0–7 I_{OL} = 5.3 mA, DSACK0–1, R/W, DS, OE, RMC, BG, BGACK, BERR I_{OL} = 7 mATXD1–4 I_{OL} =8.9 mAPB6, PB7, HALT, RESET, BR (Output)	V_{OL}	—	0.5 0.5 0.5 0.5 0.5	V
Input Capacitance All I/O Pins	C_{in}	—	20	pF
Load Capacitance (except CLKO1–2)	C_L	—	100	pF
Load Capacitance (CLKO1-2)	C_Lc		50	pF
Power (For 5 Volt Version)	Vcc	4.75	5.25	V
Power (For 3.3 Volt Version)	Vcc	3.0	3.6	V
Minimum Vcc Ramp up time on power on reset	^{t}Ranp	4	—	mSec

10.6 AC POWER DISSIPATION

Typical Curent Drain

Mode of Operation	Symbol	System Clock Frequency	BRGCLK Clock Frequency	SyncCLK Clock Frequency	Typ	Unit
Normal Mode (Rev A[1] and Rev B[2])	I_{DD}	25 MHz	25 MHz	25 MHz	250	ma
Normal Mode (Rev C[3] and Newer)	I_{DD}	25 MHz	25 MHz	25 MHz	237	ma
Normal Mode	I_{DD}	33 MHz	33 MHz	33 MHz	327	ma
Low Power Mode	I_{DDSB}	Divide by 2 12.5 MHz	Divide by 16 1.56 MHz	Divide by 2 12.5 MHz	150	ma
Low Power Mode	I_{DDSB}	Divide by 4 6.25 MHz	Divide by 16 1.56 MHz	Divide by 4 6.25 MHz	85	ma
Low Power Mode	I_{DDSB}	Divide by 16 1.56 MHz	Divide by 16 1.56 MHz	Divide by 4 6.25 MHz	35	ma
Low Power Mode	I_{DDSB}	Divide by 256 97.6 KHz	Divide by 16 1.56 MHz	Divide by 4 6.25 MHz	20	ma
Low Power Mode	I_{DDSB}	Divide by 256 97.6 KHz	Divide by 64 390 KHz	Divide by 64 390 KHz	13	ma
Low Power Stop VCO Off [5]	I_{DDSP}				0.5	ma
PLL Supply Current PLL Disabled PLL Enabled	I_{DDPD} I_{DDPE}				TBD TBD	

NOTES:
1. Rev A mask is C63T
2. Rev B masks are C69T, and F35G
3. Current Rev C masks are E63C, E68C and F15W
5. EXTAL frequency is 32KHz

All measurements was taken with only CLKO1 enabled, Vcc = 5.0V, V_{IL} = 0V and V_{IH} = Vcc

Maximum Power Dissipation

System Frequency	Vcc	Max P_D	Unit	Mask
25MHz	5.25 V	1.80	W	REV A[1] and REV B[2]
25MHz	5.25 V	1.45	W	REV C[3] and Newer
25MHz	3.6 V	0.65	W	REV C[3] and Newer
33MHz	5.25 V	2.00	W	REV C[3] and Newer

NOTES:
1. Rev A mask is C63T
2. Rev B masks are C69T, and F35G
3. Current Rev C masks are E63C, E68C and F15W

10.7 AC Electrical Specifications Control Timing

(GND = 0 Vdc, TA = 0 to 70°C; The electrical specifications in this document are preliminary; See Figure 10-2)

Num.	Characteristic	Symbol	3.3 V or 5.0 V		5.0 V		Unit
			25 MHz		33.34 MHz		
			Min	Max	Min	Max	
	System Frequency	f_{sys}	dc[1]	25.00		33.34	MHz
	Crystal Frequency	f_{XTAL}	25	6000	25	6000	kHz
	On-Chip VCO System Frequency	f_{sys}	20	50	20	67	MHz
	Start-up Time With external clock (oscillator disabled) or after changing the multiplication factor MF.	t_{pll}		2500			clks
	CLKO1–2 stability	Δ_{CLK}	TBD	TBD			%
1	CLKO1 Period	t_{cyc}	40	—	30	—	ns
1A	EXTAL Duty Cycle, MF	t_{dcyc}	40	60	40	60	%
1C	External Clock Input Period	t_{EXTcyc}	40	—	30	—	ns
2, 3	CLKO1 Pulse Width (Measured at 1.5v)	t_{CW1}	19	—	14	—	ns
2A, 3A	CLKO2 Pulse Width (Measured at 1.5v)	t_{CW2}	9.5	—	7	—	ns
4, 5	CLKO1 Rise and Fall Times (Full Drive)	t_{Crf1}	—	2	—	2	ns
4A, 5A	CLKO2 Rise and Fall Times (Full Drive)	t_{Crf2}	—	2	—	2	ns
5B	EXTAL to CLKO1 Skew—PLL enabled (MF<5)	t_{EXTP1}		a		a	ns
5C	EXTAL to CLKO2 Skew—PLL enabled (MF<5)	t_{EXTP2}		a		a	ns
5D	CLKO1 to CLKO2 Skew	t_{CSKW}		a		a	ns

Note: 1. Note that the minimum VCO frequency and the PLL default values put some restrictions on the minimum system frequency.

a: The following calculation should be used to determine the actual value for specifications 5B, 5C and 5D.

5B : 25Mhz +/-(0.9ns + 0.25 x (rise time)) (1.4ns@rise=2ns; 1.9ns@rise=4ns)

33Mhz +/-(0.5ns + 0.25 x (rise time)) (1ns@rise=2ns; 1.5ns@rise=4ns)

5C : 25/33Mhz +/-(2ns + 0.25 x (rise time)) (2.5ns@rise=2ns; 3ns@rise=4ns)

5D : 25Mhz +/-(3ns + 0.5 x (rise time)) (4ns@rise=2ns; 5ns@rise=4ns)

33Mhz +/-(2.5ns + 0.5 x (rise time)) (3.5ns@rise=2ns; 4.5ns@rise=4ns)

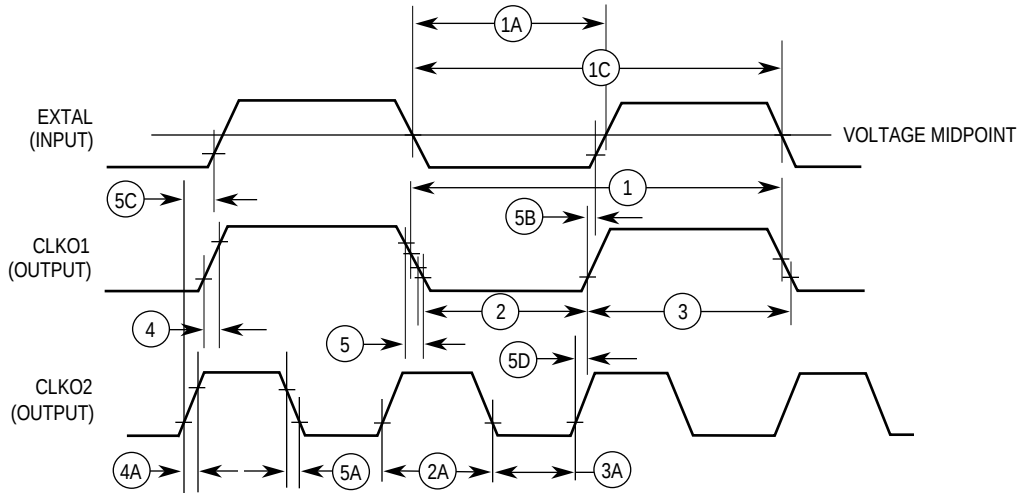


Figure 10-2. Clock Timing

10.8 EXTERNAL CAPACITOR FOR PLL

Characteristic	Symbol	3.3v		5.0 V		Unit
		Min	Max	Min	Max	
PLL external capacitor (XFC to VCCSYN) MF<5 (recommended value MF × 400 pF) MF>4 (recommended value MF × 540 pF)	c_{XFC}	 MFX680 MFX760	 MFX960 MFX1940	 MFX340 MFX380	 MFX480 MFX970	 pF pF

Note: 1. MF – multiplication factor.

Examples:

1. MODCK1 pin = 0, MF = 1 $\Rightarrow c_{XFC}$ = 400 pF
2. MODCK1 pin = 1, crystal is 32.768 KHz (or 4.192 MHz), initial MF = 401, initial frequency = 13.14 MHz, later on MF is changed to 762 to support a frequency of 25 MHz.

   Minimum c_{XFC} is: 762 x 380 = 289 nF, maximum c_{XFC} is: 401 x 970 = 390 nF. The recommended c_{XFC} for 25 MHz is: 762 x 540 = 414 nF.

   289 nF < c_{XFC} < 390 nF and closer to 414 nF. The proper available value for c_{XFC} is 390 nF.
3. MODCK1 pin = 1, crystal is 32.768 KHz (or 4.192 MHz), initial MF = 401, initial frequency = 13.14 MHz; later, MF is changed to 1017 to support a frequency of 33.34 MHz.

   Minimum c_{XFC} is: 1017 x 380 = 386 nF. Maximum c_{XFC} is: 401 x 970 = 390 nF $\Rightarrow$ 386 nF < c_{XFC} < 390 nF.

   The proper available value for c_{XFC} is 390 nF.

3A. In order to get higher range, higher crystal frequency can be used (i.e. 50 KHz), in this case:

   Minimum c_{XFC} is: 667 x 380 = 253 nF. Maximum c_{XFC} is: 401 x 970 = 390 nF $\Rightarrow$ 253 nF < c_{XFC} < 390 nF.

10.9 BUS OPERATION AC TIMING SPECIFICATIONS

(The electrical specifications in this document are preliminary; See Figure 10-3–Figure 10-19)

Num.	Characteristic	Symbol	3.3 V/5.0 V		5.0V		Unit
			25.0 MHz		33.34MHz		
			Min	Max	Min	Max	
6	CLKO1 High to Address, FC, SIZ, RMC Valid	t_{CHAV}	0	15	0	12	ns
6A	CLKO1 High to Address Valid (GAMX=1)	t_{CHAV}	0	20	0	15	ns
7	CLKO1 High to Address, Data, FC, SIZ, RMC High Impedance	t_{CHAZx}	0	40	0	30	ns
8	CLKO1 High to Address, FC, SIZ, RMC Invalid	t_{CHAZn}	0	—	0	—	ns
9	CLKO1 Low to AS, DS, OE, WE, IFETCH, IPIPE, IACK Asserted	t_{CLSA}	3	20	3	15	ns
9[10]	CLKO1 Low to CS/RAS Asserted	t_{CLSA}	4	16	4	12	ns
9B[11]	CLKO1 High to CS/RAS Asserted	t_{CHCA}	4	16	4	12	ns
9A[2,10]	AS to DS or CS/RAS or OE Asserted (Read)	t_{STSA}	–6	6	-5.625	5.625	ns
9C[2,11]	AS to CS/RAS Asserted	t_{STCA}	14	26	9	21	ns
11[10]	Address, FC, SIZ, RMC Valid to AS, CS/RAS, OE, WE (and DS Read) Asserted	t_{AVSA}	10	—	8	—	ns
11A[11]	Address, FC, SIZ, RMC Valid to CS/RAS Asserted	t_{AVCA}	30	—		—	ns
12[12]	CLKO1 Low to AS, DS, OE, WE, IFETCH, IPIPE, IACK Negated	t_{CLSN}	3	20	3	15	ns
12[16]	CLKO1 Low to CS/RAS Negated	t_{CLSN}	4	16	4	12	ns
12A[13,16]	CLKO1 High to CS/RAS Negated	t_{CHCN}	4	16	4	12	ns
12B	CS negate to WE negate (CSNTQ=1)	t_{CSTW}	15	—	12	—	ns
13[12]	AS, DS, CS, OE, WE, IACK Negated to Address, FC, SIZ Invalid (Address Hold)	t_{SNAI}	10	—	7.5	—	ns
13A[13]	CS Negated to Address, FC, SIZ Invalid (Address Hold)	t_{CNAI}	30	—		—	ns
14[10,12]	AS, CS, OE, WE (and DS Read) Width Asserted	t_{SWA}	75	—		—	ns
14C[11,13]	CS Width Asserted	t_{CWA}	35	—		—	ns
14A	DS Width Asserted (Write)	t_{SWAW}	35	—		—	ns
14B	AS, CS, OE, WE, IACK (and DS Read) Width Asserted (Fast Termination Cycle)	t_{SWDW}	35	—		—	ns
14D[13]	CS Width Asserted (Fast Termination Cycle)	t_{CWDW}	15	—	10	—	ns
15[3,10,12]	AS, DS, CS, OE, WE Width Negated	t_{SN}	35	—		—	ns
16	CLKO1 High to AS, DS, R/W High Impedance	t_{CHSZ}	—	40	—	30	ns
17[12]	AS, DS, CS, WE Negated to R/W High	t_{SNRN}	10	—	7.5	—	ns
17A[13]	CS Negated to R/W High	t_{CNRN}	30	—		—	ns
18	CLKO1 High to R/W High	t_{CHRH}	0	20	0	15	ns
20	CLKO1 High to R/W Low	t_{CHRL}	0	20	0	15	ns
21[10]	R/W High to AS, CS, OE Asserted	t_{RAAA}	10	—	7.5	—	ns

10.9 BUS OPERATION AC TIMING SPECIFICATIONS (CONTINUED)

Num.	Characteristic	Symbol	3.3 V/5.0 V 25.0 MHz		5.0V 33.34MHz		Unit
			Min	Max	Min	Max	
21A[11]	R/W High to CS Asserted	t_{RACA}	30	—		—	ns
22	R/W Low to DS Asserted (Write)	t_{RASA}	47	—	36	—	ns
23	CLKO1 High to Data-Out, Parity-Out Valid	t_{CHDO}	—	23	—	15	ns
23A	CLKO1 High to parity Valid	t_{CHPV}	—	25	—	15	ns
23B	Parity Valid to $\overline{CAS}$ low	t_{PVCL}	3	—	3	—	ns
24[12]	Data-Out, Parity-Out Valid to Negating Edge of AS, CS, WE, (Fast Termination Write)	t_{DVASN}	10	—	7.5	—	ns
25[12]	DS, CS, WE Negated to Data-Out, Parity-Out Invalid (Data-Out, Parity-Out Hold)	t_{SNDOI}	10	—	7.5	—	ns
25A[13]	CS Negated to Data-Out, Parity-Out Invalid (Data-Out, Parity-Out Hold)	t_{CNDOI}	35	—	25	—	ns
26	Data-Out, Parity-Out Valid to DS Asserted (Write)	t_{DVSA}	10	—	7.5	—	ns
27[15]	Data-In, Parity-In to CLKO1 Low (Data Setup)	t_{DICL}	1	—	1	—	ns
27B[14]	Data-In, Parity-In Valid to CLKO1 Low (Data Setup)	t_{DICL}	20	—	15	—	ns
27A	Late BERR, HALT, BKPT Asserted to CLKO1 Low (Setup Time)	t_{BELCL}	10	—	7.5	—	ns
28[18]	AS, DS Negated to DSACK, BERR, HALT Negated	t_{SNDN}	0	50	0	37.5	ns
29[4]	DS, CS, OE Negated to Data-In, Parity-In Invalid (Data-In, Parity-In Hold)	t_{SNDI}	0	—	0	—	ns
29A[4]	DS, CS, OE Negated to Data-In High Impedance	t_{SHDI}	—	40	—	30	ns
30[4]	CLKO1 Low to Data-In, Parity-In Invalid (Fast Termination Hold)	t_{CLDI}	10	—	7.5	—	ns
30A[4]	CLKO1 Low to Data-In High Impendance	t_{CLDH}	—	60	—	45	ns
31[5,15]	DSACK Asserted to Data-In, Parity-In Valid	t_{DADI}	—	32	—	24	ns
31A	DSACK Asserted to DSACK Valid (Skew)	t_{DADV}	—	10	—	7.5	ns
31B[5,14]	DSACK Asserted to Data-In, Parity-In Valid	t_{DADI}	—	35	—	26	ns
32	HALT and RESET Input Transition Time	t_{HRrf}	—	140	—		ns
33	CLKO1 High to BG Asserted	t_{CLBA}	—	20	—	15	ns
34	CLKO1 High to BG Negated	t_{CLBN}	—	20	22.5	15	ns
35[6]	BR Asserted to BG Asserted (RMC Not Asserted)	t_{BRAGA}	1	—	1	—	CLKO1
37	BGACK Asserted to BG Negated	t_{GAGN}	1	2.5	1	2.5	CLKO1
39	BG Width Negated	t_{GH}	2	—	2	—	CLKO1
39A	BG Width Asserted	t_{GA}	1	—	1	—	CLKO1
46	R/W Width Asserted (Write or Read)	t_{RWA}	100	—	75	—	ns
46A	R/W Width Asserted (Fast Termination Write or Read)	t_{RWAS}	75	—	56	—	ns
47A	Asynchronous Input Setup Time	t_{AIST}	5	—	4	—	ns
47B	Asynchronous Input Hold Time	t_{AIHT}	10	—	7.5	—	ns

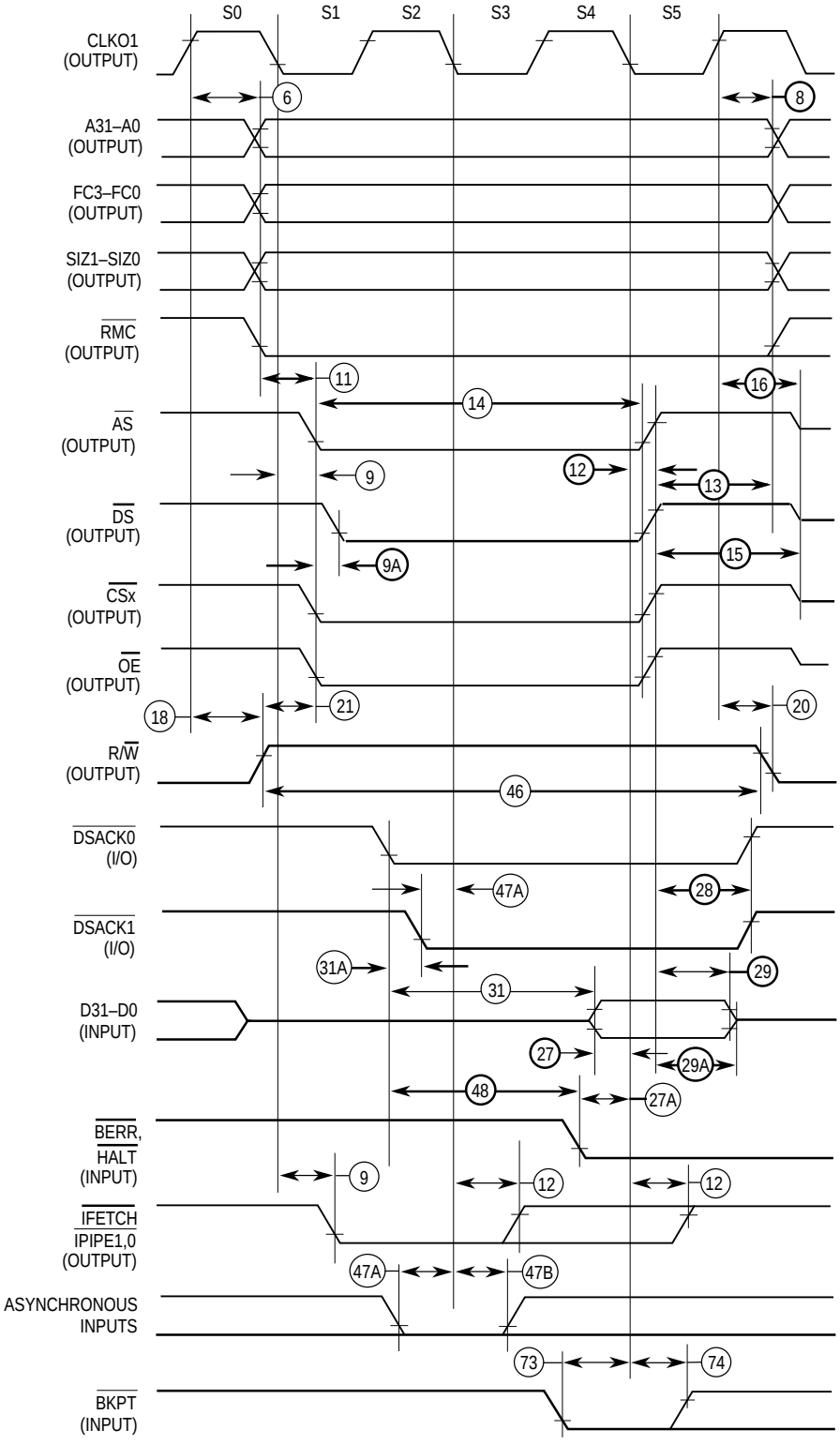
10.9 BUS OPERATION AC TIMING SPECIFICATIONS (CONTINUED)

Num.	Characteristic	Symbol	3.3 V/5.0 V 25.0 MHz		5.0V 33.34MHz		Unit
			Min	Max	Min	Max	
48[5,7]	DSACK Asserted to BERR , HALT Asserted	t_{DABA}	—	30	—	22.5	ns
49	CLKO1 high to BERR, RESETS, RESETH output Low	t_{CHRO}	—	—	—	—	ns
53	Data-Out, Parity-Out Hold from CLKO1 High	t_{DOCH}	0	—	0	—	ns
54	CLKO1 High to Data-Out, Parity-Out High Impedance	t_{CHDH}	—	20	—	15	ns
55	R/W Asserted to Data Bus Impedance Change	t_{RADC}	25	—	19	—	ns
56	RESET Pulse Width (Reset Instruction)	t_{HRPW}	512	—	512	—	CLKO1
56A	RESET Pulse Width (Input from External Device)	t_{RPWI}	20	—	20	—	CLKO1
57	BERR Negated to HALT Negated (Rerun)	t_{BNHN}	0	—	0	—	ns
58	CLKO1 High to BERR, RESETS, RESETH Driven Low	t_{CHBRL}	—	30		26	ns
58A	CLKO1 Low to RESETS Driven Low (upon Reset Instruction execution only)	t_{CLRL}	—	30		26	ns
60	CLKO1 High to BCLR0 Asserted	t_{CHBCA}	—	20	—	15	ns
61	CLKO1 High to BCLR0 Negated	t_{CHBCN}	—	20	—	15	ns
62[9]	BR Synchronous Setup Time	t_{BRSU}	5	—	3.75	—	ns
63[9]	BR Synchronous Hold Time	t_{BRH}	10	—	7.5	—	ns
64[9]	BGACK Synchronous Setup Time	t_{BGSU}	5	—	3.75	—	ns
65[9]	BGACK Synchronous Hold Time	t_{BGH}	10	—	7.5	—	ns
66	BR low to CLKO1 Rising Edge (040 comp. mode)	t_{BRCH}	5	—	5	—	ns
70	CLKO1 Low to Data Bus Driven (Show Cycle)	t_{SCLDD}	0	30	0	22.5	ns
71	Data Setup Time to CLKO1 Low (Show Cycle)	t_{SCLDS}	10	—	7.5	—	ns
72	Data Hold from CLKO1 Low (Show Cycle)	t_{SCLDH}	6	—	3.75	—	ns
73	BKPT Input Setup Time	t_{BKST}	10	—	7.5	—	ns
74	BKPT Input Hold Time	t_{BKHT}	6	—	3.75	—	ns
75	RESETH Low to Config2–0, MOD1–0, B16M valid	t_{MST}	—	500	—	500	CLKO1
76	Config2–0	t_{MSH}	0	—	0	—	ns
77	MOD1–0 Hold Time, B16M Hold Time	t_{MSH}	10	—	10	—	CLKO1
80	DSI Input Setup Time	t_{DSISU}	10	—	7.5	—	ns
81	DSI Input Hold Time	t_{DSIH}	6	—	3.75	—	ns
82	DSCLK Setup Time	t_{DSCSU}	10	—	7.5	—	ns
83	DSCLK Hold Time	t_{DSCH}	6	—	3.75	—	ns
84	DSO Delay Time	t_{DSOD}	—	t_{cyc}+ 20	—	tcyc+ 20	ns
85	DSCLK Cycle	t_{DSCCYC}	2	—	2	—	CLKO1
86	CLKO1 High to Freeze Asserted	t_{FRZA}	0	35	0	26.25	ns
87	CLKO1 High to Freeze Negated	t_{FRZN}	0	35	0	26.25	ns

10.9BUS OPERATION AC TIMING SPECIFICATIONS (CONTINUED

Num.	Characteristic	Symbol	3.3 V/5.0 V 25.0 MHz		5.0V 33.34MHz		Unit
			Min	Max	Min	Max	
88	CLKO1 High to IFETCH High Impedance	t_{IFZ}	0	35	0	26.25	ns
89	CLKO1 High to IFETCH Valid	t_{IF}	0	35	0	26.25	ns
90	CLKO1 High to PERR Asserted	t_{CHPA}	0	20	0	15	ns
91	CLKO1 High to PERR Negated	t_{CHPN}	0	20	0	15	ns
92	Minimum Vcc Ramp-Up Time At Power-On Reset	t_{RMIN}	5	-	5	-	ms

NOTES:
1. All AC timing is shown with respect to 0.8 V and 2.0 V levels unless otherwise noted.
2. This number can be reduced to 5 ns if strobes have equal loads.
3. If multiple chip selects are used, the CS width negated (#15) applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select.
4. These hold times are specified with respect to DS or CS on asynchronous reads and with respect to CLKO1 on fast termination reads. The user is free to use either hold time for fast termination reads.
5. If the asynchronous setup time (#47) requirements are satisfied, the DSACK low to data setup time (#31) and DSACK low to BERR low setup time (#48) can be ignored. The data must only satisfy the data-in to CLKO1 low setup time (#27) for the following clock cycle: BERR must only satisfy the late BERR low to CLKO1 low setup time (#27A) for the following clock cycle.
6. To ensure coherency during every operand transfer, BG will not be asserted in response to BR until after cycles of the current operand transfer are complete and RMC is negated.
7. In the absence of DSACK, BERR is an asynchronous input using the asynchronous setup time (#47).
8. During interrupt acknowledge cycles, the processor may insert up to two wait states between states S0 and S1.
9. These specs are for Synchronous Arbitration only. ASTM=1.
10. These CS specs are for TRLX=0. If RAS and RAS DD are connected together, reduce max value of RAS specification by 1.5ns.
11. These CS specs are for TRLX=1. If RAS and RAS DD are connected together, reduce max value of RAS specification by 1.5ns.
12. These CS specs are for CSNTQ=0.
13. These CS specs are for CSNTQ=1; or RAS specs for DRAM accesses.
14. These specs are read cycles with parity check and PBEE=1.
15. These specs are read cycles with parity check and PBEE=0,PAREN=1.
16. These RAS specs are for page miss case.
17. These specifications only apply to CS/RAS pins.
18. This specification applies to non fast termination cycles. In fast termination cycles, the $\overline{BERR}$ signal must be negated by 20ns after negation of $\overline{AS}$, $\overline{DS}$.



NOTE: All timing is shown with respect to 0.8-V and 2.0-V levels.

Figure 10-3. Read Cycle

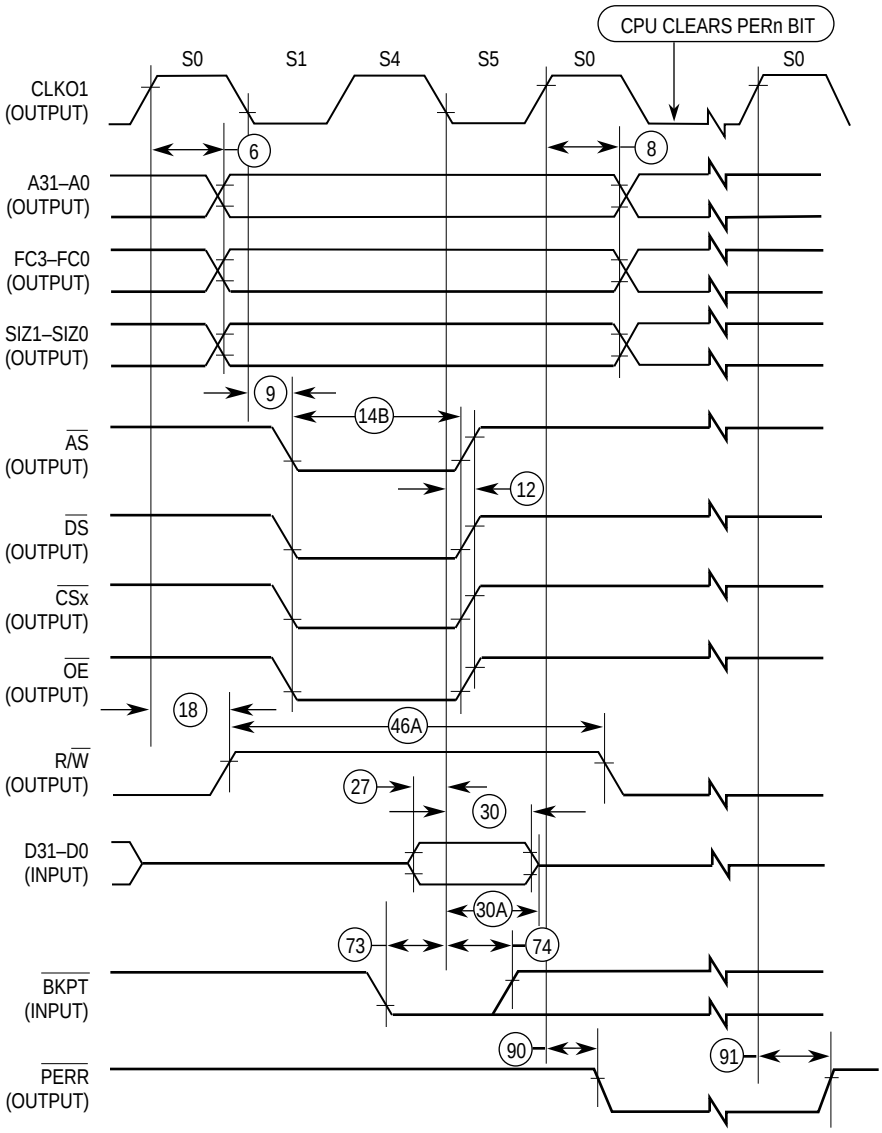
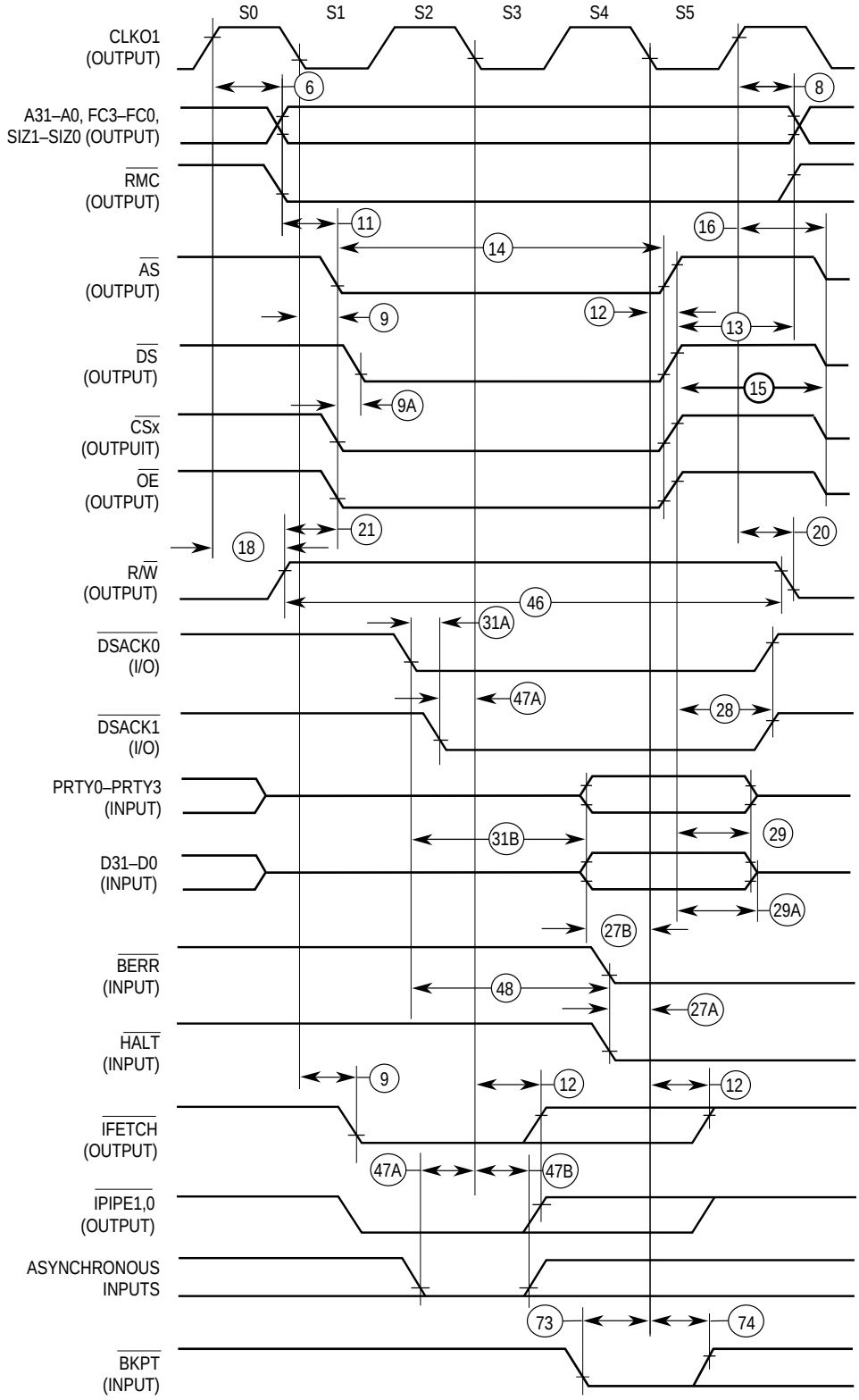


Figure 10-4. Fast Termination Read Cycle (Parity Check PAREN = 1, PBEE = 0)



NOTE: All timing is shown with respect to 0.8-V and 2.0-V levels.

Figure 10-5. Read Cycle (With Parity Check, PBEE = 1)

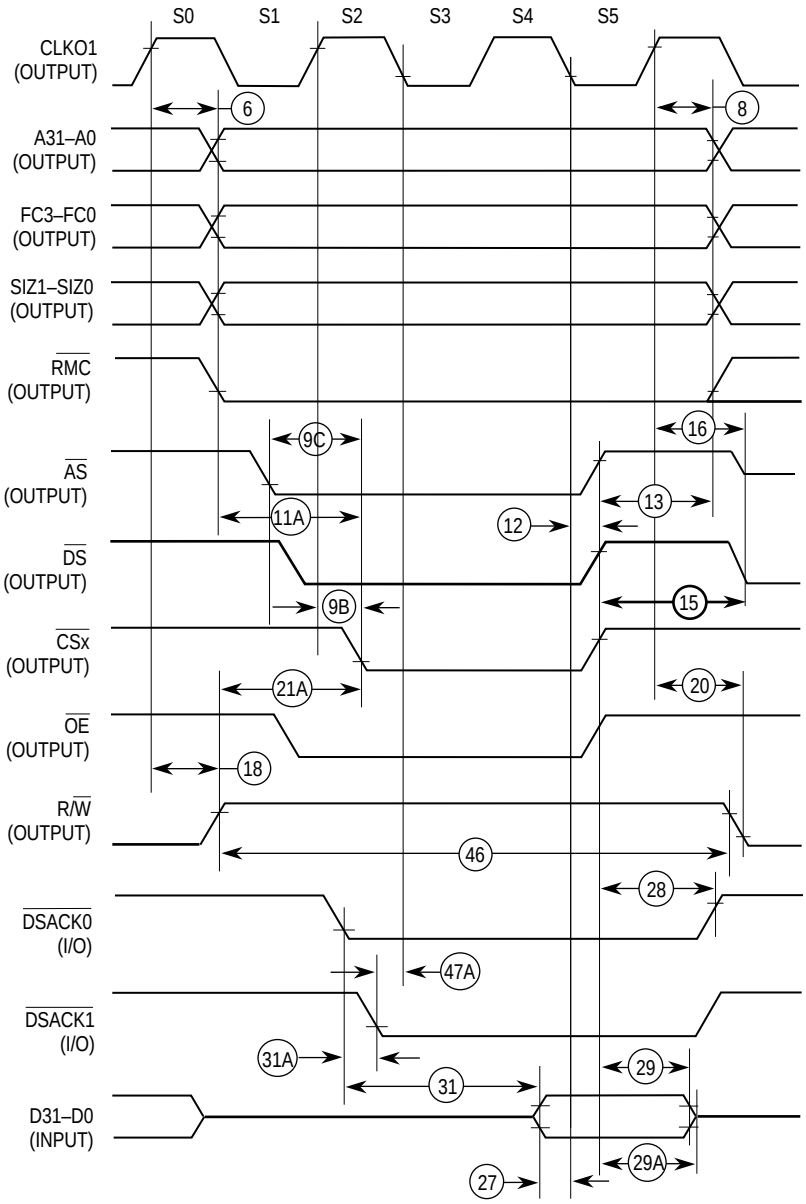
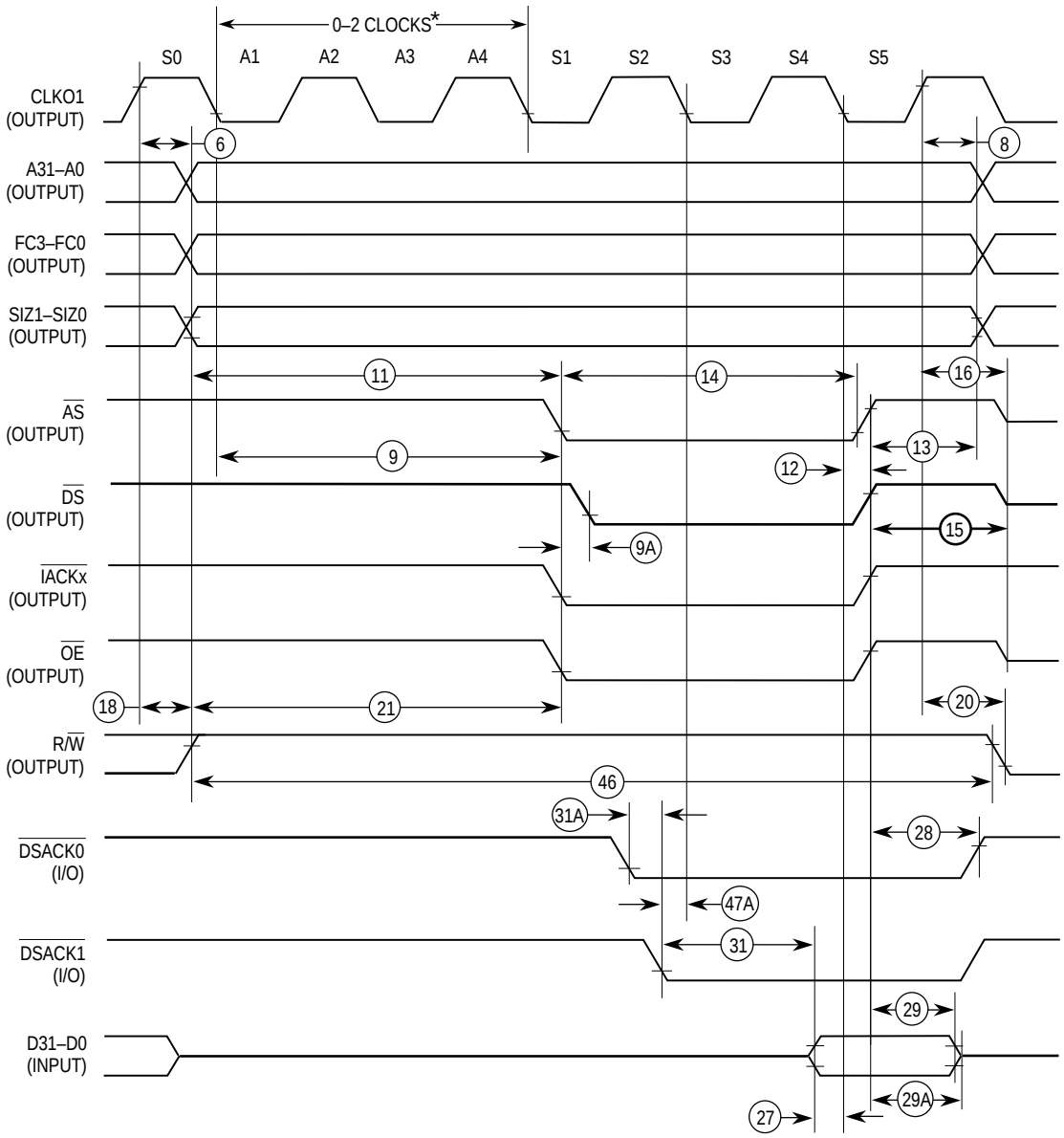
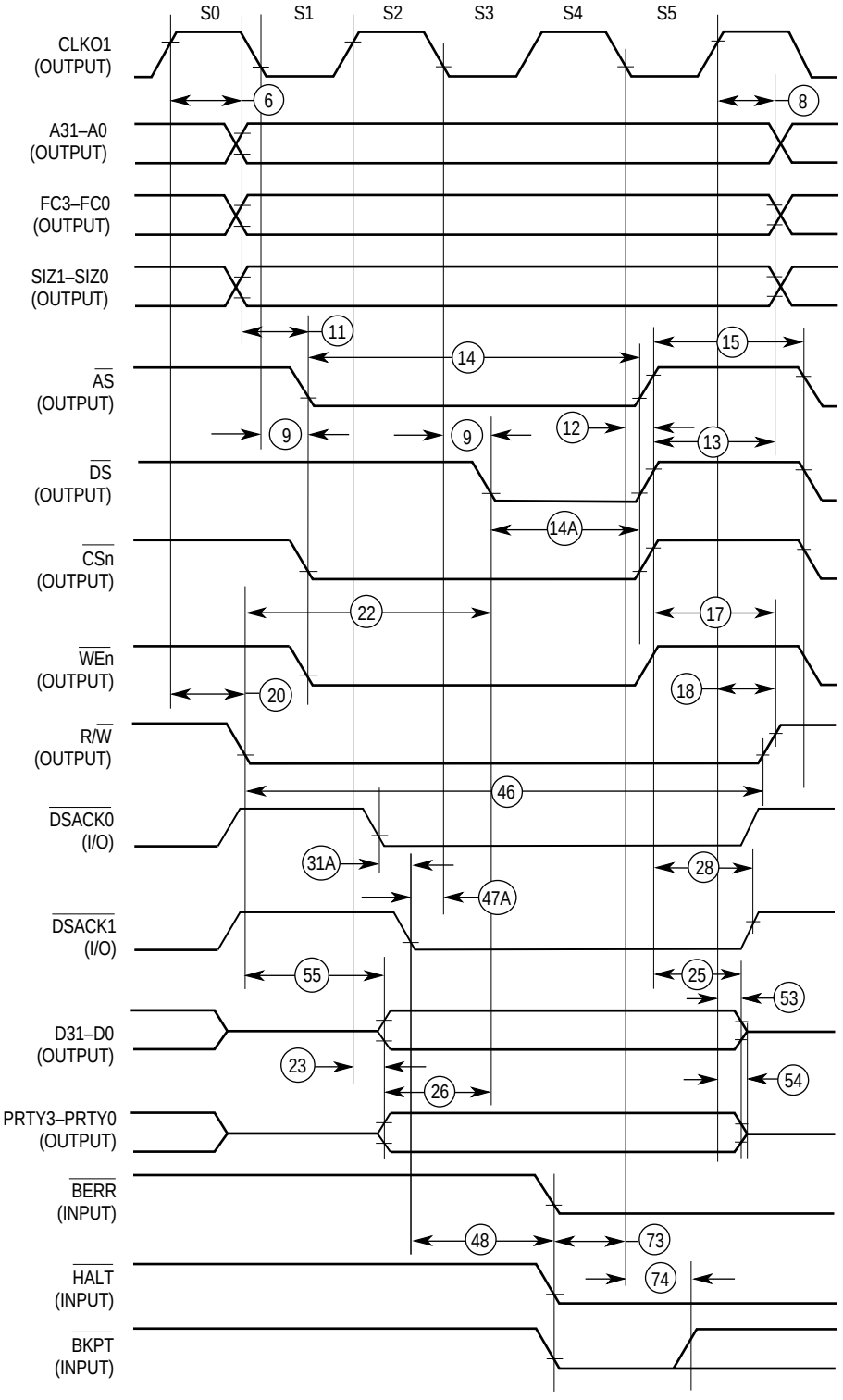


Figure 10-6. SRAM: Read Cycle (TRLX = 1)



*Up to two wait states may be inserted by the processor between states S0 and S1.

Figure 10-7. CPU32+ IACK Cycle



NOTE: All timing is shown with respect to 0.8-V and 2.0-V levels.

Figure 10-8. Write Cycle

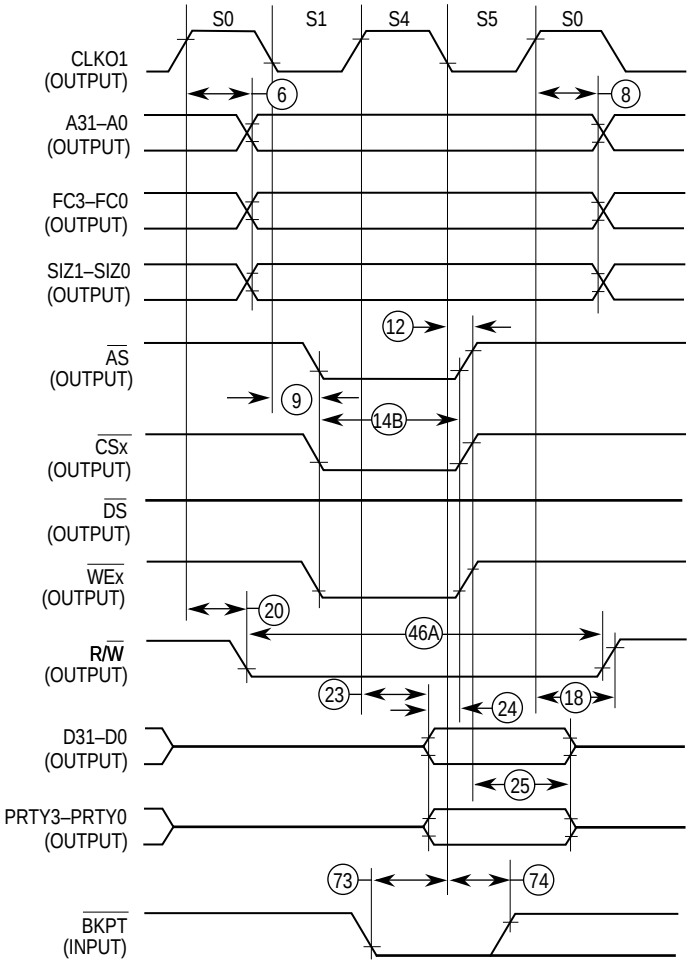


Figure 10-9. Fast Termination Write Cycle

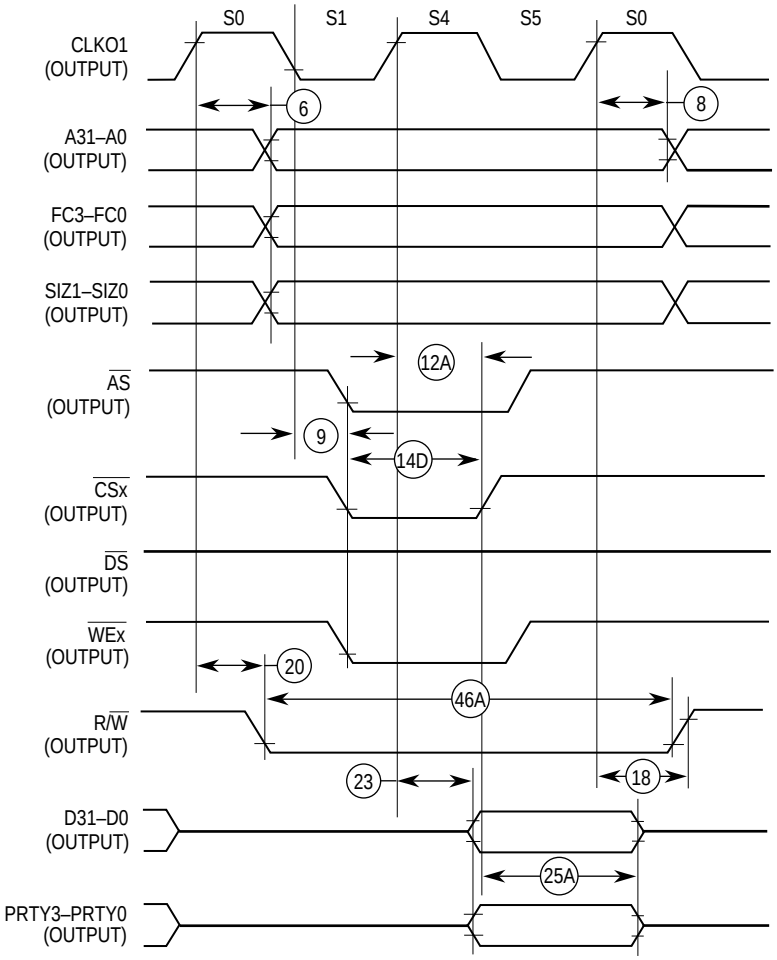
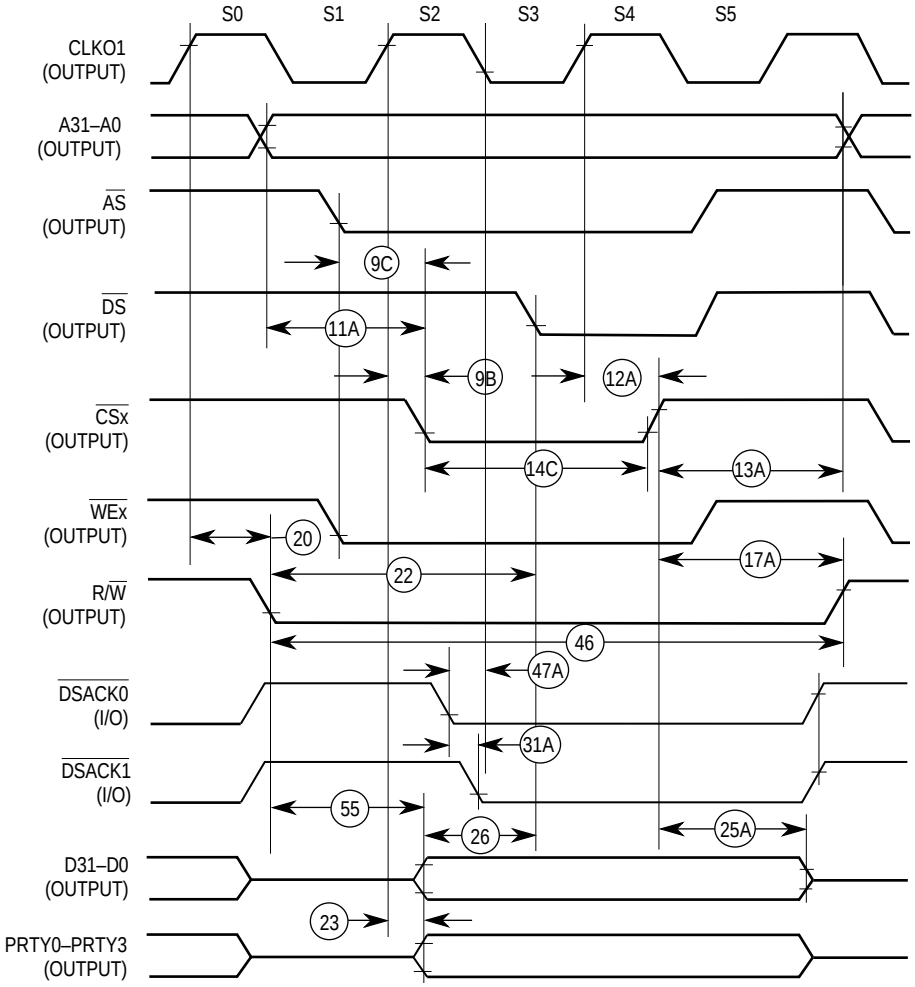


Figure 10-10. SRAM: Fast Termination Write Cycle (CSNTQ = 1)



NOTE: All timing is shown with respect to 0.8-V and 2.0-V levels.

Figure 10-11. SRAM: Write Cycle (TRLX = 1, CSNTQ = 1, TCYC = 0)

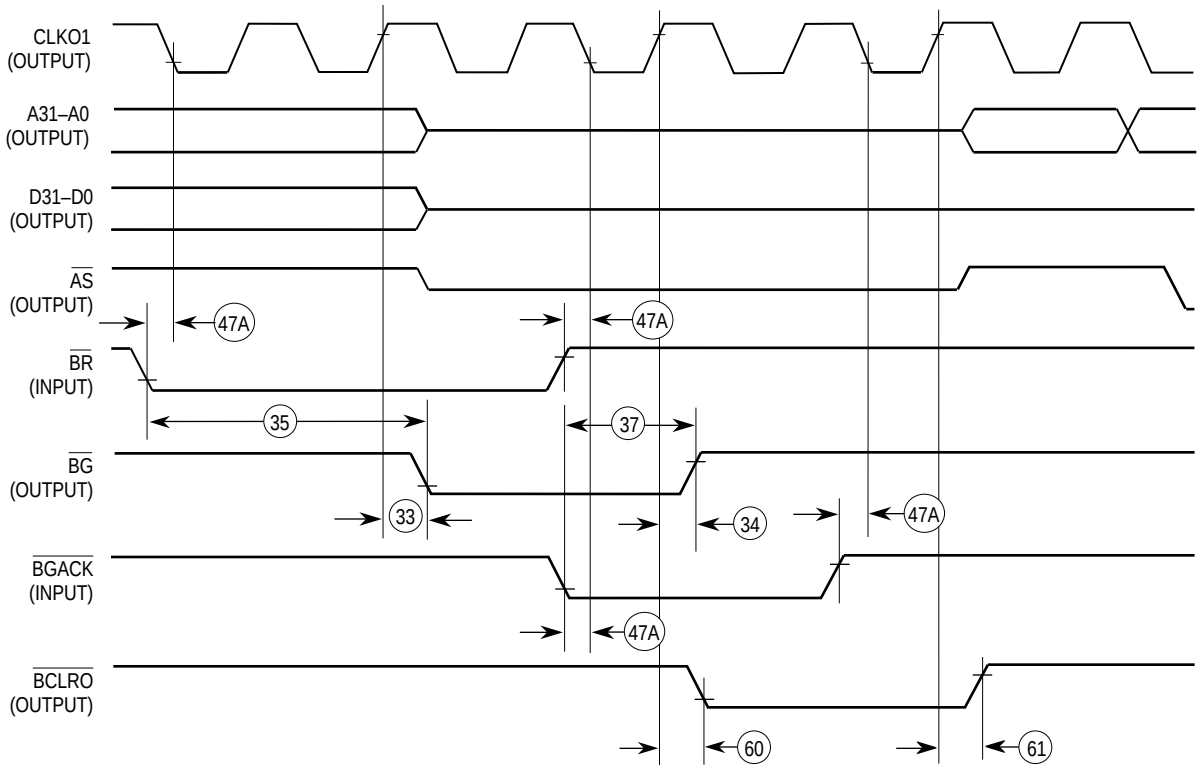


Figure 10-12. ASYNC Bus Arbitration – IDLE Bus Case

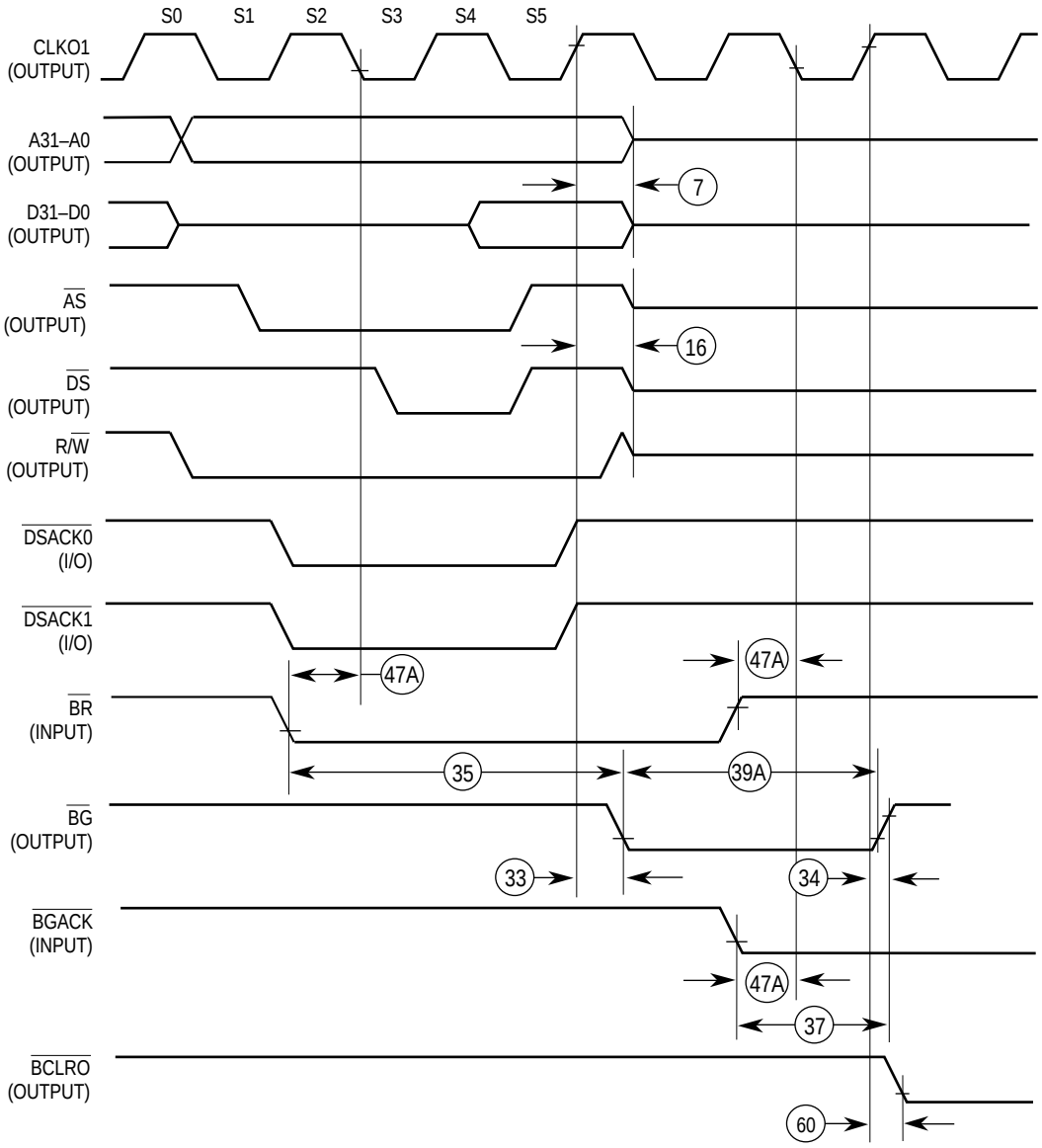


Figure 10-13. ASYNC Bus Arbitration – Active Bus Case

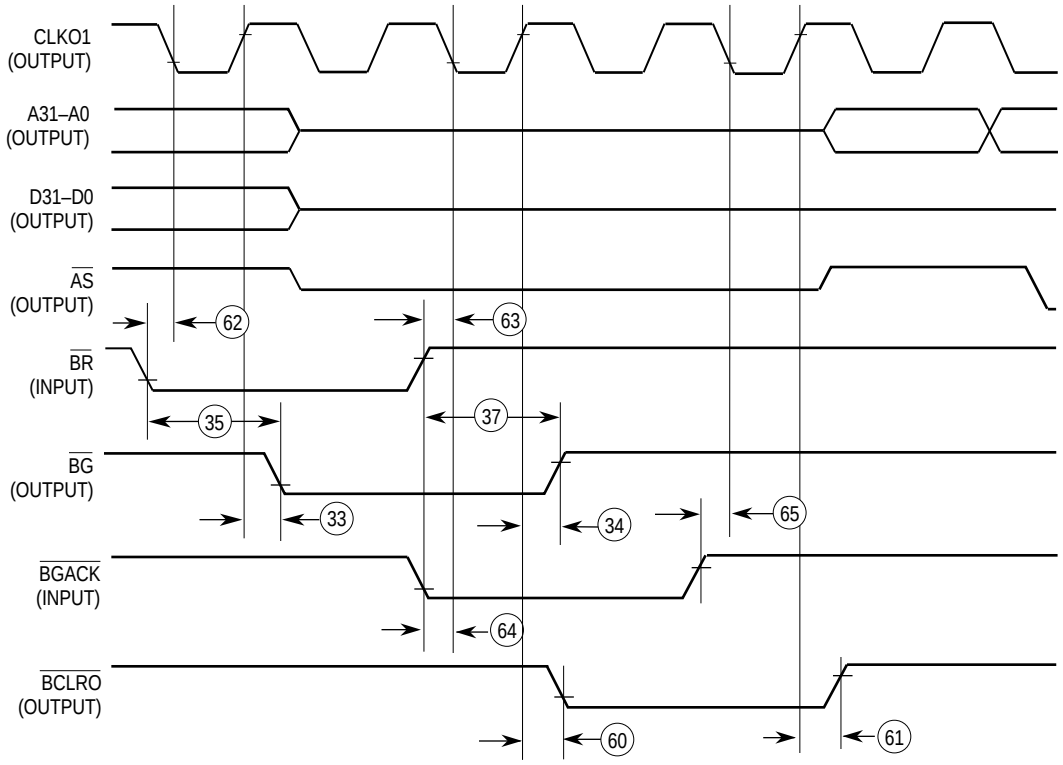


Figure 10-14. SYNC Bus Arbitration – IDLE Bus Case

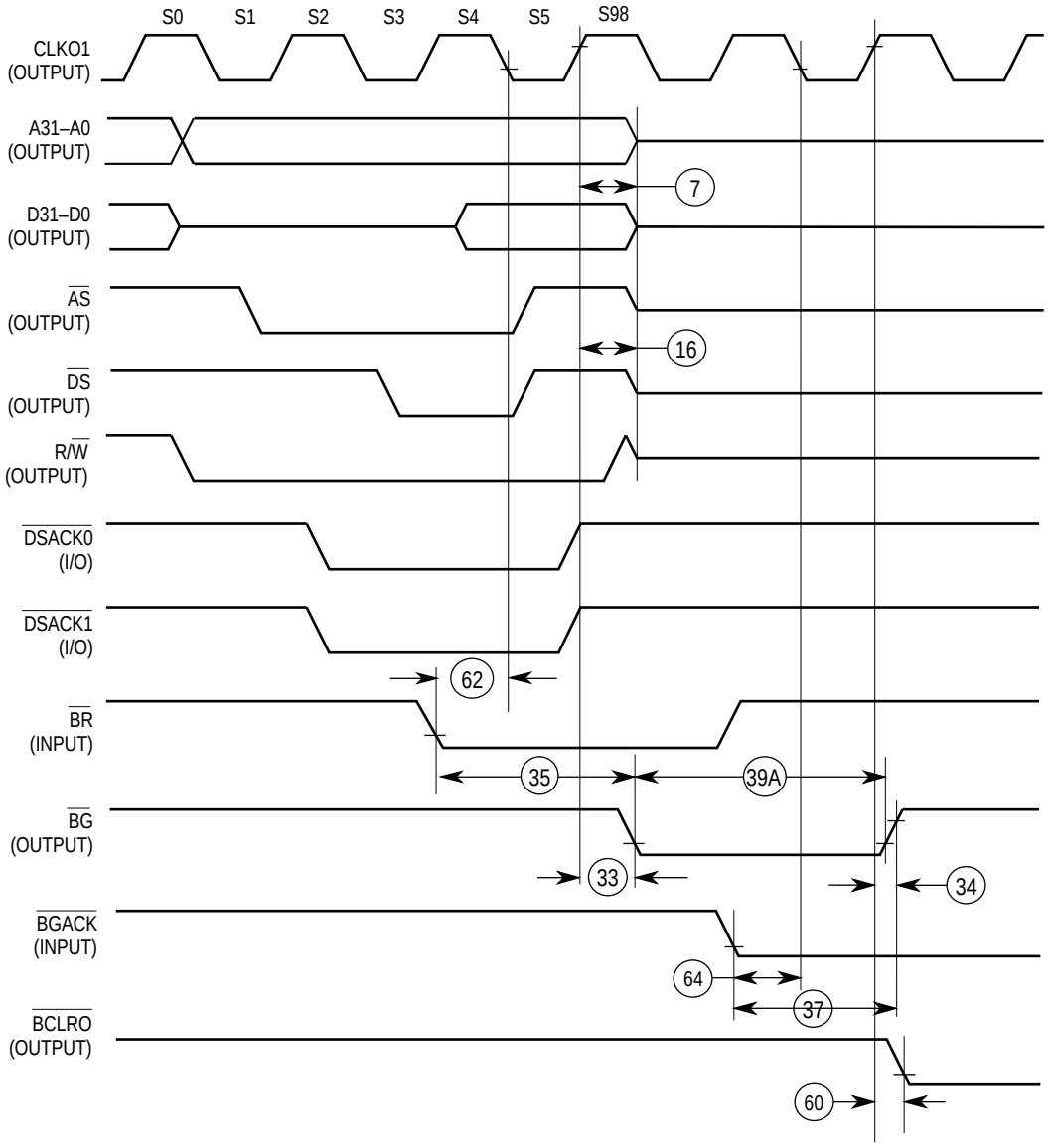


Figure 10-15. SYNC Bus Arbitration – Active Bus Case

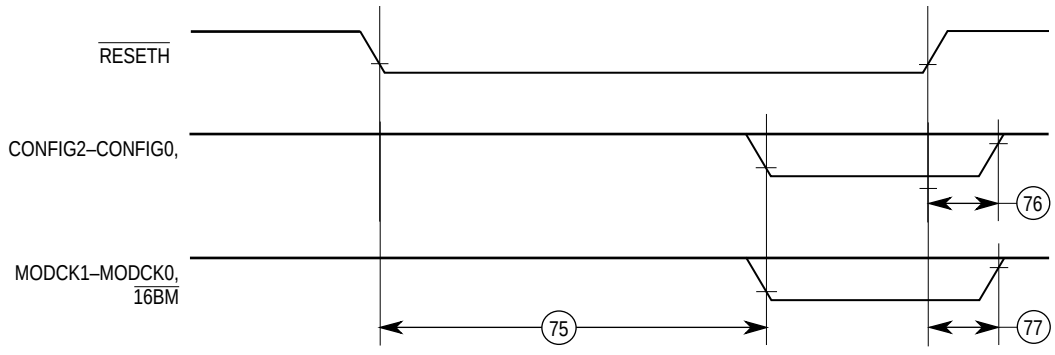


Figure 10-16. Configuration And Clock Mode Select Timing

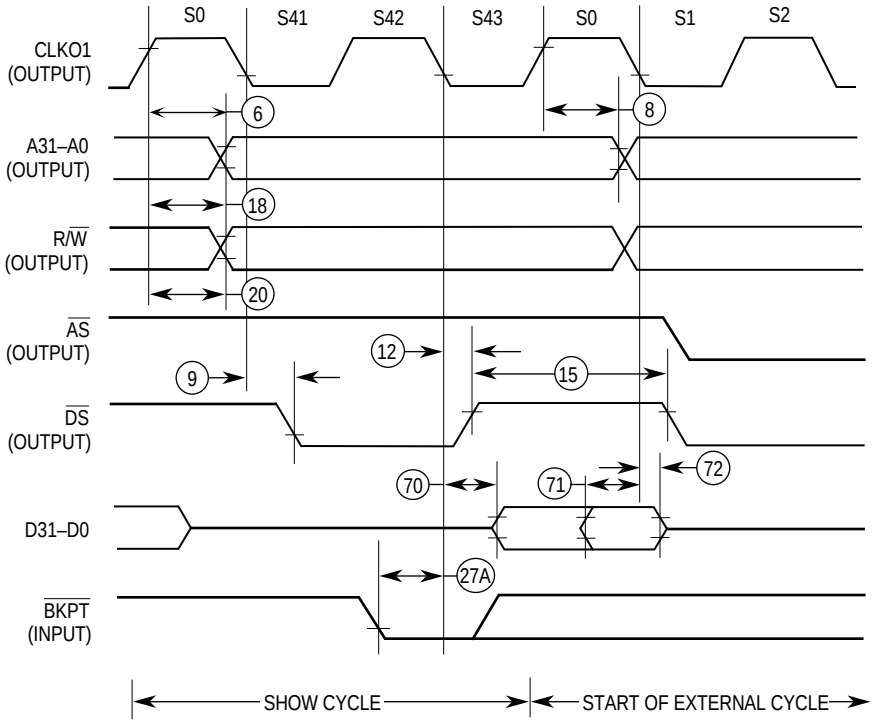


Figure 10-17. Show Cycle

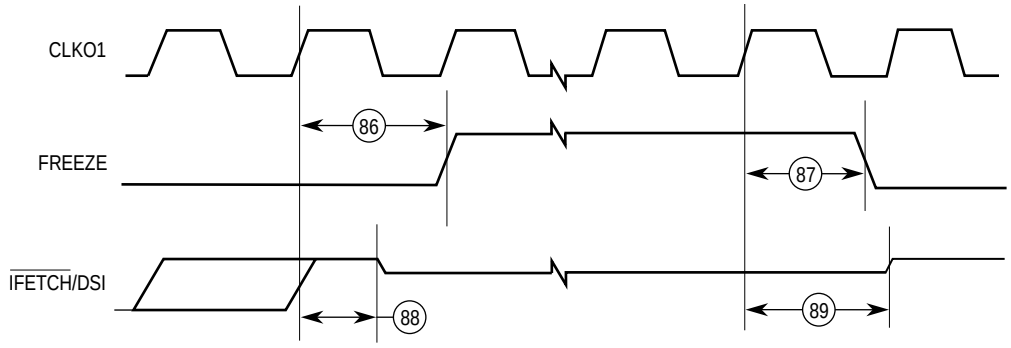


Figure 10-18. Background Debug Mode FREEZE Timing

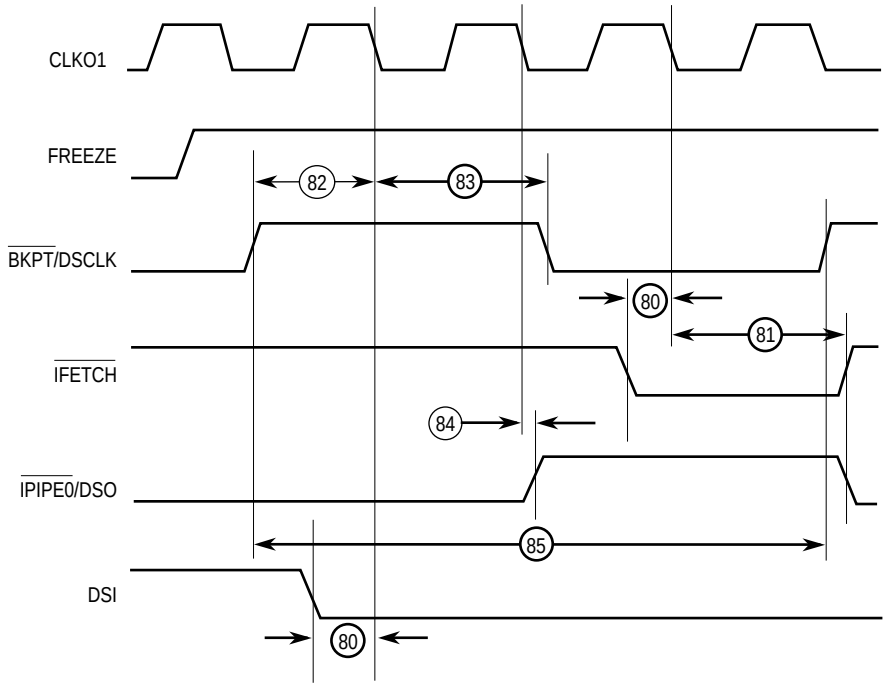


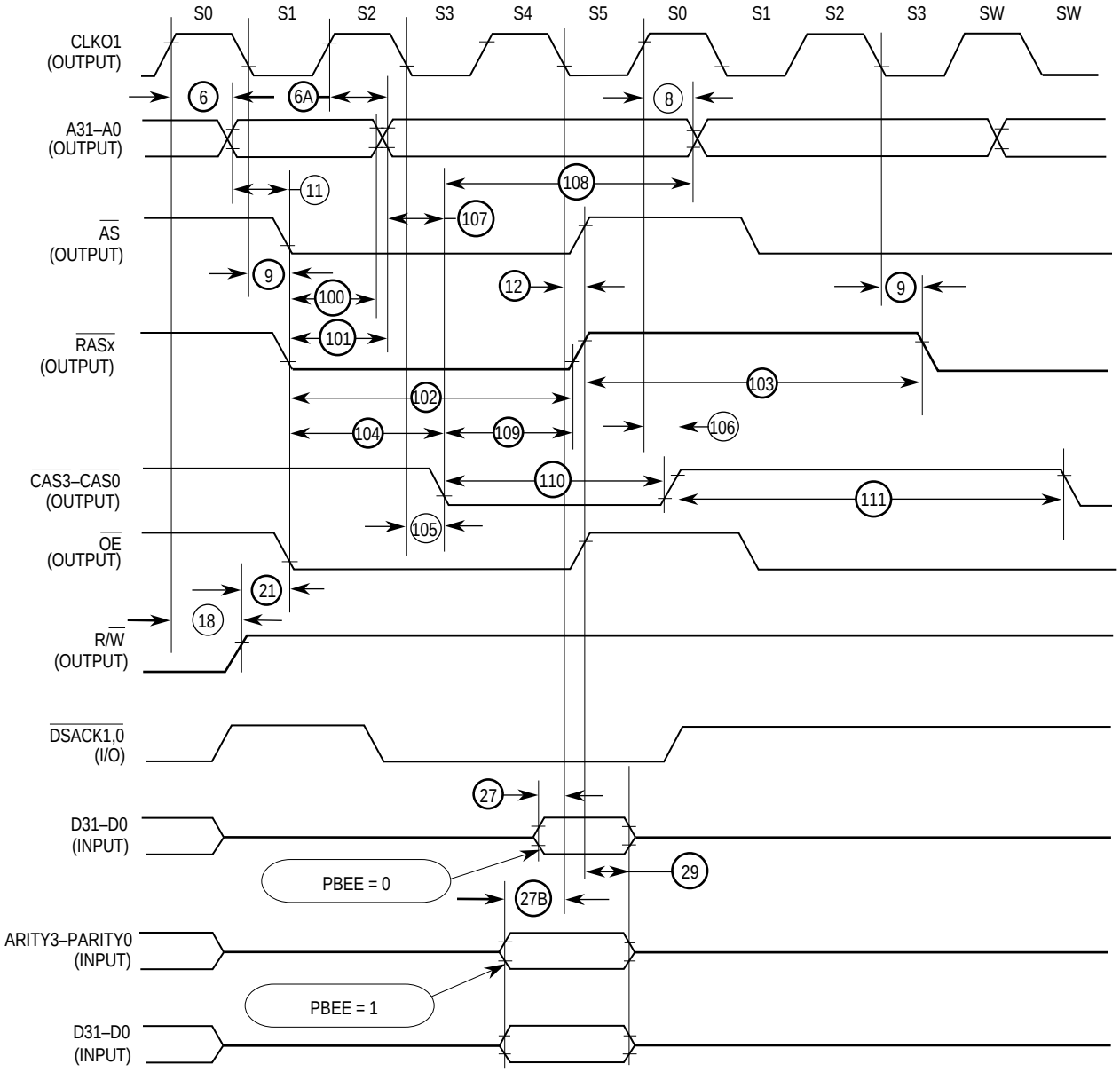
Figure 10-19. Background Debug Mode Serial Port Timing

10.10 BUS OPERATION—DRAM ACCESSES AC TIMING SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-20–Figure 10-24.)

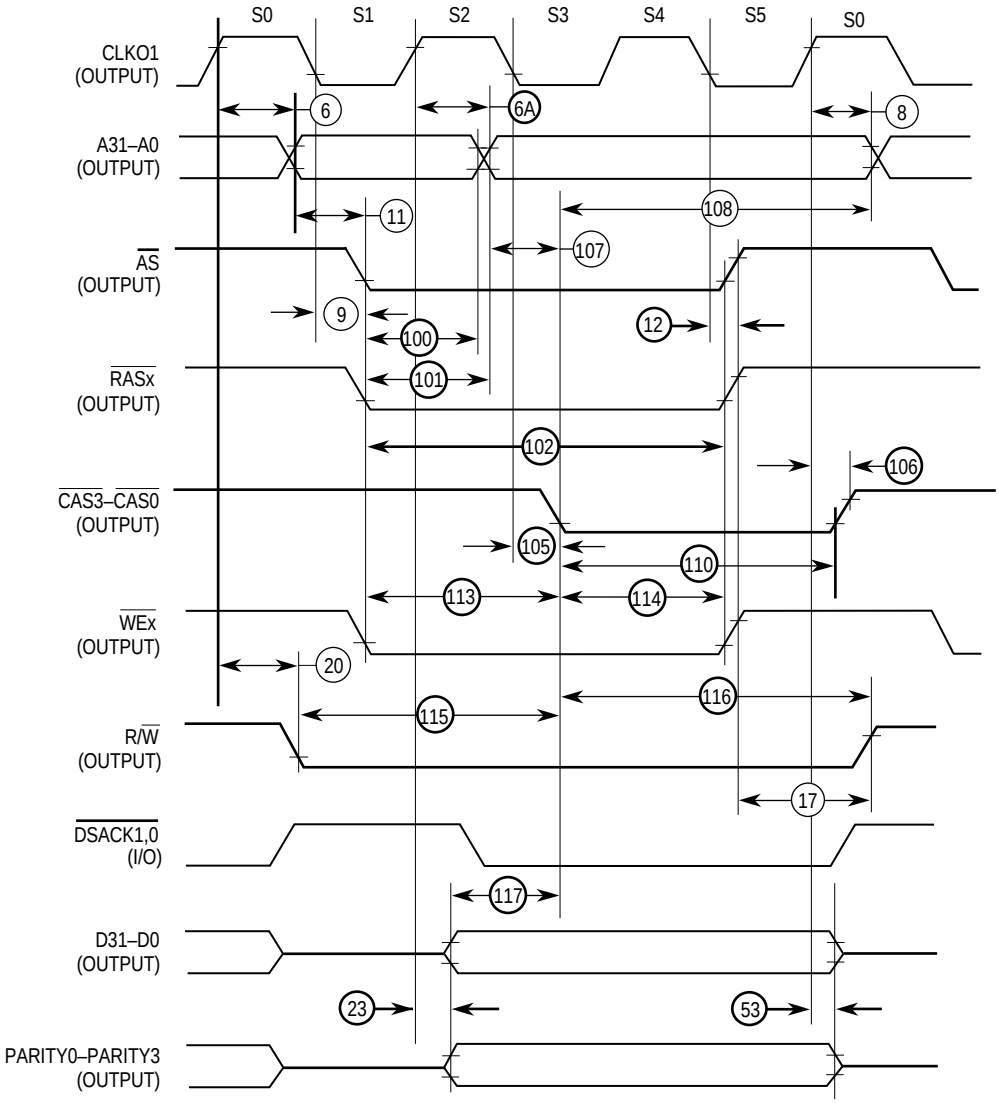
Num.	Characteristic	3.3 V or 5.0 V		5.0V		Unit
		25.0 MHz		33.34MHz		
		Min	Max	Min	Max	
100	RAS Asserted to Row Address Invalid	15		11.25		ns
101	RAS Asserted to Column Address Valid	20		15		ns
102	RAS Width Asserted	75		56.25		ns
103[1]	RAS Width Negated (Back to Back Cycles)	75		56.25		ns
104	RAS Asserted to CAS Asserted	35		26.25		ns
105	CLKO1 Low to CAS Asserted	3	13	2	10	ns
105A	CLKO1 High to CAS Asserted (Refresh Cycle)	3	13	2	10	ns
106	CLKO1 High to CAS Negated	3	13	2	10	ns
107	Column Address Valid to CAS Asserted	15		11.25		ns
108	CAS Asserted to Column Address Negated	40		30		ns
109	CAS Asserted to RAS Negated	35		27		ns
110	CAS Width Asserted	50		37.5		ns
111[1]	CAS Width Negated (Back to Back Cycles)	95		71.25		ns
111A	CAS Width Negated (Page Mode)	20		15		ns
113	WE Low to CAS Asserted	35		27		ns
114	CAS Asserted to WE Negated	35		27		ns
115	R/W Low to CAS Asserted (Write)	75		56.25		ns
116	CAS Asserted to R/W High (Write)	55		41.25		ns
117	Data-Out, Parity-Out Valid to CAS Asserted	10		7.5		ns
119	CLKO1 High to AMUX Negated	3	16	2	12	ns
120	CLKO1 High to AMUX Asserted	3	16	2	12	ns
121	AMUX High to RAS Asserted	15		11.25		ns
122	RAS Asserted to AMUX Low	15		11.25		ns
123	AMUX Low to CAS Asserted	15		11.25		ns
124	CAS Asserted to AMUX High	55		41.25		ns
125	RAS/CAS Negated to R/W change	0		0		ns

NOTES:
1.This specification is for WBTQ=0, when WBTQ=1 add another clock.



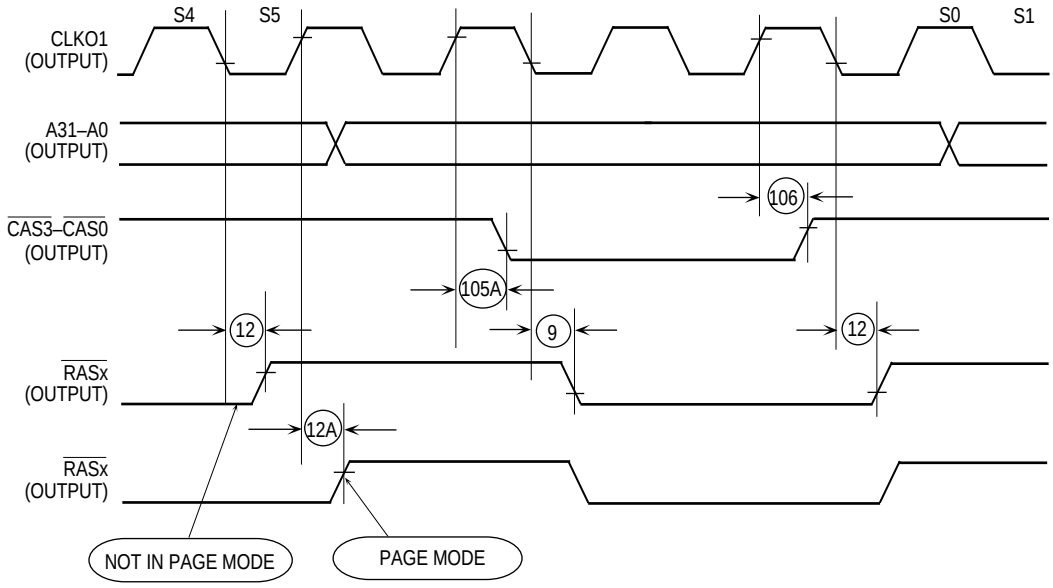
NOTE: All timing is shown with respect 0.8-V AND 2.0-V levels.

Figure 10-20. DRAM: Normal Read Cycle (Internal Mux, TRLX = 0)



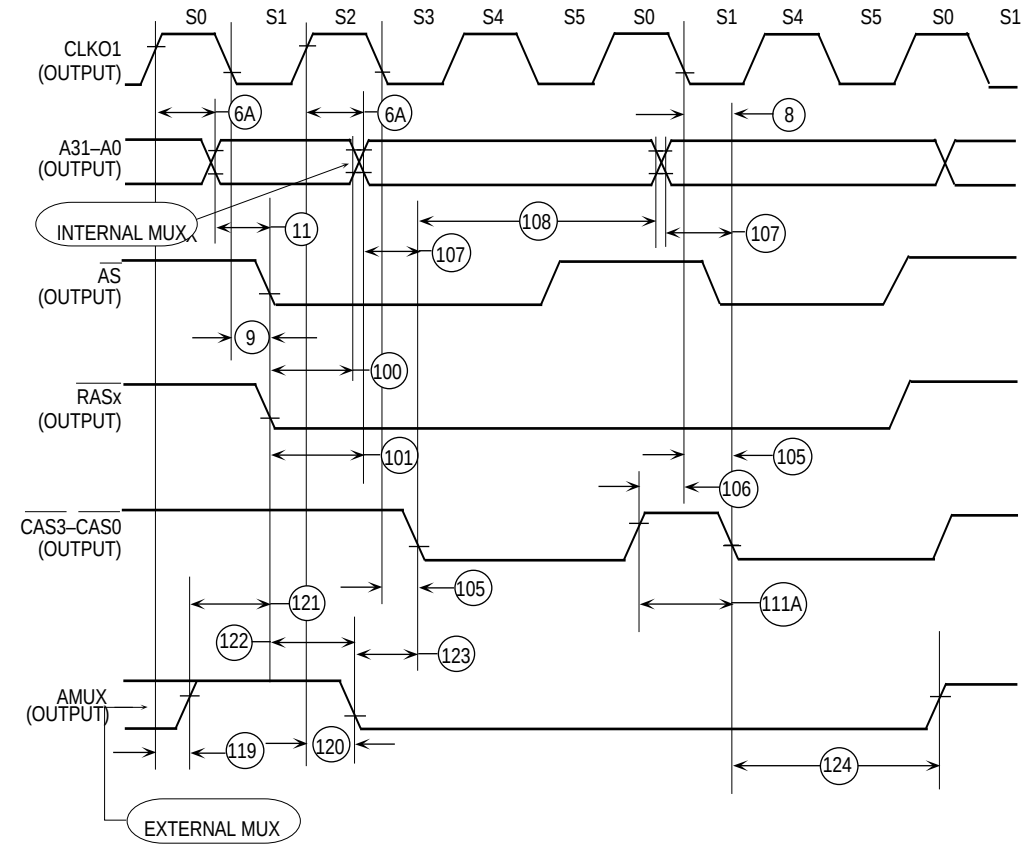
NOTE: All timing is shown with respect to 0.8-V and 2.0-V levels.

Figure 10-21. DRAM: Normal Write Cycle



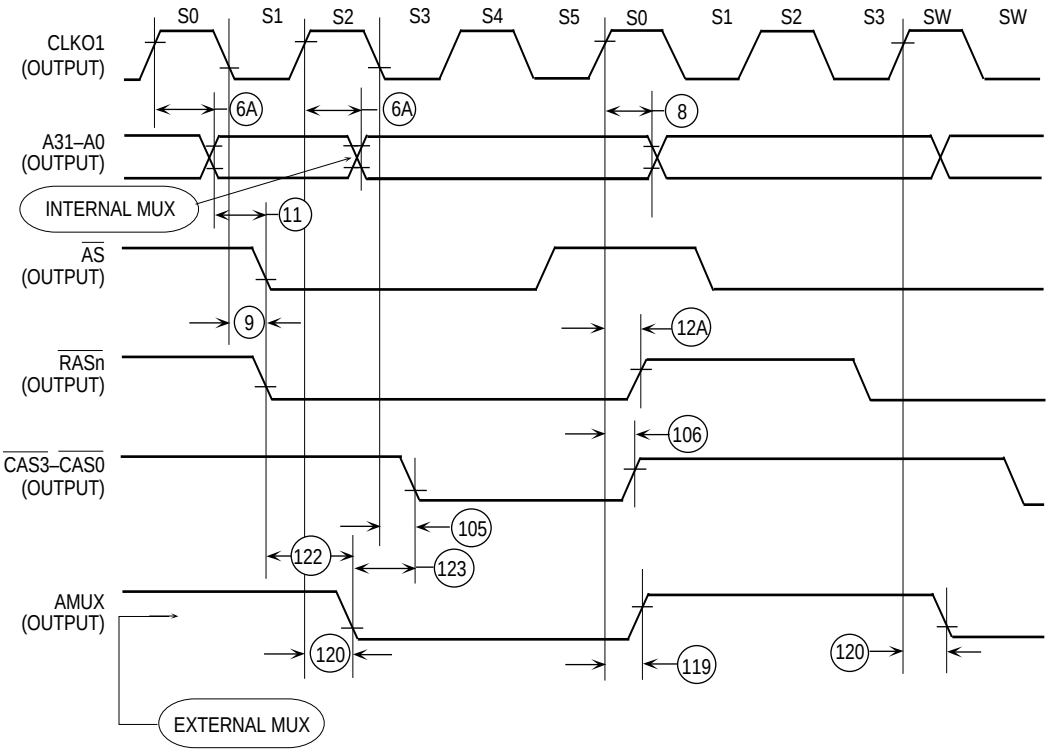
NOTE: All timing is shown with respect to 0.8-V and 2.0-V levels.

Figure 10-22. DRAM: Refresh Cycle



NOTE: All timing is shown with respect to 0.8-V and 2.0-V levels.

Figure 10-23. DRAM: Page-Mode—Page-Hit



NOTE: All timing is shown with respect to 0.8-V and 2.0-V levels.

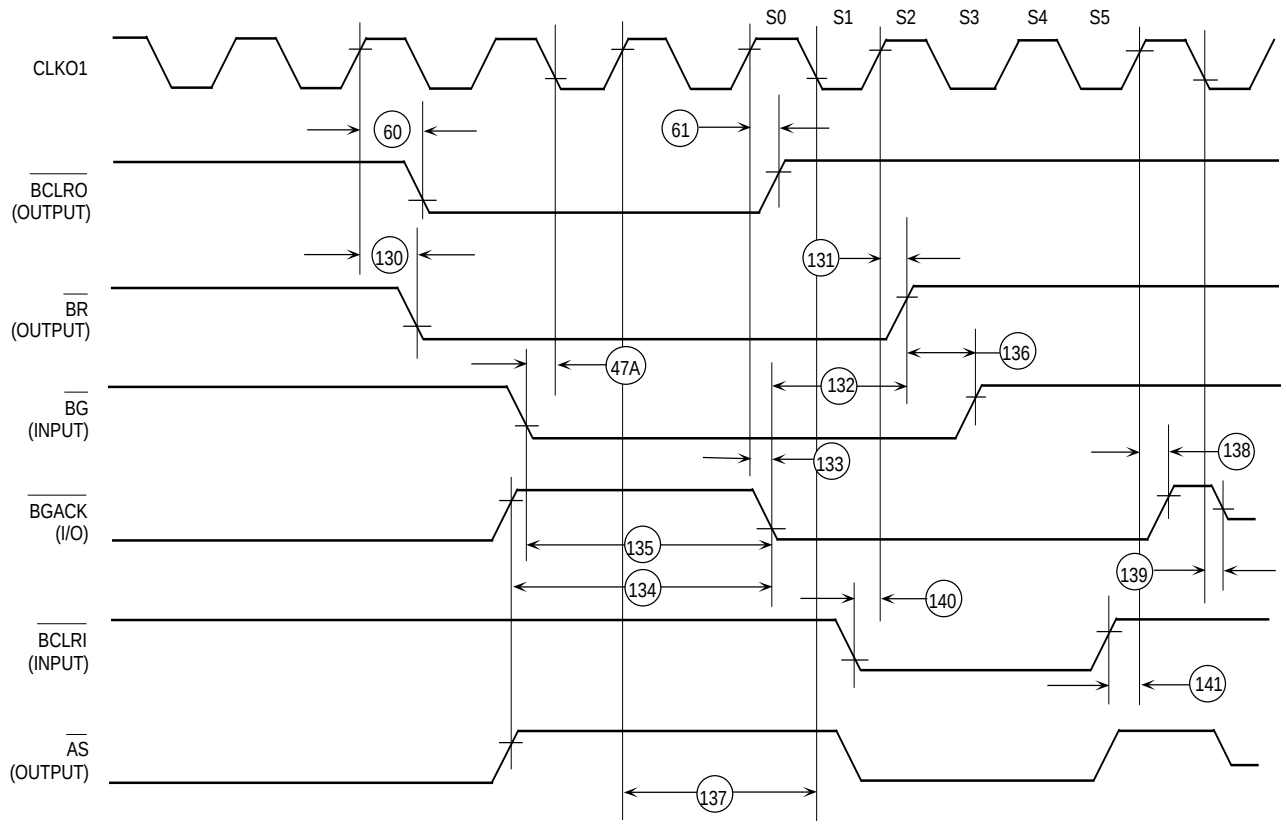
Figure 10-24. DRAM: Page-Mode—Page-Miss

10.11 030/QUICC BUS TYPE SLAVE MODE BUS ARBITRATION AC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-25 and Figure 10-26)

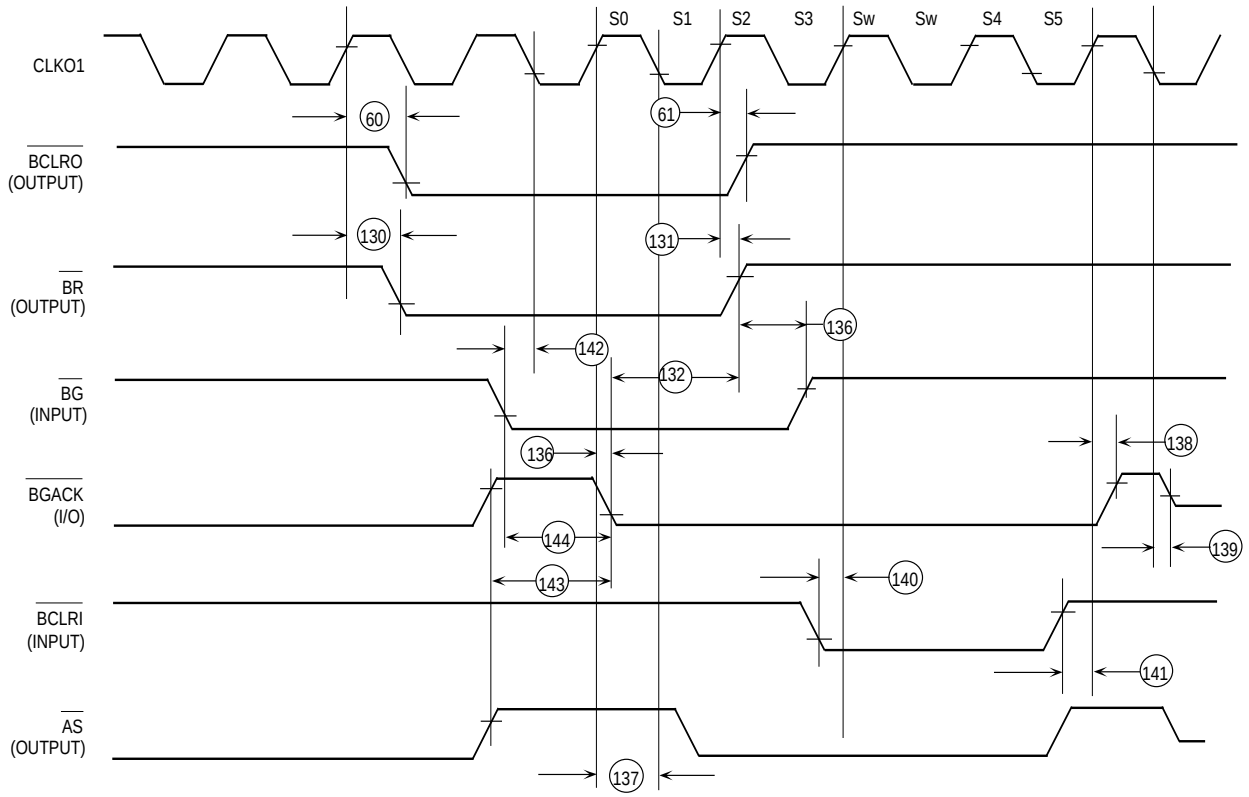
Num.	Characteristic	3.3 V or 5.0 V		5.0V		Unit
		25.0 MHz		33.34MHz		
		Min	Max	Min	Max	
130[1]	CLKO1 High to BR Asserted	—	20	—	15	ns
131	CLKO1 High to BR High Impedence	—	20	—	15	ns
132[1]	BGACK Low to BR High Impedence	20	—	15	—	ns
133	CLKO1 High to BGACK Asserted	—	20	—	15	ns
134[2]	AS and BGACK High (the latest one) to BGACK Low (when BG is asserted)	1.5	2.5 + 20	1.5	2.5 +15	clks ns
135[1,2]	BG Low to BGACK Low (no other Bus Master)	1.5	2.5 + 20	1.5	2.5 +15	clks ns
137	Clock on which BGACK Low to Clock on which AS Low.	0	—	0	—	clks
138	CLKO1 High to BGACK High	—	20	—	15	ns
139	CLKO1 Low to BGACK High Impedance	—	15	—	11.25	ns
140	BCLRI Low to CLKO1 High	15	—	11.25	—	ns
141	BCLRI High to CLKO1 High	15	—	11.25	—	ns
142[3]	BG Low to CLKO1 Low	15	—	11.25	—	ns
143[3]	AS and BGACK High (the latest one) to BGACK Low (when BG is asserted)	1	1 + 20	1	1 +15	clks ns
144[1,3]	BG Low to BGACK Low (no other bus master)	1	1 + 20	1	1 +15	clks ns

NOTES:
1. Specifications are for asynchronous arbitration (ASTM=0).
2. Specifications are for synchronous arbitration (ASTM=1).



NOTE: Diagram does not apply to MC68040 companion mode.

Figure 10-25. MC68360 Slave Mode Asynchronous Arbitration



NOTE: Diagram does not apply to MC68040 companion mode.

Figure 10-26. MC68360 Slave Mode Synchronous Arbitration

10.12 030/QUICC BUS TYPE SLAVE MODE INTERNAL READ/WRITE/ IACK ASYNCHRONOUS CYCLES AC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-27 and Figure 10-28)

Num.	Characteristic	3.3 V or 5.0 V		5.0V		Unit
		25.0 MHz		33.34MHz		
		Min	Max	Min	Max	
150	Address Valid to AS Low	10	—	8	—	ns
151	AS Valid to DS Low	0	—	0	—	ns
152	AS inactive Time	1	—	1	—	clk
153	AS High to Address Hold Time	0	—	0	—	ns
154	DS Inactive to AS Inactive	0	—	0	—	ns
155	DS Inactive Time	1	—	1	—	clk
156	DSACK Low to AS, DS High	0	—	0	—	ns
157	Data Out Valid to DSACK Low	—	15	—	11.25	ns
158	CLKO1 Low to Data-Out Valid	—	20	—	15	ns
159	DS High to Data-Out Hold Time	0	—	0	—	ns
160	DS High to Data High Impedance	—	40	—	30	ns
161	CLKO1 High to $\overline{\text{DSACK}}$ Low (Note 2)	—	20	—	15	ns
161A	CLKO1 High to $\overline{\text{DSACK}}$ Low (Note 3)	—	25			ns
162	AS High to DSACK High	—	20	—	15	ns
163	DSACK High to DSACK High Impedance	—	15	—	11.25	ns
164	R/W Valid to DS Low	0	—	0	—	ns
165	DS High To R/W High	0	—	0	—	ns
166	Data-In, MBARE Valid to DS Low	—	20	—	15	ns
167	DSACK Low to Data-In, MBARE Hold Time	0	—	0	—	ns
169	CLKO1 Low to AVECO Low	—	20	—	15	ns
170	AS High to AVECO High Impedance	—	30	—	23	ns
171	CLKO1 Low to IACK Low	—	20	—	15	ns
172	AS High to IACK High	—	30	—	23	ns

NOTES:
1Asynchronous specifications above are valid only when BSTM=0.
2. For external bus master to external memory or peripheral.
3. For external bus master to internal memory or registers.

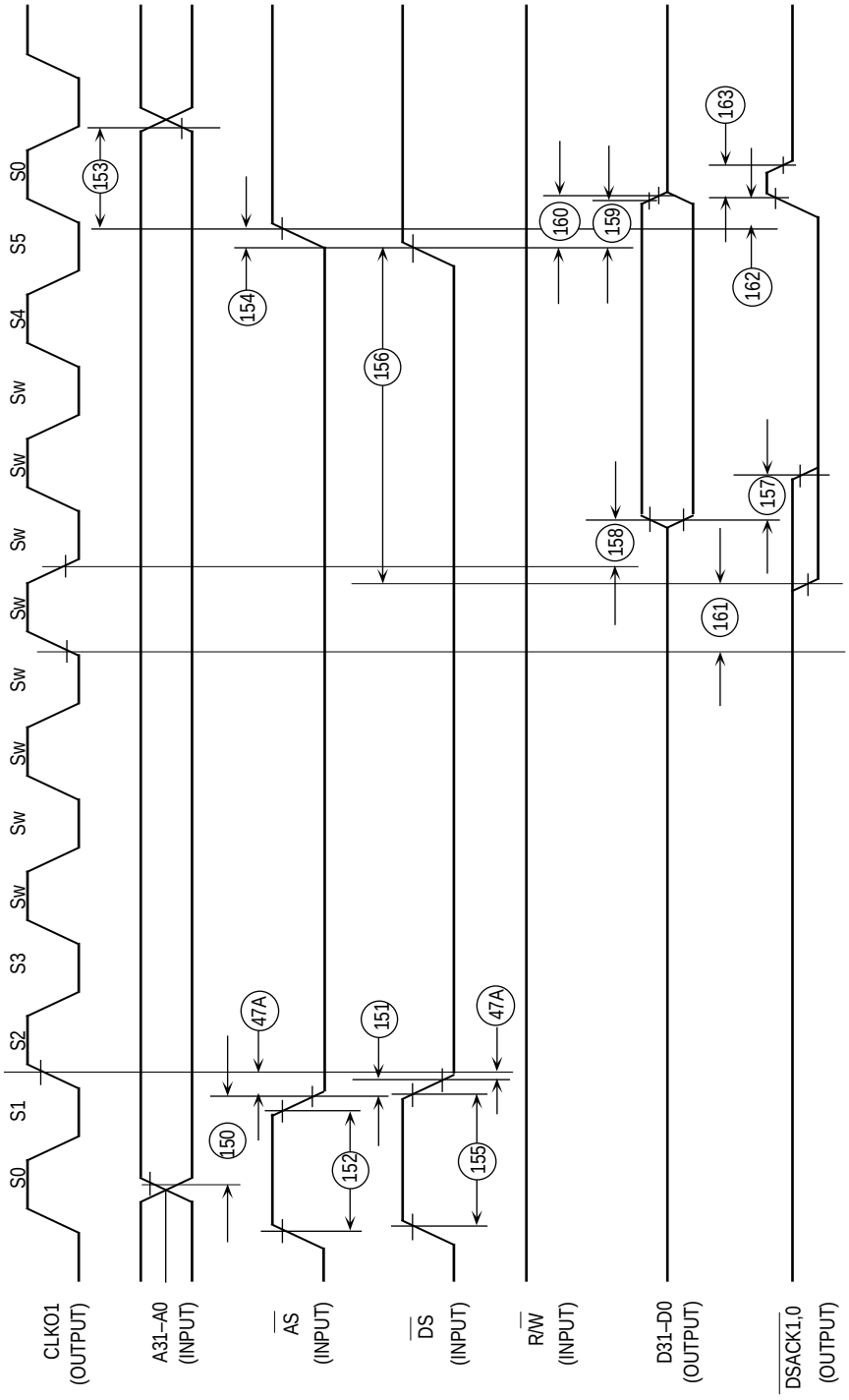


Figure 10-27. External MC68030/MC68360 Internal Asynchronous Read Cycle Timing Diagram

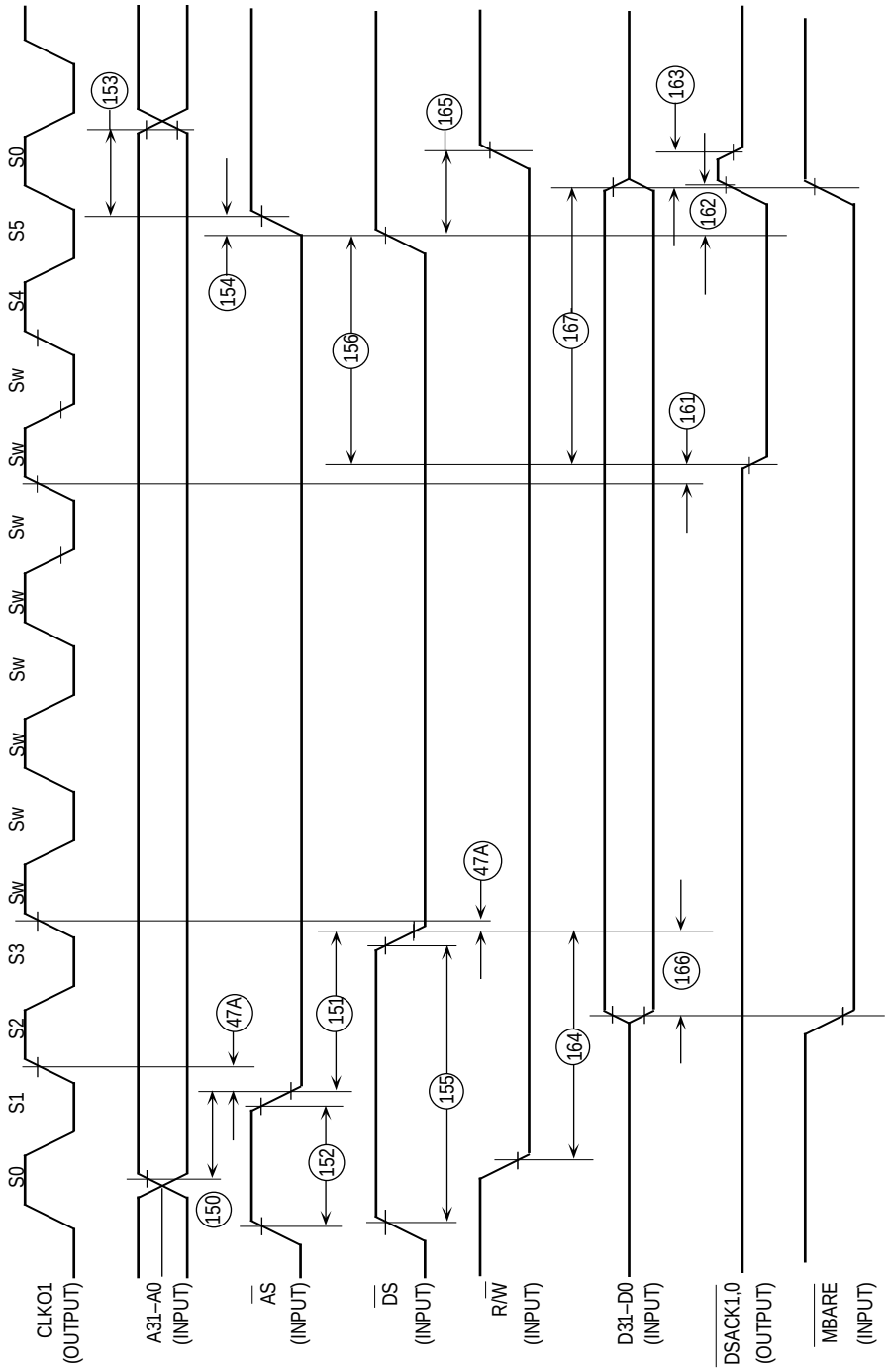


Figure 10-28. External MC68030/MC68360 Internal Registers Asynchronous Write Cycle Timing Diagram

10.13 030/QUICC BUS TYPE SLAVE MODE INTERNAL READ/WRITE/ IACK SYNCHRONOUS CYCLES AC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figures 10-29–10-32.)

Num.	Characteristic	3.3 V or 5.0 V		5.0V		Unit
		25.0 MHz		33.34MHz		
		Min	Max	Min	Max	
180	AS Low to CLKO1 High	7	—	6	—	ns
181	CLKO1 Low to AS High	—	20	—	15	ns
182	DS Low to CLKO1 High	7	—	6	—	ns
183	CLKO1 Low to DS High	—	20	—	15	ns
184	CLKO1 Low to Data-Out Valid	—	20	—	15	ns
185	R/W Low to CLKO1 High	7	—	6	—	ns
186	CLKO1 High to R/W High	—	20	—	15	ns
187	Data-In, MBARE Valid to $\overline{DS}$ Low	2	—	2	—	ns
188	CLKO1 Low to Data-In, MBARE Hold Time	10	—	7.5	—	ns
191	$\overline{AS}$ Low to $\overline{AVECO}$ Low	—	30	—	23	ns
192	$\overline{AS}$ Low to $\overline{IACK}$ Low	—	30	—	23	ns

NOTES:
1Synchronous specifications above are valid only when BSTM=1.

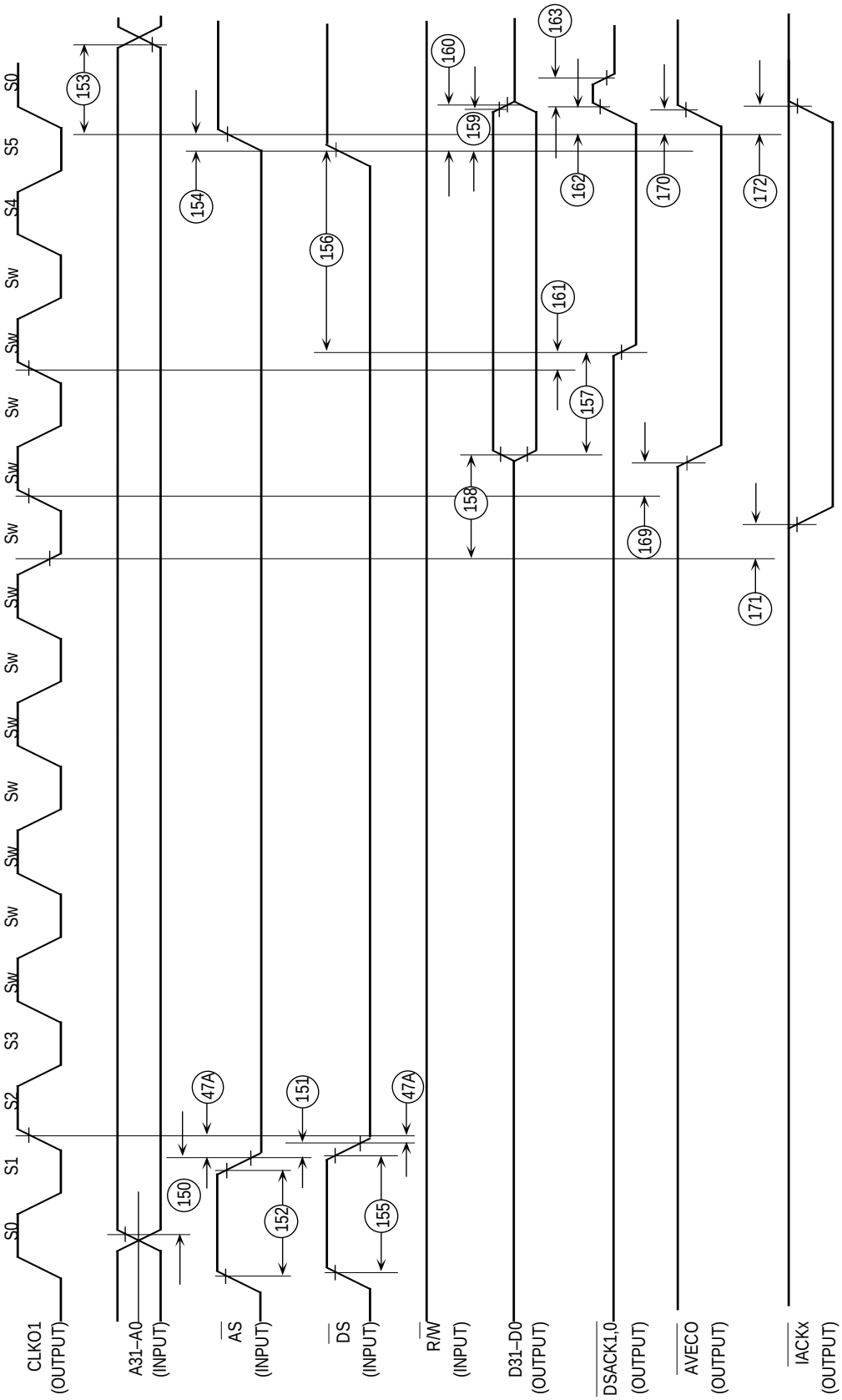
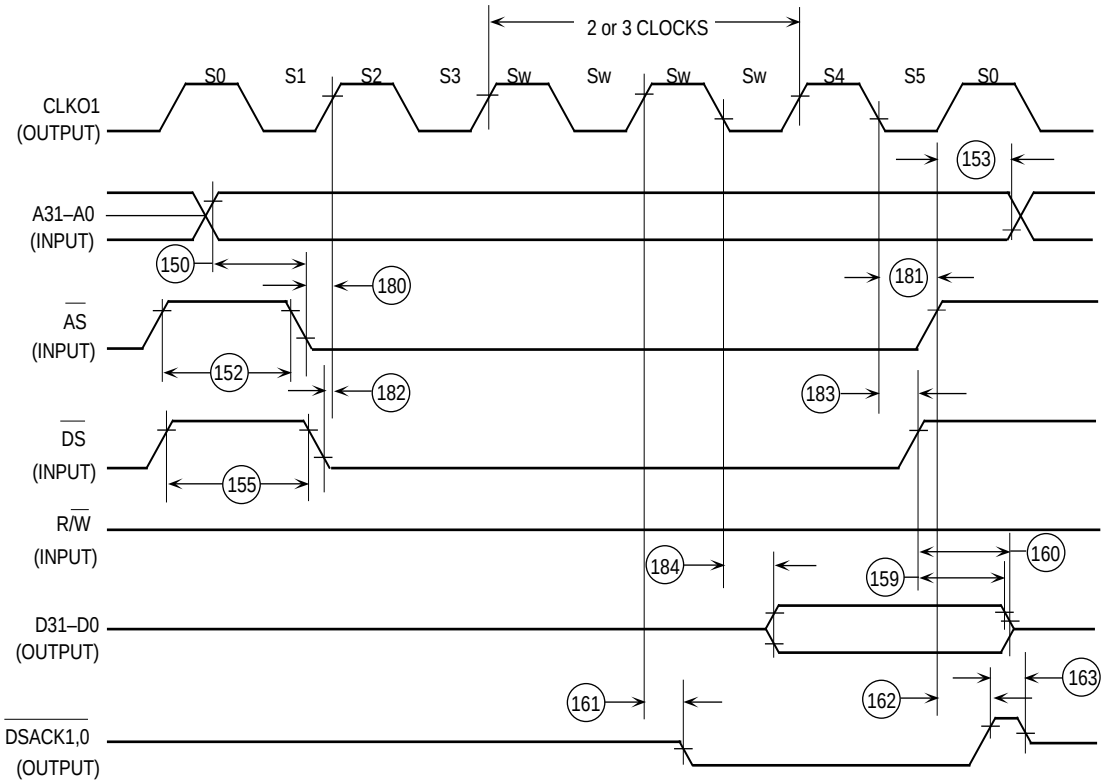
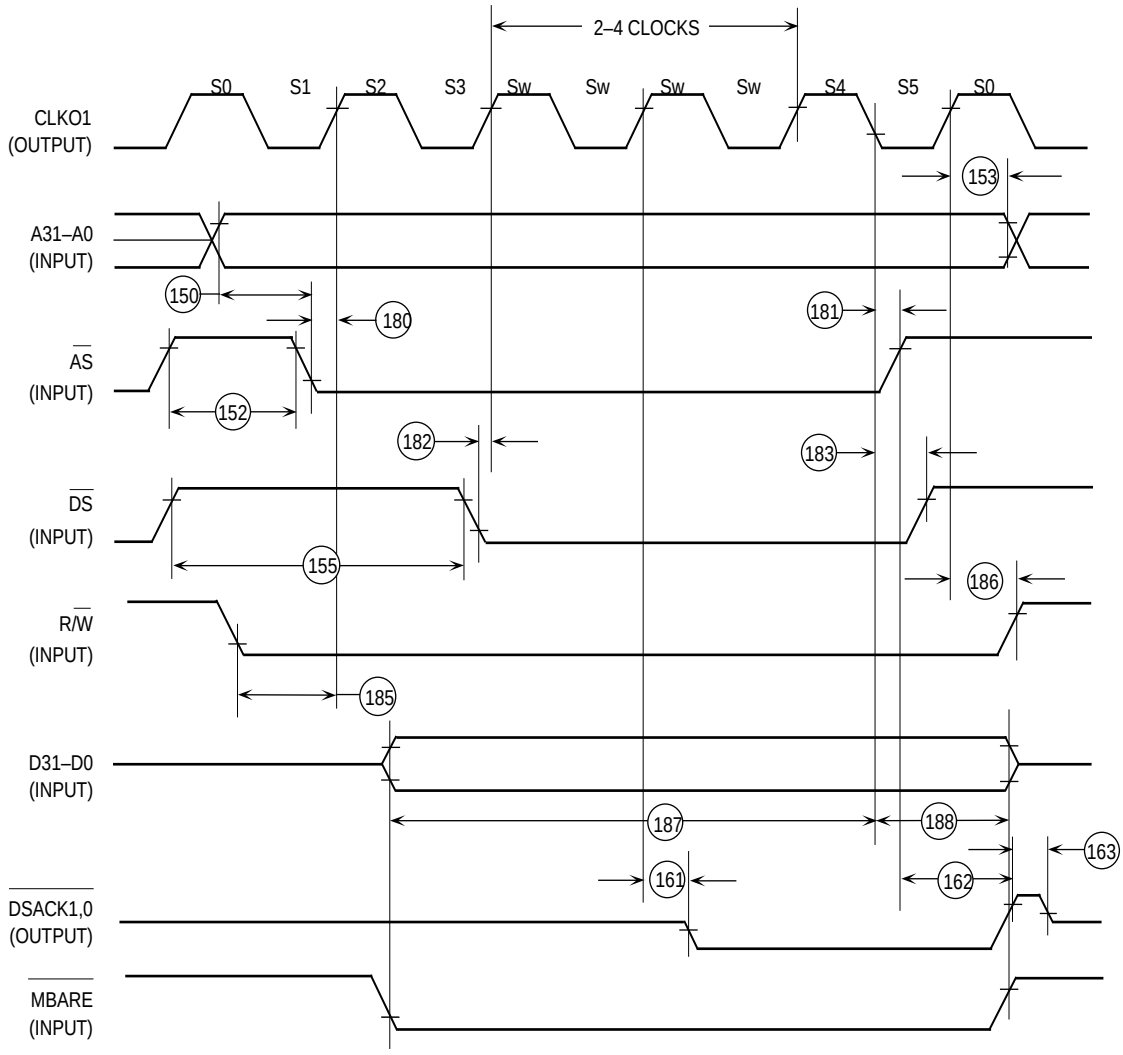


Figure 10-29. External MC68030/MC68360 Asynchronous IACK Cycle Timing Diagram



NOTE: Two wait states are inserted when reading the SIM, dual-port RAM, and CPM. Three wait states are inserted when reading the SI RAM. Additional wait states may be inserted when the SHEN1–SHEN0 =10 and one of the internal masters is accessing an internal peripheral.

Figure 10-30. External MC68030/MC68360 Internal Synchronous Read Cycle Timing Diagram



NOTE: Two wait states are inserted when writing to the SIM. Three wait states are inserted when writing to the dual-port RAM and CPM. Four wait states are inserted when writing the SI RAM. Additional wait states may be inserted when the SHEN1–SHEN0 = 10 and one of the internal masters is accessing an internal peripheral.

Figure 10-31. External MC68030/MC68360 Internal Synchronous Write Cycle Timing Diagram

NOTE

This figure represents an IACK cycle for both vectored and autovectored interrupt. If IACK is for a vectored interrupt, AVECO will not be driven. If IACK is an autovector interrupt, DSACK0-1 and data bus will not be driven.

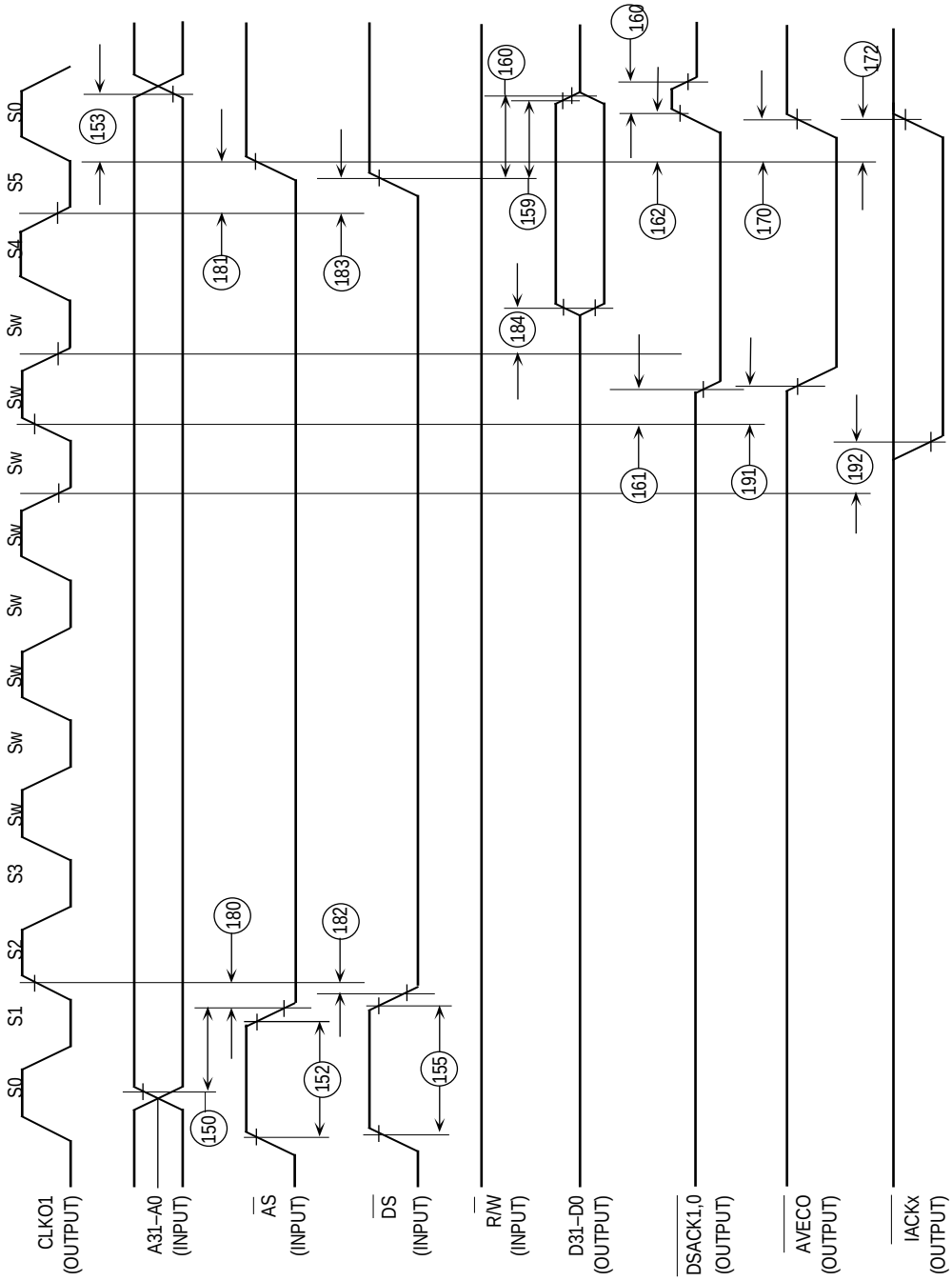


Figure 10-32. External MC68030/MC68360 Synchronous IACK Cycle Timing Diagram

10.14 030/QUICC BUS TYPE SRAM/DRAM CYCLES AC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-33–Figure 10-37)

Num.	Characteristic	3.3 V or 5.0 V		5.0V		Unit
		25.0 MHz		33.34MHz		
		Min	Max	Min	Max	
200	AS Low to OE Low (Read Cycle)	—	20	—	15	ns
201	AS High to OE High	—	20	—	15	ns
202	AS Low to WE Low (Write Cycle)	—	20	—	15	ns
203	AS High to WE High	—	20	—	15	ns
204	CLKO1 High to DSACK High	—	20	—	15	ns
205	AS Low to $\overline{CS}$ Low	—	22	—		ns
206	AS High to $\overline{CS}$ High	—	20	—	15	ns
207	AS Low to DSACK Low	—	27	—	22	ns
208	AS High to $\overline{CAS}$ High	—	20	—	15	ns
210	AS Low to BKPTO Low	—	25	—	23	ns
211	AS High to BKPTO High	—	30	—	25	ns
212	Data Valid to PRTY3–0 Valid	—	20	—	15	ns
213	Data Invalid to PRTY3–0 Invalid	5	—	3.75	—	ns
214	R/W valid to AS Low	0	—	0	—	ns
215	AS High to R/W Invalid	0	—	0	—	ns
216	AS Assert to Parity Driven	4	—	3	—	ns
217	AS Negate to Parity Invalid	4	20	—	15	ns

NOTES:
1Synchronous specifications above are valid only when BSTM=1.

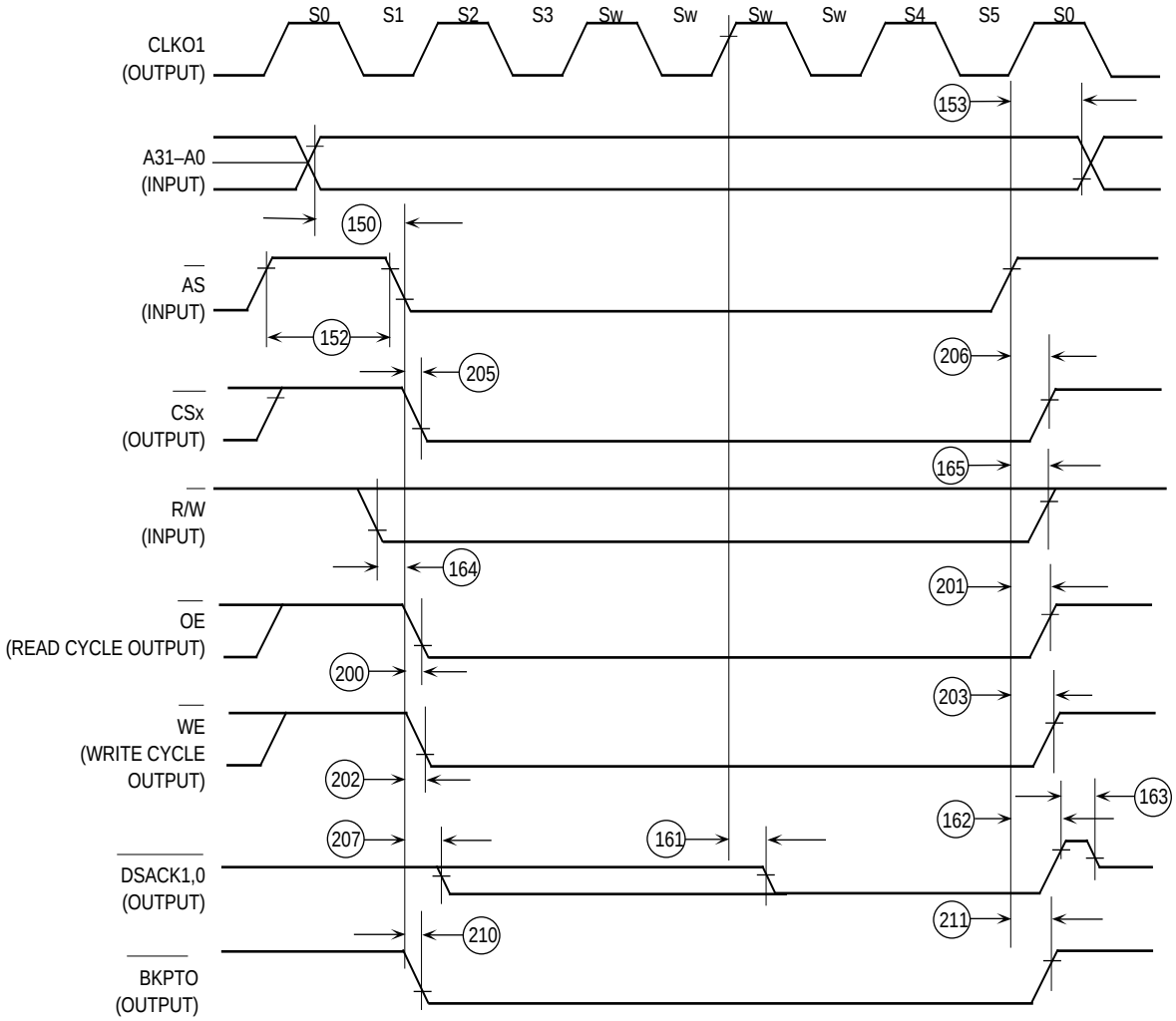


Figure 10-33. External MC68030/MC68360 SRAM Asynchronous Cycle Timing Diagram (BSTN = 0/1; SYNC = 0)

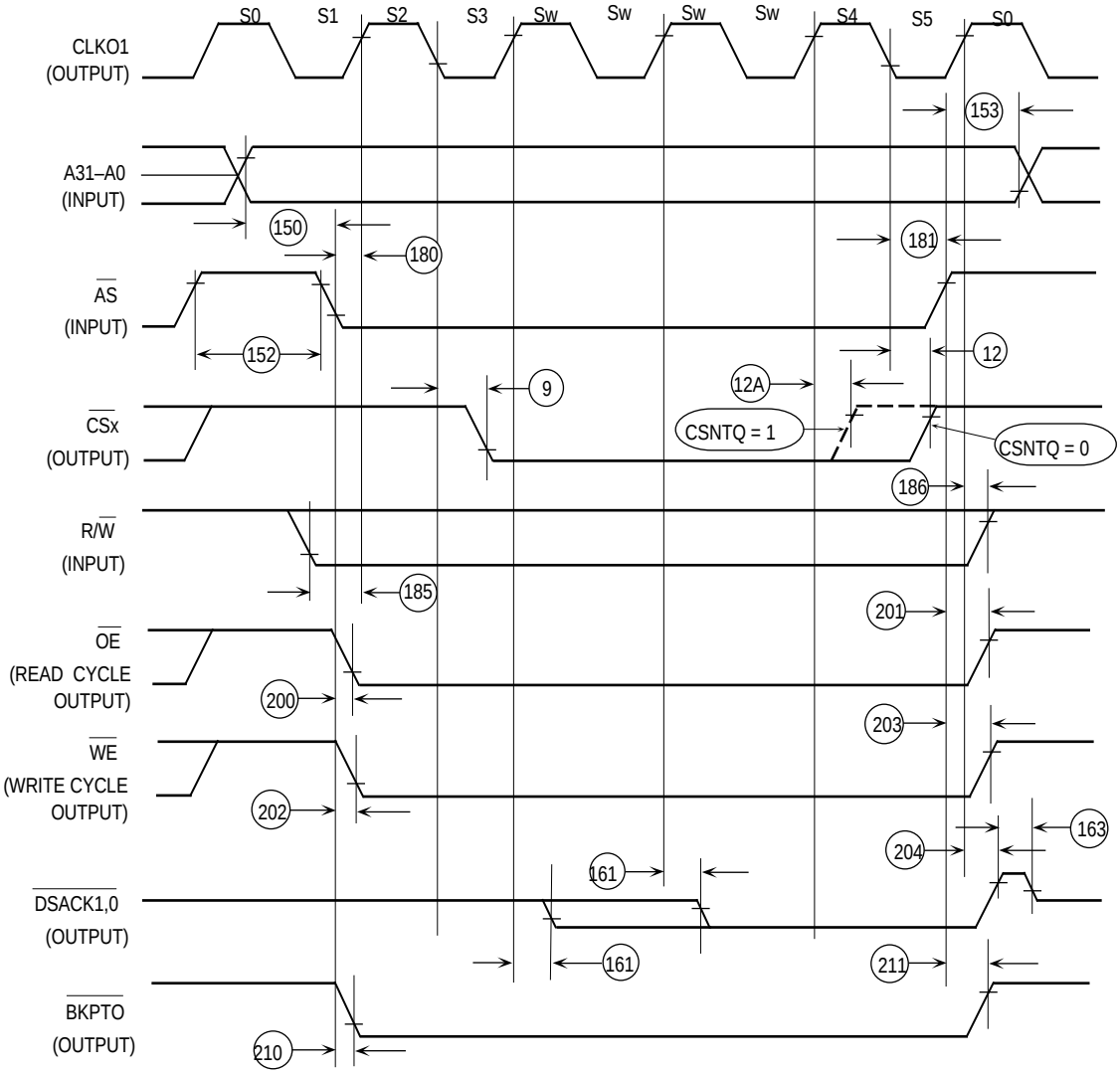


Figure 10-34. External MC68030/MC68360 SRAM Synchronous Cycle Timing Diagram (BSTM = 1; SYNC = 1)

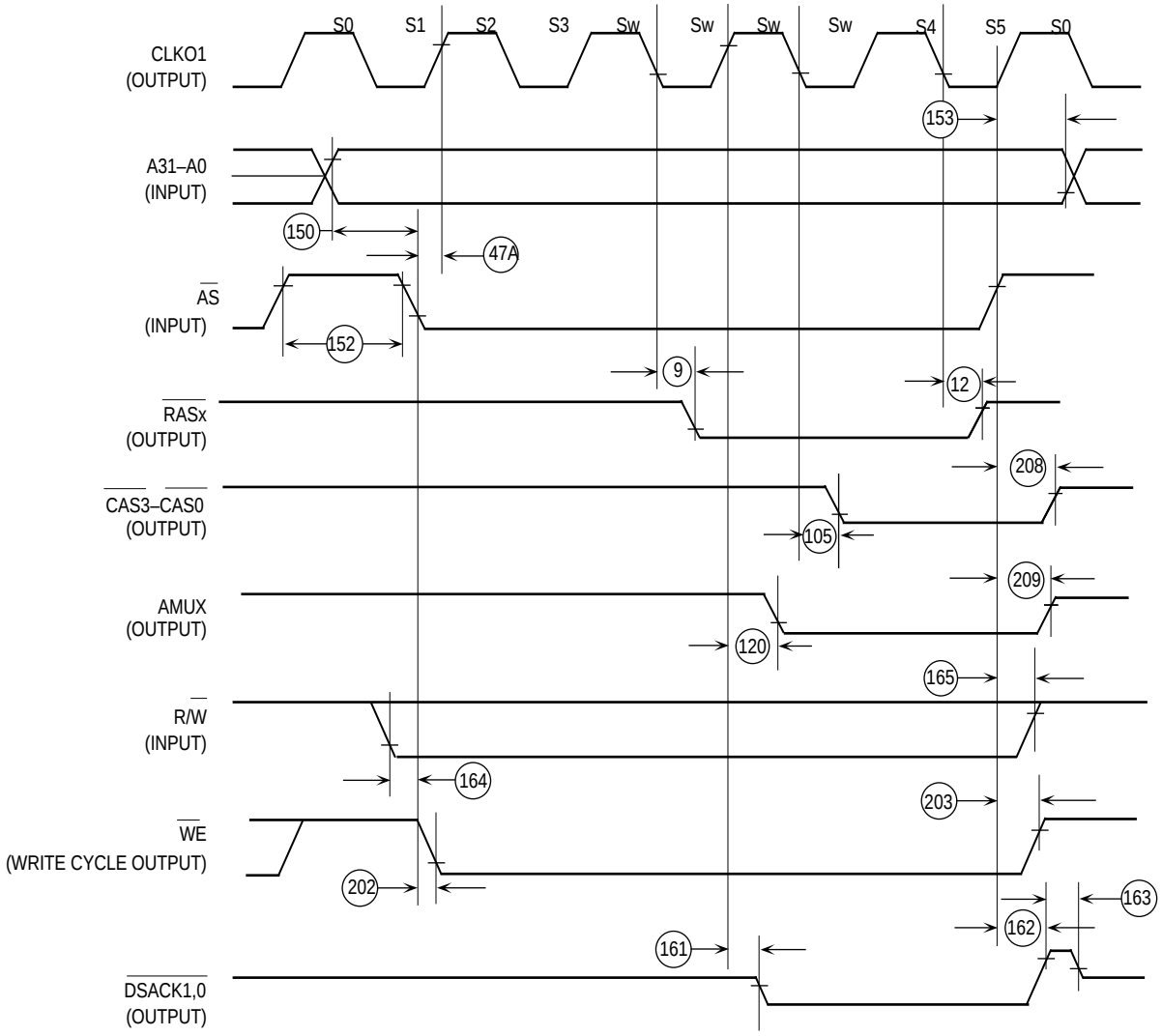


Figure 10-35. External MC68030/MC68360 DRAM Asynchronous Cycle Timing Diagram (BSTM = 0; SYNC = 0)

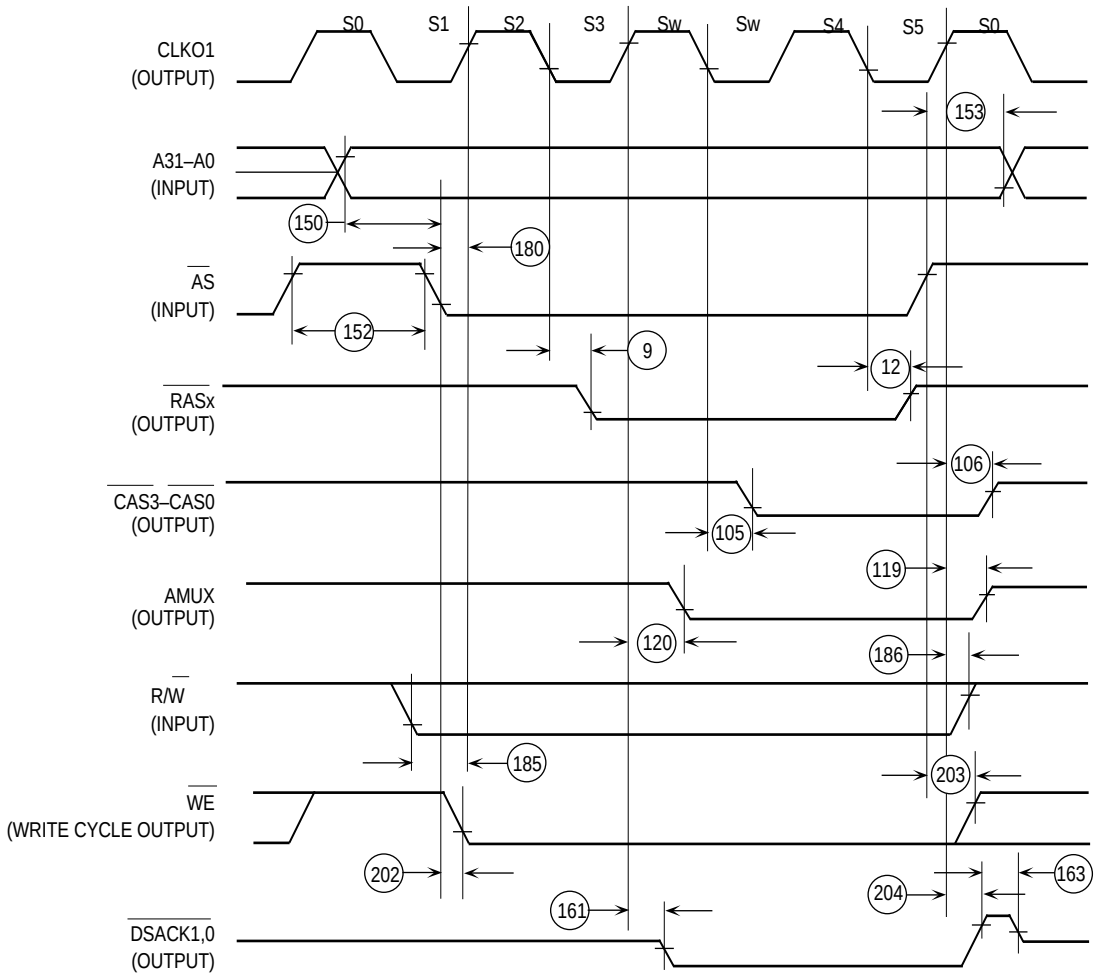


Figure 10-36. External MC68030/MC68360 DRAM Synchronous Cycle Timing Diagram (BSTM = 1; SYNC = 1)

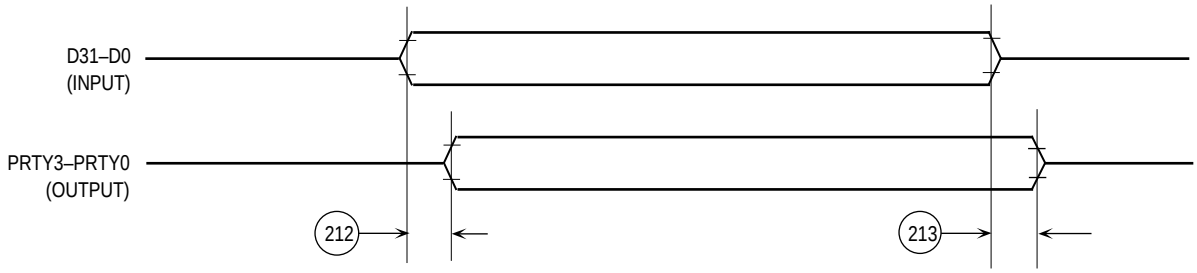


Figure 10-37. External MC68030/MC68360 Parity Bits Timing Diagram

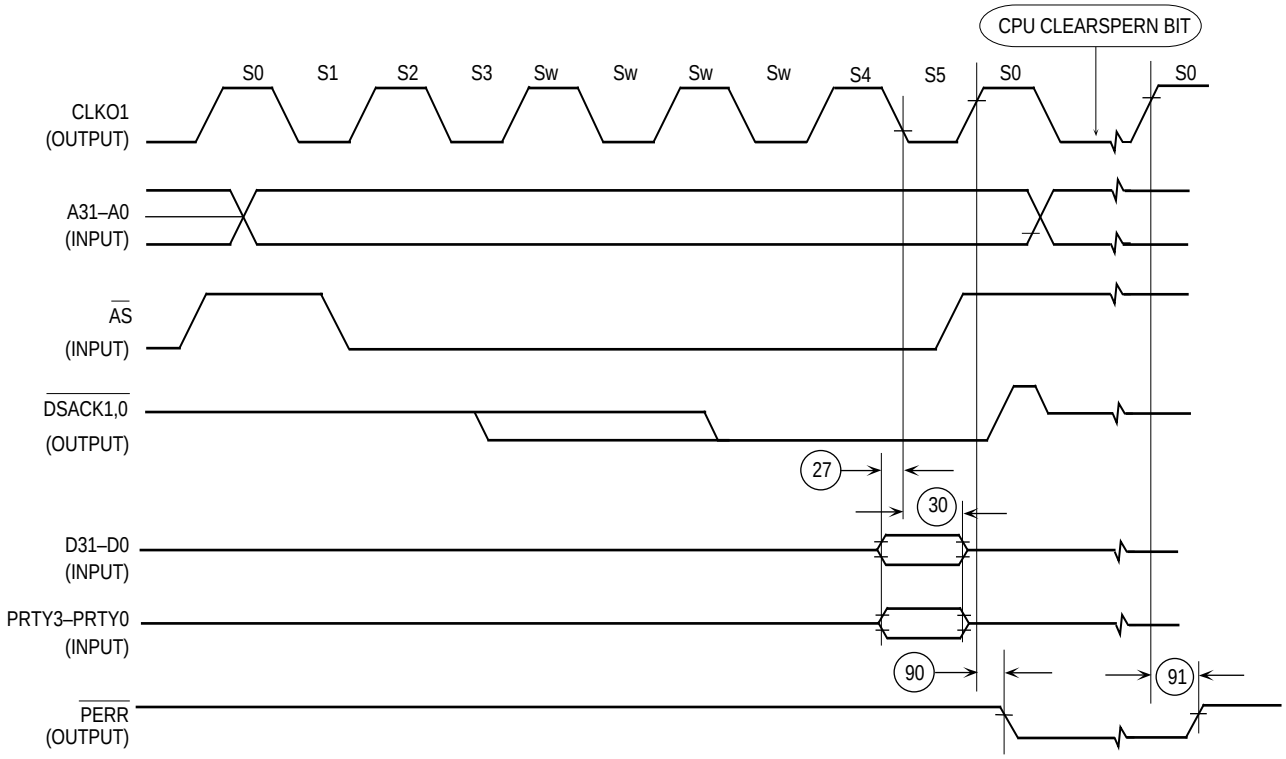


Figure 10-38. External MC68030/MC68360 Parity Bits Timing Diagram

10.15 040 BUS TYPE SLAVE MODE BUS ARBITRATION AC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-39)

Num.	Characteristic	3.3 V or 5.0 V		5.0V		Unit
		25.0 MHz		33.34MHz		
		Min	Max	Min	Max	
231	Address, Transfer Attributes High Impedance to Clock High	7	—	6	—	ns
232[1]	Clock High to BG Low	—	20	—	15	ns
233	Clock High to BG High	4	20	4	15	ns
234	BB High to Clock High (040 output)	7	—	6	—	ns
235	BB High Impedance to Clock High (040 output)	0	—	0	—	ns
236	Clock High to BB Low (360 output)	—	20	—	15	ns
237	Clock High to BB High (360 output)	—	20	—	15	ns
238	Clock Low to BB High Impedance (360 output)	—	20	—	15	ns

NOTES:
1. $\overline{BG}$ remains low until either the SDMA or the IDMA requests the external bus.

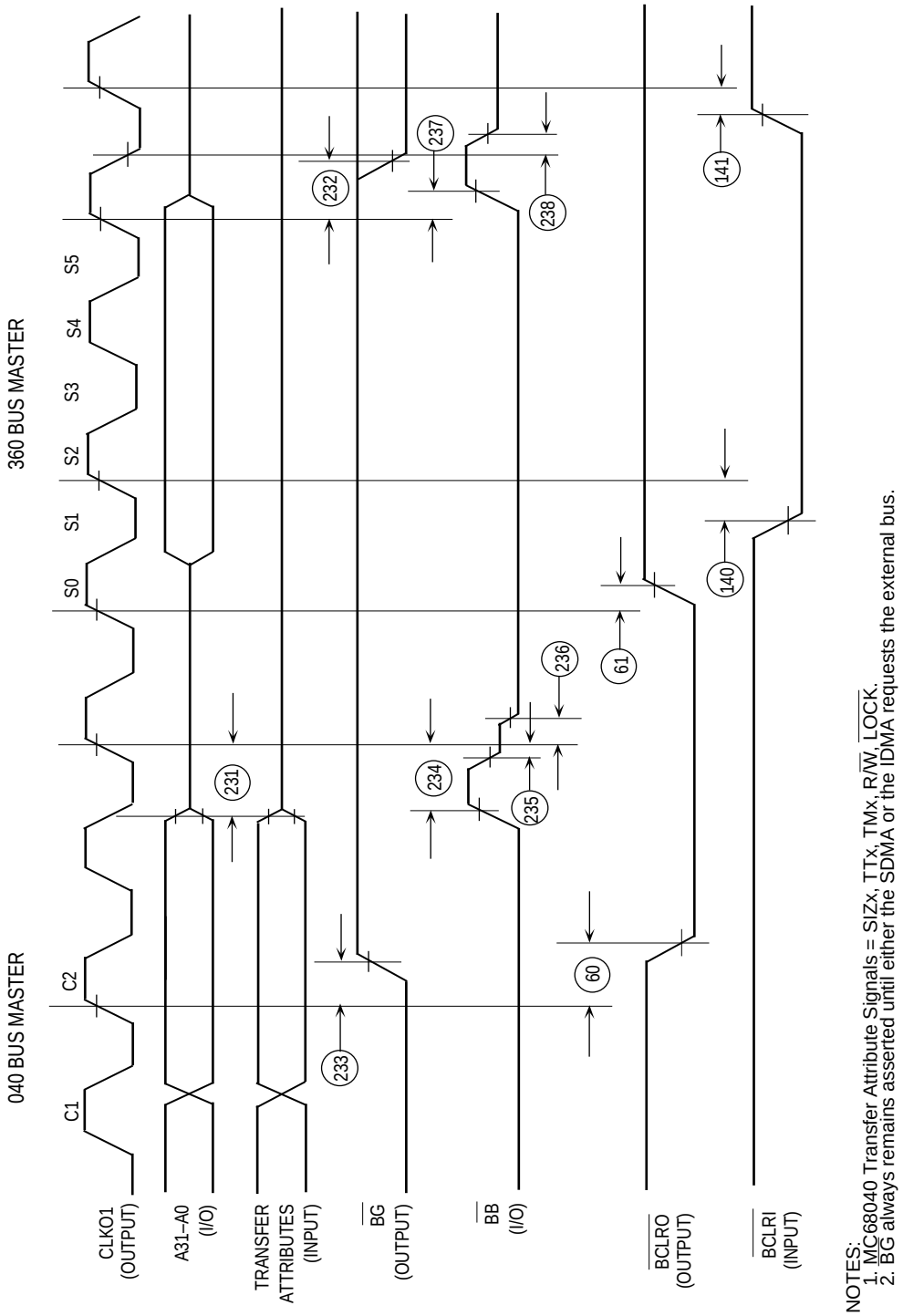


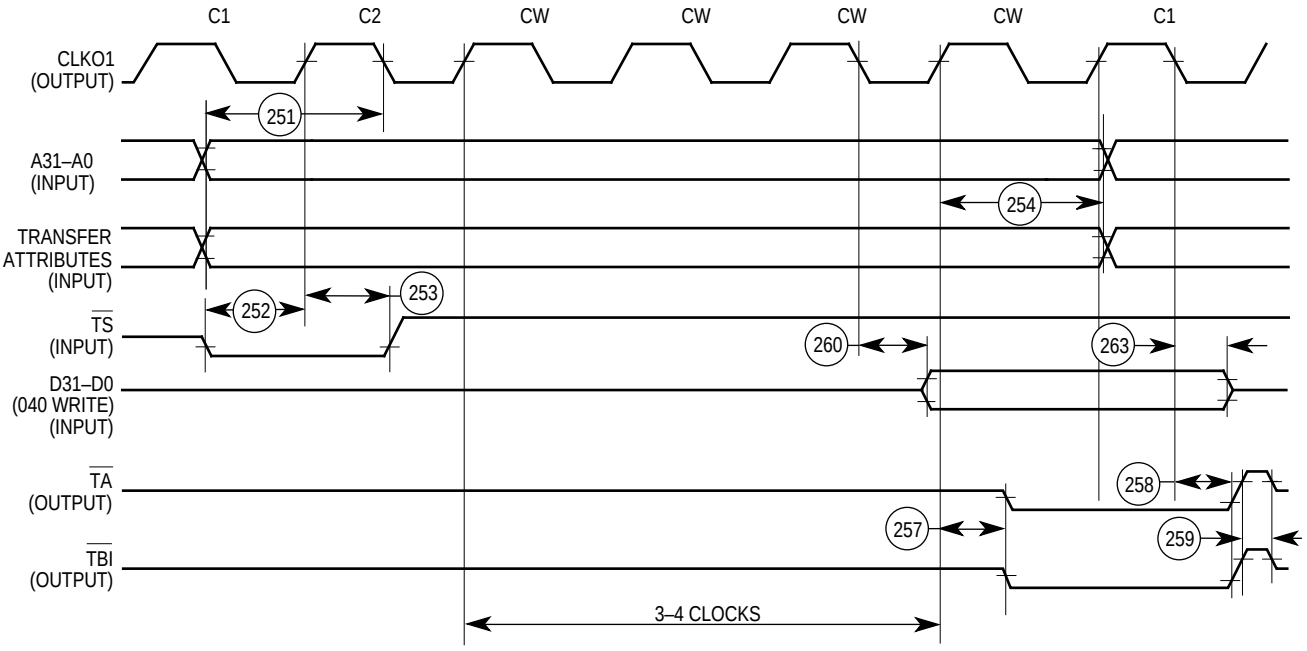
Figure 10-39. MC68040 Companion Mode Arbitration

10.16 040 BUS TYPE SLAVE MODE INTERNAL READ/WRITE/IACK CYCLES AC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-40–Figure 10-42)

Num.	Characteristic	3.3 V or 5.0 V		5.0V		Unit
		25.0 MHz		33.34MHz		
		Min	Max	Min	Max	
251[1]	Address, Transfer Attributes Valid to Clock Low	15	—	11.25	—	ns
252	TS Low to Clock High	7	—	6	—	ns
253	Clock High to TS High	5	—	3	—	ns
254	Clock High to Address, Transfer Attributes Invalid	0	—	0	—	ns
255	Data-In, MBARE Valid to Clock High (040 Write)	0	—	0	—	ns
256	Clock High to Data-In, MBARE Hold Time	0	—	0	—	ns
257	Clock High to TA, TBILow (External to External)	4	20	4	15	ns
257	Clock High to TA, TBILow (External to Internal)	4	23	4	18	ns
258[2]	Clock High to TA, TBIHigh	4	20	4	15	ns
259	TA, TBIHigh to TA, TBIHigh Impedance	—	15	—	11.25	ns
260	Clock Low to Data-Out Valid (040 Read)	—	20	—	15	ns
262	Clock Low to Data-Out Invalid	—	20	—	15	ns
263	Clock Low to Data-Out High Impedance	—	15	—		ns
264	Clock High to AVECO Low	—	20	—	15	ns
265	Clock Lowto AVECO High Impedance	—	30	—	23	ns
266	Clock Low to IACK Low	—	30	—	23	ns
267	Clock High to IACK High	—	30	—	23	ns
268	Clock Low to AVEC Low	—	30	—	23	ns

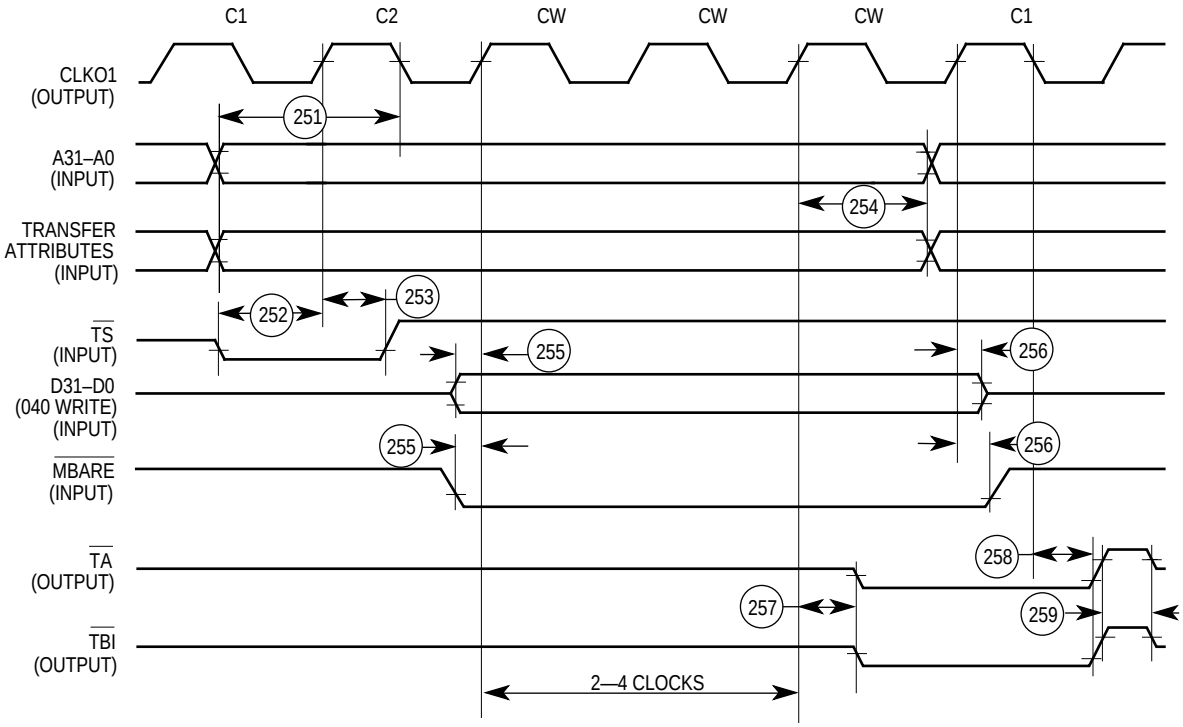
NOTES:
1.Transfer attributes signals = SIZx, TTx, TMx, R/W, and LOCK.
2.When MC68040 is accessing the internal registers, specification 258 is from clock low not clock high.



NOTES:

1. Three wait states are inserted when reading the SIM, dual-port RAM, and the CPM. Four wait states are inserted when reading the SI RAM. Additional wait states may be inserted when the SHEN1-SHEN0=10 and one of the internal masters is accessing an internal peripheral.
2. MC68040 Transfer Attribute Signals = SIZx, TTx, TMx, R/$\overline{W}$, $\overline{LOCK}$

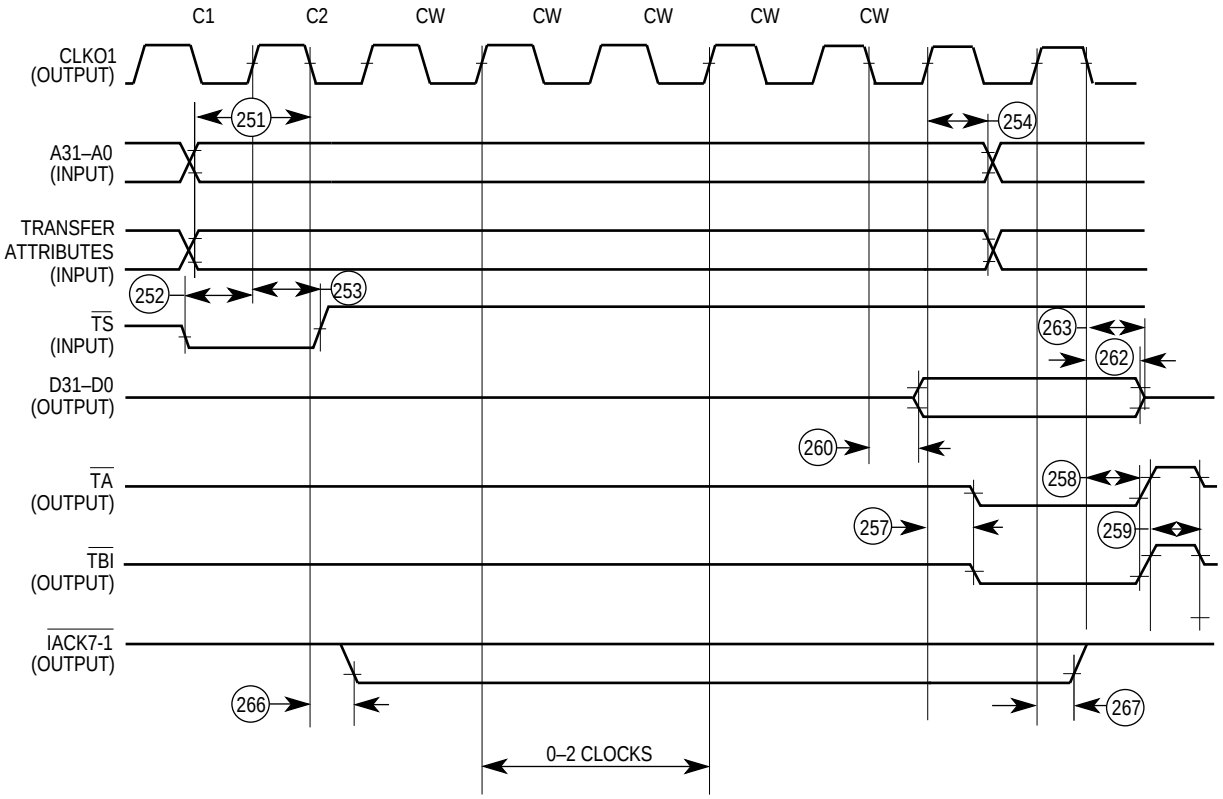
Figure 10-40. MC68040 Internal Registers Read Cycles



NOTES:

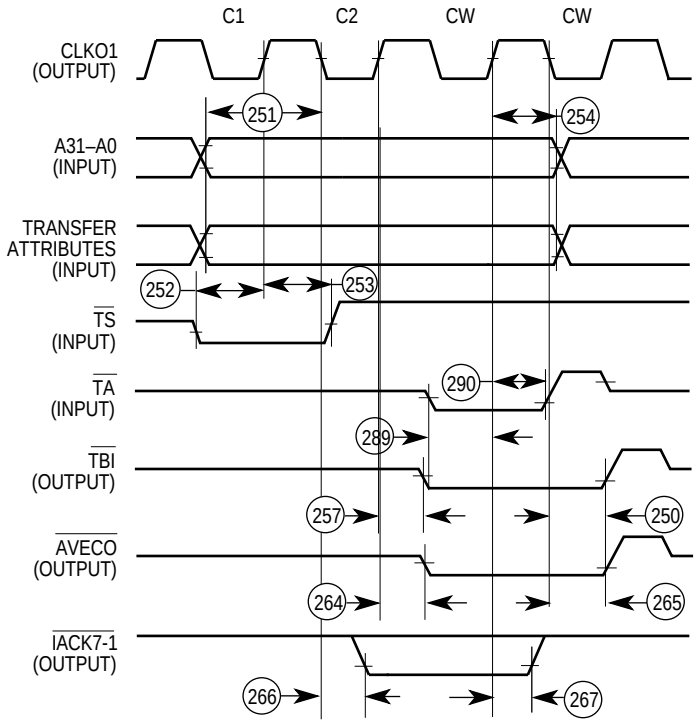
1. Two wait states are inserted when writing to the SIM. Three wait states are inserted when writing to the dual-port RAM and CPM. Four wait states are inserted when writing to the SI RAM. Additional wait states may be inserted when the SHEN1— SHEN0=10 and one of the internal masters is accessing an internal peripheral.
2. MC68040 Transfer Attribute Signals = SIZx, TTx, TMx, R/$\overline{W}$, $\overline{LOCK}$

Figure 10-41. MC68040 Internal Registers Write Cycles



NOTES:
1. MC68040 Transfer Attribute Signals = SIZx, TTx, TMx, R/$\overline{W}$, $\overline{LOCK}$.
2. Up to two wait states may be inserted for internal arbitration.

Figure 10-42. MC68040 IACK Cycles (Vector Driven)



NOTES:

1. MC68040 Transfer Attribute Signals = SIZx, TTx, TMx, R/W, LOCK.

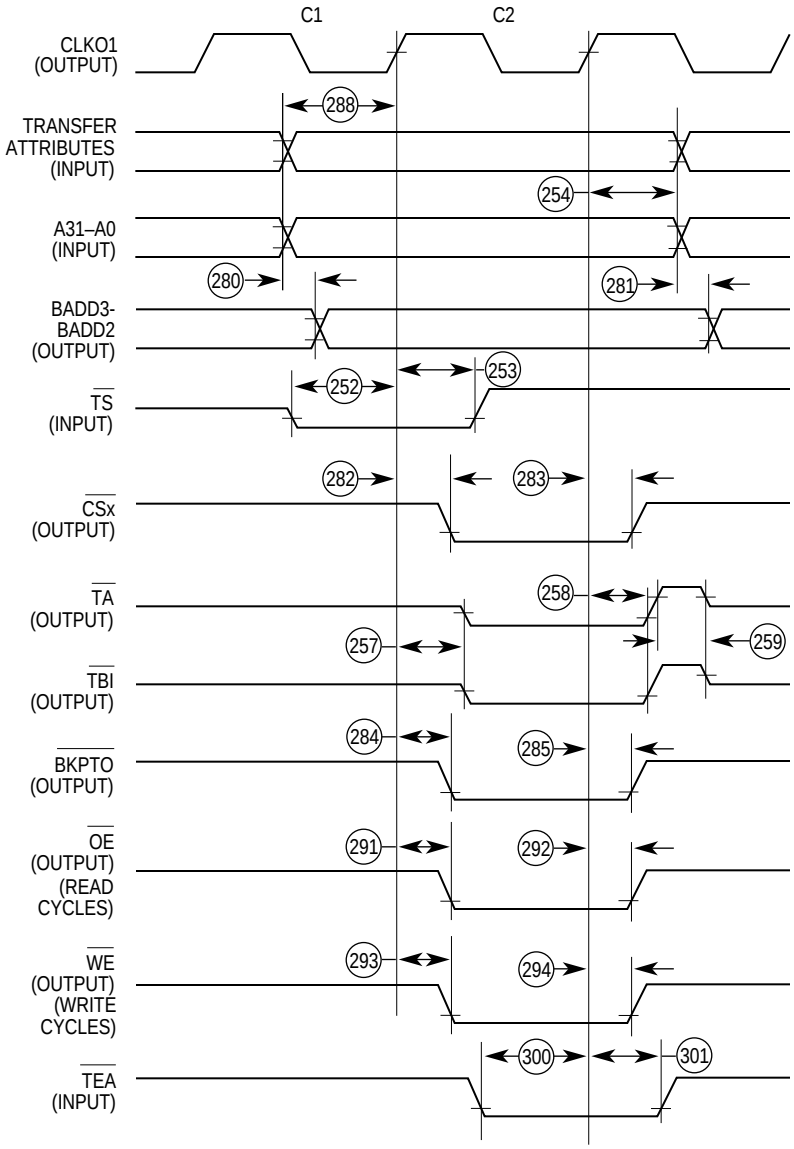
Figure 10-43. MC68040 IACK Cycles (No Vector Driven)

10.17 040 BUS TYPE SRAM/DRAM CYCLES AC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-44–Figure 10-48)

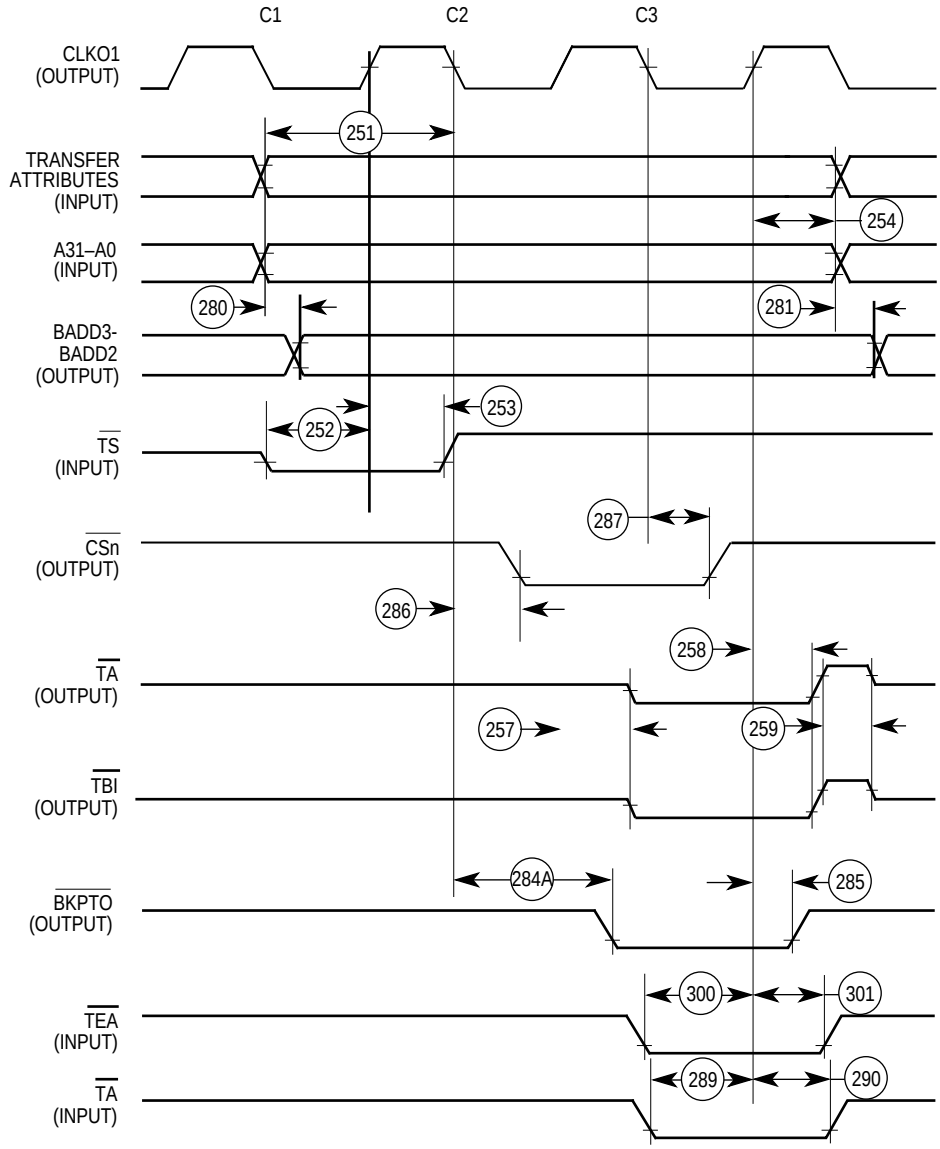
Num.	Characteristic	3.3 V or 5.0 V		5.0V		Unit
		25.0 MHz		33.34MHz		
		Min	Max	Min	Max	
280	Address Valid to BADD2–3 Valid	—	20	—	15	ns
280A	BADD2–3 Valid to CAS assertion	15	—	10	—	ns
281	Address Invalid to BADD2–3 Invalid	0	—	0	—	ns
282	Clock High to CS/RAS Low (TSS40=0)	4	16	4	12	ns
283	Clock High to CS/RAS High (CSNT40=0)	4	16	4	12	ns
284	Clock High to BRK Low	—	20	—	15	ns
284A	Clock Low to BRK Low	—	20	—	15	ns
285	Clock High to BRK High	—	20	—	15	ns
286	Clock Low to CS /RASx Low (TSS40=1)	4	16	4	12	ns
287	Clock Low to CS /RASx High (CSNT40=1)	4	16	4	12	ns
288[1]	AddressTransfer Attributes Valid to Clock High (TSS40=0)	12	—	11	—	ns
289[2]	TA Low to Clock High (External Termination)	11	—	9	—	ns
290[2]	Clock High to TA High (External Termination)	—	20	—	15	ns
291	Clock High to OE Low (Read Cycles)	—	20	—	15	ns
292	Clock High to OE High (Read Cycles)	—	20	—	15	ns
293	Clock High to WE Low (Write Cycles)	—	20	—	15	ns
294	Clock High to WE High (Write Cycles)	—	20	—	15	ns
295	Clock High to CAS Low	4	13	4	10	ns
295A	Clock High to CAS High (040 Burst Read only)	4	13	4	10	ns
296	Clock High to CAS High	4	13	4	10	ns
297	Clock Low to AMUX Low	3	16	3	12	ns
298	Clock High to AMUX High	3	16	3	12	ns
299	Clock High to BADD2–3 Valid (040 Burst Cycles)	4	20	4	15	ns
300[2]	TEA Low to Clock High	11	—		—	ns
301[2]	Clock High to TEA High	2	20	2	15	ns
302	Data, Parity Valid to Clock High (Data, Parity Set-up)	7	—	6	—	ns
303	Clock High to Data, Parity Invalid (Data, Parity Hold)	7	—	5	—	ns
305	CLKO1 High (After TS low) to Parity Valid	—	20	—	15	ns
306	CLKO1 High (After TA low) to Parity Hi-Z	4	20		15	ns

NOTES:
1.Transfer attributes signals = SIZx, TTx, TMx, R/$\overline{W}$, and $\overline{LOCK}$.
2.TEA/$\overline{TA}$ should not be asserted on a DRAM burst access, or on the same clock or before $\overline{RASx}$ / $\overline{CSx}$ is asserted



NOTE: MC68040 Transfer Attribute Signals = SIZx, TTx, TMx, R/W, LOCK

Figure 10-44. MC68040 SRAM Read/Write Cycles (TSS40 =0, CSNT40 = 0)



NOTE: MC68040 Transfer Attribute Signals = SIZx, TTx, TMx, R/$\overline{W}$, $\overline{LOCK}$

Figure 10-45. MC68040 SRAM Read/Write Cycles (TSS40 =1, CSNT40 = 1)

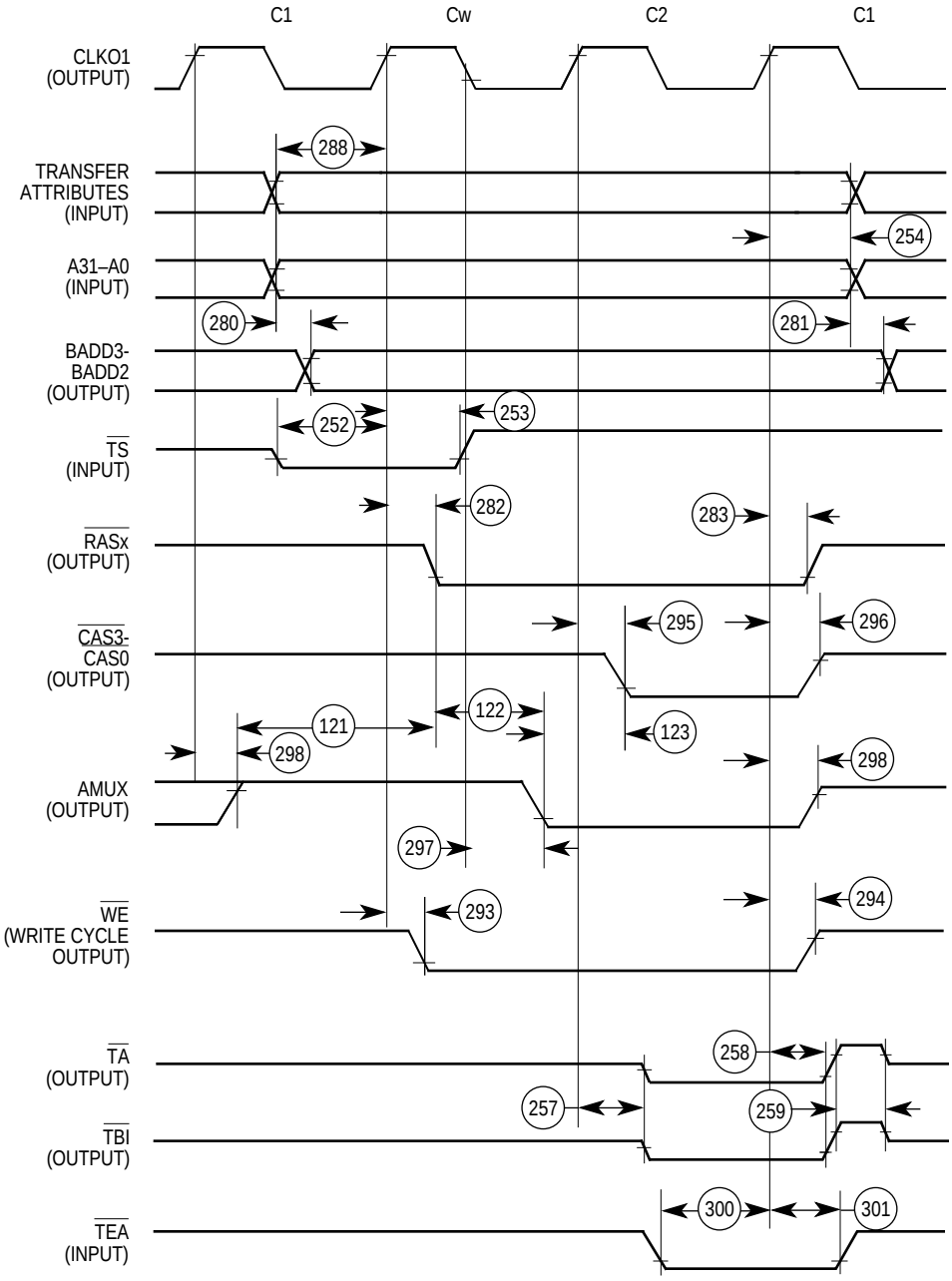


Figure 10-46. External MC68040 DRAM Cycles Timing Diagram

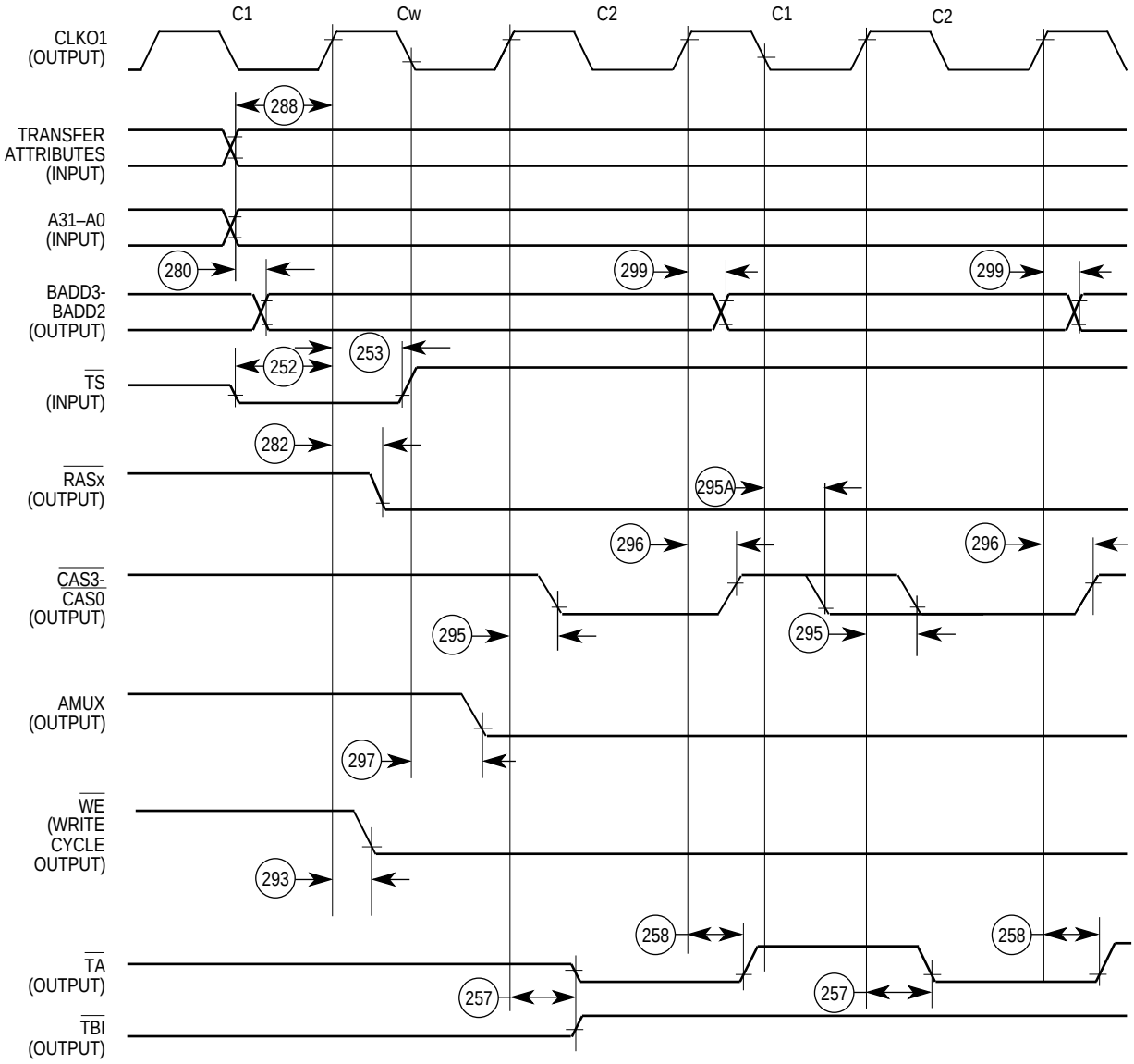


Figure 10-47. External MC68040 DRAM Burst Cycles Timing Diagram

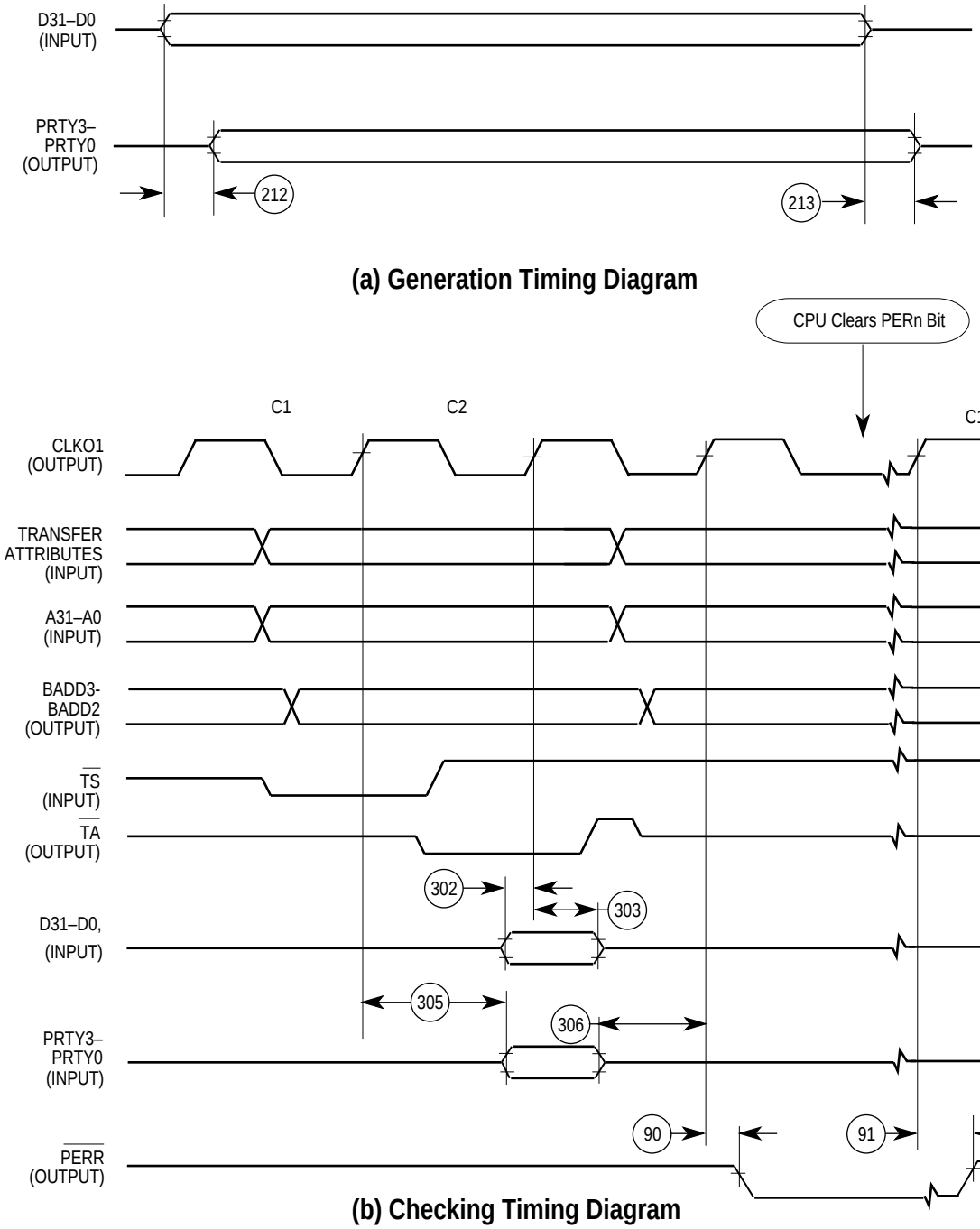


Figure 10-48. External MC68040 Parity Bit Checking Timing Diagram

10.18 IDMA AC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-49 and Figure 10-50.)

Num.	Characteristic	3.3 V or 5.0 V		5.0V		Unit
		25.0 MHz		33.34MHz		
		Min	Max	Min	Max	
1	CLKO1 Low to DACK , DONE Asserted	—	24	—	18	ns
2	CLKO1 Low to DACK , DONE Negated	—	24	—	18	ns
3[1]	DREQ Asserted to AS Asserted (for DMA Bus Cycle)	$3t_{cyc} + t_{AIST} + t_{CLSA}$				
4[1]	Asynchronous Input Setup Time to CLKO1 Low	12	—	9	—	ns
5[1]	Asynchronous Input Hold Time from CLKO1 Low	0	—	0	—	ns
6	AS to DACK Assertion Skew	0	20	0	15	ns
7	DACK to DONE Assertion Skew	–8	8	-6	6	ns
8	AS, DACK , DONE Width Asserted	70	—	52.5	—	ns
8A	AS, DACK , DONE Width Asserted (Fast Termination Cycle)	28	—	20.5	—	ns
10[1]	Asynchronous Input Setup Time to CLKO1 Low	5	—	4	—	ns
11[1]	Asynchronous Input Hold Time from CLKO1 Low	10	—	7.5	—	ns
12[2]	DREQ Input Setup Time to CLKO1 Low	20	—	15	—	ns
13[2]	DREQ Input Hold Time from CLKO1 Low	5	—	3.75	—	ns
14[2]	DONE Input Setup Time to CLKO1 Low	20	—	15	—	ns
15[2]	DONE Input Hold Time from CLKO1 Low	5	—	3.75	—	ns
16[2]	DREQ Asserted to AS Asserted	2	—	2	—	clk

NOTES:
1.These specifications are for asynchronous mode.
2.These specifications are for synchronous mode.

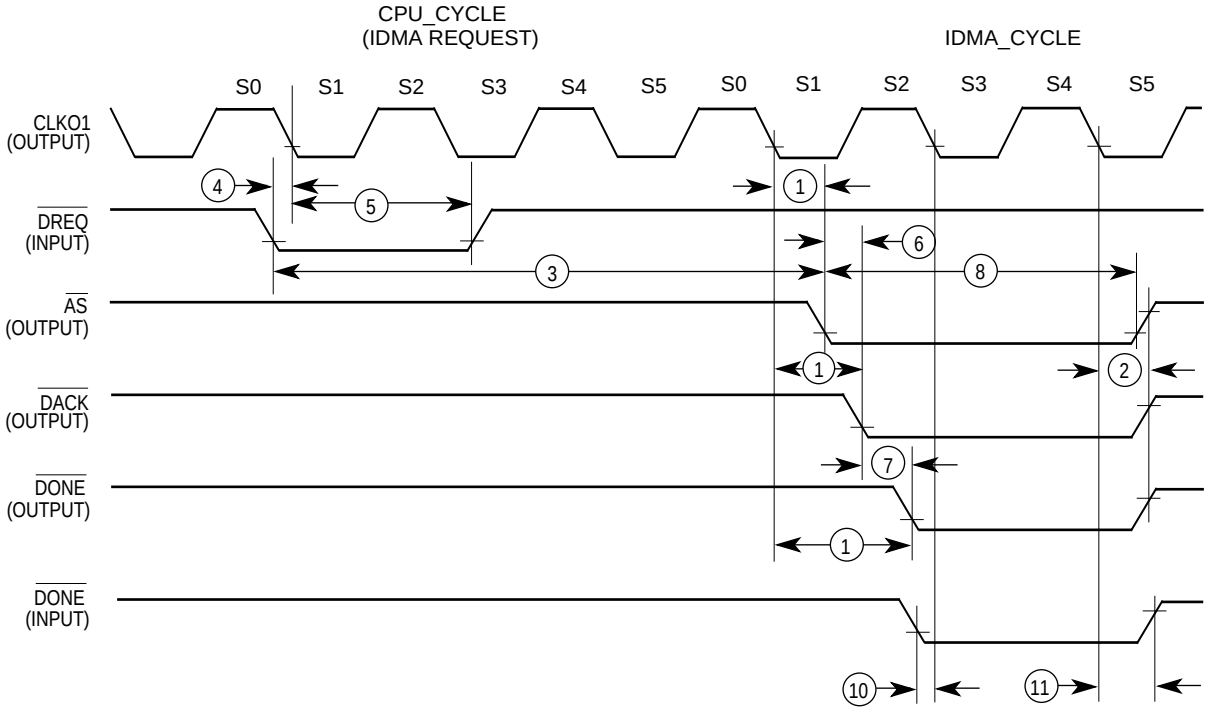


Figure 10-49. IDMA Signal Asynchronous Timing diagram

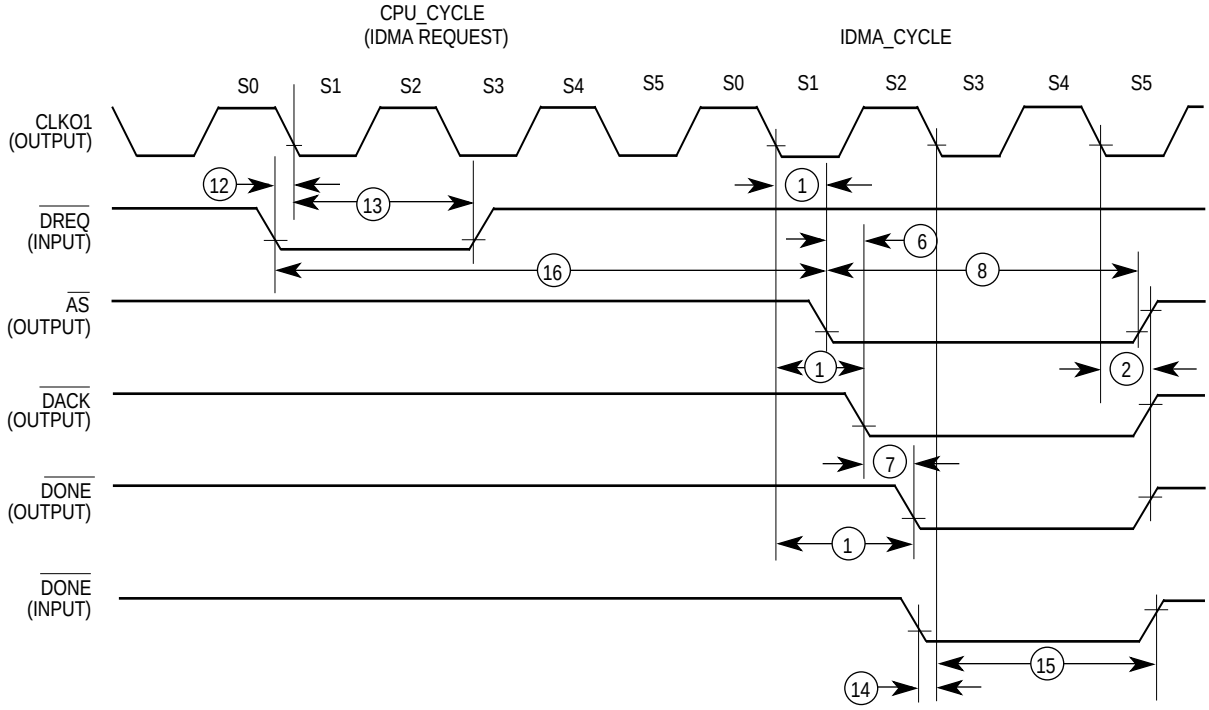


Figure 10-50. IDMA Signal Synchronous Timing diagram

10.19 PIP/PIO AC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-51– Figure 10-55)

Num.	Characteristic	3.3 V or 5.0 V		5.0 V		Unit
		25.0 MHz		33.3 MHz		
		Min	Max	Min	Max	
21	Data-In Setup Time to STBI Low	0	—	0	—	ns
22	Data-In Hold Time to STBI High	2.5– t3	—	2.5– t3	—	clk
23	STBI Pulse Width	1.5	—	1.5	—	clk
24	STBO Pulse Width	1 CLKO1 - 5ns	—	1 CLKO1 - 5ns	—	-
25	Data-Out Setup Time to STBO Low	2	—	2	—	clk
26	Data-Out Hold Time from STBO High	5	—	5	—	clk
27	STBI Low to STBO Low (Rx Interlock)	—	2	—	2	clk
28	STBI Low to STBO High (Tx Interlock)	2	—	2	—	clk
29	Data-In Setup Time to Clock Low	20	—	15	—	ns
30	Data-In Hold Time from Clock Low	10	—	7.5	—	ns
	Clock High to Data-Out Valid (CPU Writes Data, Control, or Direction)	—	25	—	25	ns

Note: t3 = Spec. 3 on Table 10.7

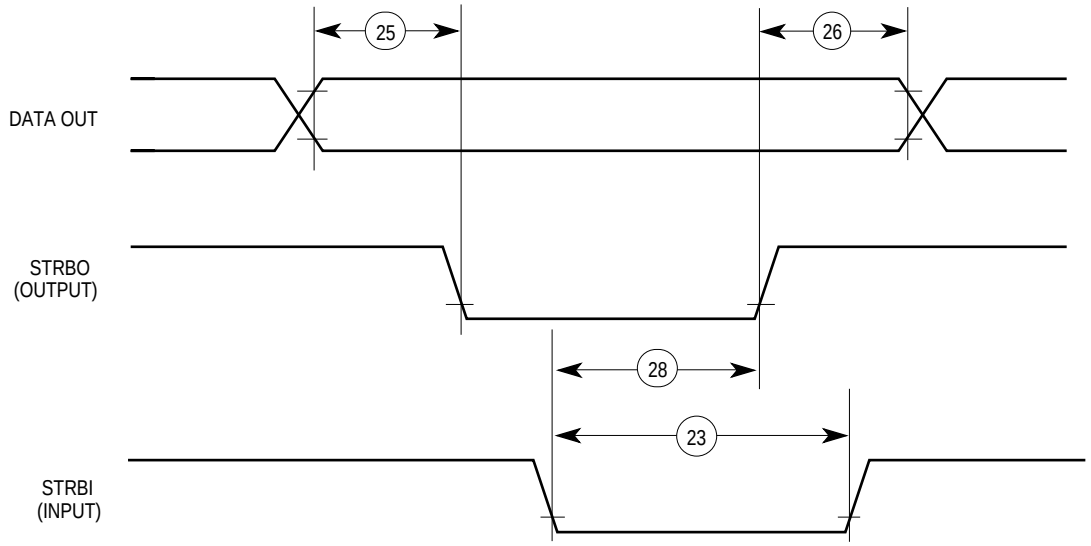


Figure 10-51. PIP Rx (Interlock Mode)

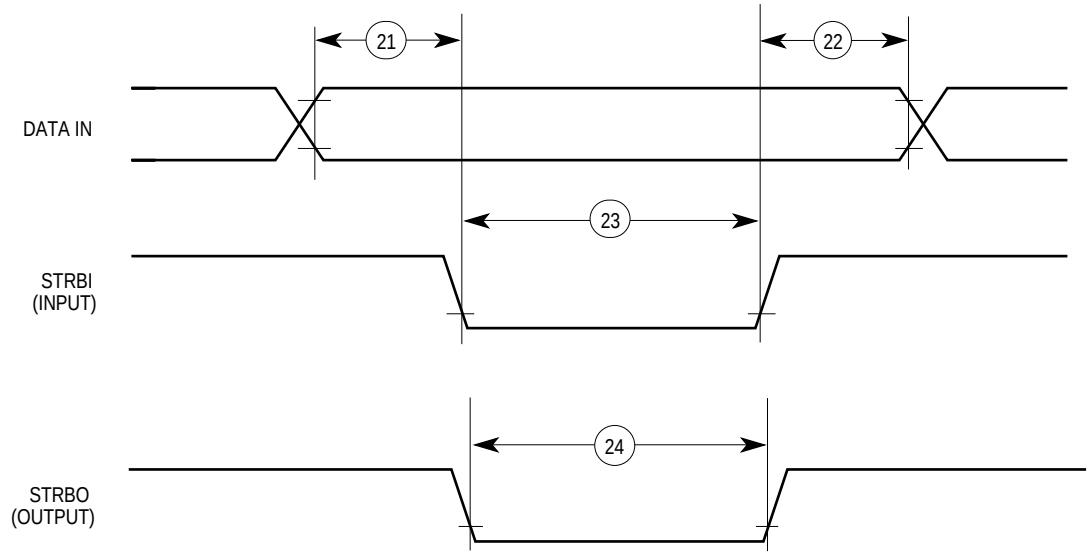


Figure 10-52. PIP Tx (Interlock Mode)

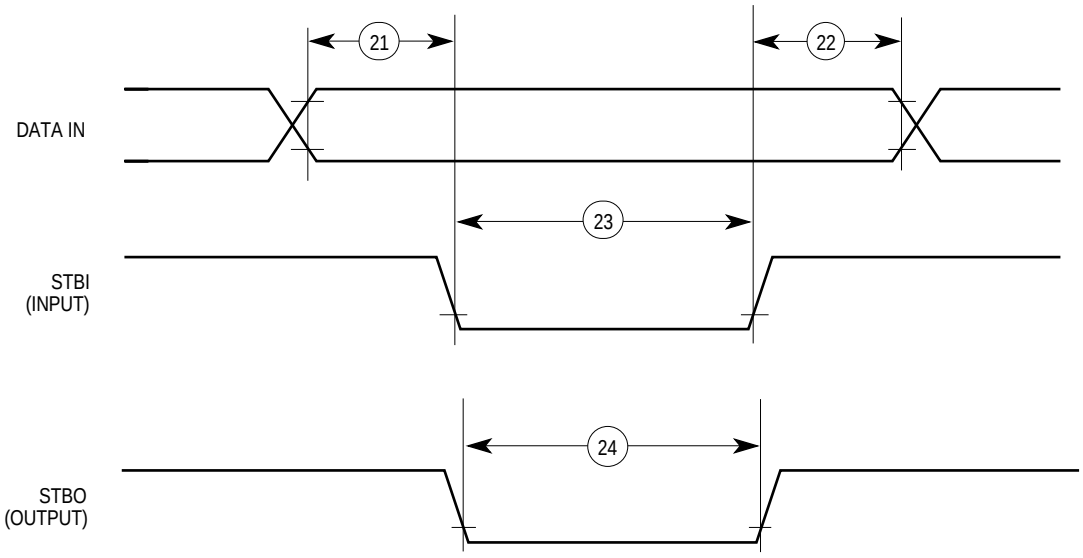


Figure 10-53. PIP Tx (Pulse Mode)

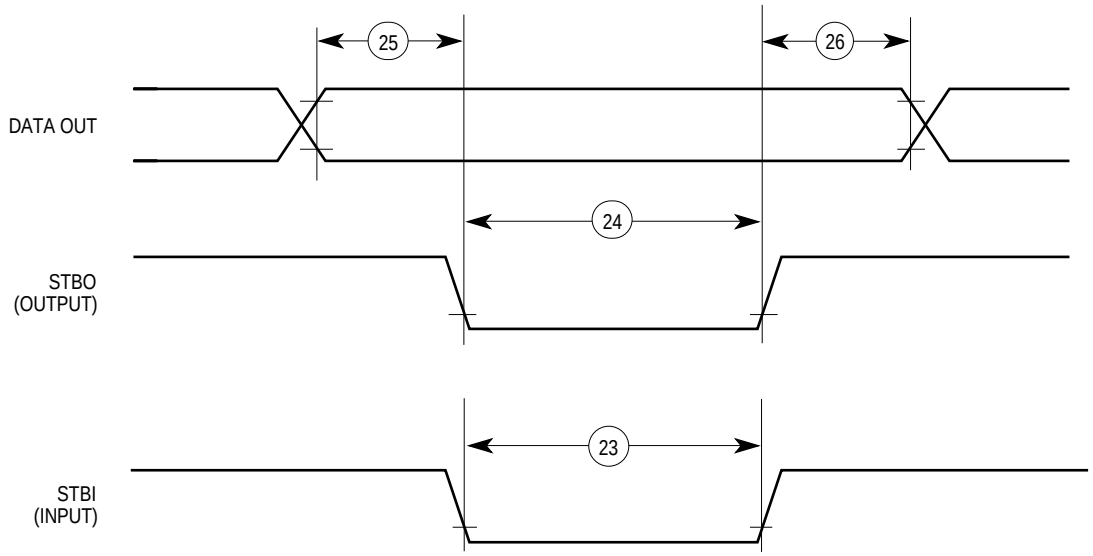


Figure 10-54. PIP Tx (Pulse Mode)

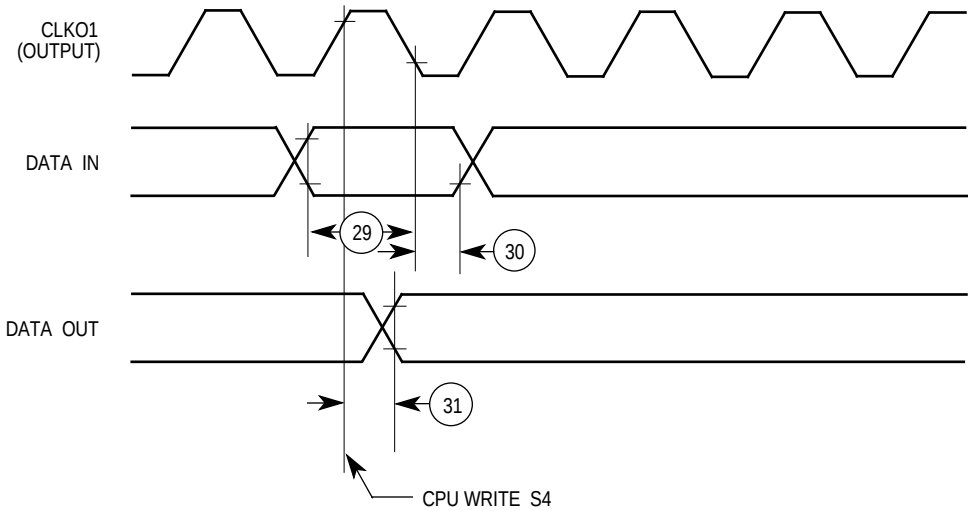


Figure 10-55. Parallel I/O Data-in/Data-Out Timing Diagram

10.20 INTERRUPT CONTROLLER AC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-56 and Figure 10-57)

Num.	Characteristic	3.3 V or 5.0 V		5.0 V		Unit
		25.0 MHz		33.34 MHz		
		Min	Max	Min	Max	
35	Port C Interrupt Pulse Width Low (Edge Triggered Mode)	70	—	55	—	ns
36	Minimum Time Between Active edges PortC	70	—	55	—	clk
37	Clock High to IOUT Valid (Slave Mode)	—	20	—	17	ns
38	Clock High to RQOUT Valid (Slave Mode)	—	20	—	17	ns

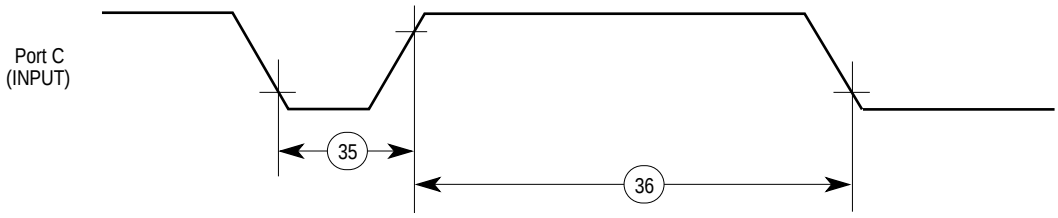


Figure 10-56. Interrupts Timing Diagram

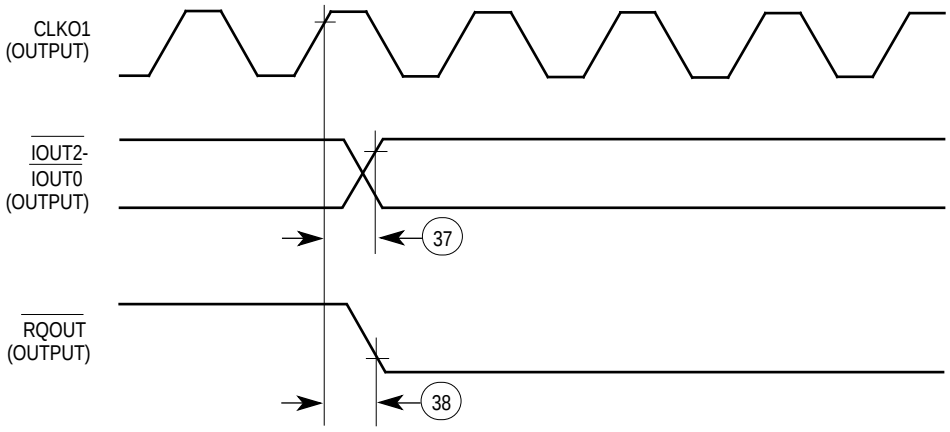


Figure 10-57. Slave Mode: Interrupts Timing Diagram

10.21 BAUD RATE GENERATOR AC ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-58)

Num.	Characteristic	3.3 V or 5.0 V 25.0 MHz		5.0 V 33.34 MHz		Unit
		Min	Max	Min	Max	
50	BRGO Rise and Fall Time	—	10	—	7.5	ns
51	BRGO Duty Cycle	40	60	40	60	%
52	BRGO Cycle	40		30		ns

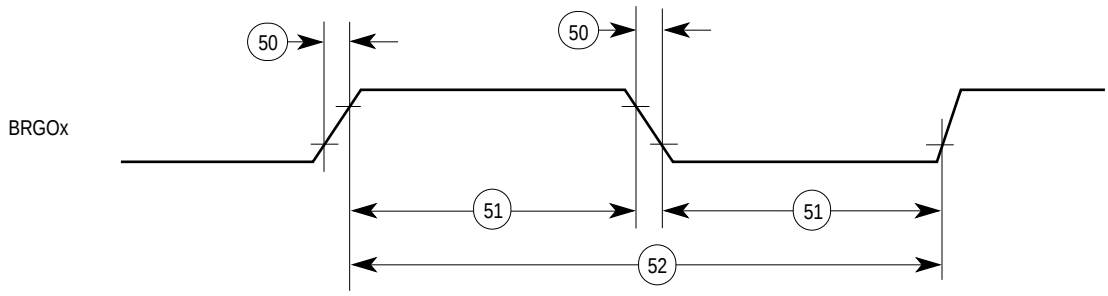


Figure 10-58. Baud Rate Generator Output Signals

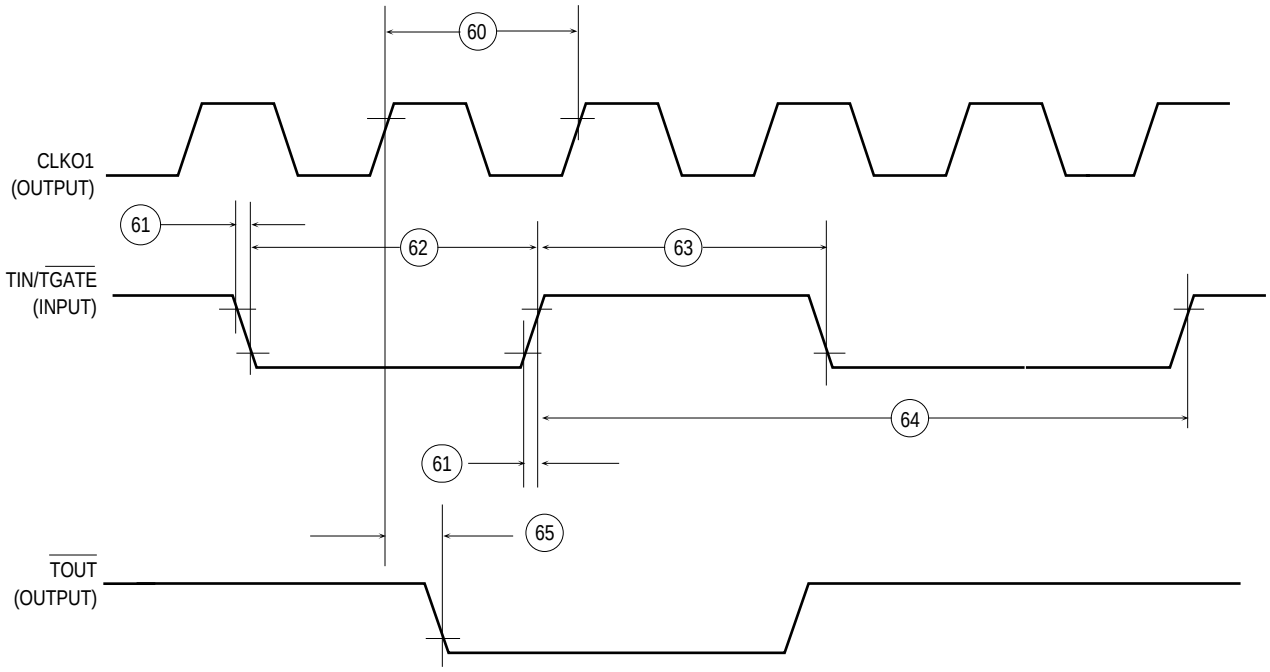


Figure 10-59. CPM General Purpose Timers

10.22 TIMER ELECTRICAL SPECIFICATIONS

The electrical specifications in this document are preliminary. See Figure 10-59)

Num.	Characteristic	Symbol	3.3 V or 5.0 V 25 MHz		5.0 V 33.34MHz		Unit
			Min	Max	Min	Max	
61	TIN/TGATE Rise and Fall Time	t_{rf}	10	—	10	—	ns
62	TIN/TGATE Low Time	—	1	—	1	—	clk
63	TIN/TGATE High Time	—	2	—	2	—	clk
64	TIN/TGATE Cycle Time	—	3	—	3	—	clk
65	CLKO1 High to TOUT Valid	t_{TO}	3	25	3	22	ns

10.23 SI ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-60–Figure 10-64.)

Num.	Characteristic	3.3 V or 5.0 V		5.0 V		Unit
		25.0 MHz		33.34 MHz		
		Min	Max	Min	Max	
70[1,3]	L1RCLK, L1TCLK Fequency (DSC=0)	—	10	—	10	MHz
71[3]	L1RCLK, L1TCLK Width Low(DSC=0)	P+10	—	P+10	—	ns
71A[2]	L1RCLK, L1TCLK Width High (DSC=0)	P+10	—	P+10	—	ns
72	L1TXD, L1ST(1–4), L1RQ, L1CLKO Rise/Fall Time	—	15	—	15	ns
73	L1RSYNC, L1TSYNC Valid to L1CLK edge (SYNC Setup Time)	20	—	20	—	ns
74	L1CLK edge to L1RSYNC, L1TSYNC Invalid (SYNC Hold Time)	35	—	35	—	ns
75	L1RSYNC, L1TSYNC Rise/Fall Time	—	15	—	15	ns
76	L1RXD Valid to L1CLK edge (L1RXD Setup Time)	42	—	42	—	ns
77	L1CLK edge to L1RXD Invalid (L1RXD Hold Time)	35	—	35	—	ns
78	L1CLK edge to L1ST(1–4) Valid	10	45	10	45	ns
78A[4]	L1SYNC Valid to L1ST(1–4) Valid	10	45	10	45	ns
79	L1CLK Edge to L1ST(1–4) Invalid	10	45	10	45	ns
80	L1CLK Edge to L1TXD Valid	10	65	10	65	ns
80A[4]	L1TSYNC Valid to L1TXD Valid	10	65	10	65	ns
81	L1CLK Edge to L1TXD High Impedance	0	42	0	42	ns
82	L1RCLK, L1TCLK Fequency (DSC=1)	—	12.5	—	16	MHz
83	L1RCLK, L1TCLK Width Low(DSC=1)	P+10	—	P+10	—	ns
83A[2]	L1RCLK, L1TCLK Width High (DSC=1)	P+10	—	P+10	—	ns
84	L1CLK Edge to L1CLKO Valid (DSC=1)	—	30	—	30	ns
85[3]	L1RQ Valid Before Falling Edge of L1TSYNC	1	—	1	—	L1TCLK
86[3]	L1GR Setup Time	42	—	42	—	ns
87[3]	L1GR Hold Time	42	—	42	—	ns
88	L1CLK edge to L1SYNC Valid (FSD = 00, CNT = 0000, BYT = 0, DSC=0)	—	0	—	0	ns

NOTES:
1. The ratio SyncCLK/L1RCLK must be greater than 2.5/1
2. Where P=1/CLKO1. Thus for a 25 MHz CLKO1 rate, P=40ns.
3. These specs are valid for IDL mode only
4. The strobes and Txd on the first bit of the frame become valid after L1CLK edge or L1SYNC, whichever is later.

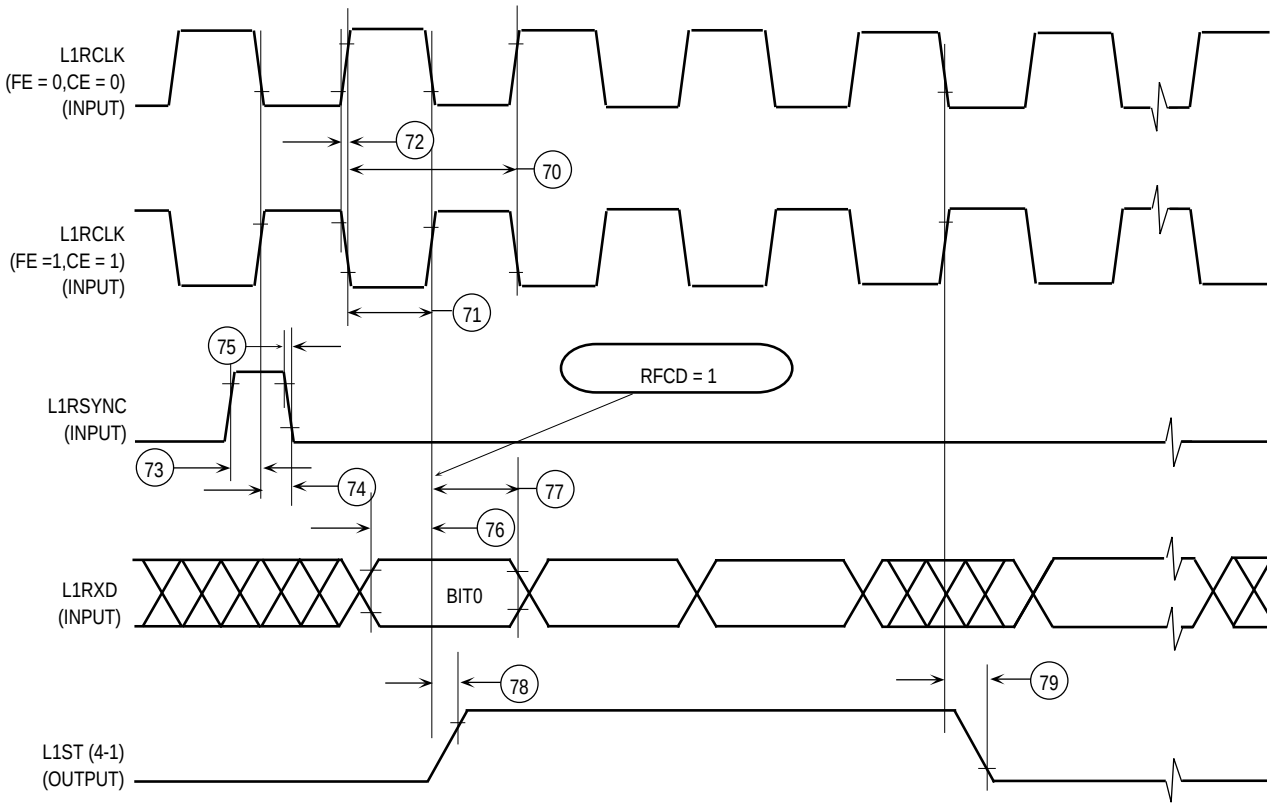


Figure 10-60. SI Receive Timing with Normal Clocking (DSC = 0)

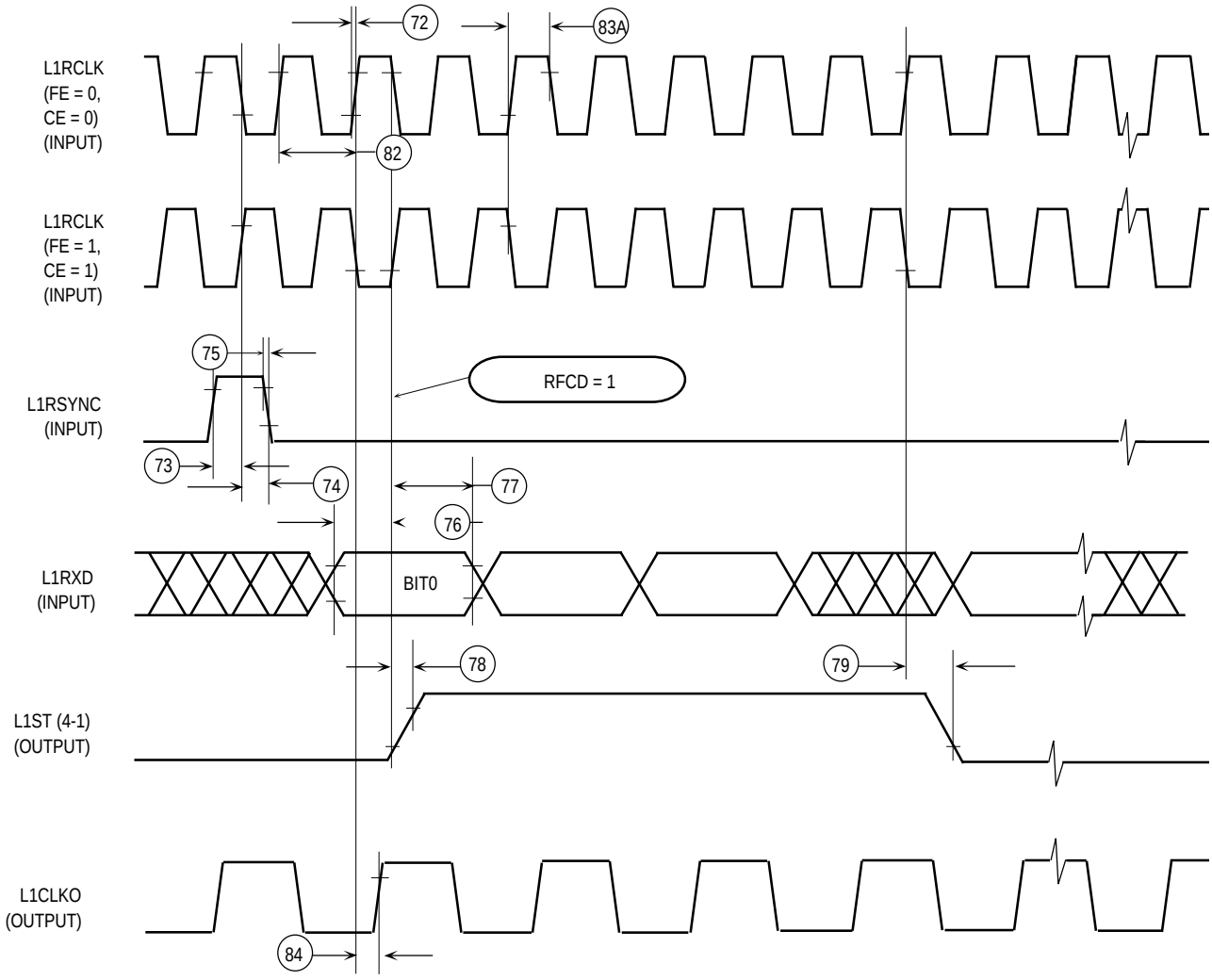


Figure 10-61. SI Receive Timing with Double Speed Clocking (DSC = 1)

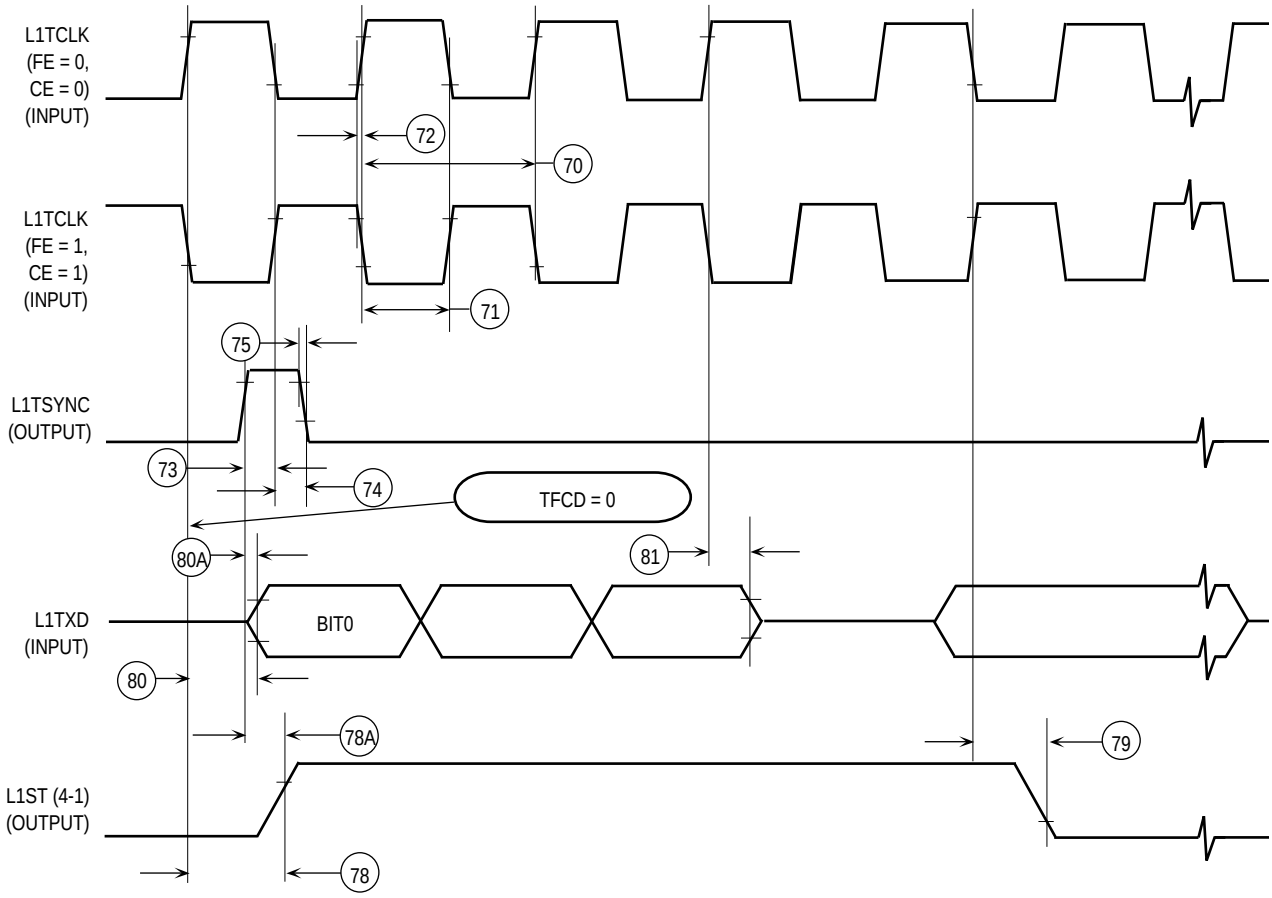


Figure 10-62. SI Transmit Timing with Normal Clocking (DSC = 0)

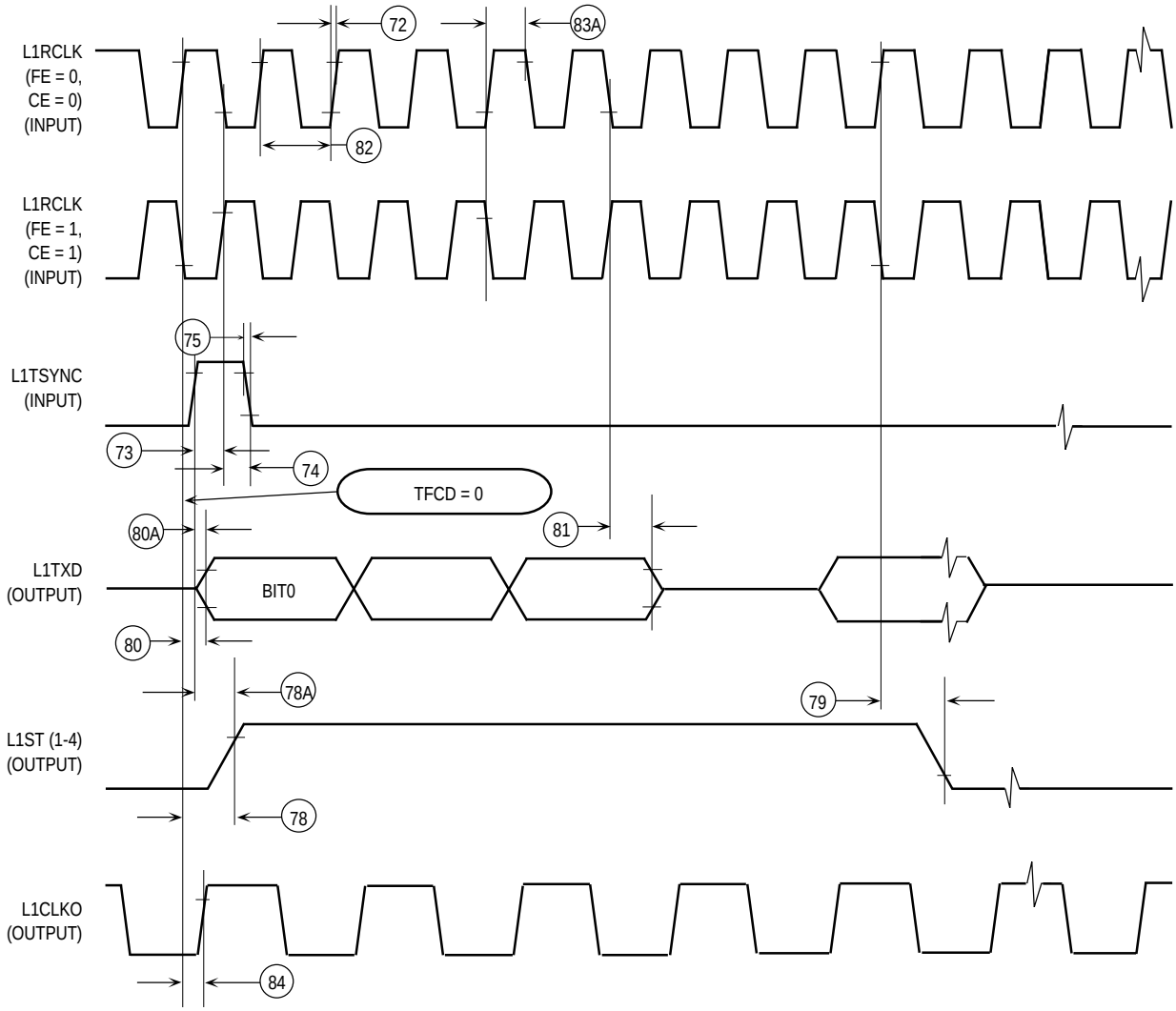


Figure 10-63. SI Transmit Timing with Double Speed Clocking (DSC = 1)

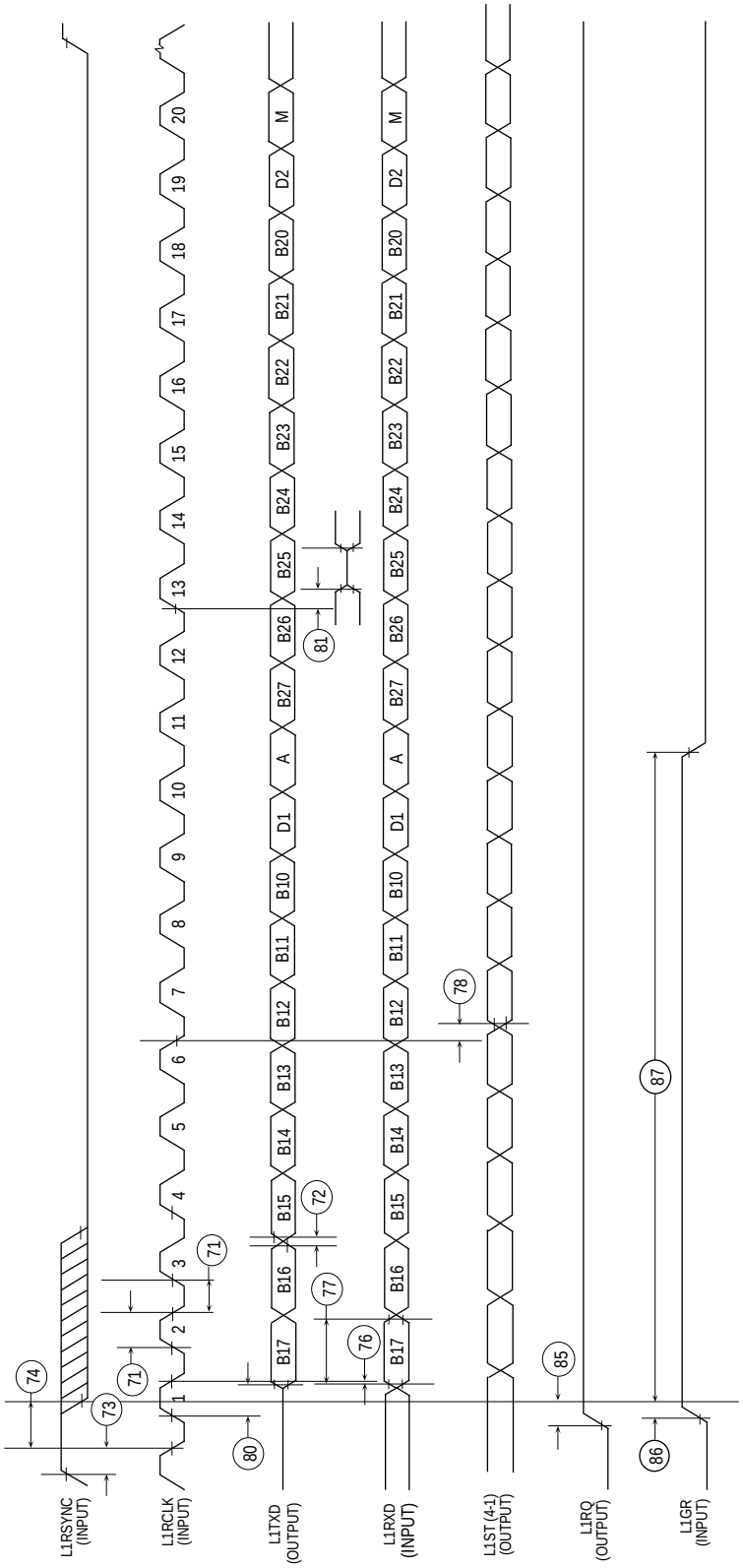


Figure 10-64. IDL Timing

10.24 SCC IN NMSI MODE—EXTERNAL CLOCK ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-65–Figure 10-67)

Num.	Characteristic	3.3 V or 5.0 V		5.0 V		Unit
		25.0 MHz		33.34 MHz		
		Min	Max	Min	Max	
100[1]	RCLK1 and TCLK1 Width High	CLKO1	—	CLKO1	—	
101	RCLK1 and TCLK1 Width Low	CLKO1 + 5nS	—	CLKO1 + 5nS	—	
102	RCLK1 and TCLK1 Rise/Fall Time	—	15	—	15	ns
103	TXD1 Active Delay (From TCLK1 Falling Edge)	0	50	0	50	ns
104	RTS1 Active/Inactive Delay (From TCLK1 Falling Edge)	0	50	0	50	ns
105	CTS1 Setup Time to TCLK1 Rising Edge	5	—	5	—	ns
106	RXD1 Setup Time to RCLK1 Rising Edge	5	—	5	—	ns
107[2]	RXD1 Hold Time from RCLK1 Rising Edge	5	—	5	—	ns
108	CD1 Setup Time to RCLK1 Rising Edge	5	—	5	—	ns

NOTES:
1.The ratio SyncCLK/RCLK1 and SyncCLK/TCLK1 must be greater or equal to 2.25/1
2.Also applies to $\overline{CD}$ and $\overline{CTS}$ hold time when they are used as an external sync signals.

10.25 SCC IN NMSI MODE—INTERNAL CLOCK ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-65–Figure 10-67)

Num.	Characteristic	3.3 V or 5.0 V		5.0 V		Unit
		25.0 MHz		33.34 MHz		
		Min	Max	Min	Max	
100[1]	RCLK1 and TCLK1 Frequency	0	8.3	0	11	MHz
102	RCLK1 and TCLK1 Rise/Fall Time	—	—	—	—	ns
103	TXD1 Active Delay (From TCLK1 Falling Edge)	0	30	0	30	ns
104	RTS1 Active/Inactive Delay (From TCLK1 Falling Edge)	0	30	40	—	ns
105	CTS1 Setup Time to TCLK1 Rising Edge	40	—	40	—	ns
106	RXD1 Setup Time to RCLK1 Rising Edge	40	—	0	—	ns
107[2]	RXD1 Hold Time from RCLK1 Rising Edge	0	—	40	—	ns
108	CD1 Setup Time to RCLK1 Rising Edge	40	—	0	30	ns

NOTES:
1.The ratio SyncCLK/RCLK1 and SyncCLK/TCLK1 must be greater or equal to 3/1
2.Also applies to $\overline{CD}$ and $\overline{CTS}$ hold time when they are used as an external sync signals.

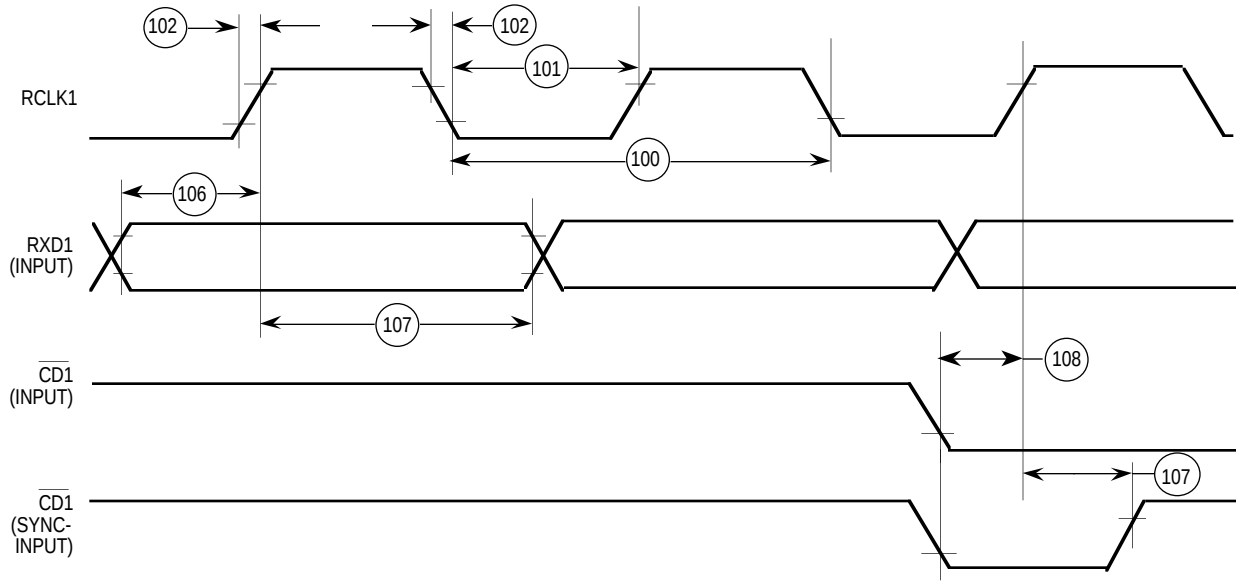


Figure 10-65. SCC NMSI Receive

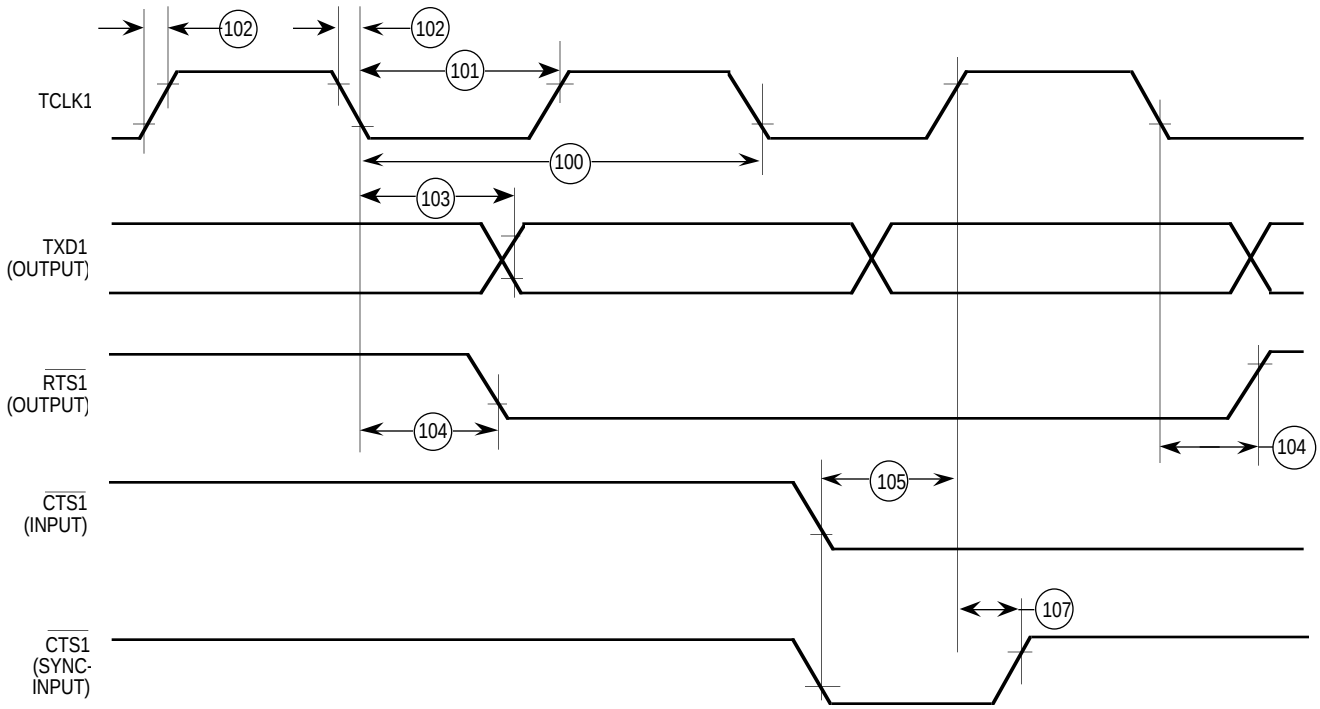


Figure 10-66. SCC NMSI Transmit

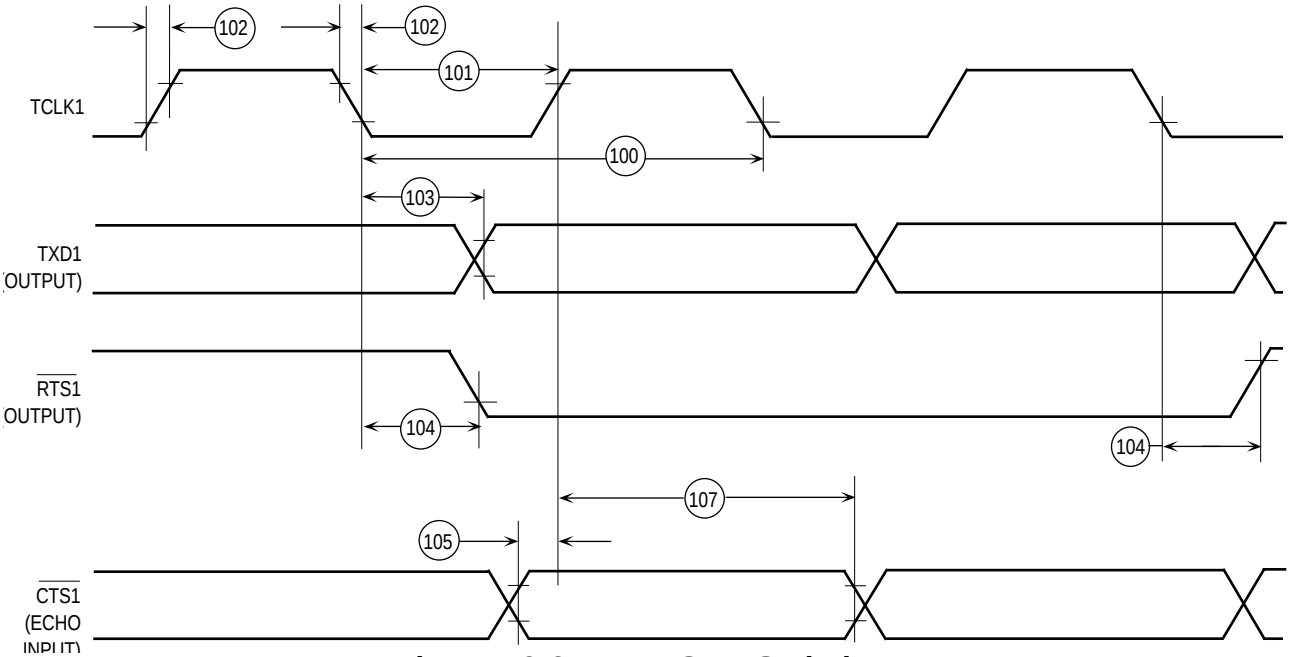


Figure 10-67. HDLC BUS Timing

10.26 ETHERNET ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-68–Figure 10-73)

Num.	Characteristic	3.3 V or 5.0 V		5.0 V		Unit
		25.0 MHz		33.34 MHz		
		Min	Max	Min	Max	
120	CLSN Width High	40	—	40	—	ns
121	RCLK1 Rise/Fall Time	—	15	—	15	ns
122	RCLK1 Width Low	CLKO1+ 5ns	—	CLKO1+ 5ns	—	
123[1]	RCLK1 Width high	CLKO1	—	CLKO1	—	
124	RXD1 Setup Time	20	—	20	—	ns
125	RXD1 Hold Time	5	—	5	—	ns
126	RENA Active Delay (from RCLK1 rising edge of the last data bit)	10	—	10	—	ns
127	RENA Width Low	100	—	100	—	ns
128	TCLK1 Rise/Fall Time	—	15	—	15	ns
129	TCLK1 Width Low	CLKO1+ 5ns	—	CLKO1+ 5ns	—	
130[1]	TCLK1 Width high	CLKO1	—	CLKO1	—	
131	TXD1 Active Delay (from TCLK1 rising edge)	10	50	10	50	ns
132	TXD1 Inactive Delay (from TCLK1 rising edge)	10	50	10	50	ns
133	TENA Active Delay (from TCLK1 rising edge)	10	50	10	50	ns
134	TENA Inactive Delay (from TCLK1 rising edge)	10	50	10	50	ns
135	RSTRT Active Delay (from TCLK1 falling edge)	10	50	10	50	ns
136	RSTRT Inactive Delay (from TCLK1 falling edge)	10	50	10	50	ns
137	RRJCT Width Low	1	—	1	—	CLKO1

138[2]	CLKO1 Low to SDACK Asserted	—	20	—	20	ns
139[2]	CLKO1 Low to SDACK Negated	—	20	—	20	ns

NOTES:
1The ratio SyncCLK/RCLK1 and SyncCLK/TCLK1 must be greater or equal to 2.25/1
2$\overline{\text{SDACK}}$ is asserted whenever the SDMA writes the incoming frame DA into memory.

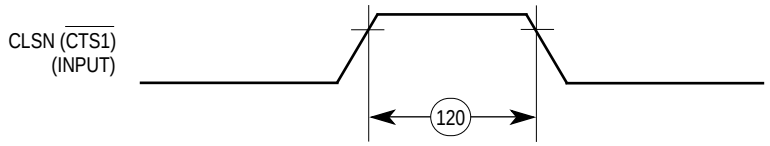


Figure 10-68. Ethernet Collision Timing

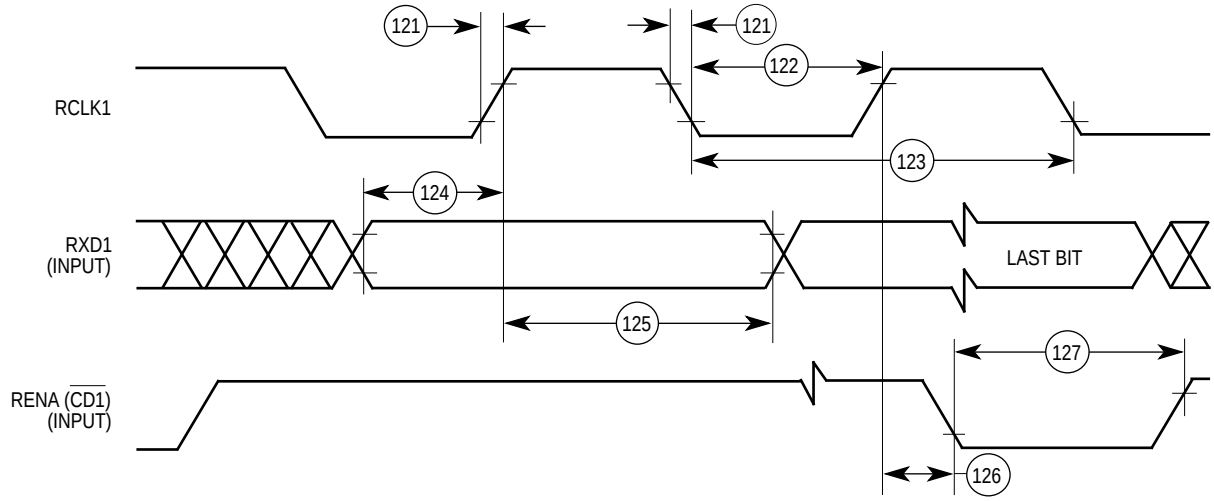
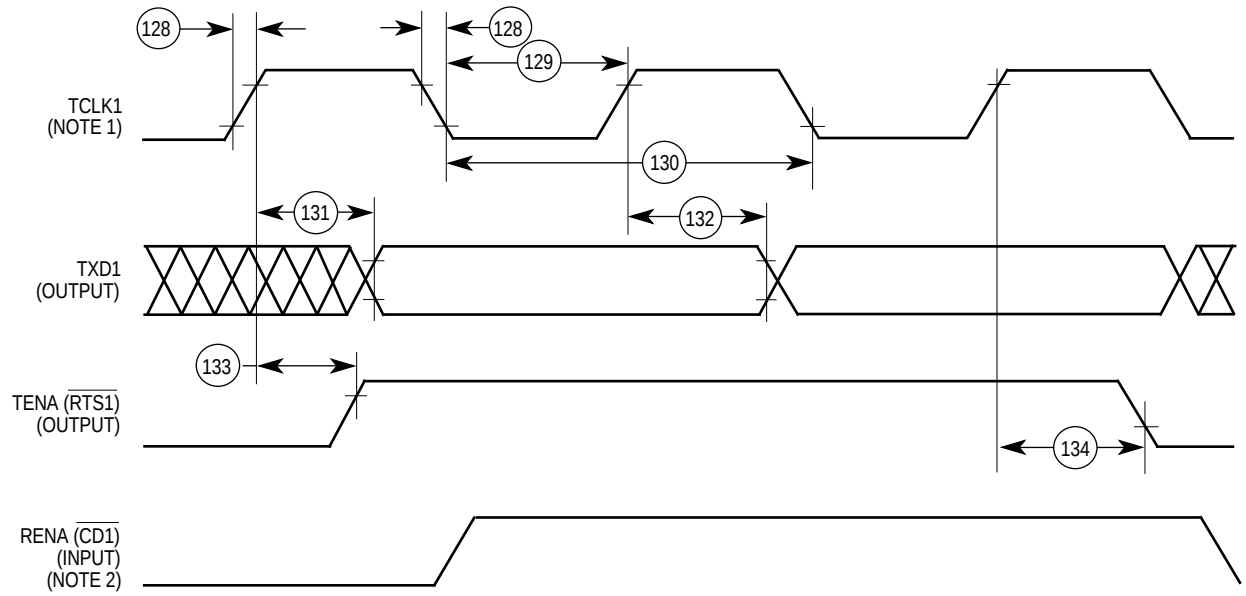
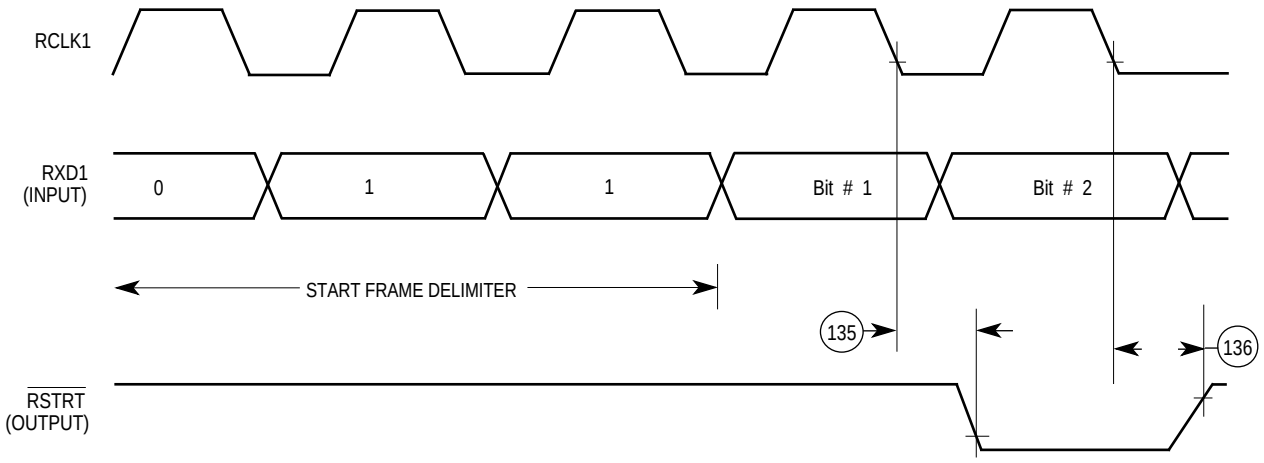


Figure 10-69. Ethernet Receive Timing



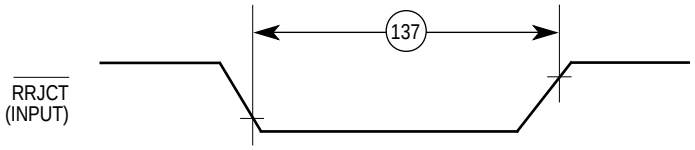
NOTES:
1. Transmit clock invert (TCI) bit in GSMR is set.
2. If RENA is deasserted before TENA, or RENA is not asserted at all during transit, then CSL bit is set in the buffer descriptor at the end of frame transmission.

Figure 10-70. Ethernet Transmit Timing



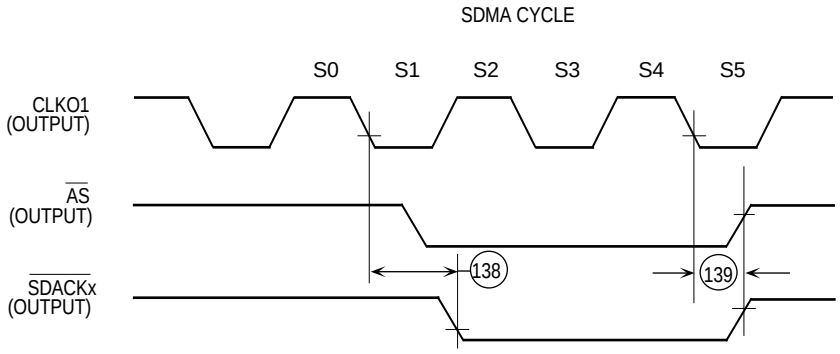
NOTE: Valid for the ethernet protocol only.

Figure 10-71. CAM Interface Receive Start Timing



NOTE: Valid for the ethernet protocol only.

Figure 10-72. CAM Interface Reject Timing



NOTE: SDACKx is asserted when the SDMA writes the received Ethernet frame into memory.

Figure 10-73. SDACK Timing Diagram

10.27 SMC TRANSPARENT MODE ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-74)

Num.	Characteristic	3.3 V or 5.0 V		5.0 V		Unit
		25.0 MHz		33.34 MHz		
		Min	Max	Min	Max	
150[1]	SMCLK Clock Period	100	—	100	—	ns
151	SMCLK Width Low	50	—	50	—	ns
151A	SMCLK Width High	50	—	50	—	ns
152	SMCLK Rise/Fall Time	—	15	—	15	ns
153	SMTXD Active Delay (from SMCLK falling edge)	10	50	10	50	ns
154	SMRXD/SYNC1 Setup Time	20	—	20	—	ns
155	SMRXD/SYNC1 Hold Time	5	—	5	—	ns

NOTES:
1.The ratio SyncCLK/SMCLK must be greater or equal to 2/1

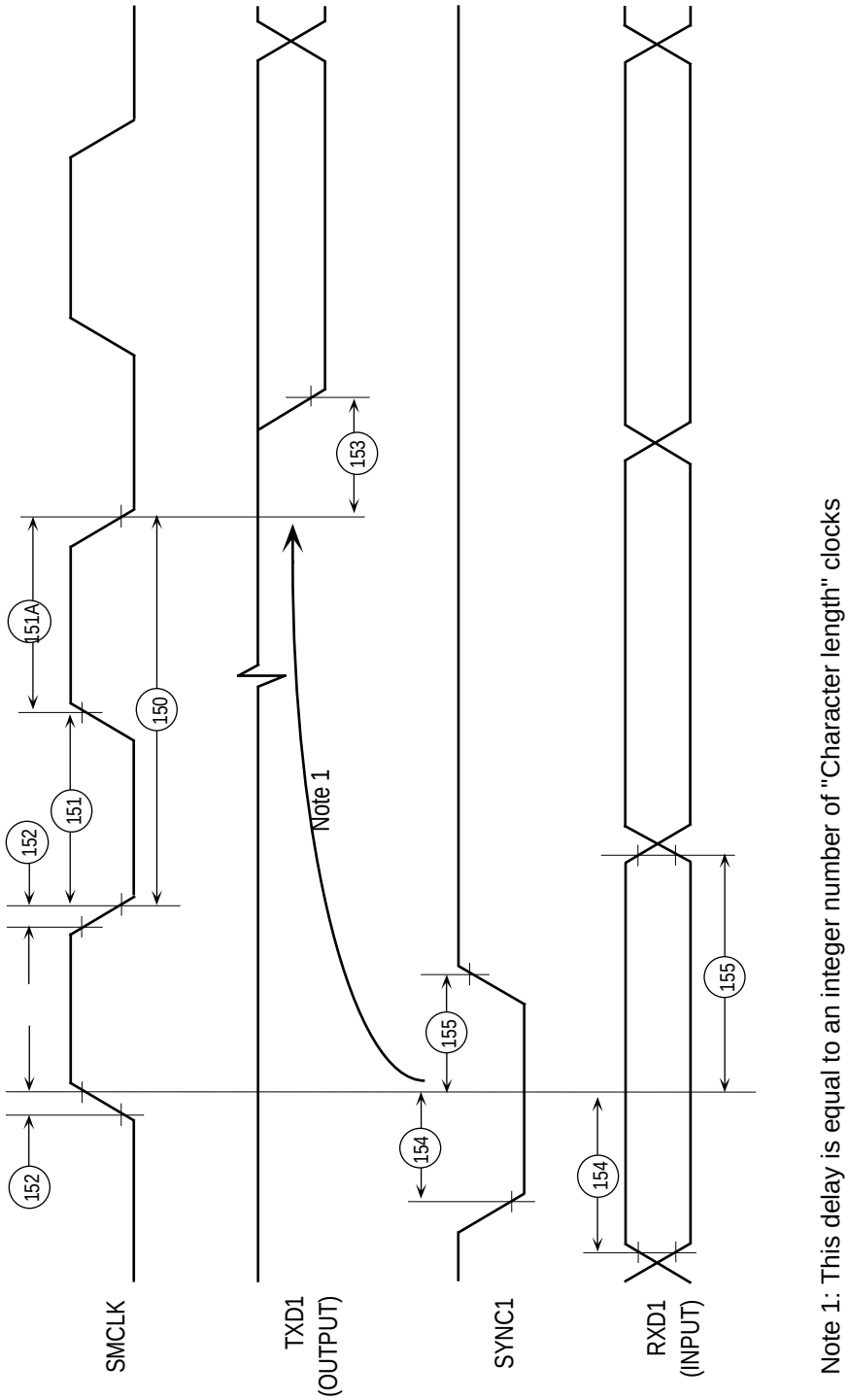


Figure 10-74. SMC Transparent

10.28 SPI MASTER ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-75 and Figure 10-76)

Num.	Characteristic	3.3 V or 5.0 V		5.0 V		Unit
		25.0 MHz		33.34 MHz		
		Min	Max	Min	Max	
160	Master Cycle Time	4	1024	4	1024	tcyc
161	Master Clock (SPICLK) High or Low Time	2	512	2	512	tcyc
162	Master Data Setup Time (Inputs)	50	—	50	—	ns
163	Master Data Hold Time (Inputs)	0	—	0	—	ns
164	Master Data Valid (after SPICLK Edge)	—	20	—	20	ns
165	Master Data Hold Time (Outputs)	0	—	0	—	ns
166	Rise Time: Output		15		15	s
167	Fall Time: Output		15		15	ns

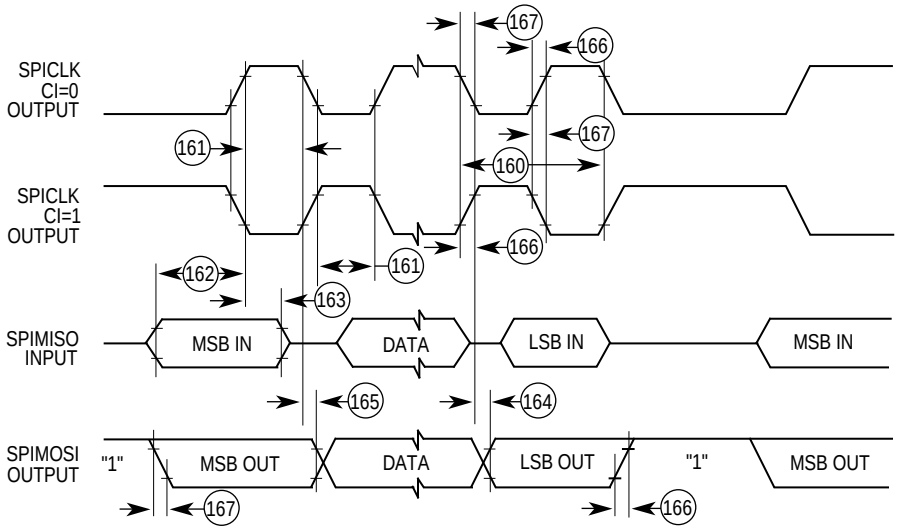


Figure 10-75. SPI Master (CP = 0)

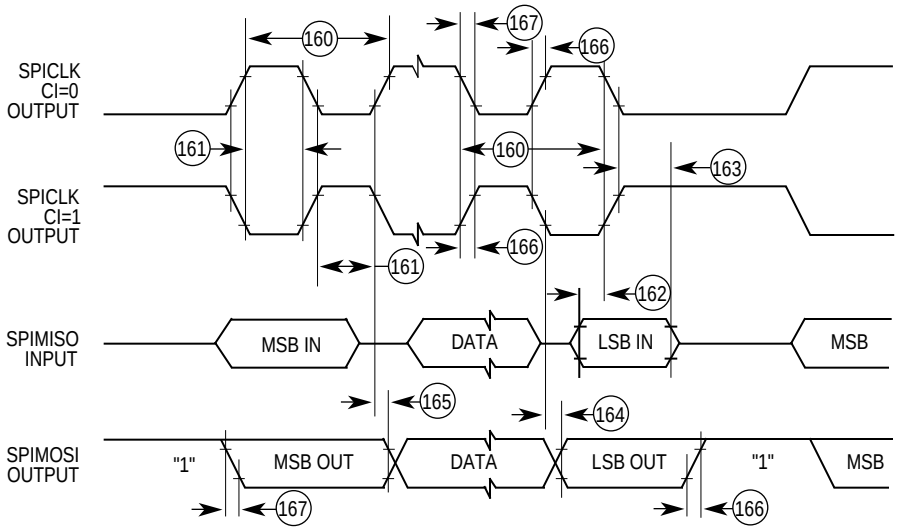


Figure 10-76. SPI Master (CP = 1)

10.29 SPI SLAVE ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-77 and Figure 10-78)

Num.	Characteristic	3.3 V or 5.0 V		5.0 V		Unit
		25.0 MHz		33.34 MHz		
		Min	Max	Min	Max	
170	Slave Cycle Time	2	—	2	—	tcyc
171	Slave Enable Lead Time	15		15		ns
172	Slave Enable Lag Time	15		15		ns
173	Slave Clock (SPICLK) High or Low Time	1	—	1	—	tcyc
174	Slave Sequential Transfer Delay (Does Not Require Deselect)	1		1		tcyc
175	Slave Data Setup Time (Inputs)	20	—	20	—	ns
176	Slave Data Hold Time (Inputs)	20	—	20	—	ns
177	Slave Access Time		50		50	ns
178	Slave SPIMISO Disable Time		50		50	ns
179	Slave Data Valid (after SPICLK Edge)	—	50	—	50	ns
180	Slave Data Hold Time (Outputs)	0	—	0	—	ns
181	Rise Time: Input		15		15	ns
182	Fall Time: Input		15		15	ns

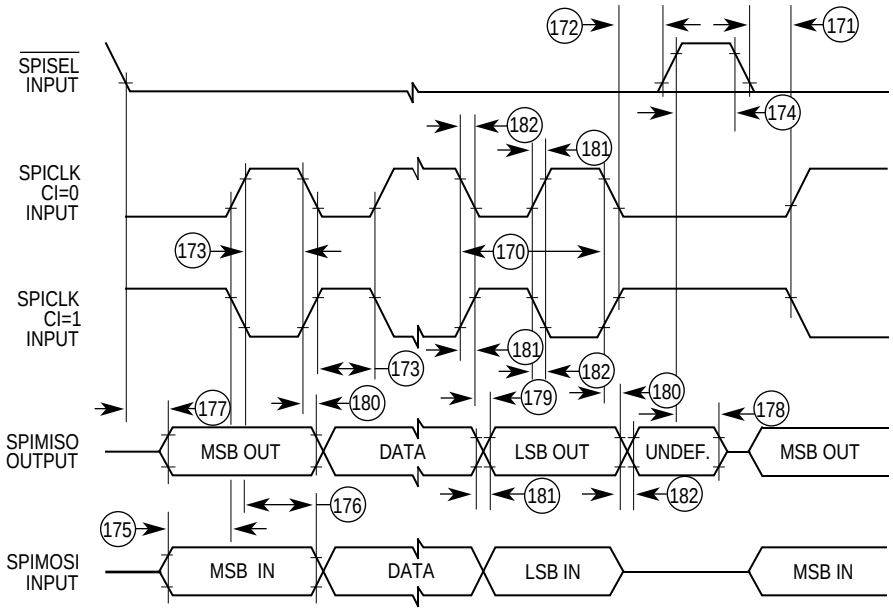


Figure 10-77. SPI Slave (CP = 0)

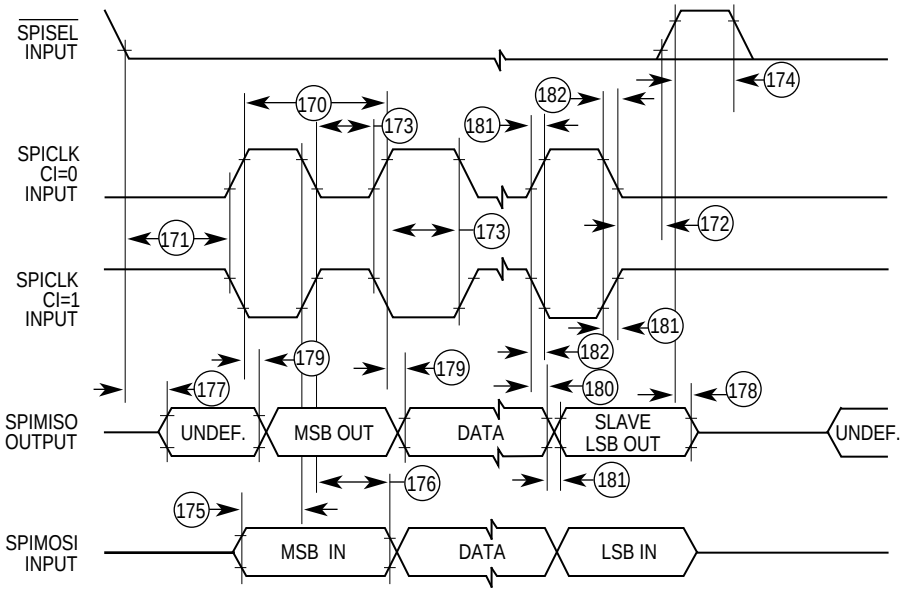


Figure 10-78. SPI Slave (CP = 1)

10.30 JTAG ELECTRICAL SPECIFICATIONS

(The electrical specifications in this document are preliminary. See Figure 10-79–Figure 10-82)

Num.	Characteristic	3.3 V or 5.0 V		5.0 V		Unit
		25.0 MHz		33.3MHz		
		Min	Max	Min	Max	
	TCK Frequency of Operation	0	25	0	25	MHz
1	TCK Cycle Time in Crystal Mode	40	—	40	—	ns
2	TCK Clock Pulse Width Measured at 1.5 V	18	—	18	—	ns
3	TCK Rise and Fall Times	0	3	0	3	ns
6	Boundary Scan Input Data Setup Time	10	—	10	—	ns
7	Boundary Scan Input Data Hold Time	18	—	18	—	ns
8	TCK Low to Output Data Valid	0	30	0	30	ns
9	TCK Low to Output High Impedance	0	40	0	40	ns
10	TMS, TDI Data Setup Time	10	—	10	—	ns
11	TMS, TDI Data Hold Time	10	—	10	—	ns
12	TCK Low to TDO Data Valid	0	20	0	20	ns
13	TCK Low to TDO High Impedance	0	20	0	20	ns
14	TRST Assert Time	100	—	100	—	ns
15	TRST Setup Time to TCK Low	40	—	40	—	ns

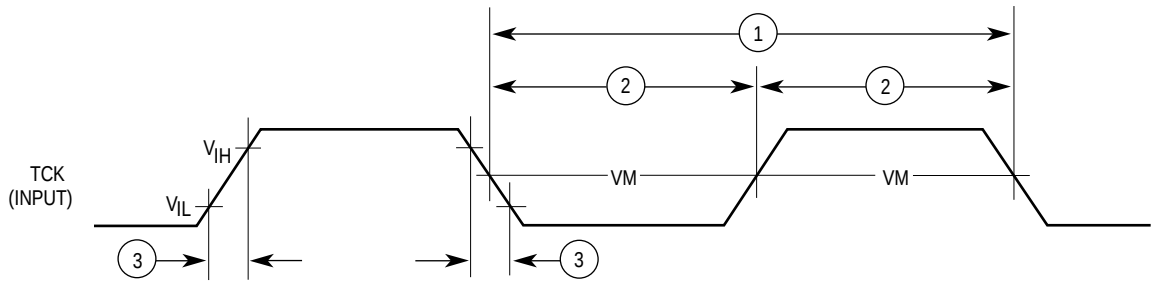


Figure 10-79. Test Clock Input Timing Diagram

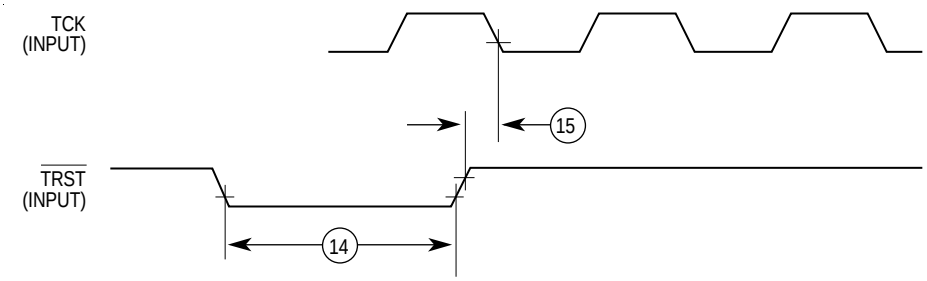


Figure 10-80. $\overline{\text{TRST}}$ Timing Diagram

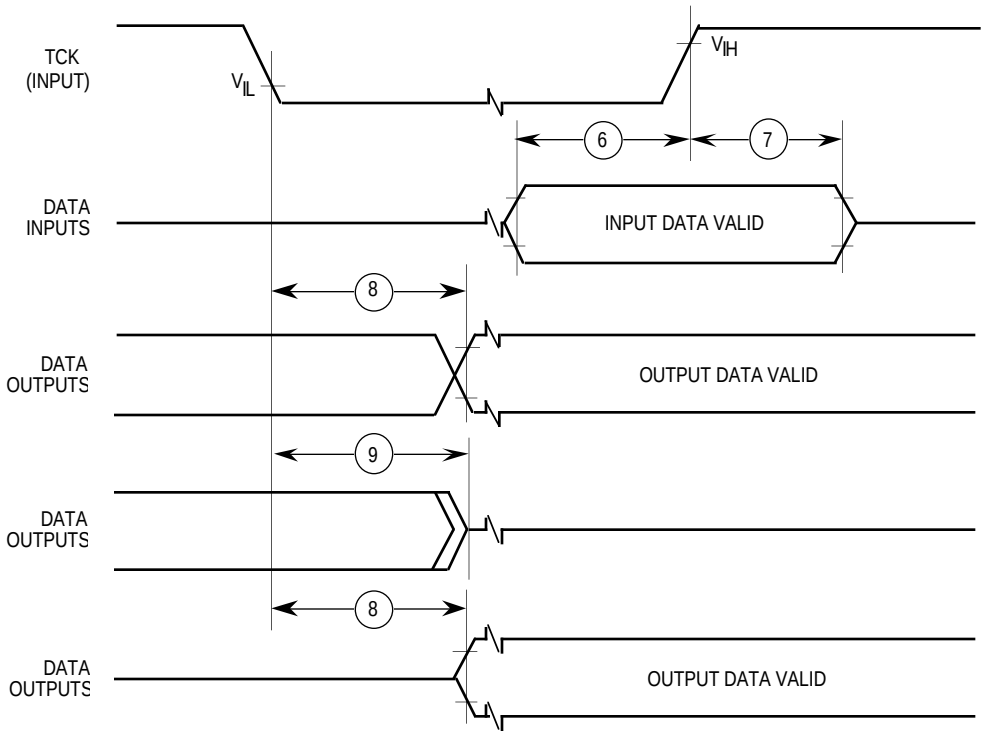


Figure 10-81. Boundary Scan (JTAG) Timing Diagram

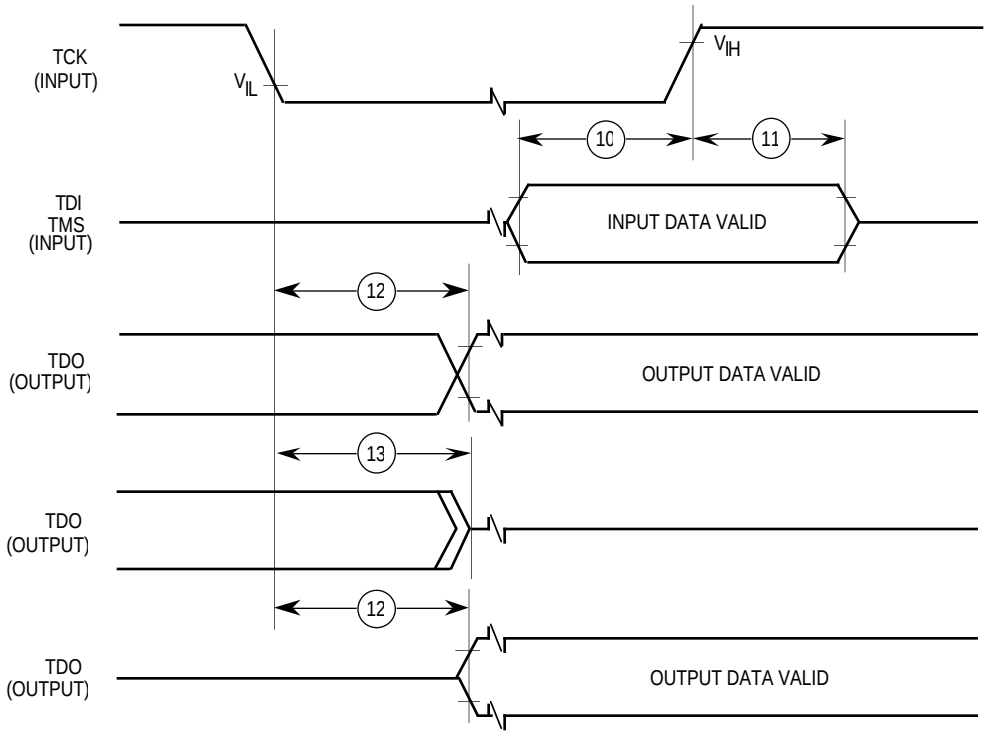


Figure 10-82. Test Access Port Timing Diagram

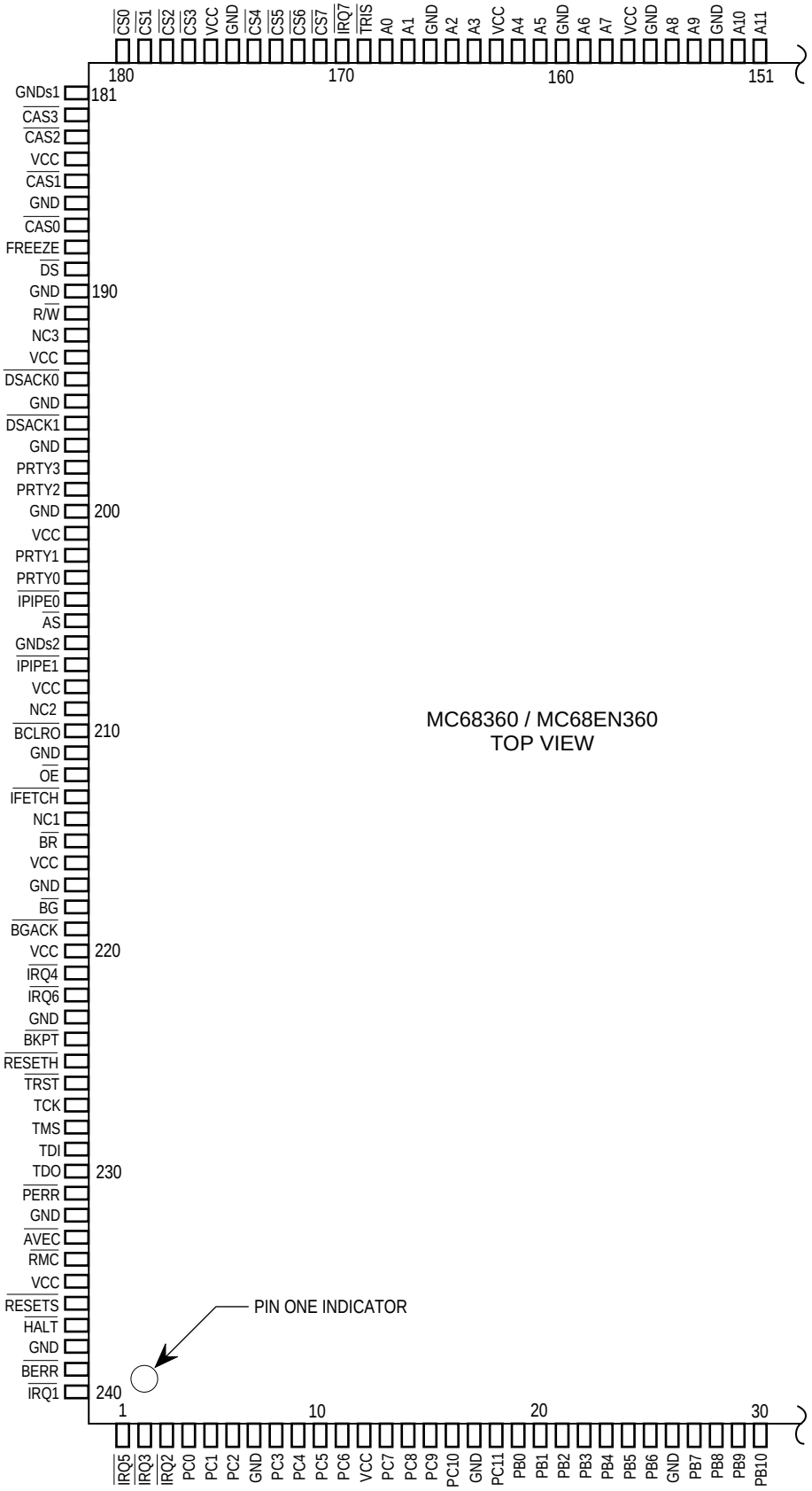
SECTION 11
ORDERING INFORMATION AND MECHANICAL DATA

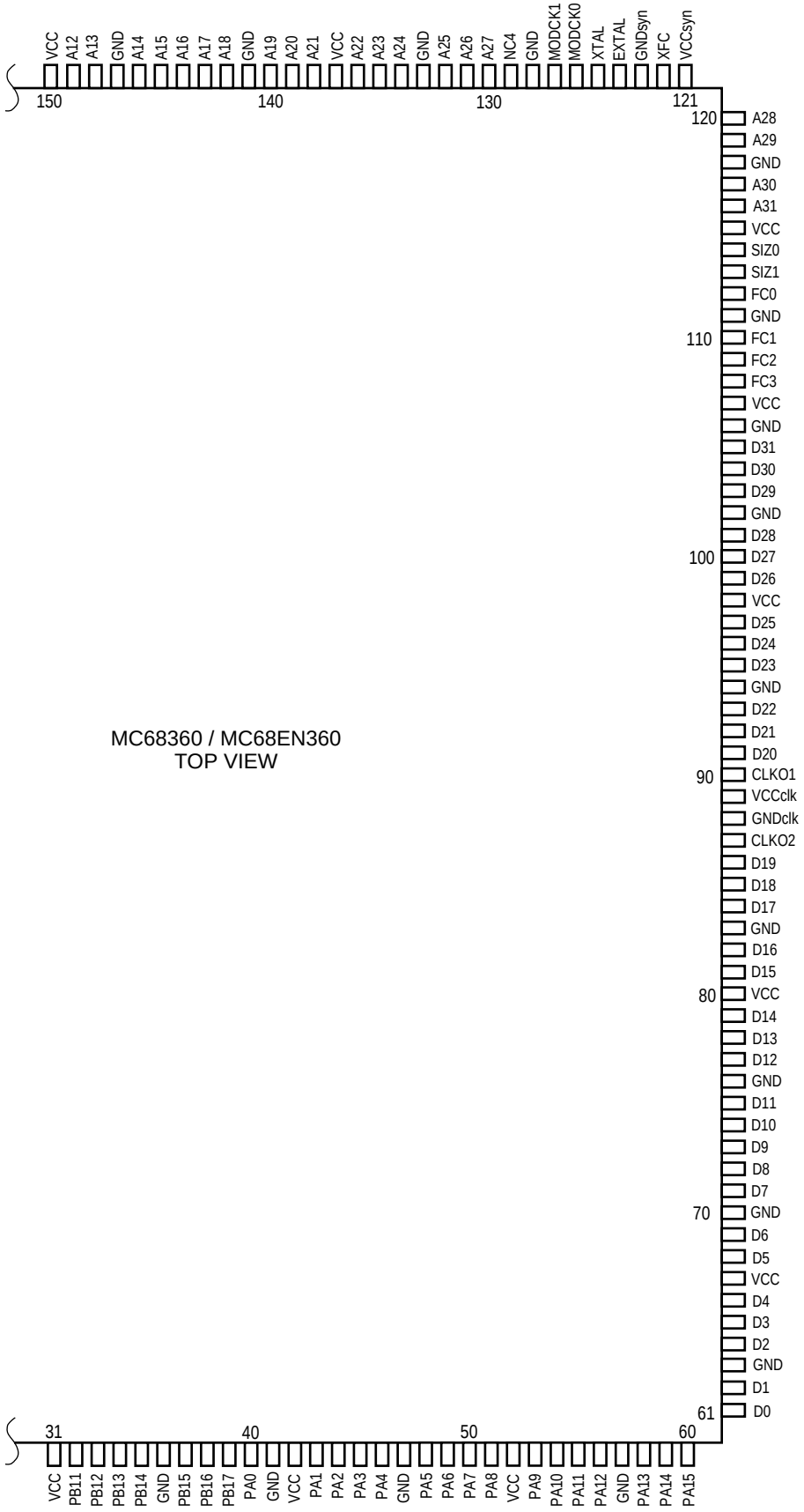
This section contains the ordering information, pin assignments, and package dimensions for the MC68360.

11.1 STANDARD ORDERING INFORMATION

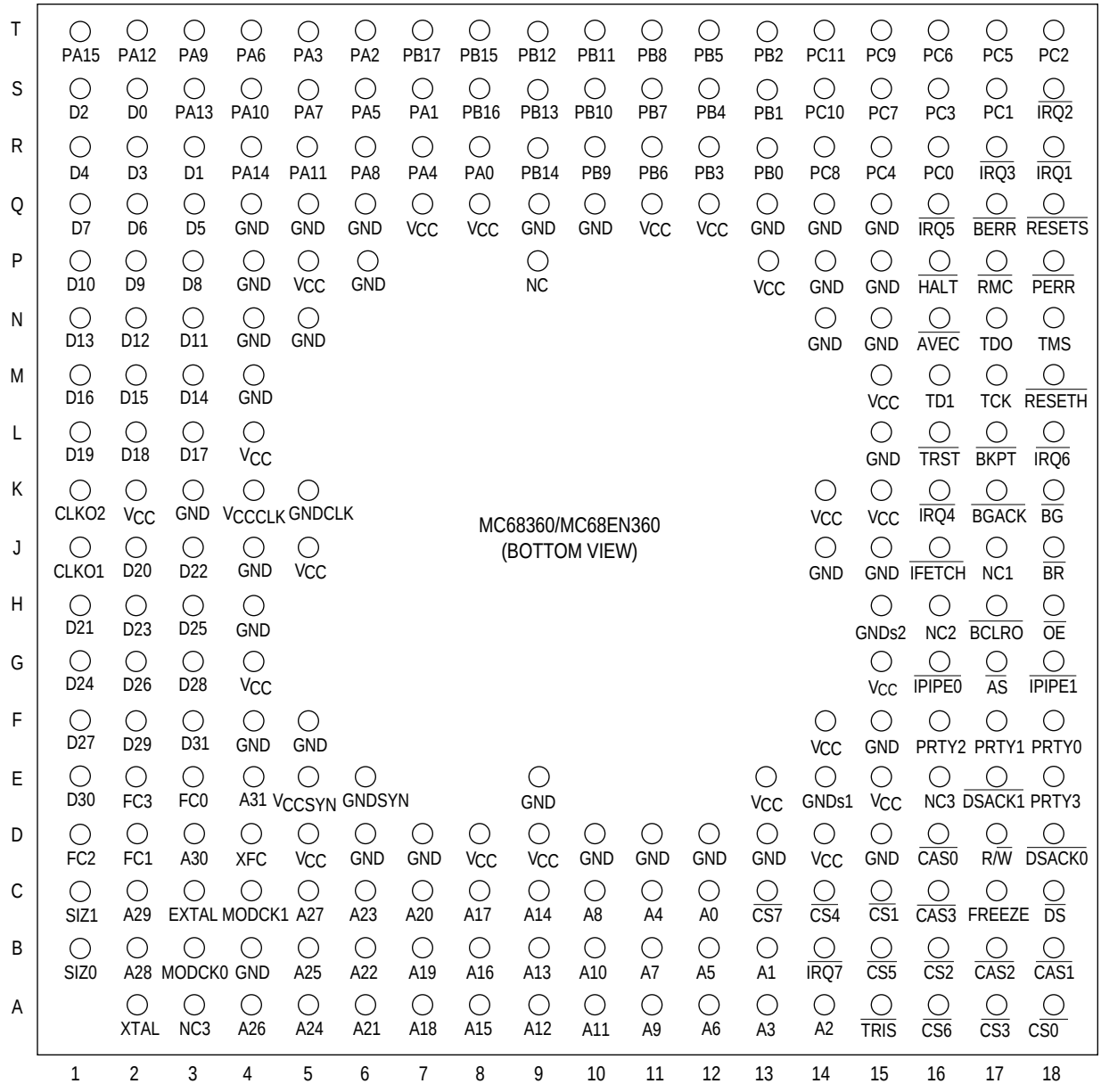
Package Type	Frequency (MHz)	Temperature	Order Number	Vcc
Quad Flat Pack (FE Suffix) Quad Flat Pack (FE Suffix) Quad Flat Pack with Ethernet Quad Flat Pack with Ethernet	0–25 0–25 0–25 0–25	0°C to 70°C –40°C to +85°C 0°C to 70°C –40°C to +85°C	MC68360FE25 MC68360CFE25 MC68EN360FE25 MC68EN360CFE25	5V +/- 5%
Pin Grid Array (RC Suffix) Pin Grid Array (RC Suffix) Pin Grid Array with Ethernet Pin Grid Array with Ethernet	0–25 0–25 0–25 0–25	0°C to 70°C –40°C to +85°C 0°C to 70°C –40°C to +85°C	MC68360RC25 MC68360CRC25 MC68EN360RC25 MC68EN360RC25	5V +/- 5%
Quad Flat Pack (FE Suffix) Quad Flat Pack with Ethernet	0–33 0–33	0°C to 70°C	MC68360FE33 MC68EN360FE33	5V +/- 5%
Pin Grid Array (RC Suffix) Pin Grid Array with Ethernet	0–33 0–33	0°C to 70°C 0°C to 70°C	MC68360RC33 MC68EN360RC33	5V +/- 5%
Ball Grid Array (ZP Suffix)) BallGrid Array with Ethernet	0–25 0–25	0°C to 70°C	MC68360ZP25 MC68EN360ZP25	5V +/- 5%
Quad Flat Pack (FE Suffix) Quad Flat Pack with Ethernet	0–25 0–25	0°C to 70°C	MC68360FE25V MC68EN360FE25V	3V +/- 0.3V
Pin Grid Array (RC Suffix) Pin Grid Array with Ethernet	0–25 0–25	0°C to 70°C	MC68360RC25V MC68EN360RC25V	3V +/- 0.3V

11.2 PIN ASSIGNMENT—240-LEAD QUAD FLAT PACK (QFP)



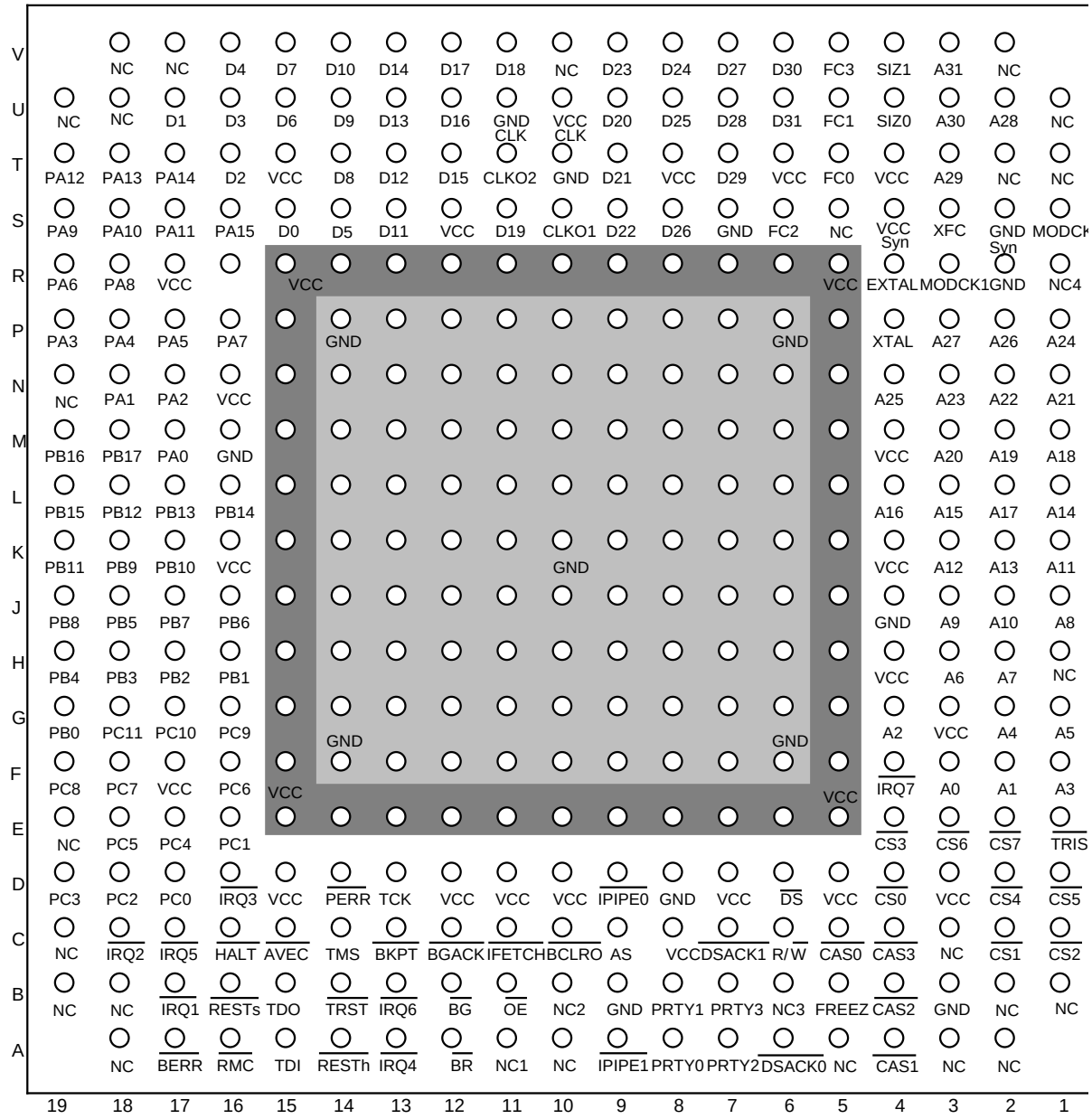

VCC A12 A13 GND A14 A15 A16 A17 A18 GND A19 A20 A21 VCC A22 A23 A24 GND A25 A26 A27 NC4 GND MODCK1 MODCK0 XTAL EXTAL GNDsyn XFC VCCsyn
150
140
130
121
120
A28
A29
GND
A30
A31
VCC
SIZ0
SIZ1
FC0
GND
110
FC1
FC2
FC3
VCC
GND
D31
D30
D29
GND
D28
100
D27
D26
VCC
D25
D24
D23
GND
D22
D21
D20
90
CLKO1
VCCclk
GNDclk
CLKO2
D19
D18
D17
GND
D16
D15
80
VCC
D14
D13
D12
GND
D11
D10
D9
D8
D7
70
GND
D6
D5
VCC
D4
D3
D2
GND
D1
61
D0
MC68360 / MC68EN360
TOP VIEW
31
40
50
60
VCC PB11 PB12 PB13 PB14 GND PB15 PB16 PB17 PA0 GND VCC PA1 PA2 PA3 PA4 GND PA5 PA6 PA7 PA8 VCC PA9 PA10 PA11 PA12 GND PA13 PA14 PA15

11.3 PIN ASSIGNMENT—241-LEAD PIN GRID ARRAY (PGA)



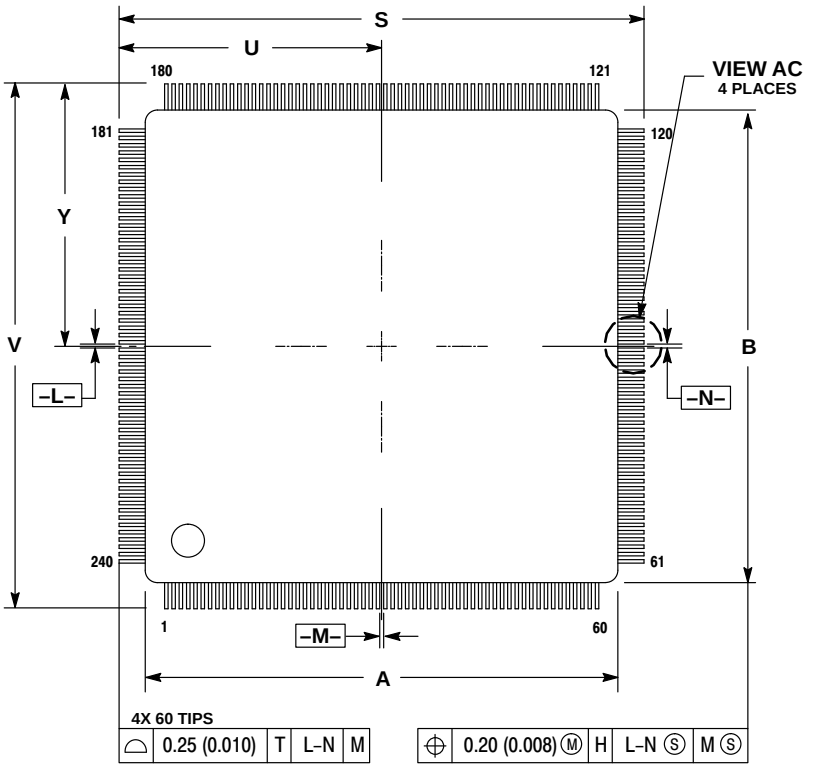
NOTE: Pin P9 "NC" is for guide purposes only.

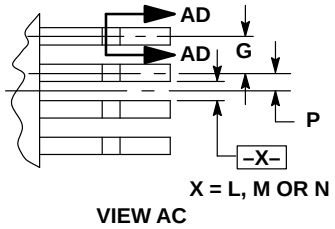
11.4 PIN ASSIGNMENT—357-LEAD BALL GRID ARRAY (BGA)



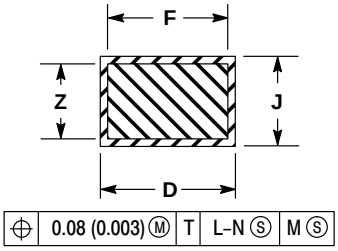
68360 OMPAC TOP VIEW
19 X 19 ARRAY 1.27mm PITCH

11.5 PACKAGE DIMENSIONS—CQFP (FE SUFFIX)

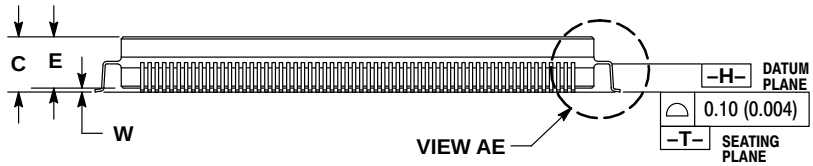


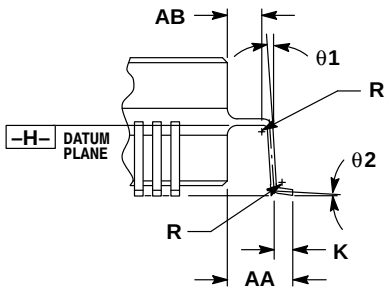


VIEW AC



SECTION AD
240 PLACES





VIEW AE

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE –H– IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE CERAMIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS –L–, –M–, AND –N– TO BE DETERMINED AT DATUM PLANE –H–.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE –T–.
6. DIMENSIONS A AND B DEFINE MAXIMUM CERAMIC BODY DIMENSIONS INCLUDING GLASS PROTRUSION AND TOP AND BOTTOM MISMATCH.

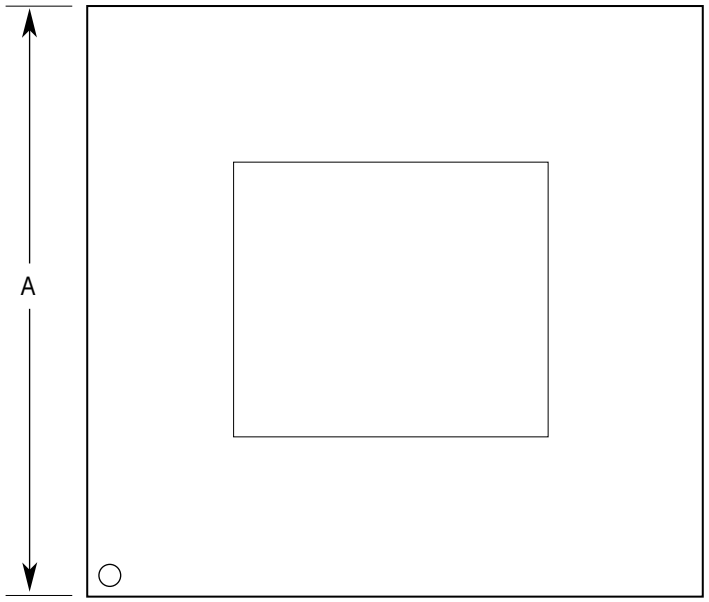
	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	30.86	31.75	1.215	1.250
B	30.86	31.75	1.215	1.250
C	3.75	4.15	0.148	0.163
D	0.18	0.30	0.007	0.012
E	3.10	3.90	0.122	0.154
F	0.17	0.23	0.007	0.009
G	0.50 BSC		0.019 BSC	
J	0.13	0.175	0.005	0.007
K	0.45	0.55	0.018	0.021
P	0.25 BSC		0.010 BSC	
R	0.15 BSC		0.006 BSC	
S	34.60 BSC		1.362 BSC	
U	17.30 BSC		0.681 BSC	
V	34.60 BSC		1.362 BSC	
W	0.25		0.010	
Y	17.30 BSC		0.681 BSC	
Z	0.12	0.13	0.005	0.005
AA	1.80 REF		0.071 REF	
AB	0.95 REF		0.037 REF	
θ1	2°	6°	2°	6°
θ2	1°	7°	1°	7°

CASE 988-01
ISSUE D

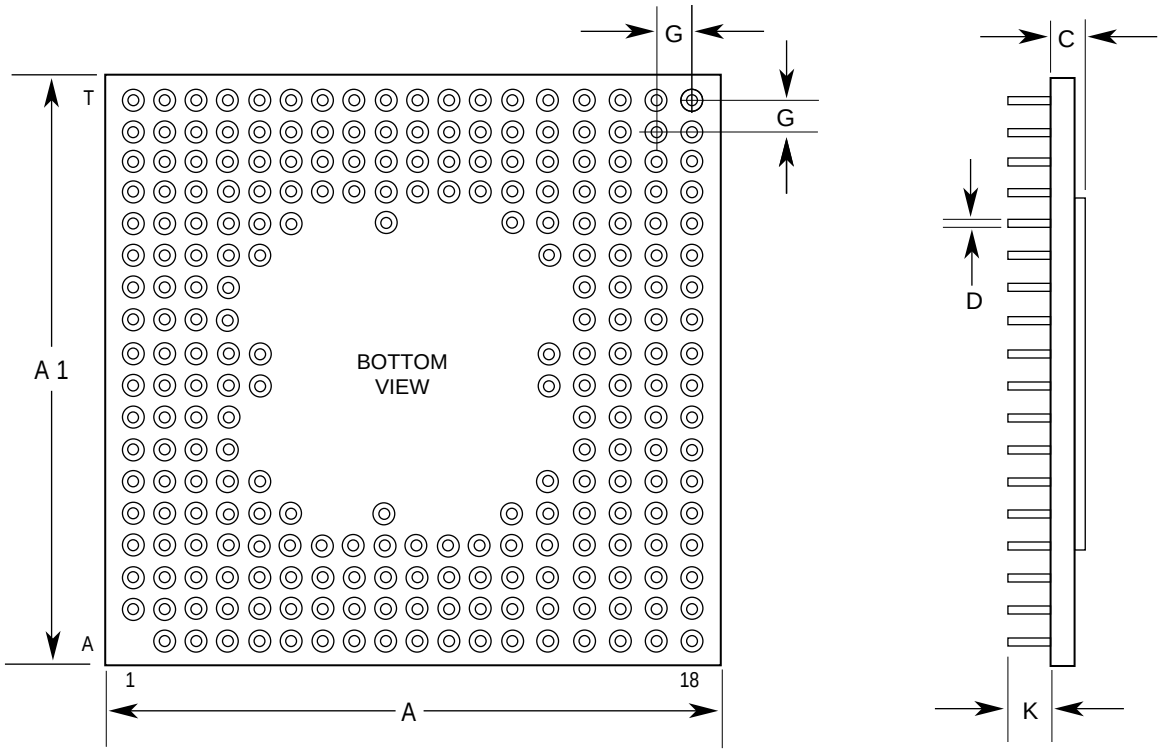
DATE 03/16/95

11.6 PACKAGE DIMENSIONS—PGA (RC SUFFIX)

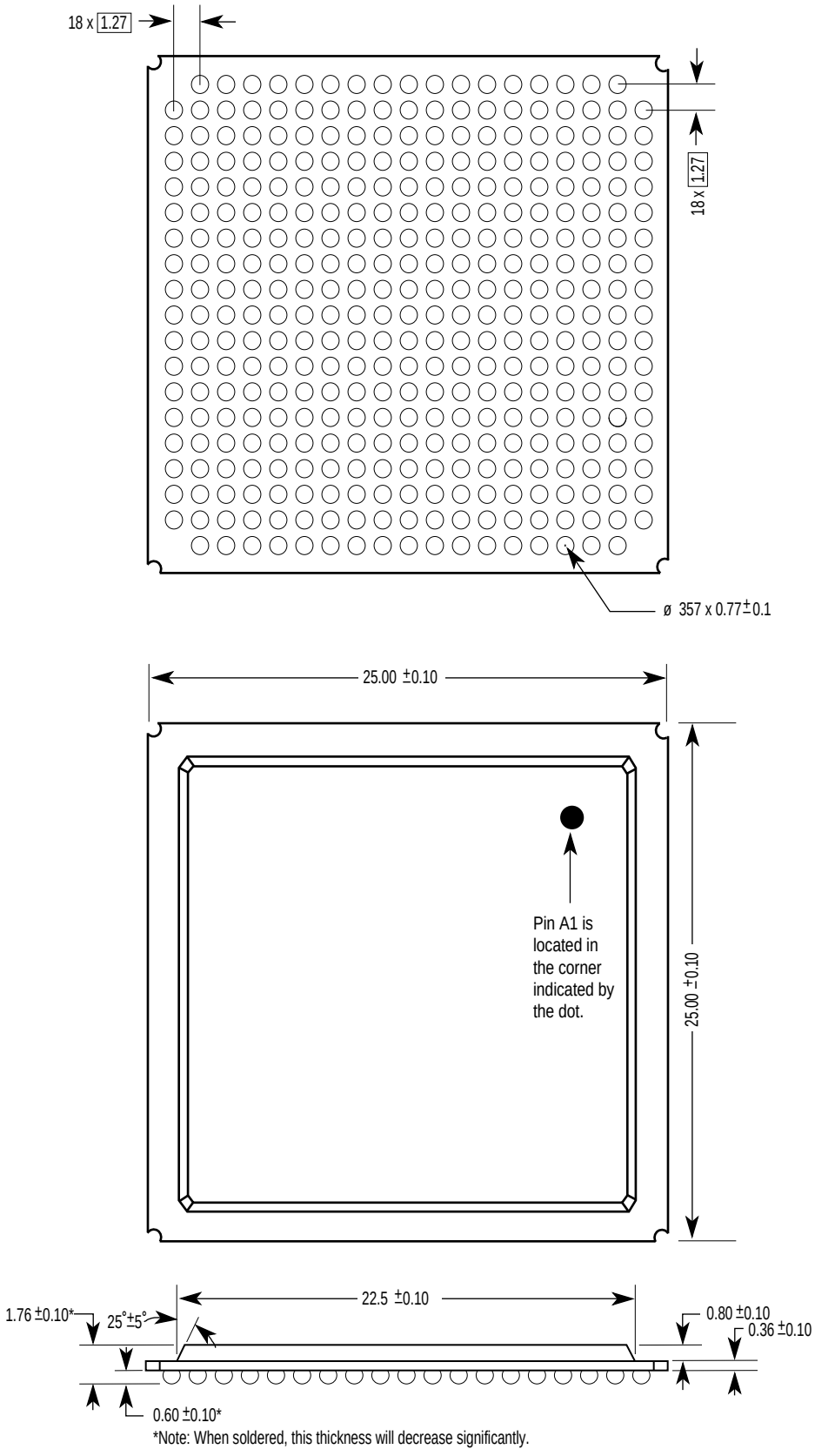
RC SUFFIX
241 LEAD PGA CASE
PRELIMINARY



DIM	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	1.840	1.880	46.74	47.75
C	0.110	0.140	2.79	3.56
D	0.016	0.020	0.41	0.51
G	0.100 BASIC		2.54 BASIC	
K	0.150	0.170	3.81	4.32



11.7 PACKAGE DIMENSIONS—BGA (ZP SUFFIX)



*Note: When soldered, this thickness will decrease significantly.

APPENDIX A
SERIAL PERFORMANCE

The QUICC at 25 MHz was designed to support unrestricted operation of the high-level data link control (HDLC) or transparent protocol running on four serial communications controllers (SCCs) simultaneously at 2.048 Mbps. The QUICC can also support one Ethernet channel at 10 Mbps and three HDLC or transparent channels at 2.048 Mbps.

The physical clocking limit of the SCCs is higher than the sustained serial bit rate. This limit is given as a 1:2.25 ratio between the sync clock (a clock generated in the clock synthesizer that can be as fast as the 25-MHz system clock) and the serial clock. For example, with a sync clock of 25 MHz, the SCCs may be clocked at 11.1 MHz. This clocking scheme allows for high-speed bursts of data bits to be handled by the SCCs for short periods of time, subject to the FIFO sizes.

When the SCCs are connected to a time-division multiplexed channel using the time-slot assigner present on the QUICC, the SCC physical clocking limit is a 1:2.5 ratio between the sync clock and the serial clock. Therefore, the SCCs may be connected to a 10.0- MHz time-division multiplexed channel with a 25-MHz QUICC. This clocking scheme allows for high-speed bursts of data bits to be handled by the SCCs for short periods of time, subject to the FIFO sizes.

Other devices that offer higher HDLC performance than the QUICC are the Motorola MC68605 1984 CCITT X.25 LAPB controller and the MC68606 CCITT Q.921 multilink LAPD controller. The MC68605 and MC68606 perform the full data-link layer protocol as well as support various transparent modes within HDLC-framed operation at speeds of at least 10 Mbps.

The performance figures listed in Table A-1 are for a 25-MHz system clock only.

High-Speed Channels		Low-Speed Channels		Comments/Restrictions
Number	**Speed**	**Number**	**Speed**	
1 HDLC	8 Mbps	—	—	
2 HDLC	4 Mbps	—	—	
3 HDLC	2.6 Mbps	—	—	
4 HDLC	2.05 Mbps	—	—	
1 ENET	10 Mbps	3 HDLC	2.05 Mbps	Minor restrictions on HDLC buffer length
1 ENET	10 Mbps	2 HDLC	2.05 Mbps	
2 ENET	10 Mbps	1 HDLC	1 Mbps	
4 UART	625 kbps	—	—	Async SCC
4 UART-S	625 kbps	—	—	Sync UART SCC
4 BISYNC	460 kbps	—	—	
1 TRAN	8 Mbps	—	—	
2 TRAN	4 Mbps	—	—	
3 TRAN	2.6 Mbps	—	—	
4 TRAN	2.05 Mbps	—	—	
2 TRAN	1.56 Mbps	—	—	SMC Channels
2 UART	100 kbps	—	—	SMC Channels

NOTES:

1. These numbers are estimates generated prior to silicon. Additional work will be performed to improve the accuracy of the SCC performance numbers and will be reported in later revisions of this manual.
2. All performance calculations assume a 25-MHz system clock and sync clock for the SCCs. The results scale linearly with different system clock speeds. A change in the sync clock will not affect performance as long as the required 1:2.25 or 1:2.5 ratio is maintained.
3. User should consult with receive and transmit clock timing of the SIA to ensure clock pulse width high and width low are satisfied for high speed serial communications.
4. In all cases, the numbers assume data is stored in external system RAM.
5. The performance is not expected to degrade based on the number of wait states in the system, as long as the system RAM has between 0 and 9 wait states.
6. The performance table is also applicable when the time-slot assigner on the QUICC is used.
7. When the performance of a high-speed channel together with a low-speed channel was measured, the high-speed channel was always SCC1.
8. Performance in *HDLC bus* mode is the same as HDLC performance, for both full duplex and half duplex operation. (Half duplex is the normal configuration of HDLC bus mode, thus HDLC half duplex performance numbers would normally be used.)
9. All results assume that the other RISC features are not operating—i.e., the RISC timer tables, the IDMA auto buffer and buffer chaining modes, the parallel interface port, and the other serial channels. If these features are operating, performance may be slightly reduced. The DRAM refresh controller has no effect on the performance.
10. Except for Ethernet, all table results assume continuous full-duplex operation. Results for half-duplex operation are roughly 2x better.
11. The SMC performance results are without the SCCs operating; otherwise, SMC performance may be reduced. Although the exact SMC performance that can be obtained is determined by many RISC utilization factors and is therefore difficult to estimate, it is expected that 9.6 kbps on both SMC UARTs could be simultaneously supported in most situations.

APPENDIX B
DEVELOPMENT TOOLS AND SUPPORT

Several software development packages are offered as a set of independent modules that provide the following features:

- QUICC Chip Evaluation
- By running the modules on the development board described in B.4 M68360QUADS Development System, it is possible to examine and evaluate the QUICC. Symbolic, user-friendly menus allow control of all parts of the QUICC.
- QUICC Simple Drivers
- Written in C, the source code of the QUICC drivers is available on the Motorola Freeware Bulletin Board (512-891-3733, 8 data, no parity, 1 stop) or through Anonymous FTP to freeware.aus.sps.mot.com under /pub/mcu360. These drivers were developed to support the needs of the general user. Although they are based on the regular QUICC drivers, they provide call-oriented interfaces and self-contained QUICC initialization routines and interrupt handling for a given protocol.
- QUICC Chip Drivers
- Written in C, the source code of the QUICC drivers is available on electronic media. These drivers were developed to support the needs of the protocol implementations.
- Protocol Implementations
- Modules implementing common ISO/OSI layer 2 and 3 protocols are available under license. These are in the form of source code written in C.
- Portability
- All of the higher layer software modules may be ported from source to different QUICC implementations and combined with user-developed code. Software interface documentation is available for all provided modules.

B.1 MOTOROLA SOFTWARE MODULES

Chip driver routines written in C illustrate initialization of the QUICC, interrupt handling, and the management of data transmission and reception on all channels.

In addition to the chip drivers, protocol modules are provided. Layer 2 modules include LAPB and LAPD. The layer 3 module is the X.25 packet layer protocol.

Since the modules require some minimal operating system services, the EDX operating system kernel is provided. EDX is the kernel implemented on the QUADS board (see B.4 M68360QUADS Development System). Use of EDX with the protocol modules is not required as long as some other operating system support is provided by the user.

All modules communicate with each other using message passing. This technique simplifies the interfaces between the different modules and enables them to interface with other user-added modules with relative ease. Descriptions of the software interfaces are available.

Each module operates completely independent of its environment. Independence from the hardware environment is achieved by the fact that each module is individually configurable and relocatable anywhere in system memory. Calls are used for requesting resources or services from the resident operating system (memory management and message passing), which creates independence from the firmware. Modules may be used with any combination of other modules, including those added by the user (see Figure B-1).

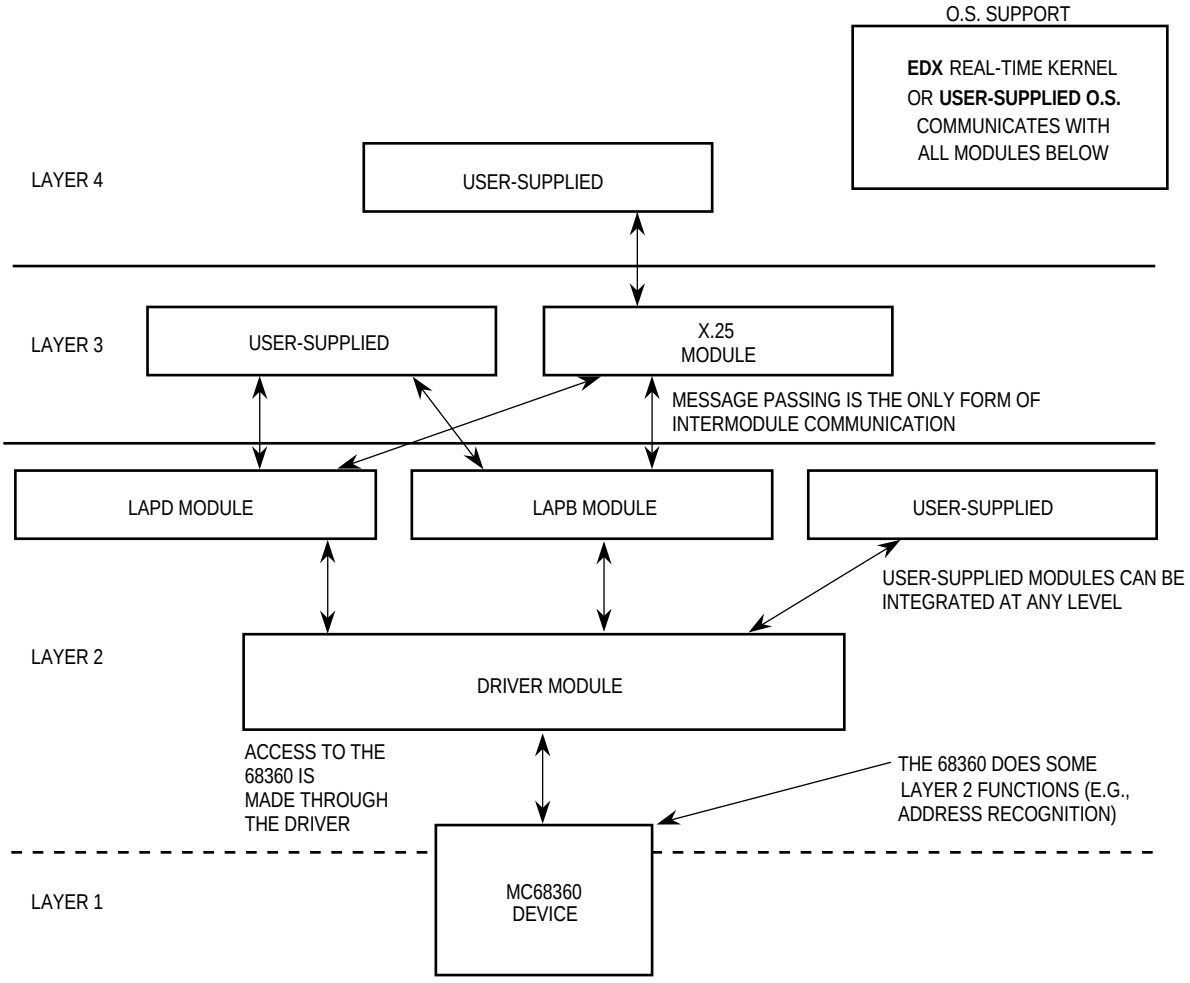


Figure B-1. Software Overview

The chip driver module and simple driver module features are as follows:

- Provides all software modules with an interface to the QUICC
- Illustrates QUICC initialization and interrupt handling
- Provides complete configuration of the QUICC on system start (or restart)
- Provides services for the QUICC serial ports

  - — Supports linked lists of frames for both transmit and receive
  - — Provides error handling and recovery for both transmit and receive
  - — Supports all serial ports in different protocol configurations
  - — Provides internal loopback (frames not sent to QUICC)
- Provides timer services
- Can be configured to send trace messages to a system log file
- Supports up to 24 serial ports

The LAPD module features are as follows:

- Fully implements 1988 CCITT Recommendation Q.920/Q.921
- Supports up to 12 physical channels (8192 logical links per channel)
- Supports management and broadcast links
- Supports user and network applications
- Uses dedicated transmit pool for fast control frame generation
- Dynamic modification of protocol parameters
- Independent of layer 1 and layer 3 implementation
- Independent of layer 2 management
- Message-oriented interface
- Independent configuration of upper and lower layer modules interfacing with each LAPD link
- Special mode for internal loopback between pairs of links
- Supports external loopback between two QUICC serial channels or on the same serial channel
- Trace option for reporting to system management each primitive issued by the LAPD module to the layer 3 or layer 2 management

The LAPB module features are as follows:

- Fully implements 1988 CCITT Recommendation X.25, chapters 2.1–2.4
- Supports up to 12 distinct physical channels with each operating as an independent station
- Modulo 8 or 128 operation
- Applicable for DTE and DCE applications
- Uses dedicated transmit pool for fast control frame generation
- Dynamic modification of protocol parameters
- Independent of layer 1 and layer 3 implementation
- Message-oriented interface
- Independent configuration of upper and lower layer modules interfacing with each LAPB link
- Special mode for internal loopback (frames are not sent to the driver)

- Supports external loopback between two QUICC serial channels or on the same serial channel
- Trace option for reporting to system management each primitive issued by the LAPB module to layer 3 or layer 2 management

The X.25 module features are as follows:

- Fully implements 1988 CCITT Recommendation X.25, chapters 3.1–7.3
- May be used with both layer 2 modules: LAPD or LAPB
- Unlimited number of layer 2 interfaces
- Supports up to 4095 logical channels for each interface
- Many DTE/DCE interface parameters (configurable for each interface):

  —DTE/DCE
  —Modulo 8 or 128 operation
  —Window size
  —Maximum receive and transmit packet lengths
- Layer 4 message fragmentation/assembly using M-BIT
- Q-BIT support
- All standard CCITT X.25 facilities
- Compatible with X.213 interface primitives
- Logical channel parameters (configurable for each interface):

  —Interface ID
  —DTE/DCE
  —Permanent virtual circuit/virtual call
  —D-BIT support
  —On-line registration support
  —TOA/NPI address mode
  —Fast select facility support
  —Logical channel (and group) numbers
  —Protocol parameters (w, T11, T12, T21, T22, N12, N13)
  —Maximum data packet length (unlimited)
- Message-oriented interface
- Independent of layer 2 and layer 4 implementation

The EDX module features are as follows:

- The EDX event-driven executive is an operating system kernel that provides:

  —Multitasking with simple task scheduling
  —Message passing between tasks
  —Memory allocation
- EDX was designed to be:

  —Fast (written mostly in M68000-family assembler)
  —Efficient (uses approximately 1.5 Kbytes of ROM)

- EDX was designed for use in communications environments. It provides "soft" real-time scheduling, not the "hard" real-time scheduling needed in many event control applications.

The approximate compiled object-code sizes are as follows:

1. Chip Drivers 35K
2. LAPD 35K
3. LAPB 30K
4. X.25 60K
5. EDX 2K

Object-code sizes may be reduced from these figures if optional module features are not compiled. A RAM scratchpad area (around 4 Kbytes) and buffer space are also required for the modules (except for EDX).

An additional user-interface module is used on the QUADS board. The user-interface module, which supports a window-based host program running on the host computer, interfaces to the QUICC and each functional protocol module. This module may be used for chip evaluation or as a tool for debugging user-developed applications. Menu options allow the user to issue specific commands or to examine the appropriate module's memory structures (or the QUICC register set and on-chip dual-port RAM). The commands may result in specific QUICC commands or in the execution of protocol-defined primitives in one of the protocol modules.

The chip driver module is available in C-language source-code form and is included in object-code form on the QUADS board. The source code requires no license.

The LAPB, LAPD, and X.25 modules are available in C-language source-code form and are also included in object-code form on the QUADS board. The code requires a license from Motorola.

NOTE

If the chip driver, LAPB, LAPD, or X.25 modules are to be used on the QUADS board, an ADI board is also required to be able to download register settings from your PC or SUN system.

The EDX kernel is available in C and assembler source-code form and is included in object-code form on the QUADS board. The source code requires a license from Motorola.

Contact the local Motorola sales office for license terms.

B.2 OTHER PROTOCOL SOFTWARE SUPPORT

Third parties also offer additional protocols that may be used with the QUICC. They include ISDN protocols such as Q.931 and general communication protocols such as TCP/IP. Contact the local sales representative for information on third-party protocols.

B.3 THIRD-PARTY SOFTWARE SUPPORT

Since the QUICC is a memory-mapped device based on a CPU32+ core, existing compilers, source-level debuggers, assemblers, and linkers designed for the CPU32 family may be successfully used. Many third parties supply such tools as well as software tools specifically designed to work with the QUADS board.

NOTE

The CPU32+ enhancements over the CPU32 are compatible with existing CPU32 software tools.

B.4 M68360QUADS DEVELOPMENT SYSTEM

The M68360 QUICC Application Development System (M68360QUADS) is a standalone board that includes the previously discussed software modules, a board-level kernel (EDX), and a monitor/debugger (CPU32BUG). It is a flexible yet lower cost alternative to full in-circuit emulation. It also provides an immediate platform for developing and testing hardware and software.

The QUADS board is a follow-on to the MC68302 ADS board. Users of the MC68302 ADS board will find the QUADS to be similar in approach and flexibility.

The QUADS board consists of a master QUICC (full Ethernet version MC68EN360), memory (1 Mbyte of DRAM SIMM with parity, 512 Kbytes of flash EPROM, 128 Kbytes of boot EPROM, and 256 bytes of EEPROM), and a slave QUICC (full Ethernet version MC68EN360) to implement serial (RS-232) and parallel (ADI) ports needed by the QUADS board. This configuration leaves all resources of the master QUICC available to the user (see Figure B-2).

The flash EPROM has on-board programming capabilities that allow the user to program their own software on the board or update the software provided by Motorola.

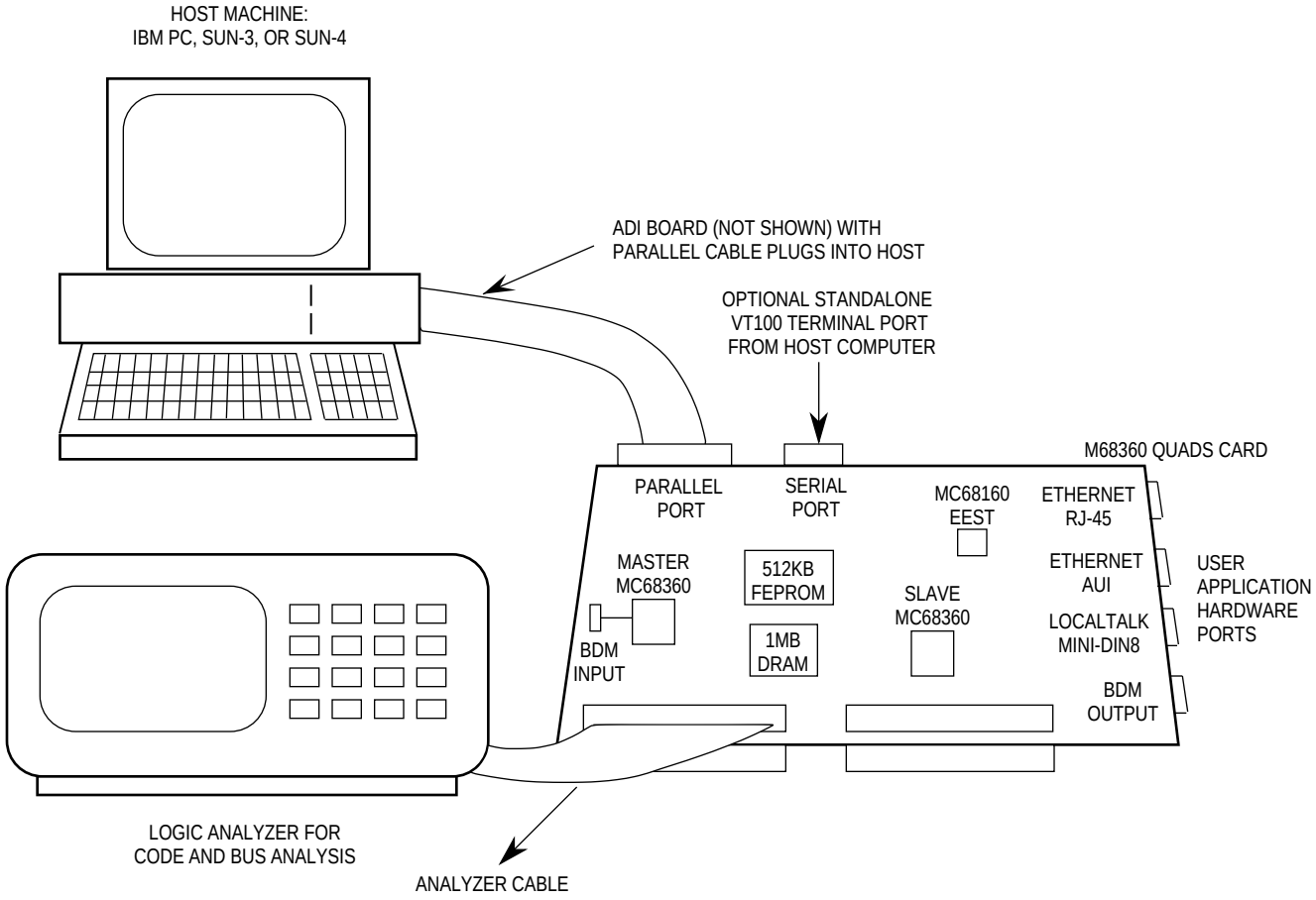


Figure B-2. QUADS System

In addition, the QUADS board provides the hardware connection for a BDM input connection to the master QUICC so that third-party BDM software may be used to control the QUADS board.

The user may wish to begin writing software on the QUADS board without waiting for the target board to be completed. The QUADS board allows this by providing the hardware connections to Ethernet (twisted-pair and AUI), LocalTalk (AppleTalk), and an RS-232 port. The Ethernet connection offers full support of twisted-pair and AUI through the use of the

MC68160 EEST device. The LocalTalk connection is made through an RS-422 transceiver. Applications not needing LocalTalk can use the RS-422 to implement other proprietary LANs. The BDM output connector allows the QUADS board to control another device using background debug mode. All these connections are made through the slave QUICC to prevent master QUICC resources from being impacted.

If the user application includes a need for the RS-232 port, the system RS-232 port may be used for the application if the host computer is connected to the QUADS board via the parallel port.

More detail of the QUADS system is shown in Figure B-3. Buffers are provided in this system so that the system functions do not affect the fanout of the expansion connector. In a real application, such buffers and the local bus arbiter are not necessarily required. The slave QUICC provides the system integration functions (such as chip selects) that would normally be provided by the QUICC's own SIM. Thus, the expansion connectors provide all the master QUICC resources without using these resources and provide a non-degraded QUICC electrical interface.

To assist with target board development, the QUADS board may be connected to the target board through the edge connectors or through a PGA connector using a flexible PC board supplied by a third party. (Adapters from PGA to PQFP are also available from third parties.) Prototype boards may therefore make use of signals available from the QUADS development board. Target board execution may be carried out with the QUICC executing from the QUADS board memory, from target board memory, or a combination of both.

The QUADS board also contains connections for use by a logic analyzer, allowing all QUICC pins to be examined. Connectors between the QUADS board and common logic analyzers are available from third parties.

To connect the QUADS board to a host computer, a parallel interface cable and host interface (ADI) board are supplied. Hosts include the IBM-PC and SUN-4 systems. Available software executing on the host allows uploading and downloading of files and data between the host and the QUADS board and allows control of the user interface module through a window-based interface program resident on the IBM-PC or SUN-4. Source-level debugging support through this interface and the serial interface is provided by third-party vendors.

NOTE

The ADI boards mentioned here are the same ADI boards used with the MC68302 ADS system. If the user already purchased an ADI board for a MC68302 development project, that same board may be used with the QUADS board as long as the ADI board supports the IBM-PC, or SUN-4. The user only has to obtain new host software (available on the BBS) to interface to the QUADS board.

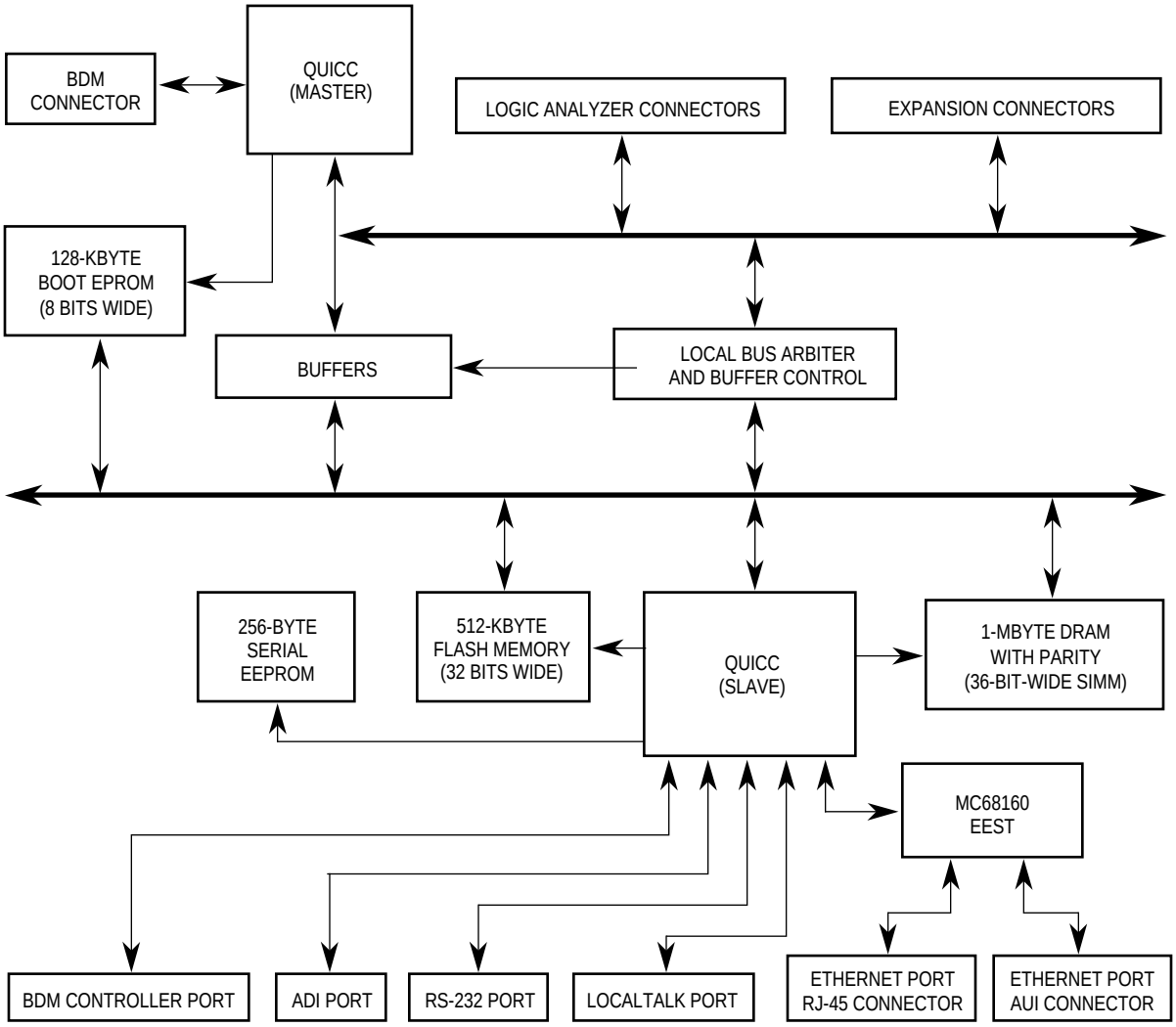


Figure B-3. QUADS Block Diagram

To connect the QUADS board to a host computer, a parallel interface cable and host interface (ADI) board are supplied. Hosts include the IBM-PC and SUN-4 systems. Available software executing on the host allows uploading and downloading of files and data between the host and the QUADS board and allows control of the user interface module through a window-based interface program resident on the IBM-PC. Source-level debugging support through this interface and the serial interface is provided by third-party vendors.

To serve as a convenient platform for software development, the QUADS board is supplied with a simple kernel and a CPU32BUG monitor/debugger. The real-time kernel (EDX) offers basic operating system services such as multitasking and memory management. The CPU32BUG monitor/debugger provides operations of memory dump and set (with optional assembly/disassembly of CPU32+ instructions), single instruction execution, breakpoints, and downloads over the serial and parallel interfaces.

Additionally, the QUADS board contains the software modules (protocol, driver, and user interface modules) in EPROM or in downloadable S-record format. An ADI board is required if the software (360sw) provided with the protocol or driver modules is to be used on the

QUADS board. That is, in order to use the .g files on the IBM/PC or SUN-4 system to download register settings to the QUADS board, an ADI card must be available to download these .g files through the serial port. The .g files can not be downloaded from the serial port on the QUADS board. Also, downloading a file to the QUADS boards is much faster through the parallel port versus the serial port.

B.5 OTHER DEVELOPMENT BOARDS

Additional third-party support includes very low-cost development boards for the QUICC as a standalone device. Additionally, third parties will include specialized development boards for such applications as the QUICC's MC68040 companion mode, which has the MC68040 and the QUICC on the same board. Contact the local sales representative for information on third-party development boards.

B.6 DIRECT TARGET DEVELOPMENT

The QUICC BDM, along with its on-chip hardware breakpoint capabilities, make it possible for some low-budget development projects to eliminate the development board or emulator. Third parties have software support that allows direct use of the QUICC background debug port. Thus, with a tool such as an IBM-PC for software development, a simple hardware connection between the PC and the QUICC BDM, and a PC software package that controls the QUICC through the BDM, an inexpensive "direct target" development environment is obtained.

APPENDIX C
RISC MICROCODE FROM RAM

The RISC processor in the QUICC has an option to execute microcode from the first 512 or 1024 bytes of the internal dual-port RAM. Motorola uses this feature to enhance existing protocols or implement additional protocols. Customers can purchase one or more of these RAM microcodes in an object-code format and download it to the QUICC dual-port RAM during system initialization.

The RAM microcode is provided by Motorola as a set of S-records that can be downloaded directly to an application development system or stored in a system EPROM. After system reset, the binary of the microcode should be copied to the QUICC dual-port RAM. The QUICC registers, including the RISC controller configuration register (RCCR), should be initialized as specified in the microcode RAM documentation. Before the RISC is used in the system, the user should issue a reset command to the communications processor command register (CR). The microcode RAM functions are available in addition to all protocols available in the standard QUICC microcode ROM.

The following microcode RAM protocols are completed, in progress, or under consideration by Motorola. Please contact the local Motorola semiconductor sales office for further information on these packages.

C.1 SIGNALING SYSTEM #7 CONTROLLER

Signalling System #7 (SS7) is a management protocol used in public switching networks. The physical, data link, and network layer functions of the SS7 protocol are called the Message Transfer Part (MTP).

The data link layer portion of the MTP (layer 2) is based upon HDLC frame formats. However, SS7 at layer 2 also includes some unique functions that are difficult to implement using an unaltered HDLC controller. These functions include: counting the number of octets by which a frame is too long; sending fill in signal units (FISUs) and link status signal units (LSSUs) continuously; maintaining the SU Error Monitor; and filtering duplicate back-to-back frames.

FISUs are 5 byte frames (three bytes plus 2 CRC bytes) that are sent continuously back-to-back when no other data needs to be transmitted. LSSUs are also sent back-to-back during the initial alignment of the protocol. LSSUs are 6 or 7 bytes long. SS7 also differs from other HDLC-based protocols in that the closing flag of one frame can be the opening flag of the next frame. Since these characteristics can make SS7 implementations very demanding, the SS7 controller on the QUICC provides additional help in these areas.

This controller only performs the lowest layers of the SS7 protocol stack. The upper layers must be performed in software by the CPU in the system. The software to perform the upper layer protocol is available from third party providers.

The SS7 controller contains the following key features:

- Uses either NMSI or TDM interface and a variety of data encoding schemes.
- Flexible data buffers with multiple buffers per signal unit allowed.
- Separate interrupts for received signal units and transmitted buffers.
- Maintenance of good frame counter, bad frame counter, and SU Error Monitor.
- Standard HDLC features:
  - —Flag/Abort/Idle generation/detection
  - —Zero insertion/deletion
  - —16-bit CRC-CCITT generation/checking
  - —Detection of non-octet aligned signal units
  - —Programmable number of flags between signal units
  - —Detection of long SUs.
  - —Discard short (less than 5 octets) signal units.
  - —Automatic Fill-In Signal Unit transmission.
  - —Automatic Link Status Signal Unit retransmission.
  - —Automatic discard of identical FISUs and LSSUs.
  - —Octet Counting Mode support.
  - —Command to force a reset of filtering state.
  - —Command to force entry into octet counting mode.
  - —Ability to permanently disable octet counting mode.
  - —Ability to disable FISU and LSSU filtering.
  - —Ability to force entry into octet counting mode if a receiver overrun occurs.
  - —Consumes 1280 bytes of the QUICC’s internal memory.

C.1.1 Performance

At 25Mhz, an aggregate SS7 bandwidth of 6 Mbps divided among the 4 SCCs consumes 100% of the processing power of the RISC communications engine. If only a percentage of the total available SS7.

bandwidth is used, the remaining RISC processing power can be used to run other protocols on other channels. Table C-1 shows possible QUICC configurations.

Table C-1. SS7 Configuration

SS7 Channels	Risc Bandwidth Consumed (est)	Possible Configuration of Other Channels
1 x 64 Kbit/s	1%	1 x 10 Mbit Ethernet, 2 x 2.048 Mbit HDLC
2 x 64 Kbit/s	2%	2 x 4 Mbit HDLC, 2 x 19.2 Kbit SMC UART
3 x 64 Kbit/s	3%	1 x 6 Mbit HDLC, 2 x 19.2 Kbit SMC UART

C.2 MULTIPLE GCI CONTROLLER

The MC68360 Quad Integrated Communications Controller (QUICC) can handle connections over two basic rate ISDN links using its Serial Management Controllers. For connection to more than two ISDN links, the Multiple GCI (MGCI) RAM-based microcode provides General Circuit Interface (GCI) handling functions for up to 16 basic rate ISDN subscriber lines. MGCI can be used in any application where multiple ISDN lines need to be terminated, such as 2, 4, 8, or 16-subscriber ISDN line cards and video servers.

Associated with each ISDN subscriber line is a monitor channel, a C/I channel, two B-channels, and one D-channel. MGCI handles only the monitor and C/I channels and has no effect on the B and D channels.

C.2.1 Typical Application

Figure C-1 shows a typical 16-subscriber ISDN line card using MGCI in an ISDN switch. Each subscriber line is terminated on the line card by a U or S/T line transceiver (e.g. MC145572 or MC145574). The multiplexer logic converts between the 16 individual basic rate GCI circuits and time division multiplex links carrying the D, B and control channels. The B channels are routed to the main system via the system interface while the D-channels are handled on the board. The C/I and monitor channels are used on the board to carry control and status information between the MC68360 and the line transceivers.

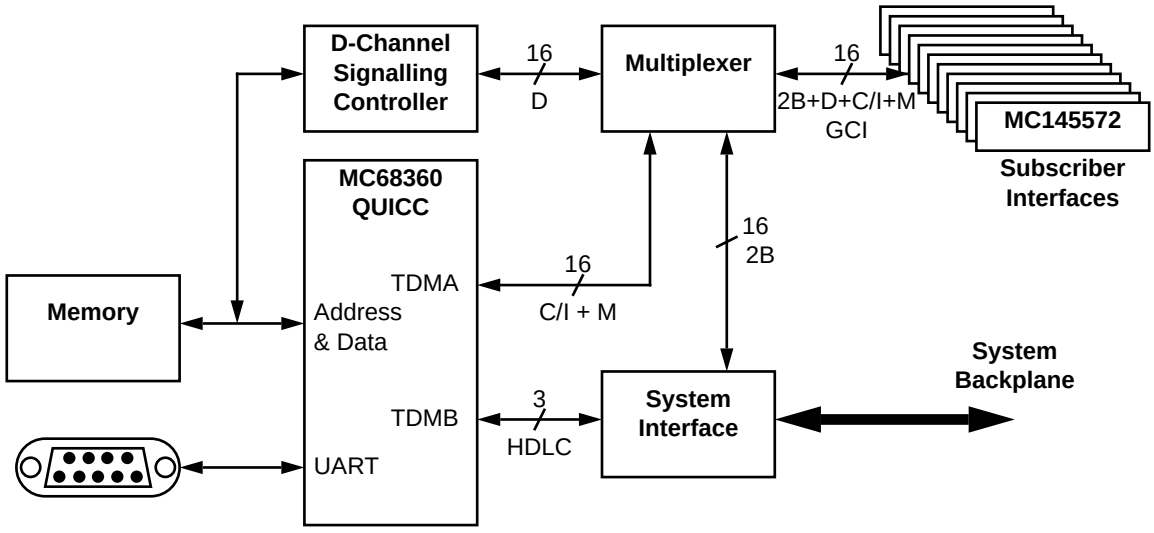


Figure C-1. Typical 16-SUbscriber ISDN Line

C.2.2 MGCI Controller Key Features

- Implements ISDN GCI and analog GCI monitor channel and C/I channel protocols.
- Handles GCI monitor message reception and transmission automatically.
- Handles four and six bit C/I code reception and transmission.
- User programmable C/I receiver detection mechanism: single change or double last look.

- All receive data is timestamped.
- Handles 2, 4, 8, or 16 ISDN lines with full C/I and monitor channel functions.
- Handles up to 32 ISDN lines with only C/I channels.
- Simultaneous detection of multiple C/I code changes and transmission changes.
- Maskable interrupts generated for many events.
- Handles GCI monitor messages from 1 to 64 Kilobytes in length.
- Monitor receiver can be locked into a particular channel.
- Monitor receiver and transmitter include timers to prevent lockup due to inactivity.
- Operates independently of user CPU activity.
- Consumes 1280 bytes of the QUICC's internal memory.

C.2.3 Performance

At 25Mhz, an MGCI serial bit rate of 2 Mbps on one SCC consumes 20 - 25% of the processing power of the RISC communications engine. An MGCI SCC operating at a serial bit rate of 2Mbps carries 32 x 64Kbit channels, Table C-2 shows possible QUICC configuration.

Table C-2. MGCI SCC Configuration

MGCI SCCs	Risc Bandwidth Consumed (est)	Possible Configuration of Other Channels
1 x 2 Mbit/s	25%	3 x 2 Mbit HDLC, 2 x 9.6 Kbit SMC UART
2 x 2 Mbit/s	50%	2 x 2 Mbit HDLC, 2 x 9.6 Kbit SMC UART

C.3 ATOM1/ATM CONTROLLER

ATOM1 provides physical layer ATM functions by converting one or more of the QUICC's Serial Communication Controllers (SCCs) into an ATM cell transmitter and receiver. The microcode provides the user with basic cell streaming facilities (cell reception and transmission) and event indications. The primary application of ATOM1 is intended to be plesiochronous digital hierarchy (PDH) and synchronous digital hierarchy (SDH) E1 and DS1 ATM equipment. Such equipment is used for signalling and low rate data transfer.

Figure C-2 shows an ATM / frame relay interworking system as may be used in remote bridging applications. Using the Ethernet channel available on the QUICC, remote LAN bridging equipment can be constructed to link remote Ethernet LANs over E1 telecommunications links.

C.3.1 Key Features

- Cell transmission and reception for all AAL protocols.
- Transmit and receive data buffers located in main memory.
- Microcode constructs the cell header and appends user defined payload on transmit.
- Microcode verifies cell headers and strips HEC before passing cell to the user on receive.

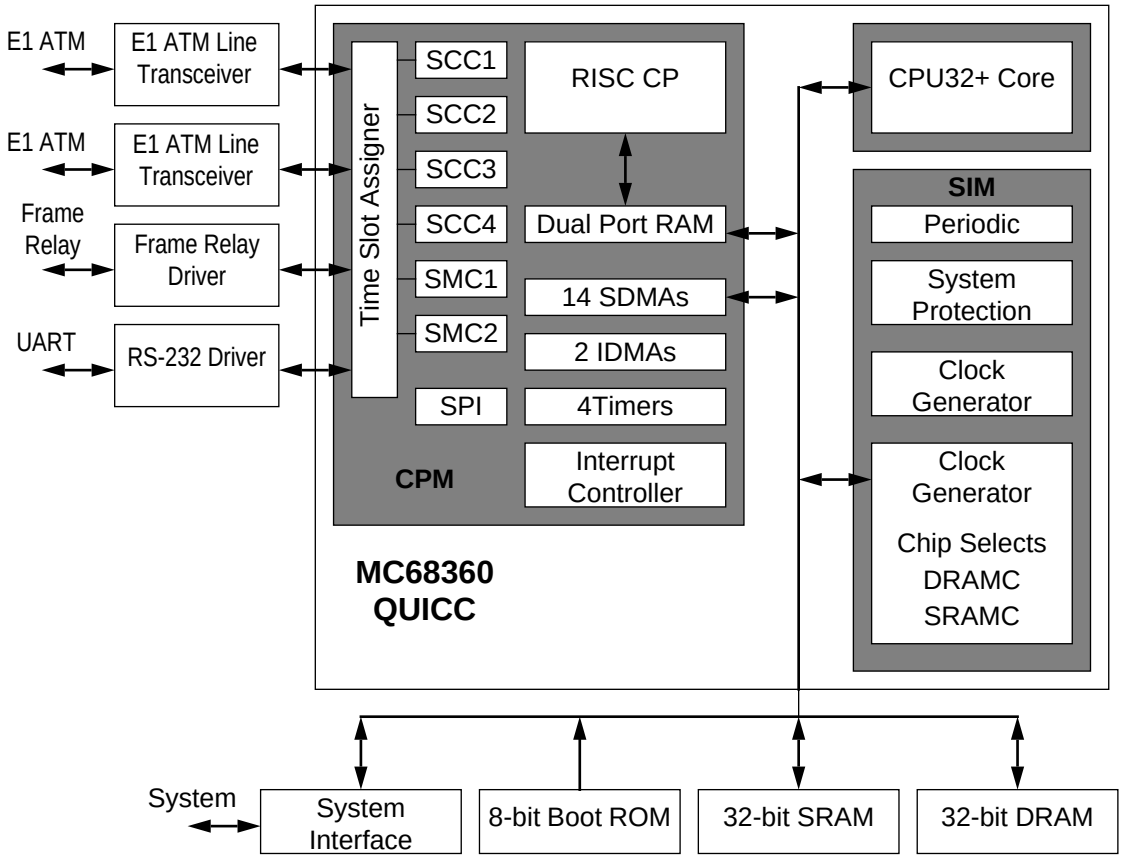


Figure C-2. ATM/Frame Relay Interwork System

- Empty cells transmitted when there are no pending data transfers.
- Empty cells and cells with non-matching headers are automatically discarded on receive.
- Scrambling option is provided utilizing the self-synchronizing $X^{43} + 1$ scrambling polynomial.
- Incoming cells with incorrect HECs are received and marked.
- Bandwidth reservation mechanism in the transmitter to allow mixing of data and isochronous services.
- CAM support on reception for handling many connections.

  —Consumes 1280 bytes of the QUICC's internal memory.

C.3.2 Performance

At 25 Mhz, an aggregate ATOM1 bandwidth of 10 Mbps divided among the 4 SCCs consumes 100% of the processing power of the RISC communications engine. If only a percentage of the total available ATOM1 bandwidth is used, the remaining RISC processing power can be used to run other protocols on other channels. Table C-3 shows the possible QUICC configuration.

Table C-3. ATOM1 Configuration

ATOM1 Channels	Risc Bandwidth Consumed (est)	Possible Configuration of Other Channels
1 x 6 Mbit/s with scrambling	90%	3 x 64Kbit HDLC or Transparent
1 x 10 Mbit/s non-scrambling	90%	2 x 64Kbit HDLC or Transparent
2 x 2 Mbit/s with scrambling	40%	2 x 2.5 Mbit HDLC or Transparent, 9.6 Kbit SMC UART
2 x 2 Mbit/s non-scrambling	40%	1 x 10 Mbit Ethernet, 9.6 Kbit SMC UART

C.4 ASYNCHRONOUS HDLC FOR PPP

Asynchronous HDLC is a frame-based protocol (see Figure 3), defined by the Internet Engineering Task Force (IETF) "Request For Comments #1549" which uses HDLC framing techniques in conjunction with UART-type characters. This protocol is typically used as the physical layer for the Point-to-Point (PPP) protocol. While this protocol can be implemented by the UART controller on the QUICC in conjunction with the CPU32+, it is more efficient and less compute-intensive for the CPU to allow the Communications Processor Module (CPM) of the QUICC to perform the framing and transparency functions of the protocol.

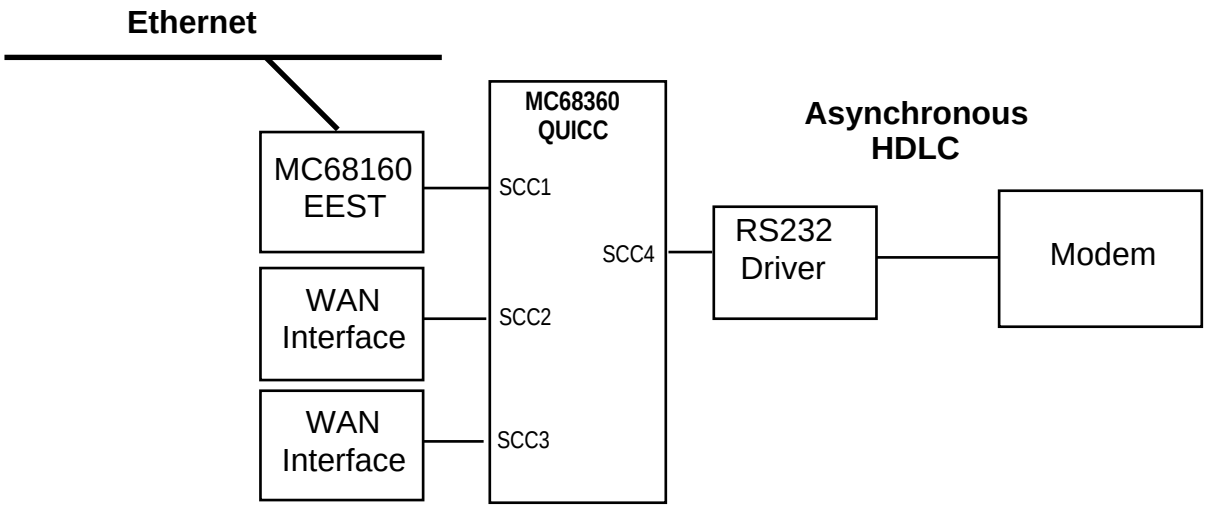


Figure C-3. Asynchronous HDLC Block Diagram

C.4.1 Key Features

- Flexible data buffer structure which allows an entire frame or a section of a frame to be transmitted and received.
- Separate interrupts for received frames and transmitted buffers.
- Automatic 16-bit CRC generation and checking (CRC-CCITT).
- Automatic generation of opening and closing flags.

- Reception of frames with only one “shared” flag.
- Automatic generation and stripping of transparency characters according to the Internet Engineering Task Force RFC 1549 utilizing transmit and receive control character maps.
- Automatic transmission of the ABORT sequence (0x7D,0x7E) after the STOP TRANSMIT command is issued.
- Automatic transmission of IDLE characters between frames and characters.
- Consumes 768 bytes of the QUICC’s internal memory.

C.4.2 Performance

At 25Mhz, an aggregate Asynchronous HDLC bandwidth of 3 Mbps divided among the 4 SCCs consumes 100% of the processing power of the RISC communications engine. If only a percentage of the total available Asynchronous HDLC bandwidth is used, the remaining RISC processing power can be used to run other protocols on other channels. Table C-4 shows the AHDLC configuration

Table C-4. AHDLC Configuration

AHDLC Channels	Risc Bandwidth Consumed (est)	Possible Configuration of Other Channels
1 x 115 Kbit/s	4%	1 x 10Mbit Ethernet, 2 x 1.5 Mbit HDLC, 9.6 Kbit SMC UART
2 x 230 Kbit/s	15%	1 x 10Mbit Ethernet, 1 x 1.5 Mbit HDLC, 9.6 Kbit SMC UART
3 x 230 Kbit/s	24%	1 x 5Mbit HDLC, 2 x 9.6 Kbit SMC UART

C.5 PROFIBUS CONTROLLER

Process field bus (PROFIBUS) is a UART- based master slave protocol that specific data between 9.6 kbps to 1.525 Mbps. The PROFIBUS protocol is mainly used in industrial process control applications. The protocol is defined in the German DIN standard 19 245.

The PROFIBUS microcode running on the QUICC RISC controller assists the core in handling some of the time-critical PROFIBUS link layer functions, leaving more of the core available for the application software.

C.5.1 Key Features

- Frame preceding IDLE sequence generation/checking
- Flexible Frame Oriented Data Buffers
- Separate interrupts for Frames and Buffers (Receive and Transmit)
- Maintenance of six 16-bit error counters
- Two Address Comparison Registers with Mask
- Frame Error, Noise Error, Parity Error Detection
- Detection of IDLE in middle of a frame
- Check Sum generation/checking

- End Delimiter (ED) generation/checking
- Init Rx/Tx, Stop TX, Restart TX and Enter Hunt Mode Commands
- Token Rotation Timer, Idle Timer, Slot Timer and Time-out Timer
- Consumes 1280 bytes of the QUICC's internal memory.

C.6 ENHANCED ETHERNET FILTERING

The enhanced Ethernet filtering microcode has been created to add a bit more flexibility to the current Ethernet Controller. It allows the user to perfectly accept or reject frames with destination addresses that are contained in a table of 24 entries. In addition to the filtering capability, the microcode adds the ability to replace the externally sampled tag byte with one that is extracted from the address table.

C.6.1 Key Features

- Can accept or reject incoming frames if the destination MAC address is contained in a list of 24 preprogrammed entries.
- Allows a shorter list to obtain better peripheral performance.
- Can optionally tag incoming accepted frames with an 8-bit tag appended to the end of the frame (rather than using a CAM).
- The filtering can be turned off to allow tagging with no frame rejection.
- Can filter on group and individual addresses.
- Is a "small" microcode (consumes 768 bytes of DPRAM).
- Filtering is only supported on one Ethernet channel (SCC1), the other channel operates normally.

C.6.2 Performance

The overhead incurred by the filtering algorithm will depend upon the number of entries in the address table. Table C-5 shows possible configurations of other channels given the load added by a number of entries in the address table.

Table C-5. Channel Configuration

Number of addresses	Possible Configuration of Other Channels
1-8	1 x 10 Mbit Ethernet, 1 x 200 Kbit HDLC
9-16	1 x 10 Mbit Ethernet, 1 x 200 Kbit HDLC
17-24	1 x 10 Mbit Ethernet

APPENDIX D
MC68MH360 PRODUCT BRIEF

The MC68MH360 is a derivative of the MC68360 QUICC.™ It is also known as the QUICC32.™

The QUICC32 has some modifications in the Communication Processor Module (CPM) hardware and a larger internal dual port RAM. These hardware changes in combination with a new ROM based microcode, called QMC (QUICC Multichannel Controller), emulates up to 32 HDLC channels within one SCC. The MH360 is ideal in primary rate ISDN applications and interfaces directly to a E1/T1 line and is capable of terminating all time slots using HDLC or transparent mode of operation. One of the time slot assigners (SI) is dedicated to the use of the QMC. The SI transfers the whole frame or selected time slots to one SCC. All other features remain unchanged and the QUICC32 is pin compatible with the other family members.

A QUICC32 in non-QMC mode has exactly the same functionality as a standard MC68360 with the exception that the Centronics and BISYNC protocols have been removed on the QUICC32 to create ROM space for the QMC protocol.

D.1 QUICC32 KEY FEATURES

The following list summarizes the key MC68MH360 QUICC32 features:

D.1.1 General

- Up to 32 Independent Communication Channels
- Arbitrary Mapping of Any of 0-31 Channels to Any of 0-31 TDM Time Slots
- Can Support Arbitrary Mapping of Any of 0-31 Channels to Any of 0-63 TDM Time Slots in Case of Common Rx & Tx Mapping
- Independent Mapping for Receive/Transmit
- Supports Either QUICC Transparent or QUICC HDLC Protocols for Each Channel
- QUICC-Like Manner for Channel Programming and Parameter Passing
- Up to 64 DMA Channels with Linear Buffer Array
- Interrupt Circular Buffer with Programmable Size and Overflow Identification
- Global Loop Mode
- Individual Channel Loop Mode
- Programmable Frame Length (Via QUICC's SI)

D.1.2 Serial Interface

- 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, User Defined
- Sub Channeling On Each Time-Slot
- Allows Independent Transmit and Receive Routing, Frame Syncs, Clocking
- Concatenation of Any, Not Necessarily Consecutive, Time Slots to Channels Independently for Receive/Transmit
- Supports H0, H11, H12 ISDN - Channels
- Allows Dynamic Allocation of Channels
- Up to 3 Additional HDLC 64 Kbps Channels at 25 Mhz System Clock when in QMC Mode
- Ethernet Support at 33 Mhz System Clock on One SCC when Operating in QMC Mode

D.1.3 System Interface

- QUICC Powerful System Support Features: CPU32+, Slave Mode, 040 Companion, Memory Controller, 2xIDMA, SIM60 Watchdogs, Bus Monitors, Timers, Low Power Modes, Breakpoint Logic, Interrupt Controller
- On-Chip Bus Arbitration for Serial DMAs with No Performance Penalty
- Efficient Bus Usage (No Bus Usage For Non-Active Channel, and for Active Channels that Have Nothing to Transmit)
- Efficient Control of the Interrupts to the CPU
- Supports External Buffer Descriptors Table
- Using On-Chip Enlarged Dual-Port RAM for Parameter Storage

D.2 QUICC ARCHITECTURE OVERVIEW

The QUICC is 32-bit controller that is an extension of other members of the Motorola M68300 family. Like other members of the M68300 family, the QUICC incorporates the intermodule bus (IMB). (The MC68302 is an exception, having an M68000 bus on chip.) The IMB provides a common interface for all modules of the M68300 family, which allows Motorola to develop new devices more quickly by using the library of existing modules. Although the IMB definition always included an option for an on-chip 32-bit bus, the QUICC is the first device to implement this option.

The QUICC consists of three modules: the CPU32+ core, the SIM60, and the CPM. Each module uses the 32-bit IMB. The MC68360 QUICC block diagram is shown in Figure D-1.

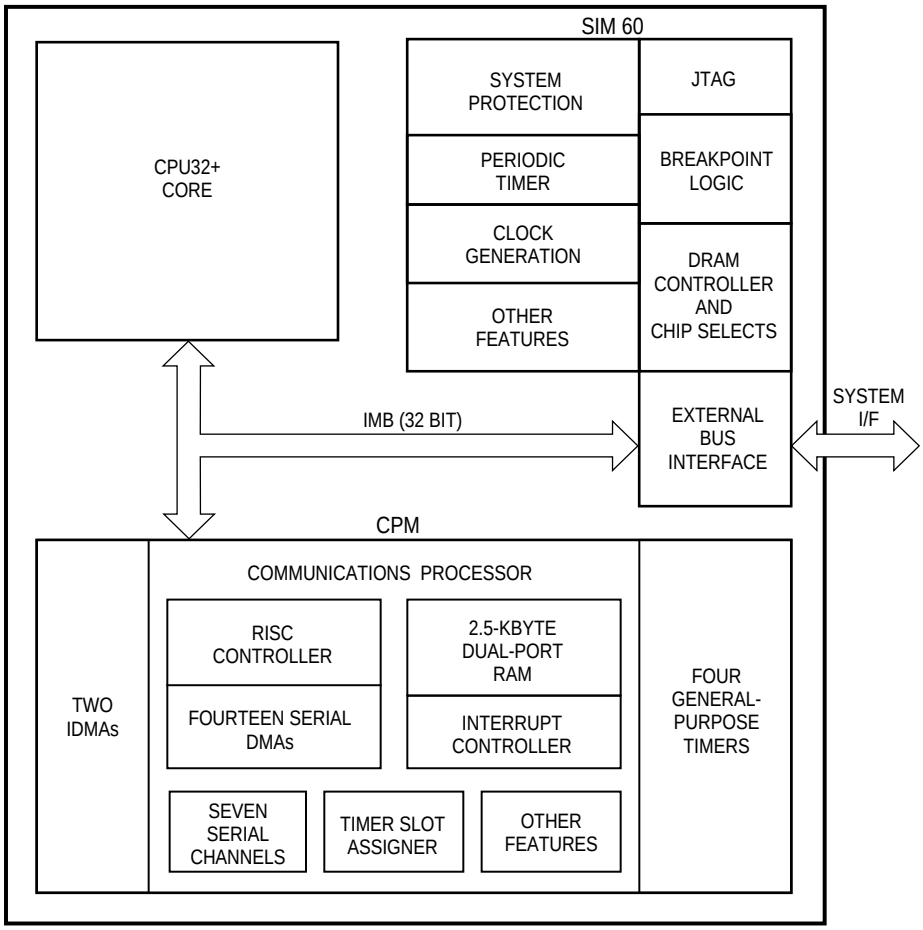


Figure D-1. QUICC Block Diagram

D.2.1 CPU32+ Core

The CPU32+ core is a CPU32 that has been modified to connect directly to the 32-bit IMB and apply the larger bus width. Although the original CPU32 core had a 32-bit internal data path and 32-bit arithmetic hardware, its interface to the IMB was 16 bits. The CPU32+ core can operate on 32-bit external operands with one bus cycle. This allows the CPU32+ core to fetch a long-word instruction in one bus cycle and to fetch two word-length instructions in one bus cycle, filling the internal instruction queue more quickly. The CPU32+ core can also read and write 32-bits of data in one bus cycle.

Although the CPU32+ instruction timings are improved, its instruction set is identical to that of the CPU32. It will also execute the entire M68000 instruction set. It contains the same background debug mode (BDM) features as the CPU32. No new compilers, assemblers, or other software support tools need be implemented for the CPU32+; standard CPU32 tools can be used.

The CPU32+ delivers approximately 4.5 MIPS at 25 MHz, based on the standard (accepted) assumption that a 10-MHz M68000 delivers 1 VAX MIPS. If an application requires more

performance, the CPU32+ can be disabled, allowing the rest of the QUICC to operate as an intelligent peripheral to a faster processor. The QUICC provides a special mode called MC68040 companion mode to allow it to conveniently interface to members of the M68040 family. This two-chip solution provides a 22-MIPS performance at 25 MHz.

The CPU32+ also offers automatic byte alignment features that are not offered on the CPU32. These features allow 16 or 32-bit data to be read or written at an odd address. The CPU32+ automatically performs the number of bus cycles required.

D.2.2 System Integration Module (SIM60)

The SIM60 integrates general-purpose features that would be useful in almost any 32-bit processor system. The term “SIM60” is derived from the QUICC part number, MC68360. The SIM60 is an enhanced version of the SIM40 that exists on the MC68340 and MC68330 devices.

First, new features, such as a DRAM controller and breakpoint logic, have been added. Second, the SIM40 was modified to support a 32-bit IMB as well as a 32-bit external system bus. Third, new configurations, such as slave mode and internal accesses by an external master, are supported.

Although the QUICC is always a 32-bit device internally, it may be configured to operate with a 16-bit data bus. Regardless of the choice of the system bus size, dynamic bus sizing is supported. Bus sizing allows 8-, 16-, and 32-bit peripherals and memory to exist in the 32-bit system bus mode and 8- and 16-bit peripherals and memory to exist in the 16-bit system bus mode.

D.2.3 Communications Processor Module (CPM)

The CPM contains features that allow the QUICC32 to excel in communications and control applications. These features may be divided into three sub-groups:

- Communications Processor (CP)
- Two IDMA Controllers
- Four General-Purpose Timers

The CP provides the communication features of the QUICC32. Included are a RISC processor, four SCCs, two SMCs, one SPI, 2.7 kbytes of dual-port RAM, an interrupt controller, a time slot assigner, three parallel ports, a parallel interface port, four independent baud rate generators, and fourteen serial DMA channels to support the SCCs, SMCs, and SPI.

The IDMAs provide two channels of general-purpose DMA capability. They offer high-speed transfers, 32-bit data movement, buffer chaining, and independent request and acknowledge logic. The RISC controller may access the IDMA registers directly in the buffer chaining modes. The QUICC32 IDMAs are similar to, yet enhancements of, the two DMA channels found on the MC68340 and the one IDMA channel found on the MC68302.

The four general-purpose timers on the QUICC32 are functionally similar to the two general-purpose timers found on the MC68302. However, they offer some minor enhancements,

such as the internal cascading of two timers to form a 32-bit timer. The QUICC32 also contains a periodic interval timer in the SIM60, bringing the total to five on-chip timers.

D.2.3.1 QUICC32 SERIAL CONFIGURATIONS. The QUICC32 offers an extremely flexible set of communications capabilities. Although a full understanding of the possibilities requires reading the appropriate sections, some of the possibilities are shown in the following diagrams. They show possible connections between QUICC32 devices. In addition, connections are often shown between QUICCs and the MC68302 to show the compatibility between these devices.

For readability, transceivers are usually omitted in the following diagrams. For local on-board communications, however, transceivers are often optional and depend on the protocol used.

Figure D-2 shows the Ethernet LAN capability of the QUICC32. An external SIA transceiver is required to complete the interface to the media. This functionality is implemented in the MC68160 enhanced Ethernet serial transceiver (EEST™). The MC68160 EEST supports connections to the attachment unit interface (AUI) or twisted-pair Ethernet formats and provides a glueless interface to the QUICC32.

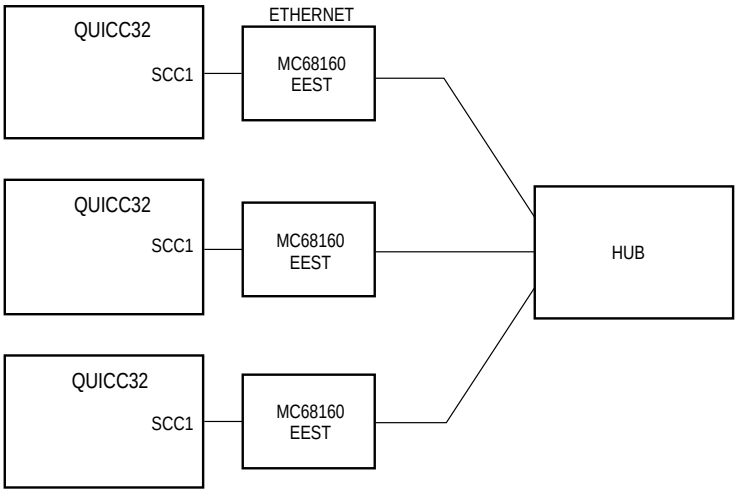
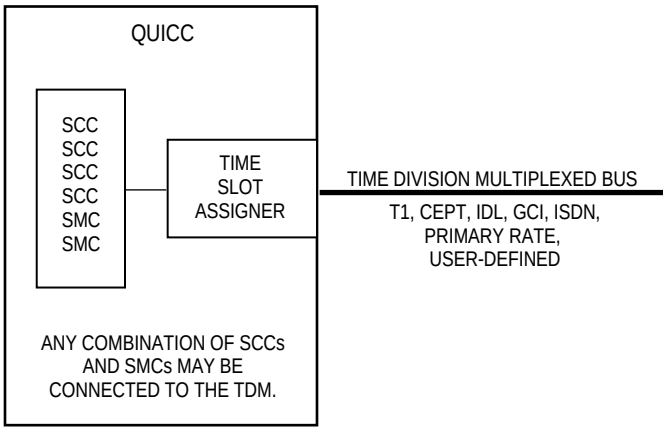


Figure D-2. Ethernet LAN Capability

Figure D-3 shows how up to six of the serial channels can connect to a TDM interface. The QUICC32 provides a built-in time-slot assigner for access to the TDM time slots. Other channels can work with their own set of pins, allowing possibilities like an Ethernet to T1 bridge, etc.

In an addition to this the QMC microcode emulates up to 32 channels within one SCC. The QUICC32 can terminate all time slots in a T1 or E1/CEPT line

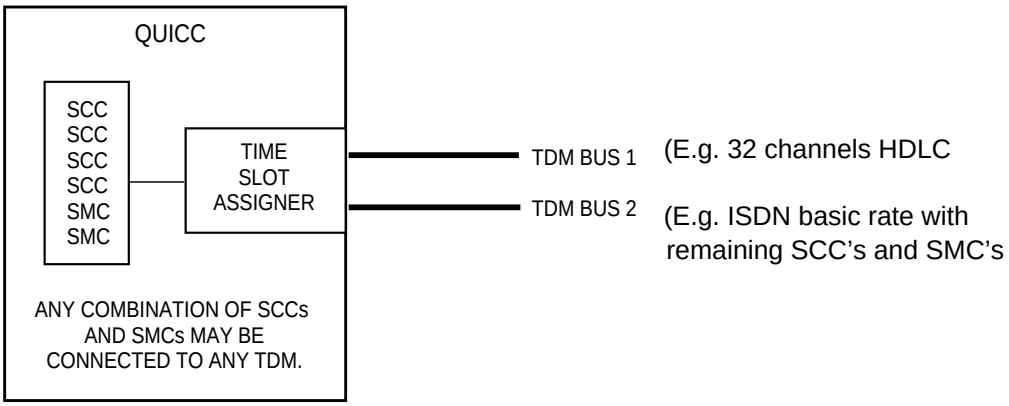


NOTE: Independent receive and transmit clocking, routing, and syncs are supported.

Figure D-3. Serial Channel to TDM Bus Implementation

Figure D-4 shows that the QUICC32 time-slot assigner can support two TDM buses. Each TDM bus can be of a different format—for example, one TDM can be a T1 line, and one can be a CEPT line. Also this technique could be used to bridge frames from basic rate ISDN to a T1/CEPT line, etc.

The QUICC32 can route channels to and from the QMC protocol to the two different TDM buses in any combination.



NOTE: Two TDM buses may be simultaneously supported with the time slot assigner.

Figure D-4. Dual TDM Bus Implementation

Figure D-5 shows a TDM application having one line termination device that extracts clocks and frame sychronization pulses. For T1/E1 line termination, devices exist that perform this function. Alternatively, this can be achieved by a DSP from the Motorola 56K family.

For line termination the QUICC32 is capable of handling up to 32 channels and it may be sufficient to omit the block labeled “Other PCM line devices”. If it is desired to incorporate other PCM line devices in the system as shown in Figure D-5, the QUICC32 can provide strobe signals to other devices that do not have a built-in time slot assigner.

Figure D-5 shows the line interface unit, LIU, being the master and providing the QUICC32 with clocks. It is also possible to let the QUICC32 be the master and provide clocks and synchronization pulses through its baud rate generators, both to itself and other devices on the TDM bus.

D.2.4 The QMC Microcode

The standard QUICC can handle one logical channel performing the protocol framework for each of its serial channels. This logical channel can be used in time division multiplexed interfaces as described above. The QMC protocol emulates up to 32 serial controllers that can operate in either HDLC mode or transparent mode within one single SCC.

The QMC microcode is ROM based and in order to create enough memory space, the QUICC32 has the Centronics and BISYNC protocols removed.

The QUICC32 has the internal dual-port RAM enlarged by 192 bytes to accommodate the parameters used by the QMC microcode.

The standard QUICC housed all the buffer descriptor table in internal RAM and the actual data areas in either internal or external RAM. The QUICC32 needs an area in external memory for its buffer descriptor. This reserved memory area shall start at an address on a 64 K boundary. The data buffers have to reside in external main memory. See Figure D-6 for memory partitioning.

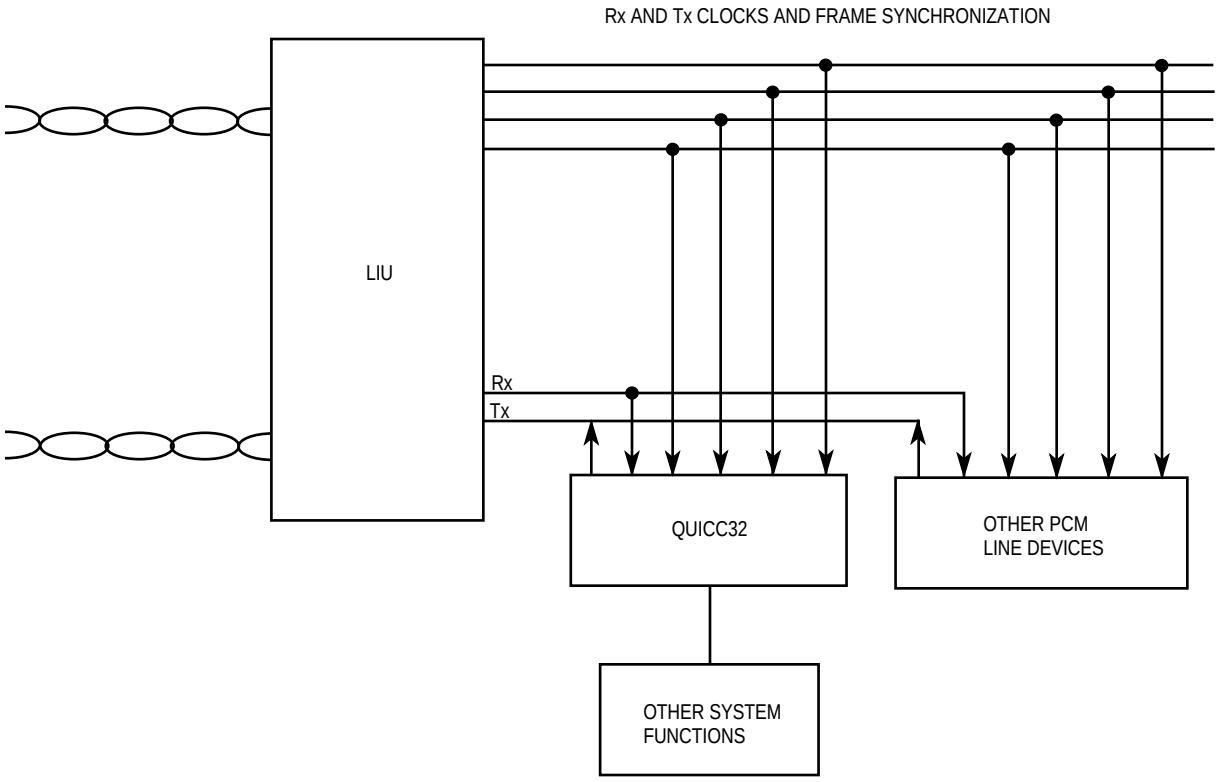


Figure D-5. TDM Connections

D.2.5 Data Flow

In a time slot environment like T1/E1, every slot occupies 8 consecutive bits creating a basic channel speed of 64 kbit/s. The number of time slots vary depending upon what interface will be used.

The QUICC32 can support up to 32 time slots with up to 8 bits, each giving a total throughput of 2.048 Mbit/s for the QMC protocol in a E1 or CEPT interface.

Through the SI RAM, the user programs either to route the whole serial stream to the SCC dedicated to QMC or use the SI RAM to gate only the designated time slots to the SCC. In this later case other serial channels in the QUICC32 can be multiplexed to the same physical interface through the SI RAM routing. External devices can also be used in designated time slots as long as they do not collide with the internal QUICC32 channels.

Any logical channel can be spread over several time slots creating super channels. The maximum configuration is to concatenate all 32 time slots into one 2,048 MBit/s channel.

Sub channels are achieved by masking out any combination of the 8 bits in every time slot. The resulting channel speed is N x 8 Kbit/S {N= 1....8}.

The physical interface from the QUICC32 is routed through the time slot assigner. Each logical channel can work in full duplex mode but does not have to be routed to the same receiver and transmitter time slot. This is how the standard QUICC operates today with each of the SCC and SMC channels. The QUICC32 can have independent receive and transmit clocks as well as frame synchronization signals. All receive and transmit channels within the QMC will have the same clock.

For diagnostic purpose each logical channel has a local loopback option. The whole incoming TDM frame can be selected to operate in global echo mode. For these diagnostic modes the clock and synchronization signals must be common for the receiver and transmitter.

D.2.6 Data Management

Time slots are numbered 0 - 31, and each time slot is 8 bits long. The parameter RAM is divided into a global section and a channel specific section. In the global section the channel routing is described in two tables, one for transmission and one for reception. This allows the user to connect the physical time slots to logical channels in any order desired and also mask for sub channels or to concatenate for super channels. Whatever combination is chosen, the maximum number of supported transmit channels is 32 and the maximum number of supported receive channels is 32.

For each logical channel there is a channel specific area in the DP RAM that governs the data flow. For each channel the user can choose if data is transparent or is framed with the HDLC protocol, and what checksum algorithm is used.

These two areas in the DP RAM occupy the whole space. Therefore the buffer descriptors for each logical channel have been moved out into the main memory. The buffer descriptors for all other protocols remain unchanged in internal memory. Depending on how many QMC channels are used, the space for buffer descriptors in the dual-port memory may vary.

.

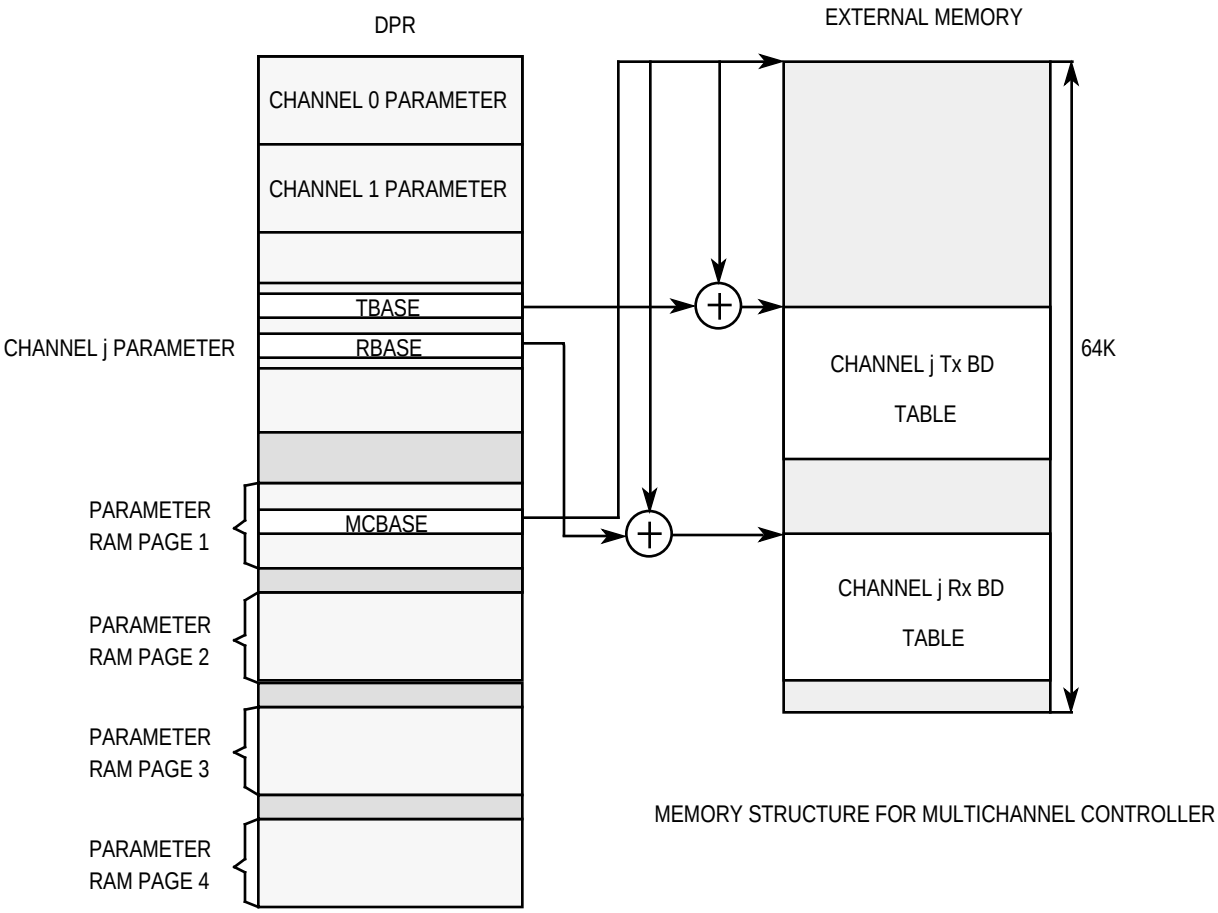


Figure D-6. FQMC Memory Map

D.2.7 Performance

The QUICC32 with the QMC protocol is designed to handle data rates from 8 kbps to 2.048 Mbps.

The QMC can support up to 32 channels.

These are the outer boundaries. The user is free to program channel routing in the QMC and the SI RAM to create any other TDM system.

Beside the support for QMC, the remaining SCC channels in the QUICC32 can, in the 25 MHz version, support a data rate of 64 kbps each with any protocol framework. These remaining channels can be used independently or also multiplexed through the SI RAM.

The 33 MHz version of QUICC32 can support one Ethernet channel with 32 QMC channels.

A more detailed performance table will be provided at a later date.

D.2.8 Development Support

No new development tools will be needed for the QUICC32.

Only the replacement of actual silicon is needed. All Motorola and third party support tools will work as before. Motorola will provide simple device drivers for developers to have an easy start on the software.

D.2.9 Ordering Information

The following table identifies the packages and operating frequencies available for the MC68MH360.

Table D-1. MC68MH360 Package/Frequency Availability

Package Type	V_{CC}	Frequency (MHz)	Temperature	Order Number
Quad Flat Pack (FE Suffix) Quad Flat Pack (CFE Suffix)	5 V	0–25 0–25	0°C to 70°C –40°C to +85°C	MC68MH360FE25 MC68MH360CFE25
Pin Grid Array (RC Suffix) Pin Grid Array (CRC Suffix)	5 V	0–25 0–25	0°C to 70°C –40°C to +85°C	MC68MH360RC25 MC68MH360CRC25
Quad Flat Pack (FE Suffix	5 V	0–33	0°C to 70°C	MC68MH360FE33
Pin Grid Array (RC Suffix)	5 V	0–33	0°C to 70°C	MC68MH360RC33
Quad Flat Pack (FE Suffix	3.3 V	0–25	0°C to 70°C	MC68MH360FE25V
Pin Grid Array (RC Suffix)	3.3 V	0–25	0°C to 70°C	MC68MH360RC25V

The documents listed in the following table contain detailed information on the MC68360. These documents may be obtained from the Literature Distribution Centers at the addresses listed at the bottom of this page.

Table D-2. Documentation

Document Title	Order Number	Contents
MC68MH360 Specification Addendum	Contact local field sales office	Preliminary design information
M68000 Family Programmer's Reference Manual	M68000PM/AD	M68000 Family Instruction Set
The 68K Source	BR729/D	Independent vendor listing supporting software and development tools

INDEX

A

A/D Converters 7-312
A/D Field 5-68
A/D Register 5-70, 5-71
A26–A0 2-1
A31–A28 2-1
Accessing the HDLC Bus 7-192
Achieving Synchronization in Transparent Mode 7-223
ACK 7-333
Address
  Ethernet Address Recognition 7-252
Address Bus 2-1
Address Error Exception 5-42, 5-46
Address Multiplexing 6-58
Address Register 5-6, 5-7
Address Strobe 4-4
ADI B-9
A-Line 5-44, 5-56, 5-59
Alternate Function Code 5-6
AMUX 2-7, 2-9, 6-48, 6-49
AppleTalk 1-10, 7-116, 7-117, 7-170, 7-196
  AppleTalk Controller 7-196
  AppleTalk Memory Map and Programming Model 7-198
  Connecting the QUICC to LocalTalk 7-199
  GSMR Programming 7-199
  HDLC Frames 7-196
  LocalTalk 7-196
  LocalTalk Frame Format 7-197
  PSMR Programming 7-200
  QUICC AppleTalk Hardware Connection 7-198
AppleTalk Controller 7-196
AppleTalk Memory Map and Programming Model 7-198
Applications 9-1
Arbitration Level 6-33
Arbitration Synchronous Timing Mode 6-31
Architectural Approach 1-6
Architecture Overview 1-4
AS 2-8, 2-13, 4-3, 6-30, 6-63, 7-41, 7-44, 7-58
Asynchronous 4-20
Asynchronous Protocols 7-134
ATEMP 5-62, 5-72
Auto Baud 7-166
Auto Buffer 7-34
Auto Buffer Example 7-56
Autovector 4-6, 4-38
AVEC 2-8, 4-6, 4-38, 4-41, 6-8, 6-26, 6-48
AVECO 6-26, 6-48
AVR 6-34

B

B0 5-49, 5-51, 5-52, 5-53, 5-54, 5-57
B1 5-49, 5-51, 5-53
Background 5-14, 5-26, 5-34
Background Processing State 5-59
Bank Parity 6-62
Base Address 3-1
BCLRI 4-58, 6-25, 6-33
BCLRIID 6-25
BCLRO 2-9, 2-10, 6-25, 6-31, 6-33, 7-44, 7-51, 7-58
BCR 7-31
BDLE 7-204
BDLE-BISYNC DLE Register 7-208
BERR 2-10, 4-5, 4-20, 4-41, 7-51
BERR Signal 5-40
BG 2-9, 2-10, 4-49, 4-53, 6-26, 6-31, 6-32, 7-44
BGACK 2-9, 4-49, 6-26, 6-31, 7-44
BGND 5-60, 5-61, 5-62
Bidirectional Centronics 7-331
Big-Endian 7-126, 7-321
BISYNC 7-114, 7-115, 7-200
  BISYNC Channel Frame Reception 7-202

BISYNC Channel Frame Transmission 7-201
BISYNC Command Set 7-204
BISYNC Control Character Recognition 7-206
BISYNC Controller 7-200
BISYNC Frames 7-200
BISYNC Memory Map 7-203
Data Length 7-213
Error-Handling 7-209
LRC 7-210
Nibblesync 7-201
Parity 7-211
Programming the BISYNC Controller 7-217
Reverse Data 7-211
SCC BISYNC Example 7-218
Sum Check 7-210
Transmitting and Receiving 7-208
BISYNC Channel Frame Reception 7-202
BISYNC Channel Frame transmission 7-201
BISYNC Command Set 7-204
BISYNC Control Character Recognition 7-206
BISYNC Controller 7-200
BISYNC Frames 7-200
BISYNC Memory Map 7-203
BISYNC Receive Buffer Descriptor 7-212
BISYNC Transmit Buffer Descriptor 7-213
Bit Manipulation Instructions 5-23
Bit Manipulation Timing Table 5-97
BKAR 6-44
BKCR 6-44
BKPT 2-11, 4-31
BKPT Instruction 5-61
BKPT Signal 5-60, 5-63, 5-66, 5-67
BKPT_TAG 5-68
BKPTO 6-26
BR 2-9, 2-10, 4-49, 4-52, 6-26, 6-31, 6-32, 6-56, 6-70, 7-44, 9-12, 9-50
BR040ID 6-30
Break 7-166
Break Support 7-151
Breakpoint 4-31
Breakpoint Exception 5-39, 5-42, 5-43
Breakpoint Exception 5-49
Breakpoint Instruction 5-59
Breakpoint Instruction 5-26, 5-43
Breakpoint Logic 6-20
BRG 7-86, 7-101
BRGC 7-106
BRGCLK 6-17, 7-104, 7-314
BRGO 7-101, 7-108, 7-364
Broadcast Address 7-257
BSTM 6-30, 6-68
BSYNC 7-204
BSYNC-BISYNC SYNC Register 7-207
Buffer
Auto Buffer 7-34
BISYNC Receive Buffer Descriptor 7-212
BISYNC Transmit Buffer Descriptor 7-213
Buffer Chaining 7-34
CP 7-123
Ethernet Receive Buffer Descriptor 7-258
Ethernet Transmit Buffer Descriptor 7-261
HDLC Receive Buffer Descriptor 7-179
HDLC Transmit Buffer Descriptor 7-183
PIP 7-341
SCC Bufer Descriptors 7-122
Single Buffer 7-34
SMC Transparent Receive Buffer Descriptor 7-299
SMC Transparent Transmit Buffer Descriptor 7-300
SMC UART Receiver Buffer Descriptor 7-283
SMC UART Transmit Buffer Descriptor 7-286
SPI Buffer Descriptor Ring 7-324
SPI Receive Buffer Descriptor 7-324
SPI Transmit Buffer Descriptor 7-326
Transfer Receive Buffer Descriptor 7-228
Transparent Transmit Buffer Descriptor 7-230
UART Receiver Buffer Descriptor 7-159
UART Transmit Buffer Descriptor 7-163
Buffer Chaining 7-34
Buffer Descriptors 7-10

Burst 6-72, 6-77, 6-78
Burst Access Support 6-61
Burst EPROM 9-38
Burst Length 6-76
Burst SRAM 9-41
Bus
  Address Bus 2-1
  Arbitration Synchronous Timing Mode 6-31
  Asynchronous 4-20
  Autovector 4-38
  Breakpoint 4-31
  Bus Error 4-5, 4-43
  Bus Exception 4-41
  Byte Port 4-12
  Data Bus 2-5
  Double Bus Fault 4-48
  Double Bus Fault Monitor 6-9
  Dynamic Bus Sizing 4-6, 4-19
  Fast Termination 4-21
  Halt Termination 4-41
  IMB 1-4, 10
  Intermodule Bus 1-4, 10
  Internal Accesses 4-59
  Interrupt Acknowledge 4-36
  LPSTOP 4-35
  Misaligned Operands 4-11
  Normal Termination 4-41
  Port Sizes 4-8
  Read Cycle 4-23
  Read-Modify-Write 4-28
  Retry 4-44
  Retry Termination 4-41
  Show Cycles 4-62
  Spurious Interrupt 4-41
  State Diagram 4-55
  Synchronous 4-21
  System Bus 2-1, 4-1
  Wait States 4-26, 4-28
  Word Port 4-9, 4-14
  Write Cycle 4-26
Bus Arbitration 4-49, 4-55, 9-35
Bus Clear 6-25
Bus Control Signals 2-7
Bus Controller 5-83
Bus Error 4-5, 4-43
Bus Error Exception 5-40
Bus Exceptions 4-41, 4-59, 7-51
Bus Monitor 6-4, 6-8
Bus Operation 4-1
Bus Signal 2-1
Busy Signal 7-335
Byte Port 4-12

C

C/I Channel 7-307
C/I Channel Receive Buffer Descriptor 7-310
C/I Channel Transmit Buffer Descriptor 7-311
Cache Modes on the MC68Ec040 9-51
Caching 9-36
Calculate Effective Address Timing Table 5-91
CALL 5-70
CALL Command 5-63
CAM Interface 7-243
Carrier Sense 7-116
CAS 2-7, 2-13, 6-48, 6-60, 6-63, 6-67
Cascaded Mode 7-19
CCITT I.460 7-97
CD 7-63, 7-101, 7-114, 7-118, 7-128, 7-129, 7-130, 7-187, 7-199, 7-219, 7-223, 7-229, 7-233, 7-240, 7-266, 7-358
CDVCR 6-12, 6-42
Centronics 1-13, 7-331, 7-336
CEPT 1-15, 14
Change of Flow 5-84, 5-85
Changing Privilege Level 5-35
Character Length 7-282, 7-298
Chip Driver Module B-2
Chip Selects 9-24
CICR 7-378
CIMR 7-381
CIPR 7-380
CISR 7-381
CLK 2-10, 2-13, 7-86, 7-101
CLK2 7-104
CLK6 7-104
CLKO 2-10
CLKO Power Pins 6-19
CLKO1 6-18
CLKO2 6-18
CLKOCR 6-12, 6-39

Clock
BRGCLK 6-17
CLKO 6-18
CLKO2 6-18
Clock Divider 7-107
Clock Glitch 7-139
Clock Synthesizer 9-14
External Components 6-14
General System Clock 6-16
Glitch Detect 7-139
Low-Power Divider 6-15
Oscillator 6-12
Oscillator Prescaler 6-13
PLL 6-14
QUICC Internal Clock Signals 6-15
SIM60 6-12
SIMCLK 6-18
SyncCLK 6-17
Clock 7-137
Clock Divider 7-107
Clock Edge 7-80
Clock Generation 6-12
Clock Glitch 7-139
Clock Synthesizer 9-14
Clocking 9-34
Clocking and Pin Functions 7-314
CLOSE Rx BD 7-148, 7-176, 7-206, 7-227, 7-251, 7-280, 7-297, 7-323
CMAR 7-33
CMR 7-28
Code Compatibility 5-5, 5-10
CODEC 7-91, 7-313
Collision Handling 7-254
Command 7-148
CLOSE Rx BD 7-148, 7-176, 7-206, 7-227, 7-251, 7-280, 7-297, 7-323
Command Set 7-5
ENTER HUNT COMMAND 7-176
ENTER HUNT MODE 7-205, 7-227, 7-251, 7-280, 7-296, 7-297
GRACEFUL STOP TRANSMIT 7-120, 7-148, 7-175, 7-205, 7-226, 7-250
INIT RX PARAMETERS 7-149, 7-176, 7-206, 7-280, 7-297, 7-323
INIT TX AND RX PARAMETERS Command 7-307
INIT TX PARAMETERS 7-148, 7-205, 7-227, 7-251, 7-280, 7-297, 7-323
INIT_IDMA 7-38
Opcodes 7-6
RESET BCS CALCULATION 7-205, 7-218
RESTART TRANSMIT 7-120, 7-148, 7-175, 7-205, 7-226, 7-251, 7-280, 7-297
SEND BREAK 7-280
Sending A Preamble 7-280
SET GROUP ADDRESS 7-251
SPI 7-323
STOP TRANSMIT 7-120, 7-147, 7-226, 7-250, 7-279, 7-297
STOP Transmit 7-175, 7-204
TIMEOUT 7-308
TRANSMIT ABORT REQUEST Command 7-307
Command Execution Latency 7-8
Command Format 5-68
Command Set 7-5
Commands
Command Execution Latency 7-8
Commands in GCI Mode 7-307
Communication Processor Module 1-6, 7-1
Companion Mode 9-31
Compatibility Issues 1-7
Compiler 9-18
Condition Code Register 5-12, 5-17
Condition Test Instructions 5-26
Conditional Branch Instruction Timing Table 5-98
CONFIG 2-9, 2-12, 2-14
CONFIG0 2-9
CONFIG1 2-9
CONFIG1/BCLRO/RAS2DD 6-49
CONFIG2 2-12
Connecting the QUICC to Ethernet 7-239
Connecting the QUICC to LocalTalk 7-199
Control Instruction Timing Table 5-99
CP 7-123
CPIC Programming Model 7-378
CPM 1-6, 7-1, 9-17
CPM Block Diagram 7-2
CPM Interrupt Controller 7-370
CPM Sub-Module Base Addresses 3-3
CPU Space 4-31

CPU32 1-5, 11
CPU32+ 5-1
  Core 1-5
  Instruction Set 5-8
  Programming Model 5-6
  Registers 5-7
    DFC 5-7
    Program Counter 5-7
    SFC 5-7
    Status Register 5-7
    Vector Base Register 5-7
  Serial Logic 5-63, 5-65
CR 7-5
CRC16 7-210
CS 2-10, 6-30, 6-48, 6-67, 6-71
CS0 6-25, 9-5
CSMA/CD 7-235
CSNT40 6-73
CSNTQ 6-73
CSR 7-32
CTS 7-63, 7-101, 7-114, 7-118, 7-120, 7-128, 7-129, 7-130, 7-187, 7-190, 7-192, 7-199, 7-219, 7-223, 7-233, 7-240, 7-266, 7-358
Current Instruction Program Counter 5-58
Cycle Length 6-77

D

D31–D16 2-5
DACK 7-28, 7-33, 7-39
DAPR 7-31
Data Encoding 7-135
Data Length 7-213
Data Movement Instructions 5-19
Data Registers 5-7
Data Strobe 4-4
DBcc instruction 5-3
DEC 7-126
Delayed RTS Mode 7-194
Deriving SWT Timeout 6-36
Deterministic Opcode Tracking 5-59, 5-60, 5-80
Development Tools and Support B-1
DFC 4-36
DHR 7-33
Differential Biphase-L 7-118
Differential Manchester 7-118, 7-136
Digital Phase-Locked Loop (DPLL) 7-135
Disable CPU32+ 6-23
Disabling the SCCs 7-139
Disabling the SMCs on the Fly 7-274
DONE 7-28, 7-33, 7-52
Double Bus Fault 4-48, 5-40
Double Bus Fault Monitor 6-4, 6-9
Double Drive RAS Lines 6-63
DPLL 7-117, 7-118
DPLL Error 7-182
DPLL Operation 7-136
DPLL Receive Block Diagram 7-137
DPLL Transmit Block Diagram 7-137
DPRBASE 3-2
DRAM Banks 6-58
DRAM Bus Error 6-63
DRAM Controller
  Address Multiplexing 6-58
  Bank Parity 6-62
  Bus Error 6-63
  Controller Overview 6-58
  Devices 9-9, 9-46
  Normal Access 6-60
  Overview 6-58
  Page Size 6-65
  Port Size 6-65
  SIMM 9-8, 9-45
  DRAM-SRAM Performance Summary 6-78
  External Master Support 6-63
  Global Memory Register 6-64
  Memory Controller Block Diagram 6-53
  Option Register 6-74
  Page Mode 6-75
  Programming Model 6-64
  Refresh 6-64
  Refresh Cycle 6-64
  Refresh Operation 6-62
DRAM Controller Overview 6-58
DRAM Counter
  Burst Access Support 6-61
  Double Drive RAS Lines 6-63
  Page Mode Support 6-60
DRAM Device 9-9, 9-46
DRAM Normal Access 6-60
DRAM Page Size 6-65
DRAM Port Size 6-65

DRAM SIMM 9-8, 9-45
DRAM-SRAM Performance Summary 6-78
DREQ 7-28, 7-33, 7-40
DS 2-8, 4-4, 6-30
DSACK 2-8, 4-21, 4-41, 6-63, 6-65, 6-67, 7-43
DSCLK 2-11, 5-63
DSI 2-11, 5-63
DSO 2-11, 5-65, 5-66
DSR 7-121
Dual Address Destination Write 7-46
Dual Address Mode 7-45
Dual Address Packing 7-46
Dual Address Source Read 7-45
Dual Address Transfer 7-46
Dual-Port RAM 7-8, 7-9, 9-15 , C-1
  Block Diagram 7-9
  CPM Sub-Module Base Addresses 3-3
  Memory Map 3-2
  Parameter RAM 7-10
  SCC Buffer Descriptors 7-122
  SCC Memory Structure 7-123
  SCC parameter RAM 7-124
  SMC Memory Structure 7-270
  SMC Parameter RAM 7-270
  SMC parameter RAM 7-270
  Transparent Memory Map 7-225
Dump Memory Block 5-70, 5-75
Dynamic Bus Sizing 4-6, 4-19
Dynamic Changes 7-140

E

EDX B-1, B-4
EEPROM 1-12, 7-312, 9-7, 9-45
EEST™ 1-9, 13, 7-236
Effects of Wait States on Instruction Timing 5-86
Enhanced Ethernet Serial Transceiver 1-9, 13
ENTER HUNT MODE 7-148, 7-176, 7-205, 7-227, 7-251, 7-280, 7-296, 7-297
EPROM 9-5, 9-38
Error Stack Frames 5-56, 5-57
Error-Handling 7-154, 7-209
Error-Handling Procedure 7-255
Ethernet 7-114, 7-116, 7-117, 7-178, A-1
  Broadcast Address 7-257
  CAM Interface 7-243
  Collision Handling 7-254
  Connecting QUICC to Ethernet 7-239
  Connecting the QUICC to Ethernet 7-240
  EEST 7-236
  Error-Handling Procedure 7-255
  Ethernet Address Recognition Flowchart 7-253
  Ethernet Block Diagram 7-237
  Ethernet Channel Frame Reception 7-242
  Ethernet Channel Frame Transmission 7-241
  Ethernet Controller 7-235
  Ethernet Memory Map 7-246
  Ethernet on QUICC 7-236
  Ethernet Programming Model 7-250
  Ethernet/802.3 Frame Format 7-235
  First-Time User 7-238
  Hash Table Algorithm 7-253
  Heartbeat 7-257, 7-263
  Internal and External Loopback 7-255
  Interpacket Gap Time 7-254
  Late Collision 7-263
  Learning Ethernet on the QUICC 7-238
  MC68160 7-236
  Nonoctet Aligned 7-261
  PB15–PB8 pins 7-243
  Promiscuous 7-257
  Retry Count 7-263
  Sample One Byte 7-243
  SCC Ethernet Example 7-266
  Serial CAM Interface 7-244
  Short Frame 7-261
  Short Frame Padding 7-262
  SIA 7-236
  Tag Byte 7-245, 7-258
  TCP/IP 7-235
Ethernet Address Recognition 7-252
Ethernet Address Recognition Flowchart 7-253
Ethernet Block Diagram 7-237
Ethernet Channel Frame Reception 7-242
Ethernet Channel Frame Transmission 7-241
Ethernet Command Set 7-250

Ethernet Controller 7-235
Ethernet Memory Map 7-246
Ethernet on QUICC 7-236
Ethernet Programming Model 7-250
Ethernet Receive Buffer Descriptor 7-258
Ethernet Transmit Buffer Descriptor 7-261
EVT 9-16
Exception
Bus Exception 4-41
Exception Handler 5-34, 5-39, 5-46, 5-47, 5-49, 5-51, 5-52, 5-53, 5-54, 5-55
Exception Priorities 5-38
Exception Processing 5-34, 5-39, 5-46
Exception Processing 5-36, 5-38, 5-39, 5-41, 5-43, 5-47, 5-52, 5-57, 5-82
Exception Vectors 5-36, 5-43, 5-59
EXTAL 2-11
External Burst Requests 7-40
External Bus Interface Control 6-21
External Bus Masters 9-18
External Components 6-14
External Cycle Steal 7-42
External Exception 5-36
External Master Support 6-63
External Master Wait State 6-67
External Sync 7-115
External Synchronization Signals 7-223

F

Fast Termination 4-21
Fast-Termination Option 7-50
Fault Address Register 5-63
Fault Correction 5-63
Fault Types 5-51, 5-52
FC 2-5
FC3–FC0 4-3
FCR 7-31
Fetch Effective Address Timing Table 5-90
FIFO 7-3, 7-179, 7-209
FIFO Latency 7-114
FIFO Length 7-114
FIFO Width 7-114
Fill Memory Block 5-76
First-Time User 7-238
Flag Status 7-186
Flash EPROM 9-5, 9-41
F-Line Instructions 5-44
FM0 7-116, 7-118, 7-136, 7-197
FM1 7-116, 7-118, 7-136
FORCE_BGND 5-67
Format Error Exception 5-43, 5-48
Four-Word Stack Frame 5-56
Fractional Stop Bits 7-153
Frame Sync Delay 7-81
Frames Threshold 7-174
FREEZE 5-62, 5-66, 6-31, 7-60
Freeze 2-12
Freeze Support 6-11
Frequency Multiplication 6-14
Function Codes 2-5, 4-3

G

GADDR 7-249
GCI 7-80, 7-96, 7-268
SMC 7-305
GCI Controller 7-305
GCIDCL 7-97
General System Clock 6-16
General-Purpose Chip-Select Overview 6-56
General-purpose Timers 7-17
Glitch 7-185
Glitch Detect 7-112, 7-139
Global (Boot) Chip-select 6-58
Global Chip Select 6-25
Global Memory Register 6-64
Glueless System 1-8
GMR 6-56, 6-64, 9-11, 9-49
GND 2-13
GNDCLK 6-19
GNDS 2-13
GNDSYN 2-13, 6-19
GO 5-63, 5-70, 5-77
GRACEFUL STOP TRANSMIT 7-120, 7-148, 7-175, 7-205, 7-226, 7-250
Grant Support 7-86
Grouped 7-380
GSMR 7-111
GSMR programming 7-196, 7-199

H

HADDR 7-173
HALT 2-10, 4-20, 4-41, 4-46, 7-44, 7-51, 7-

58
Halt Termination 4-41
Hardware Breakpoint 5-37, 5-39, 5-43, 5-56
Hash Table Algorithm 7-253
HDLC 7-114, 7-116, 7-169, 7-178, A-1
DPLL Error 7-182
FIFO 7-177, 7-179
Flag Status 7-186
Frames Threshold 7-174
Glitch 7-185
HDLC Address Recognition Example 7-174
HDLC Channel Frame Reception Processing 7-172
HDLC Channel Frame Transmission Processing 7-171
HDLC Command Set 7-175
HDLC Controller 7-169
HDLC Error-Handling 7-176
HDLC Example 7-187
HDLC Framing Structure 7-171
HDLC Memory map 7-172
HDLC Programming Model 7-174
HDLC Rx BD 7-180
Idle Sequence Status 7-186
Nonoctet Aligned 7-177
HDLC Address Recognition Example 7-174
HDLC Bus 7-178
Accessing the HDLC Bus 7-192
Delayed RTS Mode 7-194
GSMR Programming 7-196
HDLC Bus Controller 7-189
HDLC Bus Controller Example 7-196
HDLC Bus Key Features 7-192
HDLC Bus Memory Map and Programming 7-196
HDLC Bus Multi-Master Configuration 7-191
HDLC Bus Single-Master Configuration 7-191
Non-Symmetrical Duty Cycle 7-193
PSMR Programming 7-196
T1.605 7-190
Transmission Line Configuration 7-194
TSA Transmission line Configuration 7-195
HDLC Bus 1-10, 7-189, A-2
HDLC Bus Collision Detection 7-193
HDLC Bus Controller 7-189
HDLC Bus Controller Example 7-196
HDLC Bus Key Features 7-192
HDLC Bus Memory Map and Programming 7-196
HDLC Bus Multi-Master Configuration 7-191
HDLC Bus Single-Master Configuration 7-191
HDLC Channel Frame Reception Processing 7-172
HDLC Channel Frame Transmission Processing 7-171
HDLC Command Set 7-175
HDLC Controller 7-169
HDLC Error-Handling 7-176
HDLC Example 7-187
HDLC Frames 7-196
HDLC Interrupt 7-185
HDLC Memory Map 7-172
HDLC Programming Model 7-174
HDLC Receive Buffer Descriptor 7-179
HDLC Rx BD 7-180
HDLC Transmit Buffer Descriptor 7-183
Heartbeat 7-257, 7-263
Highest Priority 7-380
HMASK 7-173

I

I/O Pins 7-333
IACK 2-7, 2-8, 6-48
IACK5 4-39
IADDR 7-250
ICCR 7-26
IDL 7-80, 7-90, 7-268
IDL Interface Example 7-91
IDL Interface Programming 7-95
Idle Sequence Status 7-186
Idle Status 7-167
IDMA 6-31, 7-59
IDMA (Independent DMA Controller
External Cycle Steal 7-42
IDMA Controllers 7-24
IDMA (Independent DMA Controller) 7-35, 7-36
Auto Buffer Example 7-56

Dual Address Destination Write 7-46
Dual Address Mode 7-45
Dual Address Packing 7-46
Dual Address Source Read 7-45
Dual Address Transfer Example 7-46
External Burst Requests 7-40
External Cycle Steal 7-42
Fast-Termination Option 7-50
IDMA Bus Arbitration 7-43
IDMA Channels 7-24
IDMA Controller Block Diagram 7-25
IDMA Examples 7-55
IDMA Operand Transfers 7-45
Requesting IDMA Transfers 7-39
Single Address Mode 7-48
Single Address Transfer 7-48
IDMA Bus Arbitration (Normal Operation) 7-44
IDMA Buffer Descriptors 7-36
IDMA Bus Arbitration 7-43, 7-44
IDMA Channels 7-24
IDMA Controller Block Diagram 7-25
IDMA Controllers 7-24
IDMA Examples 7-55
IDMA Operand Transfers 7-45
IDMA Registers 7-26
IEEE 802.3 7-235
IFETCH 2-11, 5-60, 5-63, 5-80, 5-82
IMB 1-4, 10
Immediate Arithmetic/logic Instruction Table 5-95
IN 5-49, 5-50, 5-52
INIT RX PARAMETERS 7-149, 7-176, 7-206, 7-227, 7-251, 7-297, 7-323
INIT RX PARAMETERS 7-280
INIT TX AND RX PARAMETERS Command 7-307
INIT TX PARAMETERS 7-148, 7-205, 7-227, 7-251, 7-280, 7-297, 7-323
INIT_IDMA 7-38
Initialization 9-10
In-Line Synchronization Pattern 7-223
Instruction Execution Overlap 5-85, 5-86
Instruction Execution Time Calculation 5-86
Instruction Pipeline Operation 5-85
Intel 7-126, 7-272, 7-321
Interface Multiple QUICCs to an MC68EC040 9-51
Interface Signals 7-33
Interlocked Data Transfers 7-333
Intermodule Bus 1-4, 10
Internal Accesses 4-59
Internal and External Loopback 7-255
Internal Exceptions 5-36
Internal Memory 3-1
Internal Registers Memory Map 3-4
Interpacket Gap Time 7-254
Interrupt 6-26, 7-370, 9-16, 9-23, 9-34
Autovector 4-6, 4-38
CPM Interrupt Controller 7-370
Grouped 7-380
HDLC Interrupt 7-185
Highest Priority 7-380
Interrupt Acknowledge 4-36
Interrupt Arbitration 6-8, 6-33
Interrupt Handler Examples 7-382
Interrupt In-Service Register 7-381
Interrupt Mask Register 7-381
Interrupt Pending Register 7-380
Interrupt Service Mask 6-33
Interrupt Source Priorities 7-373
Interrupt Vector 6-8
Interrupt Vector Generation 7-376
Interrupts from the SCCs 7-128
IRQx 7-379
Masking Interrupt Sources 7-375
Nested Interrupts 7-374
Periodic Interrupt Request Level Encoding 6-37
QUICC Interrupt Structure 7-372
RXF Interrupt 7-174
SCC Interrupt Handling 7-130
Slave Mode 6-26
SMC Interrupt Handling 7-305
SMC UART 7-289
Spread 7-380
Spurious Interrupt 4-41
UART Interrupt Events Example 7-165
Vector Base Address 7-380
Interrupt Acknowledge 4-36
Interrupt Arbitration 6-8, 6-33
Interrupt Exceptions 5-38
Interrupt Generation 6-6
Interrupt Handler Examples 7-382

Interrupt Handling 7-291, 7-331
Interrupt In-Service Register 7-381
Interrupt Level 7-371
Interrupt Mask Register 7-381
Interrupt Pending Register 7-380
Interrupt Service Mask 6-33
Interrupt Source Priorities 7-373
Interrupt Structure 6-7
Interrupt Vector 6-8
Interrupt Vector Generation 7-376
Interrupts from the SCCs 7-128
Inverted HDLC 7-116
IOM-2 7-268
IOUT 6-49
IOUT2–IOUT0 6-26
IPEND 9-35
IPIPE 2-11, 2-13, 5-60, 5-63, 5-80, 5-82, 6-49
IRQ 2-7, 6-26, 7-379
IRQ1 6-49
ISDN 7-178
  Devices 7-312

J

Jitter 7-115

L

L1CLKOx 7-96
L1GRx 7-93
L1RCLKx 7-93, 7-96
L1RQx 7-79, 7-93
L1RSYNCx 7-80, 7-93, 7-96
L1RXD 7-73, 7-269
L1RXDx 7-93, 7-96
L1STA1 7-73
L1STB1 7-73
L1TXD 7-73, 7-269
L1TXDx 7-93, 7-96
LAPB 7-170, A-1, B-1, B-3
LAPD 7-170, A-1, B-1, B-3
Larger QUICC System 1-9
Late Collision 7-263
Learning Ethernet on the QUICC 7-238
Line Fill 6-78
Little-Endian 7-126, 7-272, 7-321
LocalTalk 1-10, 7-117, 7-196
LocalTalk Frame Format 7-197
Low Power in Normal Operation 6-12
Low-Power Divider 6-15
Low-Power Stop 6-11
LPSTOP 4-35
LRC 7-210

M

M68000 Code 9-18
M68360 QUADS Development System B-6
M68HC05 7-312
M68HC11 7-312
Manchester 7-118, 7-136, 7-235
Masking Interrupt Sources 7-375
Master 7-314
Master Mode 7-315
Master-Slave QUICC 1-17
MAX_IDL 7-145, 7-277
MAXD1 7-249
MAXD2 7-249
MBAR 3-1, 3-2, 4-36, 6-1, 6-3, 6-24, 6-27, 9-14
MBARE 6-25, 6-29
MC145554 7-313
MC68030 4-1, 6-23
MC68030-Type 6-54
MC68040 6-66, 6-77, 9-31
  BR040ID 6-30
  Burst 6-77, 6-78
  Burst Length 6-76
  Bursts 6-72
  Line Fill 6-78
MC68040 Companion Mode 1-5, 1-18, 6-23, 12
MC68160 1-9, 7-236, 13
MC68195 1-10
MC68195 LocalTalk Adaptor 7-197
MC68302 1-1, 1-6, 9-18
MC68332 1-7
MC68340 1-6, 4-1, 12
MC68360 1-1
MC68605 A-1
MC68606 A-1
MC68EC040 1-18, 4-1, 6-30, 6-52, 9-31
MC68EC040 to QUICC Interface 9-32, 9-33
MC68HC11 1-12
MCM36100S 9-8, 9-45

MCM54260 9-9, 9-46
MCM62940A 9-41
MCR 6-29, 9-10
Memory Controller 6-50
Memory Controller Block Diagram 6-53
Memory Interfaces 9-4, 9-37
Memory Map 3-2
MFLR 7-173, 7-248
Microbus Controller 5-83
Microcode 7-4, C-1
Microcode Revision Number 7-5
Microsequencer Operation 5-85
MINFLR 7-248
Minimum QUICC System 1-8
Minimum System Configuration 9-1
Misaligned Accesses 5-1
Misaligned Operands 4-11
Misc Base 3-3
MODCK 2-11
MODCK1–MODCK0 6-19
Monitor Channel 7-309
Monitor Channel Reception 7-307
Monitor Channel Transmission 7-306
Mono-Sync 7-115
Motorola Byte Ordering 7-126, 7-272, 7-321
Motorola Mode 7-126
Motorola Software Modules B-1
MOVE Instruction Timing Table 5-92
MOVEM Faults 5-49, 5-51, 5-52
MOVEP Faults 5-51
MRBLR 7-127, 7-273, 7-322
MSTAT 6-56, 6-69
Multidrop Mode 7-149
Multiprocessor System 5-57

N

NC 2-10, 2-13
Nested Interrupts 7-374
Nibblesync 7-201
NMARC 7-174
NMSI 7-62, 7-117, 7-269
    BRGOx 7-101
Nonoctet Aligned 7-177, 7-261
Non-Symmetrical Duty Cycle 7-193
Normal Asynchronous Mode 7-143
Normal Termination 4-41
NRZ 7-118, 7-135
NRZI 7-118
NRZI Mark 7-116, 7-135
NRZI Space 7-116, 7-135

O

OE 2-9, 4-4, 6-48, 6-49
On-the-Fly Changes 7-140
Opcode Tracking in Loop Mode 5-59
Open-Drain 7-358
Operand Faults 5-52
Operation Field 5-68
Option Register 6-74
OR 6-56, 9-12, 9-50
Oscillator 6-12
Oscillator Prescaler 6-13
Output Enable 4-4

P

P1284 7-331
PADAT 7-360
PADDR 7-249
PADIR 7-360
PADS 7-248
Page Mode 6-75
Page Mode Support 6-60
PAODR 7-360
Parallel I/O 7-129, 7-358
Parallel I/O Port
    Port A 7-359
    Port A Examples 7-361
    Port A Registers 7-360
    Port B Example 7-366
    Port B Pin Functions 7-363
    Port C Registers 7-368
    Port D 6-22
Parallel Interface 1-13
Parallel Interface Port 7-331
Parameter RAM 7-10
Parity 6-57, 6-68, 6-70, 6-71, 7-211
Parity Enable 7-159
PB15–PB8 pins 7-243
PBDAT 7-357, 7-365
PBDIR 7-357, 7-365
PBODR 7-358, 7-365
PBPAR 7-357
PCDAT 7-369

PCDIR 7-369
PCPAR 7-369
PCSO 7-369
PEPAR 6-48, 9-11, 9-15, 9-49
Periodic Interrupt Request Level Encoding 6-37
Periodic Interrupt Timer 6-5, 6-10
PICR 6-37
Pin Differences 6-26
PIP 7-332, 7-338, 7-341, 7-364
  BUSY signal 7-335
  Centronics 7-336
  I/O Pins 7-333
  Interlocked Data Transfers 7-333
  Parallel Interface Port 7-331
  PIP Block Diagram 7-332
  Port B 7-333
  Port B Pin Functions 7-363
  Port B Registers 7-357, 7-365
  Port E 6-23
  Programming Model 7-338
  Pulse Data Transfers 7-334
  Pulsed Handshake Timing 7-336
PIP Block Diagram 7-332
PIPC 7-339
PIPE 7-341
PIPM 7-342
PIT 6-10
PITR 6-38
PLL 6-14
PLL Power Pins 6-19
PLLCR 6-12, 6-40
Port A 7-359
Port A Examples 7-361
Port A Registers 7-360
Port B 7-333
Port B Example 7-366
Port B Pin Functions 7-363
Port B Registers 7-357, 7-365
Port C Registers 7-368
Port D 6-22
Port E 6-23
Port Sizes 4-8
Postincrement 5-5, 5-12
Power-On Reset 9-3
Predecrement 5-5, 5-12
Prefetch Controller 5-84
Prefetch Faults 5-51
Priority of the RISC 7-3
Privilege Violation 5-56
PROFIBUS C-6
Profibus 7-142
Program Control Instructions 5-24
Program Counter 5-2, 5-56, 5-58, 9-13
Programmer's Model 6-27
Programming Model 6-64, 7-317, 7-338
Programming SI RAM Entries 7-72
Programming the BISYNC Controller 7-217
Promiscuous 7-220, 7-257
PRTY 2-6
PSMR 7-120, 7-156, 7-175, 7-178, 7-210, 7-228, 7-256
PSMR Programming 7-196, 7-200
PTPR 7-341
Pulse Data Transfers 7-334
Pulsed Handshake Timing 7-336

Q

Q.921 A-1
QUad Integrated Communication Controller 1-1, 9
QUADS B-6
QUICC AppleTalk Hardware Connection 7-198
QUICC Block Diagram 1-4, 11
QUICC Ethernet Serial CAM Interface 7-244
QUICC Functional Signal Groups 2-2
QUICC Internal Clock Signals 6-15
QUICC Interrupt Structure 7-372
QUICC Key Features 1-1
QUICC Memory Map 3-1, 3-2, 6-4
QUICC System Configuration 6-55

R

R/W 4-3
R/W Field 5-68
RA 2-10
RAM
  DRAM Banks 6-58
  SI RAM 7-68
  SPI Parameter RAM Memory Map 7-320
  SRAM Banks 6-56

RAM Microcode 7-4, 7-10, 7-235
RAS 2-6, 6-30, 6-60, 6-63, 6-64, 6-65
RAS1DD 2-6, 2-11
RAS2 2-10
RAS2DD 2-9, 6-25, 6-49
RBASE 7-125, 7-271, 7-320
RBPTR 7-127, 7-273, 7-322
RCCM 7-204
RCCR 7-4
RCLK 7-101
Read A/D Register 5-71
Read Cycle 4-23
Read Memory Location 5-73
Read System Register Command 5-62
Read-Modify-Write 4-28
Read-Modify-Write Cycle 5-50
Read-Modify-Write Faults 5-51
Real-Time Clock 7-312
Receiver 7-275
Receiver Shortcut 7-276
Refresh 6-64
Refresh Cycle 6-64
Refresh Operation 6-62
Register Field 5-69
Register Map 3-1
Registers 3-1
  AVR 6-34
  BCR 7-31
  BDLE 7-204
  BDLE-BISYNC DLE Register 7-208
  BKAR 6-44
  BKCR 6-44
  BR 6-56, 6-70
  BRGC 7-106
  BSYNC 7-204
  BSYNC-BISYNC SYNC Register 7-207
  CDVCR 6-12, 6-42
  CICR 7-378
  CIMR 7-381
  CIPR 7-380
  CISR 7-381
  CLKOCR 6-12, 6-39
  CMAR 7-33
  CMR 7-28
  CR 7-5
  CS 6-71
  CSR 7-32
  DAPR 7-31
  DFC 4-36
  DHR 7-33
  DPRBASE 3-2
  DSR 7-121
  FCR 7-31
  GADDR 7-249
  GMR 6-56, 6-64
  GSMR 7-111
  HADDR 7-173
  HMASK 7-173
  IACK 6-48
  IADDR 7-250
  ICCR 7-26
  Internal Registers Memory Map 3-4
  MAXD1 7-249
  MAXD2 7-249
  MBAR 3-1, 3-2, 6-1, 6-3, 6-27
  MBARE 6-29
  MCR 6-29
  MFLR 7-173, 7-248
  MINFLR 7-248
  MRBLR 7-127, 7-273, 7-322
  MSTAT 6-56, 6-69
  NMARC 7-174
  OR 6-56
  PADAT 7-360
  PADDR 7-249
  PADIR 7-360, 7-365
  PADS 7-248
  PAODR 7-360
  PAPAR 7-361
  PBDAT 7-358, 7-365
  PBDIR 7-357
  PBODR 7-358, 7-365
  PBPAR 7-357, 7-366
  PCDAT 7-369
  PCDIR 7-369
  PCPAR 7-369
  PCSO 7-369
  PEPAR 6-48
  PIPC 7-339
  PIPE 7-341
  PIPM 7-342
  PITR 6-38
  PLLCR 6-12, 6-40
  PSMR 7-120, 7-156, 7-175, 7-178, 7-

210, 7-228, 7-256
PTPR 7-341
RBASE 7-125, 7-271, 7-320
RBPTR 7-127, 7-274, 7-322, 7-323
RCCM 7-204
RET_Lim 7-248
RFCR 7-125, 7-272, 7-321
RFTHR 7-174
RSR 6-34
RTER 7-14
RTMR 7-14
SAPR 7-30
SCCE 7-128, 7-164, 7-184, 7-216, 7-232, 7-264
SCCM 7-129, 7-186, 7-217, 7-233, 7-265
SCCS 7-129, 7-187, 7-217, 7-233, 7-265
SCCs 7-167
SDAR 7-61
SDCR 7-59
SDSR 7-61
SFC 4-36
SICMR 7-87
SICR 7-86
SIGMR 7-77
SIMODE 7-78
SIRP 7-88
SISTR 7-87
SMCE 7-288, 7-311
SMCM 7-290
SMCMR 7-270, 7-281, 7-298, 7-308
SPCOM 7-315, 7-319
SPIE 7-328
SPIM 7-329
SWIV 6-35
SWSR 6-39
SYPCR 6-35
TADDR 7-250
TBASE 7-125, 7-271, 7-320
TBPTR 7-127, 7-175, 7-274
TCN 7-22
TCR 7-22
TER 7-22
TFCR 7-125, 7-272, 7-321
TGCR 7-20
TMR 7-21
TODR 7-121, 7-175, 7-200
TRR 7-22
Registers PICR 6-37
Released Write 5-41
Requesting IDMA Transfers 7-39
RESET 4-65, 6-25
Instruction 5-35, 5-40, 5-70
BCS Calculation 7-205, 7-218
Signal 5-40
Reset 4-63, 9-14
Exception 5-39
Peripherals 5-70, 5-79
Power-On Reset 9-3
Status 6-3
Strategy 9-34
Value 3-5
RESETH 2-10, 4-64
RESETS 2-10, 4-64
RESTART TRANSMIT 7-120, 7-148, 7-175, 7-205, 7-226, 7-251, 7-280, 7-297
RET_Lim 7-248
Retry 4-44
Count 7-263
Termination 4-41
Return Program Counter 5-58, 5-63
Returning from BDM 5-63
Reverse Data 7-112, 7-211
RFCR 7-125, 7-272, 7-321
RFTHR 7-174
RISC Controller 7-3
RISC Microcode from RAM C-1
RISC Processor C-1
RISC Timer Initialization Sequence 7-14
RISC Timer Table Application 7-16
RISC Timer Tables 7-11
RM Bit 5-50
RMC 2-9, 4-3, 4-28, 4-54
RQOUT 6-26, 6-49
RR bit 5-50
RRJCT 7-244
RS-232 7-142
RSR 6-34
RSREG 5-70
RSTRT 7-243
RTE 5-16, 5-26, 5-50
RTE Instruction 5-53, 5-55
RTER 7-14
RTMR 7-14

RTS 7-63, 7-101, 7-114, 7-119, 7-130, 7-187, 7-219, 7-224, 7-233, 7-240, 7-266, 7-358, 7-364
RW Bit 5-50
RXD 7-101, 7-358

S

S/T 7-96
S/T Transceiver 7-91
Sample One Byte 7-243
SAPR 7-30
Save and Restore Operations Timing Table 5-101
Saving Power 7-141
SCC 7-101, 7-109, A-1
  Asynchronous Protocols 7-134
  Digital Phase-Locked Loop (DPLL) 7-135
  Disabling the SCCs 7-139
  Dynamic Changes 7-140
  FIFOs 7-3
  On-the-Fly Changes 7-140
  Saving Power 7-141
  SCC Receiver Full Sequence 7-140
  SCC Receiver Shortcut Sequence 7-141
  SCC Transmitter Full Sequence 7-140
  SCC Transmitter Shortcut Sequence 7-140
  Switching Protocols 7-141
  SyncCLK 7-108
  Synchronous Protocols 7-130
SCC BISYNC Example 7-218
SCC Block Diagram 7-111
SCC Buffer Descriptors 7-122
SCC Ethernet Example 7-266
SCC FIFOs 7-3
SCC Initialization 7-129
SCC Interrupt Handling 7-130
SCC Memory Structure 7-123
SCC Parameter RAM 7-124
SCC Receiver Full Sequence 7-140
SCC Receiver Shortcut Sequence 7-141
SCC Timing control 7-130
SCC Transmitter Full Sequence 7-140
SCC Transmitter Shortcut Sequence 7-140
SCC Transparent Example 7-233
SCC UART Example 7-167
SCC UART Rx BD Example 7-160
SCCE 7-128, 7-164, 7-184, 7-216, 7-232, 7-264
SCCM 7-129, 7-167, 7-186, 7-217, 7-233, 7-265
SCCS 7-129, 7-187, 7-217, 7-233, 7-265
SCCs 7-167, 9-26
SCIT 7-81, 7-96, 7-98, 7-268
SCIT Programming 7-98
SCP 1-13, 7-93, 7-313
SCSI 9-54
SDACK 7-245
SDAR 7-61
SDCR 7-59
SDLC 1-11, 7-170
SDMA 6-31
SDMA Channel
  SDMA Data Paths 7-58
SDMA Channels 7-57
SDMA Data Paths 7-58
SDMA Registers 7-59
SDSR 7-61
SEND BREAK 7-280
Sending a Preamble 7-153, 7-280
Sending Transparent Frames Between QUICCs 7-225
Serial Communication Controllers 7-109
Serial Interface 5-62, 5-63
Serial Interface with Time Slot Assigner 7-62
Serial Management Controllers 7-268
Serial Performance A-1
Serial Peripheral Interface 7-312
SET GROUP ADDRESS 7-251
SET TIMER 7-14
SFC 4-36
Shadow RAM 7-76, 7-87
Short Frame 7-261
Short Frame Padding 7-262
Show Cycles 4-62
SI 7-269
SI (Serial Interface
  Clock Edge 7-80
  Shadow RAM 7-76
  Strobes 7-88
SI (Serial Interface)
  BRG 7-86

CCITT 1.460 7-97
CLK 7-86
Frame Sync Delay 7-81
GCI 7-80, 7-96
GCIDCL 7-97
Grant Support 7-86
IDL 7-80, 7-90
IDL Interface Example 7-91
IDL Interface Programming 7-95
L1CLKOx 7-97
L1GRx 7-93
L1RCLKx 7-93, 7-96
L1RQx 7-79, 7-93
L1RSYNCx 7-80, 7-93, 7-96
L1RXDx 7-73, 7-93, 7-96
L1STA1 7-73
L1STB1 7-73
L1TXDx 7-73, 7-93, 7-96
Programming SI RAM Entries 7-72
S/T 7-96
SCIT 7-96, 7-98
SCIT Programming 7-98
SCP 7-93
Shadow RAM 7-87
SI RAM 7-68
SI RAM Dynamic Changes 7-75
SI RAM Programming Example 7-75
SMC 7-99
SPI 7-93
Strobes 7-73
TSA 7-170
SI GCI Programming 7-98
SI GCI Support 7-96
SI RAM 7-68
SI RAM Dynamic Changes 7-75
SI RAM Programming Example 7-75
SI Registers 7-77
SIA 7-236
SICMR 7-87
SICR 7-86
SIGMR 7-77
Signal 2-2
Bus Control Signals 2-7
Parity 2-6
Signal Descriptions 2-1
Signals 4-64
16BM 2-6
A 6-48
A26–A0 2-1
A31–A28 2-1
ACK 7-333
Address Bus 2-1
AMUX 2-7, 2-9, 6-48, 6-49
AS 2-8, 2-13, 4-3, 6-30, 6-63, 7-41, 7-44, 7-58
AVEC 2-8, 4-6, 4-41, 6-26, 6-48
AVECO 6-26, 6-48
BCLRI 4-58, 6-25, 6-33
BCLRIID 6-25
BCLRO 2-9, 2-10, 6-25, 6-31, 6-33, 7-44, 7-51, 7-58
BERR 2-10, 4-6, 4-20, 4-41, 7-51
BG 2-9, 2-10, 4-49, 4-53, 6-26, 6-31, 6-32, 7-44
BGACK 2-9, 4-49, 6-26, 6-31, 7-44
BKPT 2-11, 4-31
BKPTO 6-26
BR 2-9, 2-10, 4-49, 4-52, 6-26, 6-31, 6-32, 7-44
BRG 7-101
BRGO 7-108, 7-364
BUSY Signal 7-335
CAS 2-7, 6-48, 6-60, 6-63, 6-67
CASx 2-13
CD 7-63, 7-101, 7-114, 7-118, 7-128, 7-129, 7-130, 7-187, 7-199, 7-219, 7-223, 7-229, 7-233, 7-240, 7-266, 7-358
Chip-Select 2-6
CLK 2-11, 2-13, 7-101
CLK2 7-104
CLK6 7-104
CLKO 2-10
CLKO Power Pins 6-19
CONFIG 2-9, 2-12, 2-14
CONFIG0 2-9
CONFIG1 2-9
CONFIG1/BCLRO/RAS2DD 6-49
CONFIG2 2-12
CS 2-6, 2-10, 6-30, 6-48, 6-67
CS0 9-5
CSO 6-25
CTS 7-63, 7-101, 7-114, 7-118, 7-120, 7-128, 7-129, 7-130, 7-187, 7-

190, 7-192, 7-199, 7-219, 7-223, 7-233, 7-240, 7-266, 7-358
D31–D16 2-5
DACK 7-28, 7-33, 7-39
DONE 7-28, 7-33, 7-52
Double Drive RAS Lines 6-63
DREQ 7-28, 7-33, 7-40
DS 2-8, 4-4, 6-30
DSACKx 2-8, 4-21, 4-41, 6-63, 6-65, 6-67, 7-43
DSCLK 2-11
DSI 2-11
DSO 2-11
EXTAL 2-11
FC 2-5
FC–FC0 4-3
FREEZE 6-31, 7-60
Freeze 2-12
Function Code 2-5
GND 2-13
GNDCLK 6-19
GNDS 2-13
GNDSYN 2-13, 6-19
HALT 2-10, 4-20, 4-41, 6-25, 7-44, 7-51, 7-58
IACK 2-7, 2-8
IACK5 4-39
IFETCH 2-11
Interface Signals 7-33
IOUT2–IOUT0 6-26
IPEND 9-35
IPIPE 2-11, 2-13, 6-49
IRQ1 6-49
IRQx 2-7, 6-26, 7-379
L1CLKOx 7-97
L1GRx 7-93
L1RCLKx 7-93, 7-96
L1RQx 7-79, 7-93
L1RSYNCx 7-80, 7-93
L1RXDx 7-73, 7-93, 7-96, 7-269
L1STA1 7-73
L1STB1 7-73
L1TXDx 7-73, 7-93, 7-96, 7-269
MBARE 6-25
MODCK 2-11
MODCK1–MODCK0 6-19
NC 2-10, 2-13
OE 2-9, 4-4, 6-48, 6-49
PLL Power Pins 6-19
PRTY 2-6
QUICC Functional Signal Groups 2-2
R/W 4-3
RA 2-10, 6-66
RAS 2-6, 6-30, 6-60, 6-63, 6-64, 6-65
RAS1DD 2-6, 2-11
RAS2 2-10
RAS2DD 2-9, 6-25, 6-49
RCLK 7-101
RESET 6-25
RESETH 2-10
RESETS 2-10, 4-64
RMC 2-9, 4-3, 4-28, 4-54
RQOUT 6-26, 6-49
RRJCT 7-244
RSTRT 7-243
RTS 7-63, 7-101, 7-114, 7-119, 7-130, 7-187, 7-219, 7-224, 7-233, 7-240, 7-266, 7-358, 7-364
RXD 7-101, 7-358
SDACK 7-245
SIZ 2-8, 4-3, 4-8, 4-25
SMCLK 7-103, 7-269, 7-294
SMRXD 7-103, 7-269
SMSYN 7-269, 7-293
SMTXD 7-103, 7-269
SPICLK 7-315
SPIMISO 7-314, 7-315
SPIMOSI 7-314
SPISEL 7-324, 7-326, 7-327
STB 7-333
STBI 7-333
STBO 7-333
TA 6-63, 9-32
TBI 9-32
TCK 2-12
TCLK 7-101
TDI 2-12
TDM 7-358
TDO 2-12
TGATE 7-19
TIN 7-18
TMS 2-12
TOUT 7-18
TRIS 2-12, 6-26

TRST 2-12
TS 6-27, 6-60, 6-63, 6-68, 9-32
TXD 7-101, 7-358
VCC 2-13
VCCCLK 6-19
VCCSYN 2-13, 6-19
WE 2-9, 4-27, 6-48
WE3–WE0 2-1
XFC 2-11, 6-19
XTAL 2-10, 2-11
SIM (System Integration Module)
Breakpoint Logic 6-20
Bus Monitor 6-4, 6-8
Clock Generation 6-12
CSNT40 6-73
CSNTQ 6-73
Double Bus Fault Monitor 6-4
DRAM Controller Overview 6-58
External Bus Interface Control 6-21
Freeze Support 6-11
General-Purpose Chip-Select Overview 6-56
Interrupt Generation 6-6
Interrupt Structure 6-7
Low Power in Normal Operation 6-12
Low-Power Stop 6-11
MBAR 6-1
Memory Controller 6-50
Option Register 6-74
Periodic Interrupt Timer 6-5, 6-10
Programming Model 6-64
QUICC Internal clock signals 6-15
QUICC Memory Map 6-4
QUICC System Configuration 6-55
Reset Status 6-3
SIM40 6-1
Simultaneous SIM60 Interrupt Sources 6-8
Slave (Disable CPU32+) Mode 6-23
Software Watchdog Timer 6-5
Spurious Interrupt Monitor 6-5, 6-8
System Configuration 6-5
System Configuration and Protection 6-3
TRLXQ 6-74
SIM40 6-1
SIM60 1-5, 4-20, 6-1, 6-12
SIMCLK 6-18
SIMODE 7-78
Simple Driver Module B-2
Simultaneous SIM60 Interrupt Sources 6-8
Single Address Mode 7-48
Single Address Transfer Example 7-48
Single Buffer 7-34
SIRP 7-88
SISTR 7-87
SIZ 2-8, 4-3, 4-8, 4-25
Slave 7-314
Slave Disable CPU32+) Mode 6-23
Slave Mode 2-14, 7-316
BR 9-50
Burst EPROM 9-38
Burst SRAM 9-41
Bus Arbitration 4-55, 9-35
Bus Clear 6-25
Bus Exceptions 4-59
Cache Modes on the MC68EC040 9-51
Disable CPU32+ 6-23
DRAM Devices 9-46
DRAM SIMM 9-45
EEPROM 9-45
EPROM 9-38
Flash EPROM 9-41
Global Chip Select 6-25
GMR 9-49
Interfacing Multiple QUICCs to an MC68EC040 9-51
Interrupts 6-26, 9-34
MC68EC040 to QUICC Interface 9-32
Memory Interfaces 9-37
OR 9-50
PEPAR 9-49
Pin Differences 6-26
Reset Strategy 9-34
Software Configuration 9-48
SRAM 9-41
Strategy 9-34
Slave Mode 6-26, 9-31
SMC 7-99, 7-103, 7-276, 7-291, 7-305, A-2
C/I Channel 7-307
C/I Channel Receive Buffer Descriptor 7-310
C/I Channel Transmit Buffer 7-311
Character Length 7-282, 7-298
Commands in GCI Mode 7-307

Disabling the SMCs on the Fly 7-274
ENTER HUNT MODE 7-296
GCI 7-268
GCI Çontroller 7-305
IDL 7-268
Interrupt Handling 7-291
IOM-2 7-268
L1RXD 7-269
L1TXD 7-269
MAX_IDL 7-277
Monitor Channel 7-309
Monitor Channel Reception 7-307
Monitor Channel Transmission 7-306
NMSI 7-269
Receiver 7-275
Receiver Shortcut 7-276
SCIT 7-268
SI 7-269
SMC GCI Mode Register 7-308
SMC Memory Structure 7-270
SMC Transparent NMSI Example 7-303
SMC UART Memory Map 7-277
SMC UART Mode Register 7-281
SMCLK 7-269, 7-294
SMCM 7-312
SMRXD 7-269
SMSYN 7-269, 7-293
SMTXD 7-269
Switching Protocols 7-276
Synchronization with the SMSYNx Pin 7-294
Synchronization with the TSA 7-296
T1 7-268
TDM 7-268
Transmitter 7-275
Transmitter Shortcut 7-275
Transparent Command Set 7-297
Transparent Controller 7-291
Transparent Error-Handling Procedure 7-298
Transparent Protocol 7-268
Transparent Reception Processing 7-293
Transparent Transmission 7-292
Transparent Transmitter 7-292
Transparent TSA Example 7-304
UART 7-268, 7-276
UART Command Set 7-279
UART Error-Handling Procedure 7-281
UART Example 7-290
UART Programming Model 7-279
UART Reception 7-279
UART Rx BD Example 7-286
UART transmitter 7-278
SMC Block Diagram 7-269
SMC Buffer Descriptors 7-270
SMC GCI Mode Register 7-308
SMC Interrupt Handling 7-305
SMC Memory Structure 7-270
SMC Parameter RAM 7-270
SMC Transparent Mode Register 7-298
SMC Transparent Receive Buffer Descriptor 7-299
SMC Transparent Transmit Buffer Descriptor 7-300
SMC UART 7-276, 7-289
SMC UART Memory Map 7-277
SMC UART Mode Register 7-281
SMC UART Receive Buffer Descriptor 7-283
SMC UART Transmit Buffer Descriptor 7-286
SMCE 7-288, 7-311
SMCLK 7-103, 7-269, 7-294
SMCM 7-290, 7-312
SMCMR 7-270, 7-281, 7-298, 7-308
SMRXD 7-103, 7-269
SMSYN 7-269, 7-293
SMTXD 7-103, 7-269
Snooping 9-36
Software Breakpoints 5-43
Software Compatibility 1-7
Software Configuration 9-10, 9-48
Software Test-Drive 9-13
Software Watchdog Timer 6-5, 6-9
SPCOM 7-315, 7-319
Special Status Word 5-42, 5-48
Special-Purpose MOVE Instruction Table 5-92
SPI 1-12, 1-13, 7-91, 7-93, 7-312, 7-323
Clocking and Pin Functions 7-314
Interrupt Handling 7-331
Master Mode 7-315
Programming Model 7-317

Slave 7-314
Slave Mode 7-316
SPI Baud Rate Generator 7-314
SPI Block Diagram 7-313
SPI FIFO 7-3
SPI Master Example 7-329
SPI Mode Register 7-317
SPI Slave Example 7-330
SPICLK 7-315
SPIMOSI 7-314
SPISEL 7-324, 7-326, 7-327
Start Transmit 7-320
Transfer Format 7-319
Transmit/Receive 7-315
SPI Baud Rate Generator 7-314
SPI Block Diagram 7-313
SPI Buffer Descriptor Ring 7-324
SPI FIFO 7-3
SPI Master Example 7-329
SPI Mode Register 7-317
SPI Parameter RAM Memory Map 7-320
SPI Receive Buffer Descriptor 7-324
SPI Slave Example 7-330
SPI Transmit Buffer Descriptor 7-326
SPICLK 7-315
SPIE 7-328
SPIM 7-329
SPIMISO 7-314, 7-315
SPIMOSI 7-314
SPISEL 7-324, 7-326, 7-327
SPIMISO 7-314
Spread 7-380
Spurious Interrupt 4-41
Spurious Interrupt Monitor 6-5, 6-8
SRAM 9-6, 9-41
SRAM Banks 6-56
SRAM Port Size 6-75
S-Records 7-169
Stack Frames 5-56
Stack Pointer 5-57, 5-72, 9-13
Start Transmit 7-320
State Diagram 4-55
Status Register 5-56, 5-58
STB 7-333
STBI 7-333
STBO 7-333
STOP TRANSMIT 7-120, 7-147, 7-175, 7-204, 7-226, 7-250, 7-279, 7-297
Stopped Processing State 5-34
Strategy 9-34
Strobes 7-73, 7-88
Sum Check 7-210
Supervisor Data Space 6-32
Supervisor Mode 9-13
Switching Protocols 7-141, 7-276
SWIV 6-35
SWSR 6-39
SYNC 6-67, 6-68
Sync 7-115
SyncCLK 6-17, 7-67, 7-108
Synchronization with the SMSYNx Pin 7-294
Synchronization with the TSA 7-296
Synchronous 4-21
Synchronous Mode 7-144
Synchronous Protocols 7-130
Synchronous UART 7-158
SYPCR 6-35
System Configuration 6-5
System Configuration and Protection 6-3
System Control Instructions 5-25
System Integration Module 1-5, 6-1
System Protection 9-15

T

T1 7-116, 7-268
T1 Line 1-15, 14
T1.605 7-190
TA 6-63, 6-66, 9-32
TADDR 7-250
Tag Byte 7-245, 7-258
TBASE 7-125, 7-271, 7-320
TBI 9-32
TBPTR 7-127, 7-175, 7-274, 7-323
TCK 2-12
TCLK 7-101
TCN 7-22
TCP/IP 7-235
TCR 7-22
TDI 2-12
TDM 7-268, 7-358
TDM Channels 7-170
TDM Interface 1-15, 13
TDO 2-12

TER 7-22
TFCR 7-125, 7-272, 7-321
TGATE 7-19
TGCR 7-20
Thermal Characteristics 10-2
Timeout 7-308
Timer Block Diagram 7-17
Timer Examples 7-23
Timers 7-17, 9-25
  Cascaded Mode 7-19
  Deriving SWT Timeout 6-36
  General-Purpose Timer 7-17
  RISC Timer Tables 7-11
  Timer Block Diagram 7-17
  Timer Examples 7-23
  TM_BASE 7-13
  Wake-Up Timer 7-151
Time-Slot 1-15, 14
TIN 7-18
TM_BASE 7-13
TMR 7-21
TMS 2-12
TODR 7-121, 7-175, 7-200
Totally Transparent 7-220
TOUT 7-18
TP Bit 5-49
TR Bit 5-49, 5-51, 5-52, 5-53
Trace 5-37, 5-39
Trace Exception 5-39, 5-42, 5-45, 5-46, 5-49, 5-51, 5-56
Trace Mode 5-7
Trace on Instruction Execution 5-59
Transfer Format 7-319
Transition to BDM 5-62
Transmission Line Configuration 7-194
TRANSMIT ABORT REQUEST Command 7-307
Transmit on Demand 7-121
Transmit/Receive Process 7-315
Transmitter 7-275
Transmitter Shortcut 7-275
Transmitting and Receiving 7-208
Transparent 7-115, 7-116, A-1
  4-Bit SYNC 7-223
  Achieving Synchronization in Transparent Mode 7-223
  External Synchronization Signals 7-223
  In-Line Synchronization Pattern 7-223
  PIP 7-338
  Promiscuous 7-220
  SCC Transparent Example 7-233
  Sending Transparent Frames Between QUICCs 7-225
  SMC 7-291
  Totally Transparent 7-220
  Transparent Channel Frame Reception Processing 7-222
  Transparent Channel Frame Transmission Processing 7-221
  Transparent Command Set 7-226
  Transparent Controller 7-220
  Transparent Error-Handling Procedure 7-227
  Transparent Receive Buffer Descriptor 7-228
Transparent Channel Frame Reception Processing 7-222
Transparent Channel Frame Transmission Processing 7-221
Transparent Command Set 7-226, 7-297
Transparent Controller 7-220, 7-291
Transparent CRC 7-112
Transparent Error-Handling Procedure 7-227, 7-298
Transparent Memory Map 7-225
Transparent Mode Register 7-228
Transparent NMSI Example 7-303
Transparent Protocol 7-268
Transparent Receive Buffer Descriptor 7-228
Transparent Reception Processing 7-293
Transparent Transmit Buffer Descriptor 7-230
Transparent TSA Example 7-304
TRAP instruction 5-39
TRIS 2-12, 6-26
TRLXQ 6-74
TRR 7-22
TRST 2-12
TS 6-26, 6-60, 6-63, 6-68, 9-32
TSA 7-170
  Using the TSA 7-195
TSA Transmission Line Configuration 7-195
TXD 7-101, 7-358

U

UART 7-105, 7-114, 7-117, 7-118, 7-268, 7-276, A-2
- Auto Baud 7-166
- Break 7-166
- Break Support 7-151
- Error-Handling 7-154
- Fractional Stop Bits 7-153
- Idle Status 7-167
- MAX_IDL 7-145
- Multidrop Mode 7-149
- Normal Asynchronous Mode 7-144
- Parity Enable 7-159
- SCC UART Example 7-167
- SCC UART Rx BD Example 7-160
- Sending a Preamble 7-153
- SMC 7-276
- Synchronous Mode 7-144
- Synchronous UART 7-158
- UART Address Recognition 7-149
- UART Character Format 7-142
- UART Command Set 7-147
- UART Control Characters 7-150
- UART Controller 7-141
- UART Memory Map 7-145
- UART Mode Register 7-156
- UART Multidrop Operation 7-150
- UART Programming Model 7-147
- XOFF 7-152, 7-169
- XON 7-152, 7-169

UART Character Format 7-142
UART Command Set 7-147, 7-279
UART Controller 7-141
UART Encoding/Decoding 7-138
UART Error-Handling Procedure 7-281
UART Example 7-290
UART Interrupt Events Example 7-165
UART Key Features 7-143
UART LAN 1-11
UART Memory Map 7-145
UART Mode Register 7-156
UART Multidrop Operation 7-150
UART Programming Model 7-147, 7-279
UART Receive Buffer Descriptor 7-159
UART Reception 7-279
UART Rx BD Example 7-286
UART Transmission 7-278
UART Transmit Buffer Descriptor 7-163
Unimplemented Instructions 5-10, 5-44, 5-59
UNLK 5-9, 5-17, 5-34
User Privilege Level 5-7, 5-35
Using the TSA 7-195

V

V.120 7-91
V.14 7-158
VBR 9-14
VCC 2-13
VCCCLK 6-19
VCCSYN 2-13, 6-19
Vector
- Software Watchdog Timer 6-9

Vector Base Address 7-380
Vector Base Register 5-4, 5-11, 5-36, 5-43, 5-72

W

Wait State 6-56, 6-67
Wait States 4-26, 4-28
Wake-Up Timer 7-151
Watchdog Timer 9-26
WE 2-1, 2-9, 4-27, 6-48
Word Port 4-9, 4-14
Write A/D Register 5-70, 5-71
Write Cycle 4-26
Write Memory Location 5-74
Write Pending Buffer 5-83
Write Protect 6-56
Write Protection 6-71
Write System Register 5-72

X

X.21 7-114
X.25 A-1, B-1, B-4
XFC 2-11, 6-19
XOFF 7-152, 7-169
XON 7-152, 7-169
XTAL 2-10, 2-11