MC68F375

REFERENCE MANUAL

Revised 15 October 2000

Digital DNA™ from Motorola

THE HEART OF SMART.

© Copyright 2000 MOTOROLA; All Rights Reserved

MC68F375

REFERENCE MANUAL

Revised 15 October 2000

© Copyright 2000 MOTOROLA; All Rights Reserved

Paragraph Number

TABLE OF CONTENTS

Page Number

PREFACE

Section 1
OVERVIEW DESCRIPTION

1.1 Introduction 1-1
1.2 MC68F375 Feature List 1-1
1.3 Module Descriptions. 1-3
1.3.1 Central Processing Unit Module – CPU32 1-3
1.3.2 Single Chip Integration Module – SCIM2E 1-3
1.3.3 Queued Analog to Digital Converter Module – QADC64 1-3
1.3.4 Analog Multiplexer – AMUX 1-4
1.3.5 Queued Serial Multi-Channel Communications Module – QSMCM 1-4
1.3.6 TouCAN Module 1-4
1.3.7 Enhanced Time Processing Unit – TPU3 1-4
1.3.8 DPTRAM TPU Emulation RAM Module – DPTRAM 1-5
1.3.9 1T Flash Electrically Erasable Read Only Memory – CMFI 1-5
1.3.10 Static RAM – SRAM 1-5
1.3.11 Mask Programmable Read Only Memory – ROM 1-5
1.3.12 Configurable Timer Module 9 – CTM9 1-6
1.4 Referenced Documents 1-6
1.5 MC68F375 Functional Block Diagram 1-7
1.6 MC68F375 Pin Usage. 1-8
1.7 Address Map 1-9
1.7.1 Address Bus Decoding 1-9

Section 2
SIGNAL DESCRIPTIONS

2.1 Pin Characteristics 2-1
2.2 Pinout 2-7
2.2.1 Pinout Diagram. 2-7

Section 3
CENTRAL PROCESSOR UNIT

3.1 General 3-1
3.2 CPU32 Registers 3-2
3.2.1 Data Registers 3-4
3.2.2 Address Registers 3-5
3.2.3 Program Counter 3-6
3.2.4 Control Registers 3-6
3.2.4.1 Status Register. 3-6
3.2.4.2 Alternate Function Code Registers. 3-7

Paragraph Number		Page Number

3.2.5 Vector Base Register (VBR) . . . 3-7
3.3 Memory Organization . . . 3-7
3.4 Virtual Memory . . . 3-9
3.5 Addressing Modes . . . 3-9
3.6 Processing States . . . 3-9
3.7 Privilege Levels . . . 3-10
3.8 Instructions . . . 3-10
3.8.1 M68000 Family Compatibility . . . 3-15
3.8.2 Special Control Instructions . . . 3-15
3.8.2.1 Low-Power Stop (LPSTOP) . . . 3-15
3.8.2.2 Table Lookup and Interpolate (TBL) . . . 3-15
3.8.2.3 Loop Mode Instruction Execution . . . 3-15
3.9 Exception Processing . . . 3-16
3.9.1 Exception Vectors . . . 3-16
3.9.2 Types of Exceptions . . . 3-18
3.9.3 Exception Processing Sequence . . . 3-18
3.10 Development Support . . . 3-18
3.10.1 M68000 Family Development Support . . . 3-19
3.10.2 Background Debug Mode . . . 3-19
3.10.3 Enabling BDM . . . 3-20
3.10.4 BDM Sources . . . 3-20
3.10.4.1 External $\overline{\text{BKPT}}$ Signal . . . 3-21
3.10.4.2 BGND Instruction . . . 3-21
3.10.4.3 Double Bus Fault . . . 3-21
3.10.4.4 Peripheral Breakpoints . . . 3-21
3.10.5 Entering BDM . . . 3-21
3.10.6 BDM Commands . . . 3-22
3.10.7 Background Mode Registers . . . 3-23
3.10.7.1 Fault Address Register (FAR) . . . 3-23
3.10.7.2 Return Program Counter (RPC) . . . 3-23
3.10.7.3 Current Instruction Program Counter (PCC) . . . 3-24
3.10.8 Returning from BDM . . . 3-24
3.10.9 Serial Interface . . . 3-24
3.10.10 Recommended BDM Connection . . . 3-26
3.10.11 Deterministic Opcode Tracking . . . 3-26
3.10.12 On-Chip Breakpoint Hardware . . . 3-27

Section 4
SINGLE-CHIP INTEGRATION MODULE 2 (SCIM2E)

4.1 Overview . . . 4-1
4.2 System Configuration . . . 4-2
4.2.1 SCIM Module Configuration Register . . . 4-2
4.2.2 Module Mapping . . . 4-4

Paragraph Number		Page Number

4.2.3 Interrupt Arbitration 4-4
4.2.4 Noise Reduction in Single-Chip Mode 4-4
4.2.5 Show Internal Cycles 4-5
4.2.6 FREEZE Assertion Response 4-5
4.3 System Clock 4-5
4.3.1 System Clock Sources 4-6
4.3.2 Clock Synthesizer Submodule 4-6
4.3.3 Slow Reference Mode 4-8
4.3.4 Fast Reference Mode 4-11
4.3.5 External Clock Mode 4-13
4.3.6 Clock Synthesizer Control Register 4-14
4.3.6.1 Frequency control Bits (X,W,Y) 4-15
4.3.6.2 E Clock Divide Rate (EDIV) 4-15
4.3.6.3 Loss of Clock Oscillator Disable (LOSCD) 4-15
4.3.6.4 Limp Mode (SLIMP) 4-15
4.3.6.5 Synthesizer Lock (SLOCK) 4-16
4.3.6.6 Reset Enable (RSTEN) 4-16
4.3.6.7 Low Power Stop Mode SCIM2 Clock (STSCIM) 4-16
4.3.6.8 Low Power Stop Mode External Clock (STEXT) 4-16
4.3.7 Clock Circuits Operation 4-16
4.3.7.1 Synthesizer Circuit 4-16
4.3.7.2 Phase Comparator and Filter 4-16
4.3.7.3 Lock Detect Circuit 4-18
4.3.7.4 Clock Control Circuit 4-18
4.3.7.5 Loss Of Clock Detect Circuit (LOC) 4-18
4.3.8 Basic Operation 4-18
4.3.8.1 POR Characteristics 4-19
4.3.8.2 External Clock Operating Mode 4-19
4.3.8.3 PLL Operating Mode 4-19
4.3.8.4 RSTEN Bit Operation 4-19
4.3.8.5 Reset Conditions 4-19
4.3.8.6 Low Power Operation 4-20
4.3.8.7 Loss of Reference Signal 4-22
4.4 System Protection 4-23
4.4.1 System Protection Control Register 4-24
4.4.2 Reset Status 4-25
4.4.3 Bus Monitor 4-25
4.4.4 Halt Monitor 4-25
4.4.5 Spurious Interrupt Monitor 4-26
4.4.6 Software Watchdog 4-26
4.4.6.1 Software Watchdog Service Register 4-28
4.4.7 Periodic Interrupt Timer 4-28
4.4.8 Interrupt Priority and Vectoring for the Periodic Interrupt Timer 4-29

Paragraph Number		Page Number

4.4.9 Low Power Stop Mode 4-30
4.4.9.1 Periodic Interrupt Control Register 4-30
4.4.9.2 Periodic Interrupt Timer Register 4-31
4.5 External Bus Interface 4-31
4.5.1 Bus Control Signals 4-32
4.5.1.1 Address Bus 4-32
4.5.1.2 Address Strobe 4-32
4.5.1.3 Data Bus 4-32
4.5.1.4 Data Strobe 4-34
4.5.1.5 Read/Write Signal 4-34
4.5.1.6 Size Signals 4-34
4.5.1.7 Function Codes 4-34
4.5.1.8 Data Size Acknowledge Signals 4-35
4.5.1.9 Bus Error Signal 4-35
4.5.1.10 Halt Signal 4-35
4.5.1.11 Autovector Signal 4-36
4.5.2 Dynamic Bus Sizing 4-36
4.5.3 Operand Alignment 4-37
4.5.4 Misaligned Operands 4-37
4.5.5 Operand Transfer Cases 4-37
4.6 Bus Operation 4-38
4.6.1 Synchronization to CLKOUT 4-39
4.6.2 Regular Bus Cycle 4-39
4.6.2.1 Read Cycle 4-40
4.6.2.2 Write Cycle 4-40
4.6.3 Fast Termination Cycles 4-41
4.6.4 CPU Space Cycles 4-42
4.6.4.1 Breakpoint Acknowledge Cycle 4-43
4.6.4.2 LPSTOP Broadcast Cycle 4-46
4.6.5 Bus Exception Control Cycles 4-46
4.6.5.1 Bus Errors 4-48
4.6.5.2 Double Bus Faults 4-48
4.6.5.3 Retry Operation 4-49
4.6.5.4 Halt Operation 4-49
4.6.6 External Bus Arbitration 4-50
4.6.6.1 Show Cycles 4-51
4.7 Reset 4-52
4.7.1 SCIM2E Reset Control Logic 4-52
4.7.1.1 SCIM2E Reset Control Flow 4-53
4.7.2 Reset Exception Processing 4-55
4.7.3 Reset Source Summary 4-55
4.7.4 Reset Status Register 4-56
4.7.5 Reset Timing 4-56

Paragraph Number		Page Number

4.7.6 Power-On Reset 4-57
4.7.7 Pin State During Reset 4-58
4.7.7.1 Reset States of SCIM2E Pins 4-58
4.7.7.2 Reset States of Pins Assigned to Other MCU Modules 4-59
4.7.8 Operating Configuration Out of Reset 4-60
4.7.8.1 Operating Mode Selection 4-60
4.7.8.2 Data Bus Mode Selection 4-62
4.7.8.3 Single-Chip Mode 4-64
4.7.8.4 Fully (16-bit) Expanded Mode 4-66
4.7.8.5 Breakpoint Mode Selection 4-68
4.7.8.6 Emulation Mode Selection 4-69
4.7.9 Use of the Three-State Control Pin 4-69
4.8 Interrupts 4-70
4.8.1 Interrupt Exception Processing 4-70
4.8.2 Interrupt Priority and Recognition 4-70
4.8.3 Interrupt Acknowledge and Arbitration 4-71
4.8.4 Interrupt Processing Summary 4-73
4.9 Chip Selects 4-73
4.9.1 Chip-Select Pin Assignment Register 4-75
4.9.1.1 Port C Data Register 4-78
4.9.2 Chip-Select Base Address Registers 4-78
4.9.3 Chip-Select Option Registers 4-80
4.9.4 Chip-Select Operation 4-84
4.9.4.1 Using Chip-Select Signals for Interrupt Acknowledge Cycle Termination 4-84
4.9.4.2 Chip-Select Reset Operation 4-85
4.10 General-Purpose Input/Output 4-87
4.10.1 Ports A and B 4-88
4.10.2 Port A and B Data Registers 4-88
4.10.3 Port E 4-88
4.10.3.1 Port E Data Register 4-89
4.10.3.2 Port E Data Direction Register 4-89
4.10.3.3 Port E Pin Assignment Register 4-89
4.10.4 Port F 4-90
4.10.4.1 Port F Data Register 4-91
4.10.4.2 Port F Data Direction Register 4-92
4.10.4.3 Port F Pin Assignment Register 4-92
4.10.4.4 Port F Edge-Detect Flag Register 4-93
4.10.4.5 Port F Edge-Detect Interrupt Vector 4-93
4.10.4.6 Port F Edge-Detect Interrupt Level 4-93
4.10.5 Port G 4-93
4.10.5.1 Port G and H Data Registers 4-94
4.10.5.2 Port G and H Data Direction Registers 4-94
4.10.6 Port H 4-94

Paragraph Number | Page Number

Section 5
QUEUED ANALOG-TO-DIGITAL CONVERTER MODULE-64

5.1 Overview . . . 5-1
5.2 Features . . . 5-2
5.3 QADC64 Pin Functions . . . 5-2
5.3.1 Port A Pin Functions . . . 5-4
5.3.1.1 Port A Analog Input Pins . . . 5-4
5.3.1.2 Port A Digital Input/Output Pins . . . 5-4
5.3.2 Port B Pin Functions . . . 5-4
5.3.2.1 Port B Analog Input Pins . . . 5-4
5.3.2.2 Port B Digital Input Pins . . . 5-4
5.3.3 External Trigger Input Pins . . . 5-5
5.3.4 Multiplexed Address Output Pins . . . 5-5
5.3.5 Multiplexed Analog Input Pins . . . 5-5
5.3.6 Voltage Reference Pins . . . 5-5
5.3.7 Dedicated Analog Supply Pins . . . 5-6
5.3.8 External Digital Supply Pin . . . 5-6
5.3.9 Digital Supply Pins . . . 5-6
5.4 QADC64 Bus Interface . . . 5-6
5.5 Module Configuration . . . 5-6
5.5.1 Low-Power Stop Mode . . . 5-6
5.5.2 Freeze Mode . . . 5-7
5.5.3 Supervisor/Unrestricted Address Space . . . 5-7
5.6 General-Purpose I/O Port Operation . . . 5-8
5.6.1 Port Data Register . . . 5-8
5.6.2 Port Data Direction Register . . . 5-9
5.7 External Multiplexing Operation . . . 5-9
5.8 Analog Input Channels . . . 5-11
5.9 Analog Subsystem . . . 5-11
5.9.1 Conversion Cycle Times . . . 5-12
5.9.1.1 Amplifier Bypass Mode Conversion Timing . . . 5-13
5.9.2 Front-End Analog Multiplexer . . . 5-14
5.9.3 Digital-to-Analog Converter Array . . . 5-14
5.9.4 Comparator . . . 5-14
5.9.5 Successive Approximation Register . . . 5-14
5.10 Digital Control Subsystem . . . 5-14
5.10.1 Queue Priority . . . 5-15
5.10.2 Queue Boundary Conditions . . . 5-17
5.10.3 Scan Modes . . . 5-18
5.10.3.1 Disabled Mode . . . 5-18
5.10.3.2 Reserved Mode . . . 5-18
5.10.3.3 Single-Scan Modes . . . 5-18
5.10.3.4 Continuous-Scan Modes . . . 5-21

Paragraph Number **Page Number**

5.10.4 QADC64 Clock (QCLK) Generation 5-25
5.10.5 Periodic/Interval Timer 5-29
5.11 Interrupts 5-29
5.11.1 Interrupt Operation 5-30
5.11.1.1 Polled and Interrupt-Driven Operation 5-30
5.11.2 Interrupt Sources 5-30
5.11.3 Interrupt Priority 5-31
5.11.4 Interrupt Arbitration 5-31
5.11.5 Interrupt Vectors 5-32
5.12 Programming Model 5-33
5.12.1 QADC64 Module Configuration Register 5-35
5.12.2 QADC64 Interrupt Register 5-35
5.12.3 Port A/B Data Register 5-36
5.12.4 Port Data Direction Register 5-37
5.12.5 QADC64 Control Register 0 (QACR0) 5-37
5.12.6 QADC64 Control Register 1 (QACR1) 5-38
5.12.7 QADC64 Control Register 2 (QACR2) 5-40
5.12.8 QADC64 Status Register 0 (QASR0) 5-42
5.12.9 QADC64 Status Register 1 (QASR1) 5-44
5.12.10 Conversion Command Word Table 5-45
5.12.11 Result Word Table 5-51
5.13 Analog Multiplexer Submodule 5-52
5.13.1 Signal Descriptions 5-52
5.13.1.1 External Pins (Connected to Pads) 5-52
5.13.1.2 Internal Pins (Connected to QADC64) 5-52
5.13.2 Mixed AMUX/External Multiplexing 5-54
5.13.3 Pin Connection and Performance Considerations 5-56

Section 6
QUEUED SERIAL MULTI-CHANNEL MODULE

6.1 Overview 6-1
6.2 Block Diagram 6-1
6.3 Signal Descriptions 6-2
6.4 Memory Map 6-2
6.5 QSMCM Global Registers 6-4
6.5.1 Low-Power Stop Operation 6-5
6.5.2 Freeze Operation 6-5
6.5.3 Access Protection 6-5
6.5.4 QSMCM Interrupts 6-6
6.5.5 QSMCM Configuration Register (QSMCMMCR) 6-6
6.5.6 QSMCM Test Register (QTEST) 6-7
6.5.7 QSMCM Interrupt Level Registers (QILR, QIVR, QSPI_IL) 6-7
6.6 QSMCM Pin Control Registers 6-8

Paragraph Number		Page Number

6.6.1 Port QS Data Register (PORTQS) 6-10
6.6.2 PORTQS Pin Assignment Register (PQSPAR) 6-10
6.6.3 PORTQS Data Direction Register (DDRQS) 6-12
6.7 Queued Serial Peripheral Interface 6-13
6.7.1 QSPI Registers 6-15
6.7.1.1 QSPI Control Register 0 6-16
6.7.1.2 QSPI Control Register 1 6-18
6.7.1.3 QSPI Control Register 2 6-18
6.7.1.4 QSPI Control Register 3 6-19
6.7.1.5 QSPI Status Register 6-20
6.7.2 QSPI RAM 6-21
6.7.2.1 Receive RAM 6-22
6.7.2.2 Transmit RAM. 6-22
6.7.2.3 Command RAM 6-22
6.7.3 QSPI Pins 6-23
6.7.4 QSPI Operation 6-24
6.7.4.1 Enabling, Disabling, and Halting the SPI 6-25
6.7.4.2 QSPI Interrupts. 6-26
6.7.4.3 QSPI Flow 6-26
6.7.5 Master Mode Operation 6-33
6.7.5.1 Clock Phase and Polarity 6-34
6.7.5.2 Baud Rate Selection. 6-34
6.7.5.3 Delay Before Transfer 6-35
6.7.5.4 Delay After Transfer 6-35
6.7.5.5 Transfer Length 6-36
6.7.5.6 Peripheral Chip Selects 6-36
6.7.5.7 Master Wraparound Mode 6-37
6.7.6 Slave Mode. 6-37
6.7.6.1 Description of Slave Operation. 6-39
6.7.7 Slave Wraparound Mode 6-40
6.7.8 Mode Fault 6-41
6.8 Serial Communication Interface 6-41
6.8.1 SCI Registers 6-44
6.8.2 SCI Control Register 0 6-45
6.8.3 SCI Control Register 1 6-45
6.8.4 SCI Status Register (SCxSR). 6-47
6.8.5 SCI Data Register (SCxDR) 6-49
6.8.6 SCI Pins 6-50
6.8.7 SCI Operation. 6-50
6.8.7.1 Definition of Terms 6-50
6.8.7.2 Serial Formats 6-51
6.8.7.3 Baud Clock. 6-51
6.8.7.4 Parity Checking 6-52

6.8.7.5 Transmitter Operation 6-52
6.8.7.6 Receiver Operation 6-54
6.8.7.7 Idle-Line Detection 6-55
6.8.7.8 Receiver Wake-Up 6-56
6.8.7.9 Internal Loop Mode 6-57
6.9 SCI Queue Operation 6-57
6.9.1 Queue Operation of SCI1 for Transmit and Receive 6-57
6.9.2 Queued SCI1 Status and Control Registers 6-57
6.9.2.1 QSCI1 Control Register 6-57
6.9.2.2 QSCI1 Status Register 6-58
6.9.3 QSCI1 Transmitter Block Diagram 6-59
6.9.4 QSCI1 Additional Transmit Operation Features 6-60
6.9.5 QSCI1 Transmit Flow Chart Implementing the Queue 6-62
6.9.6 Example QSCI1 Transmit for 17 Data Bytes 6-64
6.9.7 Example SCI Transmit for 25 Data Bytes 6-65
6.9.8 QSCI1 Receiver Block Diagram 6-66
6.9.9 QSCI1 Additional Receive Operation Features 6-66
6.9.10 QSCI1 Receive Flow Chart Implementing The Queue 6-69
6.9.11 QSCI1 Receive Queue Software Flow Chart 6-70
6.9.12 Example QSCI1 Receive Operation of 17 Data Frames 6-71

Section 7
CAN 2.0B CONTROLLER MODULE

7.1 Overview 7-1
7.2 Features 7-2
7.3 External Pins 7-2
7.4 TouCAN Architecture 7-3
7.4.1 TX/RX Message Buffer Structure 7-3
7.4.1.1 Common Fields for Extended and Standard Format Frames 7-4
7.4.1.2 Fields for Extended Format Frames 7-6
7.4.1.3 Fields for Standard Format Frames 7-6
7.4.1.4 Serial Message Buffers 7-6
7.4.1.5 Message Buffer Activation/Deactivation Mechanism 7-7
7.4.1.6 Message Buffer Lock/Release/Busy Mechanism 7-7
7.4.2 Receive Mask Registers 7-7
7.4.3 Bit Timing 7-8
7.4.3.1 Configuring the TouCAN Bit Timing 7-9
7.4.4 Error Counters 7-9
7.4.5 Time Stamp 7-11
7.5 TouCAN Operation 7-11
7.5.1 TouCAN Reset 7-11
7.5.2 TouCAN Initialization 7-11
7.5.3 Transmit Process 7-12

7.5.3.1 Transmit Message Buffer Deactivation 7-13
7.5.3.2 Reception of Transmitted Frames 7-13
7.5.4 Receive Process 7-13
7.5.4.1 Receive Message Buffer Deactivation 7-15
7.5.4.2 Locking and Releasing Message Buffers 7-15
7.5.5 Remote Frames 7-16
7.5.6 Overload Frames 7-16
7.6 Special Operating Modes 7-16
7.6.1 Debug Mode 7-16
7.6.2 Low-Power Stop Mode 7-17
7.6.3 Auto-Power Save Mode 7-18
7.7 Interrupts 7-19
7.8 Programmer's Model 7-20
7.8.1 TouCAN Module Configuration Register 7-23
7.8.2 TouCAN Interrupt Configuration Register 7-25
7.8.3 Control Register 0 7-26
7.8.4 Control Register 1 7-27
7.8.5 Prescaler Divide Register 7-28
7.8.6 Control Register 2 7-29
7.8.7 Free Running Timer 7-29
7.8.8 Receive Global Mask Registers 7-30
7.8.9 Receive Buffer 14 Mask Registers 7-30
7.8.10 Receive Buffer 15 Mask Registers 7-31
7.8.11 Error and Status Register 7-31
7.8.12 Interrupt Mask Register 7-33
7.8.13 Interrupt Flag Register 7-33
7.8.14 Error Counters 7-34

Section 8
TIME PROCESSOR UNIT 3

8.1 Overview 8-1
8.2 TPU3 Components 8-2
8.2.1 Time Bases 8-2
8.2.2 Timer Channels 8-2
8.2.3 Scheduler 8-2
8.2.4 Microengine 8-2
8.2.5 Host Interface 8-3
8.2.6 Parameter RAM 8-3
8.3 TPU Operation 8-3
8.3.1 Event Timing 8-3
8.3.2 Channel Orthogonality 8-3
8.3.3 Interchannel Communication 8-4
8.3.4 Programmable Channel Service Priority 8-4

8.3.5 Coherency 8-4
8.3.6 Emulation Support 8-4
8.3.7 TPU3 Interrupts 8-5
8.3.8 Prescaler Control for TCR1 8-5
8.3.9 Prescaler Control for TCR2 8-7
8.4 Programming Model. 8-8
8.4.1 TPU Module Configuration Register. 8-10
8.4.2 TPU3 Test Configuration Register 8-12
8.4.3 Development Support Control Register 8-12
8.4.4 Development Support Status Register 8-13
8.4.5 TPU3 Interrupt Configuration Register 8-14
8.4.6 Channel Interrupt Enable Register 8-14
8.4.7 Channel Function Select Registers 8-15
8.4.8 Host Sequence Registers. 8-15
8.4.9 Host Service Request Registers. 8-16
8.4.10 Channel Priority Registers 8-17
8.4.11 Channel Interrupt Status Register 8-18
8.4.12 Link Register. 8-18
8.4.13 Service Grant Latch Register 8-18
8.4.14 Decoded Channel Number Register. 8-18
8.4.15 TPU3 Module Configuration Register 2 8-18
8.4.16 TPU Module Configuration Register 3 8-20
8.4.17 TPU3 Test Registers 8-20
8.4.18 TPU3 Parameter RAM 8-21
8.5 Time Functions. 8-21

Section 9
DUAL-PORT TPU RAM (DPTRAM)

9.1 Introduction 9-1
9.2 Features. 9-1
9.3 DPTRAM Configuration and Block Diagram 9-2
9.4 Programming Model. 9-2
9.4.1 DPTRAM Module Configuration Register (DPTMCR) 9-3
9.4.2 DPTRAM Test Register 9-4
9.4.3 Ram Base Address Register (DPTBAR) 9-4
9.4.4 MISR High (MISRH) and MISR Low (MISRL) 9-5
9.4.5 MISC Counter (MISCNT) 9-6
9.5 Operation 9-6
9.5.1 Normal Operation 9-6
9.5.2 Standby Operation 9-6
9.5.3 Reset Operation 9-6
9.5.4 Stop Operation 9-7
9.5.5 Freeze Operation 9-7

9.5.6 TPU3 Emulation Mode Operation 9-7
9.6 Multiple Input Signature Calculator (MISC) 9-8

Section 10
CDR MoneT FLASH FOR THE IMB3 (CMFI)

10.1 Overview 10-1
10.1.1 Overview Description 10-1
10.1.2 Features of the CMFI 10-3
10.1.3 Glossary of terms used in the CMFI EEPROM Specification 10-4
10.2 CMFI EEPROM Interface 10-5
10.2.1 External Interface 10-6
10.3 Programmer's Model 10-7
10.4 CMFI EEPROM Control Block 10-7
10.4.1 CMFI EEPROM Module Control Block Addressing 10-8
10.4.2 Reserved Register Accesses 10-9
10.4.3 CMFI EEPROM Configuration Register (CMFIMCR) 10-9
10.4.4 CMFI EEPROM Test Register (CMFITST) 10-12
10.4.5 CMFI Base Address Registers (CMFIBAR) 10-14
10.4.6 High Voltage Control Register 10-16
10.4.7 CMFIBS CMFI Bootstrap Words [3:0] 10-19
10.4.8 Pulse Width Timing Control 10-20
10.4.9 A Technique to Determine SCLKR, CLKPE, and CLKPM 10-21
10.5 CMFI EEPROM Array Addressing 10-22
10.5.1 Read Burst Buffers 10-22
10.5.2 Program Page Buffers 10-23
10.6 Operation 10-23
10.6.1 Power On Reset 10-24
10.6.2 Master Reset 10-24
10.6.3 System Reset 10-24
10.6.4 Register Read and Write Operation 10-24
10.6.5 Array Read Operation 10-25
10.6.6 Programming 10-25
10.6.6.1 Program Sequence 10-26
10.6.6.2 Program Margin Reads 10-30
10.6.6.3 Programming Shadow Information 10-31
10.6.6.4 Over Programming 10-31
10.6.7 Erase 10-31
10.6.7.1 Erase Sequence 10-32
10.6.7.2 Erase Margin Reads 10-34
10.6.7.3 Erasing Shadow Information Words 10-34
10.6.8 Stop Operation 10-35
10.6.8.1 Low Power Stop Clock Operation 10-35
10.6.8.2 STOP Recovery 10-35

10.6.9 Background Debug Mode or Freeze Operation 10-35

Section 11
STATIC RANDOM ACCESS MEMORY (SRAM)

11.1 Introduction 11-1
11.2 Programmer's Model 11-1
11.2.1 SRAM Control Block 11-1
11.2.2 SRAM Array 11-3
11.2.2.1 SRAM Array Addressing 11-3
11.3 SRAM Module Control and Status Registers 11-3
11.3.1 Module Configuration Register (RAMMCR) 11-3
11.3.2 Array Base Address Registers (RAMBAH, RAMBAL) 11-4
11.4 Operation 11-5
11.4.1 Normal Operation 11-6
11.4.1.1 Read/Write 11-6
11.4.2 Standby Operation 11-6
11.4.2.1 Power Down 11-6
11.4.3 RESET Operation 11-6
11.4.4 STOP Operation 11-7
11.4.5 Overlay Operation 11-7

Section 12
MASK ROM MODULE

12.1 Introduction 12-1
12.2 Mask Programmable Options 12-1
12.3 Programmer's Model 12-2
12.3.1 ROM Control Block 12-2
12.3.1.1 ROM Module Control Block Addressing 12-2
12.3.2 ROM Array 12-2
12.3.2.1 ROM Array Addressing 12-3
12.4 ROM Module Control and Configuration Registers 12-3
12.4.1 Module Configuration Register (ROMMCR) 12-3
12.4.2 ROM Base Address Register (ROMBAH, ROMBAL) 12-5
12.4.3 ROM Signature High (SIGHI) 12-6
12.4.4 ROM Signature Low (SIGLO) 12-6
12.5 Bootstrap Information Words (ROMBS0–ROMBS3) 12-7
12.6 Operation 12-8
12.6.1 RESET Operation 12-8
12.6.2 Bootstrap Operation 12-9
12.6.3 Normal Operation 12-9
12.6.4 Read/Write Access 12-9
12.6.5 Emulation Mode Operation 12-10
12.6.6 Stop Operation 12-11
12.6.7 FREEZE Operation 12-11

Paragraph Number

Page Number

Section 13
CONFIGURABLE TIMER MODULE (CTM9)

13.1 Introduction 13-1
13.1.1 CTM9 Configuration 13-1
13.1.2 CTM9 Pins and Naming Convention 13-4
13.2 Free Running Counter Submodule (FCSM). 13-4
13.2.1 The FCSM Counter 13-5
13.2.2 FCSM Clock Sources 13-6
13.2.3 FCSM External Event Counting 13-6
13.2.4 The FCSM Time Base Bus Driver 13-6
13.2.5 FCSM Interrupts 13-6
13.2.6 Freeze Action on the FCSM 13-6
13.2.7 FCSM Registers 13-7
13.2.7.1 FCSMSIC — FCSM Status/Interrupt/Control Register 13-7
13.2.7.2 FCSMCNT — FCSM Counter Register 13-9
13.3 Modulus Counter Submodule (MCSM) 13-9
13.3.1 The MCSM Modulus Latch 13-10
13.3.2 The MCSM Counter 13-10
13.3.2.1 Loading the MCSM Counter Register. 13-10
13.3.2.2 Using the MCSM as a Free-Running Counter 13-11
13.3.3 MCSM Clock Sources 13-11
13.3.4 MCSM External Event Counting 13-11
13.3.5 The MCSM Time Base Bus Driver 13-11
13.3.6 MCSM interrupts. 13-11
13.3.7 Freeze Action on the MCSM 13-12
13.3.8 MCSM Registers 13-12
13.3.9 MCSMSIC — MCSM Status/Interrupt/Control Register 13-12
13.3.10 MCSMCNT — MCSM Counter Register 13-14
13.3.11 MCSMML — MCSM Modulus Latch Register 13-14
13.4 Single-Action Submodule (SASM) 13-14
13.4.1 SASM Modes of Operation. 13-16
13.4.1.1 Clearing and Using the FLAG Bits 13-16
13.4.1.2 Input Capture (IC) Mode. 13-17
13.4.1.3 Output Compare (OC) Mode 13-17
13.4.1.4 Output Compare and Toggle (OCT) Mode. 13-18
13.4.1.5 Output Port (OP) mode. 13-18
13.4.2 SASM Interrupts 13-19
13.4.3 Freeze Action on the SASM 13-19
13.4.4 SASM Registers 13-19
13.4.4.1 SICA — SASM Status/Interrupt Control Register A 13-20
13.4.4.2 SDATA — SASM Data Register A 13-22
13.4.4.3 SICB — SASM Status/Interrupt Control Register B 13-22
13.4.4.4 SDATB — SASM Data Register B 13-23

Paragraph Number		Page Number

13.5 Double-Action Submodule (DASM) 13-23
13.5.1 32-Bit Coherent Access 13-25
13.5.2 DASM Modes of Operation. 13-25
13.5.2.1 Disable (DIS) mode 13-26
13.5.2.2 Input Pulse Width Measurement (IPWM) Mode 13-26
13.5.2.3 Input Period Measurement (IPM) Mode 13-27
13.5.2.4 Input Capture (IC) Mode. 13-28
13.5.2.5 Output Compare (OCB and OCAB) Modes 13-29
13.5.2.6 Output Pulse Width Modulation (OPWM) Mode. 13-32
13.5.3 DASM interrupts 13-34
13.5.4 Freeze Action on the DASM 13-34
13.5.5 DASM Registers 13-35
13.5.5.1 DASMSIC — DASM Status/Interrupt Control Register. 13-35
13.5.5.2 DASMA — DASM Data Register A 13-37
13.5.5.3 DASMB — DASM Data Register B 13-38
13.6 Pulse Width Modulation Submodule (PWMSM). 13-38
13.6.1 Output Flip-Flop and Pin. 13-39
13.7 Time Base Bus System 13-40
13.7.1 Clock Selection. 13-40
13.7.2 The PWMSM Counter (PWMC) 13-40
13.7.3 PWMSM Period Registers and Comparator. 13-40
13.7.4 PWMSM Pulse Width Registers and Comparator 13-41
13.7.5 0% and 100% ‘Pulses’ 13-41
13.7.6 PWMSM Coherency. 13-42
13.7.7 PWMSM Interrupts 13-42
13.7.8 Freeze Action on the PWMSM 13-42
13.7.9 PWM frequency, Pulse Width and Resolution 13-42
13.7.10 PWM Frequency. 13-43
13.7.11 PWM Pulse Width. 13-44
13.7.12 PWM Period and Pulse Width Register Values 13-44
13.7.13 PWMSM Register Map and Registers 13-44
13.7.13.1 PWMSIC — Status, Interrupt and Control Register 13-45
13.7.13.2 PWMA — PWM Period Register 13-48
13.7.13.3 PWMB — PWM Pulse Width Register 13-49
13.7.13.4 PWMC — PWM Counter Register. 13-49
13.8 Bus Interface Unit Submodule (BIUSM). 13-50
13.8.1 Freeze Action on the BIUSM 13-50
13.8.2 LPSTOP Action on the BIUSM 13-50
13.8.3 STOP and WAIT Action on the BIUSM 13-50
13.8.4 BIUSM Registers 13-50
13.8.4.1 BIUMCR — BIUSM Module Configuration Register. 13-51
13.8.4.2 BIUTBR — BIUSM Time Base Register. 13-52
13.9 Counter Prescaler Submodule (CPSM) 13-52

13.9.1 Freeze Action on the CPSM 13-53
13.9.2 CPSM Registers 13-53
13.9.2.1 CPCR — CPSM Control Register 13-54
13.9.3 Clock Sources for the Counter Submodules 13-54
13.10 CTM9 Interrupts 13-55
13.11 CTM9 Function Examples 13-55
13.11.1 CTM9 Single Input Capture 13-55
13.11.2 CTM9 Input Double Edge Pulse Width Measurement 13-56
13.11.3 CTM9 Input Double Edge Period Measurement 13-57
13.11.4 CTM9 Single Output Compare 13-58
13.11.5 CTM9 Double Edge Single Output Pulse Generation 13-59
13.11.6 CTM9 Output Pulse Width Modulation With DASM 13-60
13.11.7 CTM9 Input Pulse Accumulation 13-61

Appendix A
INTERNAL MEMORY MAP

Appendix B
REGISTER GENERAL INDEX

Appendix C
REGISTER DIAGRAM INDEX

Appendix D
TPU ROM FUNCTIONS

D.1 Overview D-1
D.2 Programmable Time Accumulator (PTA) D-3
D.3 Queued Output Match TPU Function (QOM) D-5
D.4 Table Stepper Motor (TSM) D-7
D.5 Frequency Measurement (FQM) D-10
D.6 Universal Asynchronous Receiver/Transmitter (UART) D-12
D.7 New Input Capture/Transition Counter (NITC) D-15
D.8 Multiphase Motor Commutation (COMM) D-17
D.9 Hall Effect Decode (HALLD) D-19
D.10 Multichannel Pulse-Width Modulation (MCPWM) D-21
D.11 Fast Quadrature Decode TPU Function (FQD) D-28
D.12 Period/Pulse-Width Accumulator (PPWA) D-31
D.13 Output Compare (OC) D-33
D.14 Pulse-Width Modulation (PWM) D-35
D.15 Discrete Input/Output (DIO) D-37
D.16 Synchronized Pulse-Width Modulation (SPWM) D-39
D.17 Serial Input/Output Port (SIOP) D-42
D.17.1 Parameters D-43
D.17.1.1 CHAN_CONTROL D-44
D.17.1.2 BIT_D D-44
D.17.1.3 HALF_PERIOD D-44
D.17.1.4 BIT_COUNT D-44
D.17.1.5 XFER_SIZE D-44
D.17.1.6 SIOP_DATA D-44

Paragraph Number **Page Number**

D.17.2 Host CPU Initialization of the SIOP Function D-45
D.17.3 SIOP Function Performance D-45
D.17.3.1 XFER_SIZE Greater than 16 . . . . . D-46
D.17.3.2 Data Positioning D-46
D.17.3.3 Data Timing D-46

Appendix E
ELECTRICAL CHARACTERISTICS

E.1 Absolute Maximum Ratings E-1
E.2 Radiated Emissions E-1
E.3 Thermal Characteristics E-2
E.4 DC Characteristics E-2
E.5 AC Characteristics E-5
E.6 QSPI Characteristics E-20
E.7 TPU3 Characteristics E-23
E.8 QADC64 and AMUX Characteristics E-24
E.9 CTM9 Electrical Characteristics E-29
E.9.1 5 V, 16.78 MHz Operating Conditions E-29
E.9.2 5 V, 25.0 MHz Operating Conditions E-29
E.10 TouCAN Characteristics E-32
E.11 CMFI FLASH Characteristics E-33
E.11.1 EPEB0 Pin Timing E-34
E.11.2 FLASH Program/Erase Voltage Conditioning E-34
E.12 ROM Electrical Specifications E-36

INDEX

Online publishing by JABIS™, ***http://www.jabis.com***

Paragraph Number

Page Number

LIST OF FIGURES

Figure Number		Page Number
1-1	MC68F375 Block Diagram	1-7
1-2	MC68F375 Address Map	1-10
2-1	MC68F375 Pad Map	2-8
2-2	MC68F375 Ball Map	2-9
3-1	CPU32 Block Diagram	3-2
3-2	User Programming Model	3-3
3-3	Supervisor Programming Model Supplement	3-4
3-4	Data Organization in Data Registers	3-5
3-5	Address Organization in Address Registers	3-6
3-6	Memory Operand Addressing	3-8
3-7	Loop Mode Instruction Sequence	3-16
3-8	Common In-Circuit Emulator Diagram	3-20
3-9	Bus State Analyzer Configuration	3-20
3-10	Debug Serial I/O Block Diagram	3-25
3-11	BDM Serial Data Word	3-26
3-12	BDM Connector Pinout	3-26
4-1	SCIM2E Block Diagram	4-2
4-2	Slow Reference Mode	4-9
4-3	Fast Reference Mode	4-12
4-4	External Clock Mode	4-14
4-5	Crystal Oscillator and External Capacitor Configuration	4-17
4-6	LPSTOP Flowchart	4-22
4-7	System Protection	4-24
4-8	Periodic Interrupt Timer and Software Watchdog Timer	4-28
4-9	MCU Basic System	4-33
4-10	Operand Byte Order	4-37
4-11	Word Read Cycle Flowchart	4-40
4-12	Write Cycle Flowchart	4-41
4-13	CPU Space Address Encoding	4-43
4-14	Breakpoint Operation Flowchart	4-45
4-15	LPSTOP Interrupt Mask Encoding on DATA[15:0]	4-46
4-16	Bus Arbitration Flowchart for Single Request	4-51
4-17	SCIM2 Reset Control Flow	4-54
4-18	Power-On Reset	4-58
4-19	Preferred Circuit for Data Bus Mode Select Conditioning	4-63
4-20	Alternate Circuit for Data Bus Mode Select Conditioning	4-64
4-21	Basic MCU System	4-74
4-22	Chip-Select Circuit Block Diagram	4-75
4-23	CPU Space Encoding for Interrupt Acknowledge	4-85
4-24	Port F Block Diagram	4-91

Figure Number		Page Number

5-1 QADC64 Block Diagram 5-1
5-2 QADC64 Input and Output Signals 5-3
5-3 Example of Full External Multiplexing 5-10
5-4 QADC64 Module Block Diagram 5-12
5-5 Conversion Timing 5-13
5-6 Bypass Mode Conversion Timing 5-13
5-7 QADC64 Queue Operation with Pause 5-16
5-8 QADC64 Clock Subsystem Functions 5-26
5-9 QADC64 Clock Programmability Examples 5-28
5-10 QADC64 Interrupt Flow Diagram 5-30
5-11 QADC64 Interrupt Vector Format 5-33
5-12 QADC64 Conversion Queue Operation 5-46
5-13 AMUX/QADC64 Configured for Mixed Multiplexing 5-55
5-14 Analog Multiplexer Submodule Charging Current Illustration 5-57

6-1 QSMCM Block Diagram 6-2
6-2 QSPI Block Diagram 6-14
6-3 QSPI RAM 6-22
6-4 Flowchart of QSPI Initialization Operation 6-27
6-5 Flowchart of QSPI Master Operation (Part 1) 6-28
6-6 Flowchart of QSPI Master Operation (Part 2) 6-29
6-7 Flowchart of QSPI Master Operation (Part 3) 6-30
6-8 Flowchart of QSPI Slave Operation (Part 1) 6-31
6-9 Flowchart of QSPI Slave Operation (Part 2) 6-32
6-10 SCI Transmitter Block Diagram 6-42
6-11 SCI Receiver Block Diagram 6-43
6-12 Queue Transmitter Block Enhancements 6-60
6-13 Queue Transmit Flow 6-62
6-14 Queue Transmit Software Flow 6-63
6-15 Queue Transmit Example for 17 Data Bytes 6-64
6-16 Queue Transmit Example for 25 Data Frames 6-65
6-17 Queue Receiver Block Enhancements 6-66
6-18 Queue Receive Flow 6-69
6-19 Queue Receive Software Flow 6-70
6-20 Queue Receive Example for 17 Data Bytes 6-71

7-1 TouCAN Block Diagram 7-1
7-2 Typical CAN Network 7-3
7-3 Extended ID Message Buffer Structure 7-4
7-4 Standard ID Message Buffer Structure 7-4
7-5 TouCAN Interrupt Vector Generation 7-19
7-6 TouCAN Message Buffer Memory Map 7-23

8-1 TPU3 Block Diagram 8-1
8-2 TCR1 Prescaler Control 8-7
8-3 TCR2 Prescaler Control 8-8

Figure Number		Page Number

9-1 DPTRAM Configuration 9-2

10-1 Block Diagram for a CMFI EEPROM in the 256-Kbyte Configuration. 10-2
10-3 Shadow Information 10-12
10-2 10-12
10-4 Pulse Status Timing 10-19
10-5 Master Reset Configuration Timing 10-24
10-6 Program State Diagram 10-28
10-7 Erase State Diagram 10-33

11-1 SRAM Module Configuration 11-2

13-1 Configurable Timer Module, CTM9 Block Diagram 13-2
13-2 FCSM Block Diagram 13-5
13-3 MCSM Block Diagram 13-9
13-4 SASM Block Diagram 13-15
13-5 SASM Block Diagram (Channel A) 13-16
13-6 DASM Block Diagram 13-24
13-7 Input Pulse Width Measurement Example 13-27
13-8 Input Period Measurement Example 13-28
13-9 DASM Input Capture Example 13-29
13-10 Single Shot Output Pulse Example 13-31
13-11 Single Shot Output Transition Example 13-31
13-12 DASM Output Pulse Width Modulation Example 13-33
13-13 Pulse Width Modulation Submodule Block Diagram 13-39
13-14 CPSM Block Diagram 13-53
13-15 CTM9 Example — Single Edge Input Capture 13-56
13-16 CTM9 Example — Double Capture Pulse Width Measurement 13-57
13-17 CTM9 Example — Double Capture Period Measurement 13-58
13-18 CTM9 Example — Single Edge Output Compare 13-59
13-19 CTM9 Example — Double Edge Output Compare 13-60
13-20 CTM9 Example — Pulse Width Modulation Output 13-61

D-1 TPU3 Memory Map D-1
D-2 PTA Parameters D-4
D-3 QOM Parameters D-6
D-4 TSM Parameters — Master Mode D-8
D-5 TSM Parameters — Slave Mode D-9
D-6 FQM Parameters D-11
D-7 UART Transmitter Parameters D-13
D-8 UART Receiver Parameters D-14
D-9 NITC Parameters D-16
D-10 COMM Parameters, Part 1 of 2 ..... D-18
D-11 COMM Parameters, Part 2 of 2 ..... D-19
D-12 HALLD Parameters D-20
D-13 MCPWM Parameters — Master Mode D-22

Figure Number		Page Number

D-14 MCPWM Parameters — Slave Edge-Aligned Mode D-23
D-15 MCPWM Parameters — Slave Ch A Non-Inverted Center-Aligned Mode ... D-24
D-16 MCPWM Parameters — Slave Ch B Non-Inverted Center-Aligned Mode ... D-25
D-17 MCPWM Parameters — Slave Ch A Inverted Center-Aligned Mode D-26
D-18 MCPWM Parameters — Slave Ch B Inverted Center-Aligned Mode D-27
D-19 FQD Parameters — Primary Channel D-29
D-20 FQD Parameters — Secondary Channel D-30
D-21 PPWA Parameters D-32
D-22 OC Parameters D-34
D-23 PWM Parameters D-36
D-24 DIO Parameters D-38
D-25 SPWM Parameters, Part 1 of 2 ..... D-40
D-26 SPWM Parameters, Part 2 of 2 ..... D-41
D-27 Two Possible SIOP Configurations D-42
D-28 SIOP Parameters D-43
D-29 SIOP Function Data Transition Example D-47

E-1 CLKOUT Output Timing Diagram E-9
E-2 External Clock Input Timing Diagram E-9
E-3 ECLK Output Timing Diagram E-9
E-4 Read Cycle Timing Diagram E-10
E-5 Write Cycle Timing Diagram E-11
E-6 Fast Termination Read Cycle Timing Diagram E-12
E-7 Fast Termination Write Cycle Timing Diagram E-13
E-8 Bus Arbitration Timing Diagram — Active Bus Case E-14
E-9 Bus Arbitration Timing Diagram — Idle Bus Case E-15
E-10 Show Cycle Timing Diagram E-15
E-11 Chip-Select Timing Diagram E-16
E-12 Reset and Mode Select Timing Diagram E-16
E-13 Background Debugging Mode Timing — Serial Communication E-17
E-14 Background Debugging Mode Timing — Freeze Assertion E-17
E-15 ECLK Timing Diagram E-19
E-16 QSPI Timing — Master, CPHA = 0 ..... E-21
E-17 QSPI Timing — Master, CPHA = 1 ..... E-21
E-18 QSPI Timing — Slave, CPHA = 0 ..... E-22
E-19 QSPI Timing — Slave, CPHA = 1 ..... E-22
E-20 TPU Timing Diagram E-23
E-21 EPEB0 Pin Timing E-34
E-22 V_{PP} and V_{DD} Power Sequencing E-35
E-23 A Recommended External V_{PP} Pin Conditioning Circuit E-36

LIST OF TABLES

Table Number		Page Number
1-1	MC68F375 Pin Usage	1-8
2-1	Pin Characteristics	2-1
2-2	Power Connections	2-3
2-3	Output Driver Types	2-4
2-4	Signal Characteristics	2-4
2-5	Signal Functions	2-6
3-1	Unimplemented MC68020 Instructions	3-10
3-2	Instruction Set Summary	3-11
3-3	Exception Vector Assignments	3-17
3-4	BDM Source Summary	3-21
3-5	Polling the BDM Entry Source	3-22
3-6	Background Mode Command Summary	3-23
3-7	CPU Generated Message Encoding	3-26
4-1	SCIMMCR Bit Descriptions	4-3
4-2	SCIMMCR Noise Control Bits	4-4
4-3	Show Cycle Enable Bits	4-5
4-4	Effects of FREEZE Assertion	4-5
4-5	System Clock Sources	4-6
4-6	CLKOUT Frequency: Slow Reference; 32.768 KHz Reference	4-10
4-7	CLKOUT In Fast Reference Mode with 4.0 MHz Reference	4-13
4-8	Port Reset Condition	4-20
4-9	SYPCR Bit Descriptions	4-24
4-10	Bus Monitor Period	4-25
4-11	SWP Reset States	4-26
4-12	Software Watchdog Divide Ratio	4-27
4-13	PTP Reset States	4-29
4-14	Periodic Interrupt Priority	4-30
4-15	PICR Bit Descriptions	4-31
4-16	PITR Bit Descriptions	4-31
4-17	Size Signal Encoding	4-34
4-18	Address Space Encoding	4-35
4-19	Effect of $\overline{\text{DSACK}}$ Signals	4-36
4-20	Operand Alignment	4-38
4-21	$\overline{\text{DSACK}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ Assertion Results	4-47
4-22	Reset Source Summary	4-55
4-23	RSR Bit Descriptions	4-56
4-24	SCIM2E Pin States During Reset	4-59
4-25	Pins Associated with Basic Configuration Options	4-60
4-26	Mode Configuration During Reset	4-61
4-27	Fully (16-bit) Expanded Mode Reset Configuration	4-66

Table Number		Page Number

4-28 Reset Pin Function of $\overline{CS}$[10:6] 4-67
4-29 Reset Configuration for MC68F375 Memory Modules 4-67
4-30 Partially (8-bit) Expanded Mode Reset Configuration 4-68
4-31 CSPAR0 Pin Assignments 4-76
4-32 CSPAR1 Pin Assignments 4-77
4-33 Reset Pin Function of $\overline{CS}$[10:6] 4-77
4-34 Pin Assignment Field Encoding 4-77
4-35 Block Size Encoding 4-79
4-36 CSBARBT/CSBAR Bit Descriptions 4-80
4-37 CSOR Bit Descriptions 4-81
4-38 $\overline{DSACK}$ Field Encoding 4-82
4-39 Interrupt Priority Level Field Encoding 4-83
4-40 Chip-Select Base and Option Register Reset Values 4-86
4-41 $\overline{CSBOOT}$ Base and Option Register Reset Values 4-87
4-42 General-Purpose I/O Ports 4-87
4-43 Port E Pin Assignments 4-90
4-44 Port F Pin Assignments 4-92
4-45 PFPAR Pin Functions 4-92

5-1 Multiplexed Analog Input Channels 5-5
5-2 Analog Input Channels 5-11
5-3 Queue 1 Priority Assertion 5-15
5-4 QADC64 Clock Programmability 5-28
5-5 QADC64 Status Flags and Interrupt Sources 5-31
5-6 QADC64 Address Map 5-34
5-7 QADC64MCR Bit Settings 5-35
5-8 QADC64INT Bit Settings 5-36
5-9 PORTQA, PORTQB Bit Settings 5-37
5-10 DDRQA Bit Settings 5-37
5-11 QACR0 Bit Settings 5-38
5-12 QACR1 Bit Settings 5-39
5-13 Queue 1 Operating Modes 5-39
5-14 QACR2 Bit Settings 5-41
5-15 Queue 2 Operating Modes 5-42
5-16 QASR0 Bit Settings 5-43
5-17 Queue Status 5-44
5-18 QASR1 Bit Settings 5-45
5-19 CCW Bit Settings 5-49
5-20 Non-Multiplexed Channel Assignments and Pin Designations 5-50
5-21 Multiplexed Channel Assignments and Pin Designations 5-50
5-22 AMUX I/O Functionality 5-53

6-1 QSMCM Register Map 6-3
6-2 QSMCM Global Registers 6-5

Table Number		Page Number

6-3 QSMCMMCR Bit Settings 6-7
6-4 QILR Bit Settings 6-8
6-5 QIVR Bit Settings 6-8
6-6 QSPI_IL Bit Settings 6-8
6-7 QSMCM Pin Control Registers 6-9
6-8 Effect of DDRQS on QSPI Pin Function 6-10
6-9 QSMCM Pin Functions 6-11
6-10 PQSPAR Bit Settings 6-12
6-11 DDRQS Bit Settings 6-13
6-12 QSPI Register Map 6-16
6-13 SPCR0 Bit Settings 6-17
6-14 Bits Per Transfer 6-17
6-15 SPCR1 Bit Settings 6-18
6-16 SPCR2 Bit Settings 6-19
6-17 SPCR3 Bit Settings 6-20
6-18 SPSR Bit Settings 6-21
6-19 Command RAM Bit Settings 6-23
6-20 QSPI Pin Functions 6-24
6-21 Example SCK Frequencies with a 40-MHz System Clock 6-35
6-22 SCI Registers 6-44
6-23 SCCxR0 Bit Settings 6-45
6-24 SCCxR1 Bit Settings 6-46
6-25 SCxSR Bit Settings 6-48
6-26 SCxSR Bit Settings 6-50
6-27 SCI Pin Functions 6-50
6-28 Serial Frame Formats 6-51
6-29 Examples of SCIx Baud Rates 6-52
6-30 Effect of Parity Checking on Data Size 6-52
6-31 QSCI1CR Bit Settings 6-57
6-32 QSCI1SR Bit Settings 6-59

7-1 Common Extended/Standard Format Frames 7-5
7-2 Message Buffer Codes for Receive Buffers 7-5
7-3 Message Buffer Codes for Transmit Buffers 7-5
7-4 Extended Format Frames 7-6
7-5 Standard Format Frames 7-6
7-6 Receive Mask Register Bit Values 7-8
7-7 Mask Examples for Normal/Extended Messages 7-8
7-8 Example System Clock, CAN Bit Rate and S-Clock Frequencies 7-9
7-9 Interrupt Sources and Vector Addresses 7-20
7-10 TouCAN Register Map 7-22
7-11 TCNMCR Bit Settings 7-24
7-12 CANICR Bit Settings 7-26
7-13 CANCTRL0 Bit Settings 7-26
7-14 RX MODE[1:0] Configuration 7-27

Table Number		Page Number

7-15 Transmit Pin Configuration 7-27
7-16 CANCTRL1 Bit Settings 7-28
7-17 PRESDIV Bit Settings 7-29
7-18 CANCTRL2 Bit Settings 7-29
7-19 TIMER Bit Settings 7-30
7-20 RXGMSKHI, RXGMSKLO Bit Settings 7-30
7-21 ESTAT Bit Settings 7-31
7-22 Transmit Bit Error Status 7-32
7-23 Fault Confinement State Encoding 7-33
7-24 IMASK Bit Settings 7-33
7-25 IFLAG Bit Settings 7-33
7-26 RXECTR, TXECTR Bit Settings 7-34

8-1 Enhanced TCR1 Prescaler Divide Values 8-6
8-2 TCR1 Prescaler Values 8-6
8-3 TCR2 Counter Clock Source 8-7
8-4 TCR2 Prescaler Control 8-8
8-5 TPU3 Register Map 8-9
8-6 TPUMCR Bit Settings 8-11
8-7 DSCR Bit Settings 8-12
8-8 DSSR Bit Settings 8-13
8-9 TICR Bit Settings 8-14
8-10 CIER Bit Settings 8-14
8-11 CFSRx Bit Settings 8-15
8-12 HSQRx Bit Settings 8-16
8-13 HSSRx Bit Settings 8-17
8-14 CPRx Bit Settings 8-17
8-15 Channel Priorities 8-17
8-16 CISR Bit Settings 8-18
8-17 TPUMCR2 Bit Settings 8-19
8-18 Entry Table Bank Location 8-19
8-19 System Clock Frequency/Minimum Guaranteed Detected Pulse 8-20
8-20 TPUMCR3 Bit Settings 8-20
8-21 Parameter RAM Address Map 8-21

9-1 DPTRAM Register Map 9-3
9-2 DPTMCR Bit Settings 9-4
9-3 DPTBAR Bit Settings 9-5

10-1 CMFI EEPROM Module External Signals 10-6
10-2 CMFI EEPROM Memory Map with 32-Bit Word 10-7
10-3 CMFI Control Register Addressing 10-9
10-4 CMFIMCR Bit Settings 10-10
10-5 CMFITST Bit Settings 10-13
10-6 CMF Programming Algorithm (v6 and Later) 10-13

Table Number		Page Number

10-7 CMF Erase Algorithm (v6) 10-14
10-8 CMFIBAR (CMFIBAH, CMFIBAL) Bit Settings 10-15
10-9 CMFICTL1 Bit Settings 10-16
10-10 CMFICTL2 Bit Settings 10-17
10-11 System Clock Range 10-20
10-12 Clock Period Exponent and Pulse Width Range 10-21
10-13 Program Interlock State Descriptions 10-28
10-14 Results of Programming Margin Read 10-30
10-15 Register Shadow Information 10-31
10-16 Erase Interlock State Descriptions 10-33

11-1 RAMMCR Bit Settings 11-4
11-2 RASP Encoding 11-4
11-3 RAMBAH, RAMBAL Bit Settings 11-5
11-4 SRAM Array Read/Write Minimum Access Times 11-6

12-1 ROM Module Register Map 12-3
12-2 ROMMCR Bit Settings 12-4
12-3 BAR (ROMBAH, ROMBAL) Bit Settings 12-6
12-4 SIGHI Bit Settings 12-6
12-5 SIGLO Bit Settings 12-7
12-6 Minimum ROM Module Access Times 12-11

13-1 CTM9 Configuration Description 13-3
13-2 FCSM Register Map 13-7
13-3 FMSMSIC Bit Settings 13-8
13-4 MCSM Register Map 13-12
13-5 MCSMSIC Bit Settings 13-13
13-6 SASM Register Map 13-20
13-7 SICA Bit Settings 13-21
13-8 DASM Modes of Operation 13-25
13-9 DASM PWM Example Output Frequencies/Resolutions at fSYS = 16 MHz 13-34
13-10 DASM Register Map 13-35
13-11 DASMSIC Bit Settings 13-36
13-12 PWM Pulse and Frequency Ranges (in Hz) Using /2 Option (16.78 MHz) 13-43
13-13 PWM Pulse and FrequencyRanges (in Hz) Using /3 Option (16.78 MHz) 13-43
13-14 PWMSM Register Map 13-45
13-15 PWMSIC Bit Settings 13-46
13-16 PWMSM Output Pin Polarity Selection 13-48
13-17 PWMSM Clock Rate Selection 13-48
13-18 BIUSM Register Map 13-51
13-19 BIUMCR Bit Settings 13-51
13-20 CPSM Register Map 13-53

Table Number		Page Number

13-21 CPCR Bit Settings 13-54
13-22 Prescaler Division Ratio Select 13-54

A-1 TouCAN (CAN 2.0B Controller) A-2
A-2 CTM9 (Configurable Timer Module) A-4
A-3 QADC64 (Queued Analog-to-Digital Converter) A-7
A-4 CMFI (CDR MoneT FLASH FOR THE IMB3) A-8
A-5 ROM Module A-8
A-6 Overlay SRAM Modules (Static Random Access Memory) A-9
A-7 DPTRAM (Dual-Port TPU RAM) A-9
A-8 SCIM2E (Single-Chip Integration Module) A-10
A-9 SRAM Module (Static Random Access Memory) A-12
A-10 QSMCM (Queued Serial Multi-Channel Module) A-12
A-11 TPU3 (Time Processor Unit) A-14

D-1 Bank 0 Functions D-2
D-2 Bank 1 Functions D-2
D-3 QOM Bit Encoding D-5
D-4 SIOP Function Valid CHAN_Control Options D-44
D-5 SIOP State Timing D-46

E-1 Absolute Maximum Ratings E-1
E-2 Thermal Characteristics E-2
E-3 DC Characteristics E-2
E-4 Clock Control Timing E-5
E-5 AC Timing E-6
E-6 Background Debugging Mode Timing E-17
E-7 ECLK Bus Timing E-18
E-8 QSPI Timing E-20
E-9 Time Processor Unit (TPU3) Timing E-23
E-10 QADC64 Maximum Ratings E-24
E-11 QADC64 DC Electrical Characteristics (Operating) E-25
E-12 QADC64 AC Electrical Characteristics (Operating) E-26
E-13 QADC64 Conversion Characteristics (Operating) E-27
E-14 AMUX_HV Absolute Maximum Ratings E-28
E-15 AMUX_HV DC Electrical Characteristics (Operating) E-28
E-16 AMUX_HV Conversion Characteristics (Operating) E-29
E-17 CTM9 5 V, 16.78 MHz Operating Conditions E-29
E-18 CTM9 5 V, 25 MHz Operating Conditions E-29
E-19 FCSM Timing Characteristics E-30
E-20 MCSM Timing Characteristics E-30
E-21 SASM Timing Characteristics E-31
E-22 DASM Timing Characteristics E-31
E-23 PWMSM Timing Characteristics E-32
E-24 TouCAN AC Characteristics E-32
E-25 CMFI Program and Erase Characteristics E-33

Table Number		Page Number

E-26 CMFI AC and DC Power Supply Characteristics E-33
E-27 CMFI FLASH EEPROM Module Life E-34
E-28 IMB3 ROM AC/DC Characteristics E-36

Table Number	Page Number

PREFACE

This manual describes the capabilities, operation, and functions of the MC68F375 microcontroller unit. Documentation for the Modular Microcontroller Family follows the modular construction of the devices in the product line. Each device has a comprehensive user's manual which provides sufficient information for normal operation of the device. The user's manual is supplemented by module reference manuals that provide detailed information about module operation and applications. Refer to Motorola publication *Advanced Microcontroller Unit (AMCU) Literature* (BR1116/D) for a complete listing of documentation to supplement this manual.

The following conventions are used throughout the manual.

Logic level one is the voltage that corresponds to a Boolean true (1) state.

Logic level zero is the voltage that corresponds to a Boolean false (0) state.

To **set** a bit means to establish logic level one on the bit.

To **clear** a bit means to establish logic level zero on the bit.

0xXXXX denotes hexadecimal numbers, **0bXXXX** denotes binary.

A signal that is **asserted** is in its active logic state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.

A signal that is **negated** is in its inactive logic state. An active low signal changes from logic level zero to logic level one when negated, and an active high signal changes from logic level one to logic level zero.

A **specific bit or signal** within a range is referred to by mnemonic and number. For example, ADDR15 is bit 15 of the address bus. A **range of bits or signals** is referred to by mnemonic and the numbers that define the range. For example, DATA[7:0] form the low byte of the data bus.

LSB means least significant bit or bits. **MSB** means most significant bit or bits. References to low and high bytes are spelled out.

SECTION 1
OVERVIEW DESCRIPTION

1.1 Introduction

The MC68F375 is a member of the MC68300/68HC16 family of modular microcontrollers. This family includes a series of modules from which numerous microcontrollers (MCU's) are derived. These modules are connected on-chip via the intermodule bus (IMB). This section includes a list of pins that are available to each module on the chip, see **Table 1-1**.

1.2 MC68F375 Feature List

The major features of the MC68F375 are listed below. A block diagram of the device is shown in **Figure 1-1**.

- Modular architecture.
- 32-bit 68000 family CPU with upward object code compatibility (CPU32):
  - — Virtual memory implementation.
  - — Loop mode for instruction execution.
  - — Improved exception handling for controller applications.
  - — Table lookup and interpolate instruction.
- Single chip integration module (SCIM2E):
  - — External bus support.
  - — Parallel ports option on address and data bus in single chip modes and partially expanded modes.
  - — Nine programmable chip select outputs.
  - — Two special emulation-specific chip select outputs.
  - — System protection logic.
  - — System clock based on slow (~32 KHz) or Fast (~4 MHz) crystal reference or external clock operation (2X).
  - — Periodic interrupt timer, watchdog timer, clock monitor and bus monitor.
- 10-bit queued analog-to-digital converter with AMUX (QADC64):
  - — Up to 16 channels on the QADC64 (25 total including the 16 AMUX Channels).
  - — 4 automatic channel selection and conversion modes.
  - — 2 channel scan queues of variable length.
  - — 64 result registers and 3 result alignment formats.
  - — Programmable input sample time.
  - — Analog multiplexer capable of multiplexing 16 analog channels.
- Queued serial multi-channel communication module with three serial I/O subsystems (QSMCM):
  - — 2 enhanced SCI (UART): modulus baud rate generator, parity. One SCI has 16 level Rx and Tx queues.
  - — Queued SPI: 160-byte RAM, up to 32 automatic transfers, continuous cycling, 8-to-16 bits per transfer, LSB/MSB first.

— Dual function I/O port pins.

- Time processing unit (TPU3):
  — 16 channels – each is associated with a pin.
  — Each channel can perform any time function.
  — Each time function may be assigned to more than one channel.
  — Each channel has an event register comprised of: 16-bit capture register, 16-bit compare/match register, 16-bit greater-than-or-equal-to comparator.
  — Each channel can be programmed to perform match or capture operations with one or both of the two 16-bit free running timer count registers (TCR1 and TCR2).
  — TCR1 is clocked from the internal TPU3 system clock.
  — TCR2 may be clocked or gated from the external T2CLK pin.
  — All time primitives are microcoded.
  — 4 Kbytes of microstore program ROM space.
  — All channels have eight 16-bit parameter registers.
  — A hardware scheduler with three priority levels is included.
  — Resolution is one system clock period.
  — Modulus prescaler (DIV 2, 4, 6..... 62, 64)
- 6-Kbyte TPU emulation RAM (DPTRAM).
  — Can be used as system RAM or TPU microcode storage.
- 256K Byte 1T Flash EEPROM (CMFI).
  — 5 V program/erase.
  — Block 0 protected by an external pin.
- SRAM
  — External V_{STBY} pin for separate standby supply.
  — 8-Kbyte static RAM (SRAM):
  — 4 x 512-byte static RAM (SRAM):
  — 512-byte modules can overlay any 512-byte portion (on 512-byte boundaries) of the CMFI module.
- Late programmable read only memory – ROM:
  — 8-Kbyte array, accessible as bytes or words.
  — User selectable default base address.
  — User selectable bootstrap ROM function.
- CAN2.0B controller module (TouCAN™):
  — Full implementation of CAN protocol specification – Version 2.0B.
  — Standard/extended data and remote frames (up to 109/127 bits long).
  — Programmable bit rate up to 1 Mbit/sec, derived from system clock.
  — 16 Rx/Tx message buffers of 0-8 bytes data length, of which 2 buffers are configurable to work as Rx buffers with specific programmable masks.
  — Programmable global Rx masks.
  — Programmable transmit-first scheme: lowest ID or lowest buffer number.
  — Low power “SLEEP” mode, with programmable “WAKE UP” on bus activity.
  — Automatic time-stamping of received and transmitted messages.
- Configurable timer module 9 (CTM9):
  — 1 bus interface unit submodule (BIUSM).
  — 1 counter prescaler submodule (CPSM).
  — 1 free-running counter submodule (FCSM).

— 2 modulus counter submodules (MCSM).
— 4 single action submodules (SASM).
— 4 double action submodules (DASM).
— 4 dedicated PWM submodules (PWMSM)

- Package: flip chip and 217-ball 23 x 23 mm PBGA.
- Operating temperature: -40° C through 125° C
- Operating frequency: 33.00 MHz system clock at V_{DDL} = 3.3 V / V_{DDH} = 5.0 V.

1.3 Module Descriptions

A short description of each module of the MC68F375 appears in the following sections. For more details on each module, please refer to the specific module section as indicated.

1.3.1 Central Processing Unit Module – CPU32

The CPU32 is upward-compatible with the M68000 family which excels at processing calculation-intensive algorithms and supporting high level languages. All of the MC68010 and most of the MC68020 enhancements such as virtual memory support, loop mode execution, and 32-bit mathematical operations are supported. New instructions such as table lookup and interpolate and low-power stop support the specific requirements of controller applications. Refer to **SECTION 3 CENTRAL PROCESSOR UNIT**.

1.3.2 Single Chip Integration Module – SCIM2E

The MC68F375 contains the single chip integration module 2 (SCIM2E). The SCIM2E consists of several submodules: the external bus interface, the system protection submodule, the test submodule, the clock synthesizer, and the chip select submodule. Refer to **SECTION 4 SINGLE-CHIP INTEGRATION MODULE 2 (SCIM2E)**.

The MC68F375 SCIM2E includes improvements to the regular SCIM. This enhanced SCIM includes improvements to the clock synthesizer. These changes are defined in detail in **4.3.2 Clock Synthesizer Submodule**. Please refer to the *SCIM Reference Manual* (SCIMRM/AD) for more details on the characteristics of this module.

1.3.3 Queued Analog to Digital Converter Module – QADC64

The queued analog-to-digital converter 64 (QADC64) provides the microcontroller unit (MCU) with two conversion sequence control mechanisms (queues); a 16 channel expandable multiplexer; a 10-bit A/D converter; and digital port logic. Refer to **SECTION 5 QUEUED ANALOG-TO-DIGITAL CONVERTER MODULE-64**.

- The queue RAM stores the sequence of channels to convert, conversion parameters, and stores conversion results.
- The queue control logic provides the trigger/run modes, interrupt control, queue status, etc.
- The 10-bit A/D converter selects and convert the desired channel, using a successive approximation technique. The absolute accuracy (total unadjusted error) compared to an ideal transfer curve is +/-2 counts.

- The port logic provides up to 8 input-only and 8 bidirectional digital interface pins. Pins which are used as analog channels should be masked out of the digital data.

1.3.4 Analog Multiplexer – AMUX

The analog multiplexer (AMUX) submodule expands the channel capacity of the QADC64 analog-to-digital converter inputs by a maximum of 32 analog channels. 16 analog channels are bonded out on the MC68F375. The AMUX does not have an intermodule bus (IMB3) interface; control is through the QADC64. Refer to **5.13 Analog Multiplexer Submodule**.

1.3.5 Queued Serial Multi-Channel Communications Module – QSMCM

The queued serial multi-channel module (QSMCM) provides the microcontroller unit (MCU) with three serial communication interfaces divided into three submodules: the queued serial peripheral interface (QSPI) and two serial communications interfaces (SCI). These submodules communicate with the CPU via a common slave bus interface unit (SBIU). Refer to **SECTION 5 QUEUED ANALOG-TO-DIGITAL CONVERTER MODULE-64**.

The QSPI is a full-duplex, synchronous serial interface for communicating with peripherals and other MCUs. It is enhanced from the original QSM to include a total of 160 bytes of queue RAM to accommodate more receive, transmit, and control information.

The duplicate, independent, SCIs are full-duplex universal asynchronous receiver transmitter (UART) serial interface. The original QSM SCI is enhanced by the addition of an SCI, a common external baud clock source, receive and transmit buffers on one SCI.

1.3.6 TouCAN Module

The TouCAN module is a communication controller implementing the CAN protocol. It contains all the logic needed to implement the CAN2.0B protocol, supporting both standard ID format and extended ID. The protocol is a CSMA/CD type, with collision detection without loss, used mainly for vehicle systems communication and industrial applications. The module contains 16 message buffers used for transmit and receive, and masks used to qualify the received message ID before comparing it to the receive buffers. Refer to **SECTION 7 CAN 2.0B CONTROLLER MODULE**.

1.3.7 Enhanced Time Processing Unit – TPU3

The TPU3 is an intelligent, semi-autonomous co-processor designed for timing control. Operating simultaneously with the CPU, the TPU processes microinstructions, schedules and processes real-time hardware events, performs input and output, and accesses shared data without CPU intervention. Consequently, for each timer event the CPU setup and service time are minimized or eliminated.

The TPU3 can be viewed as a special-purpose microcomputer that performs a programmable series of two operations, match and capture. Each occurrence of either operation is called an event. A programmed series of events is called a function. TPU3

functions replace software functions that would require host CPU interrupt service. Refer to **SECTION 8 TIME PROCESSOR UNIT 3**.

1.3.8 DPTRAM TPU Emulation RAM Module – DPTRAM

The RAM module with TPU microcode storage support (DPTRAM) consists of a control register block and a 6-Kbyte array of static RAM which can be used as a microcode storage for TPU3 or general purpose memory. Microcode initialization is done by the host CPU through standard IMB modes of access. Refer to **SECTION 9 DUAL-PORT TPU RAM (DPTRAM)**.

The DPTRAM interface includes an IMB3 bus Interface and two[1] TPU3 interfaces. When the RAM is being used in microcode mode, the array may only be accessed by the TPU3 via a separate local bus, and not via the intermodule bus.

1.3.9 1T Flash Electrically Erasable Read Only Memory – CMFI

The MC68F375 contains an electrically erasable, programmable 256-Kbyte FLASH memory (CMFI). The primary function of the CMFI module is to serve as electrically programmable and erasable non-volatile memory (NVM) to store program instructions and/or data. It is a non-volatile solid state silicon memory device consisting of an array of isolated elements, a means for selectively adding and removing charge to the elements electrically and a means of selectively sensing the stored charge in the elements. When power is removed from the device, the stored charge of the isolated elements will be retained. Refer to **SECTION 10 CDR MoneT FLASH FOR THE IMB3 (CMFI)**.

1.3.10 Static RAM – SRAM

There are two types of SRAM in the MC68F375:

- 8K Static RAM – SRAM. This module is a fast access (2 clocks) general purpose static RAM (SRAM) for the MCU and is accessed via the IMB.
- 2K (4 x 512 Byte) Patch Static RAM – SRAM. These modules are fast access (2 clocks) general purpose static RAMs (SRAM) for the MCU with a patch option which provides a method to overlay the internal CMFI memory for emulation.

Refer to **SECTION 11 STATIC RANDOM ACCESS MEMORY (SRAM)**.

1.3.11 Mask Programmable Read Only Memory – ROM

The Mask ROM module is designed to be used with the inter-module bus (IMB3) and consequently any CPU capable of operating on the IMB. A size of 8192 (8K) bytes was selected to reside on the MC68F375 MCU. The ROM is a “late programmable” type which means that programming of the array and control register options occurs later in the processing flow, allowing reduction in cycle time between software code changes from the user to available devices.

During master reset the ROM will monitor one DATA line (DATA14) to determine if it should respond as a memory mapped ROM, or be disabled. If the state of DATA14 is

[1] Note: the MC68F375 contains only a single TPU3. The second TPU3 interface is inactive.

1, the STOP bit in the ROMMCR register will be cleared to 0 and the array will respond normally to the bootstrap address range and the ROM array base address. If DATA14 is 0, the STOP bit will be set and the bootstrap address range and the ROM array will be disabled until the STOP bit is cleared either by an IMB write or until the next master reset which occurs with DATA14 = 1. When STOP is set, the array will not respond to the bootstrap address range or the ROM Array base address in ROMBAH and ROMBAL, allowing an external device to respond to the ROM Array's address space, and/or provide bootstrap information. This allows the ROM to be disabled from outside of the device if necessary. Refer to **SECTION 12 MASK ROM MODULE**.

1.3.12 Configurable Timer Module 9 – CTM9

The configurable timer module (CTM) is a family of timer modules for the Motorola Modular Microcontroller Family (MMF). The timer architecture is modular—before the chip is manufactured the user can specify the number of time-bases (counter submodules) and channels (action submodules, or timer I/O pins) that are included. Refer to **SECTION 13 CONFIGURABLE TIMER MODULE (CTM9)**. The major blocks in the CTM9 are:

- Bus interface unit submodule (BIUSM)
- Single action submodule (SASM)
- Double action submodule (DASM)
- Pulse width modulation submodule (PWMSM)
- Counter prescaler submodule (CPSM)
- Free-running counter submodule (FCSM)
- Modulus counter submodule (MCSM)

1.4 Referenced Documents

- ***CPU32 Central Processor Unit Reference Manual* (CPU32RM/AD).**
- ***SCIM Reference Manual* (SCIMRM/AD)**.
- ***QSM Reference Manual* (QSMRM/AD or QSM section of MC68332UM/AD)**.
- ***CTM Reference Manual* (CTMRM/D)**.
- ***TPU Reference Manual* (TPURM/AD)**.

1.5 MC68F375 Functional Block Diagram

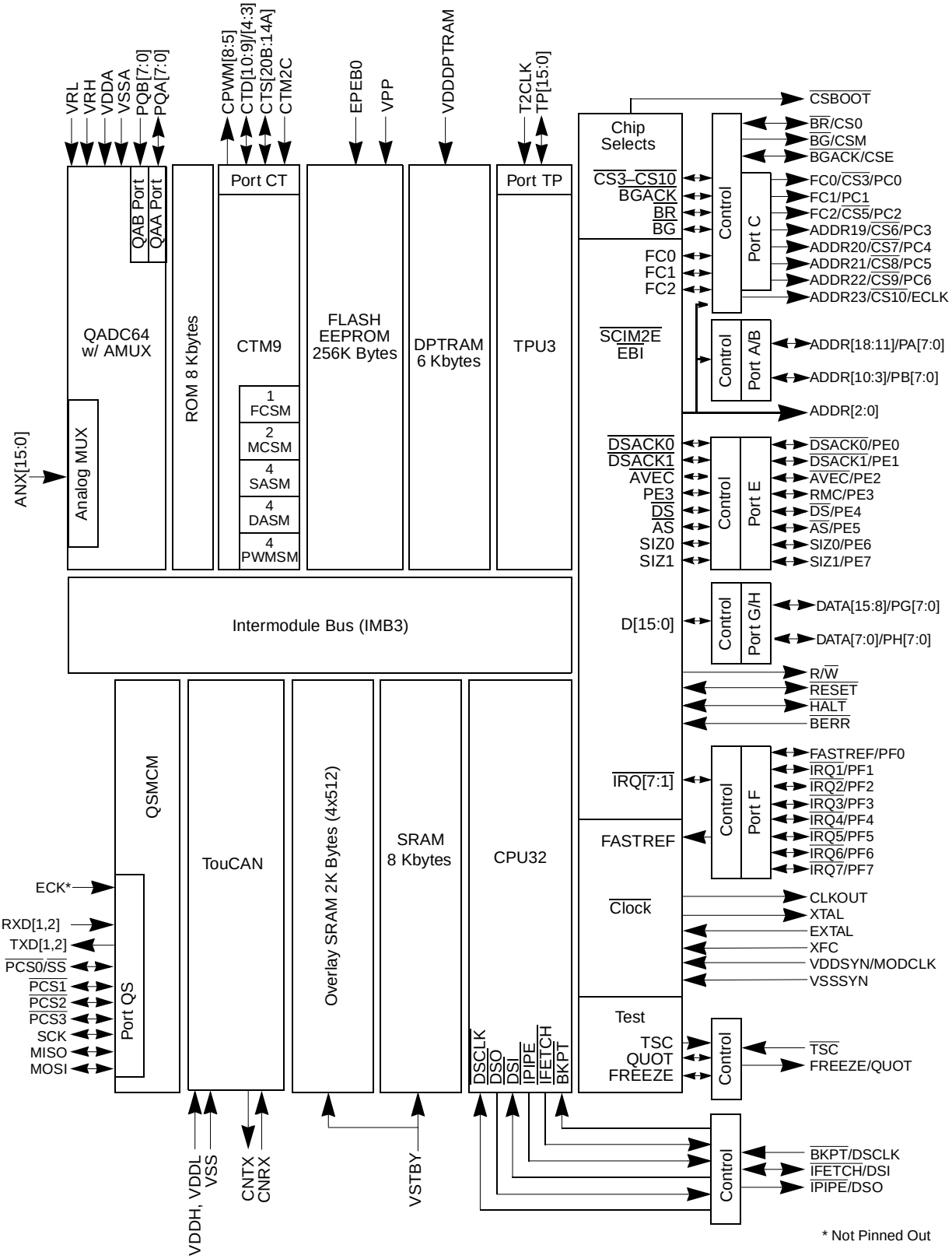


Figure 1-1 MC68F375 Block Diagram

1.6 MC68F375 Pin Usage

Table 1-1 shows the MC68F375 pin usage. For more information on an individual pin, refer to the corresponding module documentation.

Table 1-1 MC68F375 Pin Usage

Functions	Total Pins	Port Pins
CPU32		
BDM	3	
Total	**3**	
SCIM2E		
Port A, Port B	16	16
Port G, Port H	16	16
Port F	8	8
Port C	7	7
Port E	8	8
Bus Control, A[2:0], CSBOOT, TSC, RESET	13	
CLKOUT, PLL	4	
Total	**72**	**55**
QADC64/AMUX		
Channels	16	16
Analog Mux	**16**	
Total	**32**	**16**
QSMCM		
SCI1, SCI2	4	4
SPI	7	7
ECK		
Total	**11**	**11**
TPU3		
Channels	16	16
Clock	1	
Total	**17**	**16**
TouCAN		
Receive	1	
Transmit	1	
Total	**2**	
CTM9		
PWMSM	4	4
SASM	8	8
DASM	4	4
Clock	1	
Total	**17**	**16**
CMFI		
EPEB0	1	
Total	1	

Table 1-1 MC68F375 Pin Usage (Continued)

Functions	Total Pins	Port Pins
Power (V_{DDH}, V_{DDL}, V_{SS})		
I/O Pad Power (V_{DDH})	14	
Logic Power (V_{DDL})	3	
Ground (V_{SS})	16	
SRAM (V_{STBY})	1	
DPTRAM ($V_{DDDPTRAM}$)	1	
CMFI (V_{PP})	1	
SCIM2E Clock (V_{DDSYN}/V_{SSSYN})	2	
QADC64 Power (V_{DDA}/V_{SSA})	2	
QADC64 Reference (V_{RH}/V_{RL})	2	
Total	**42**	
Total	**199**	**114**

1.7 Address Map

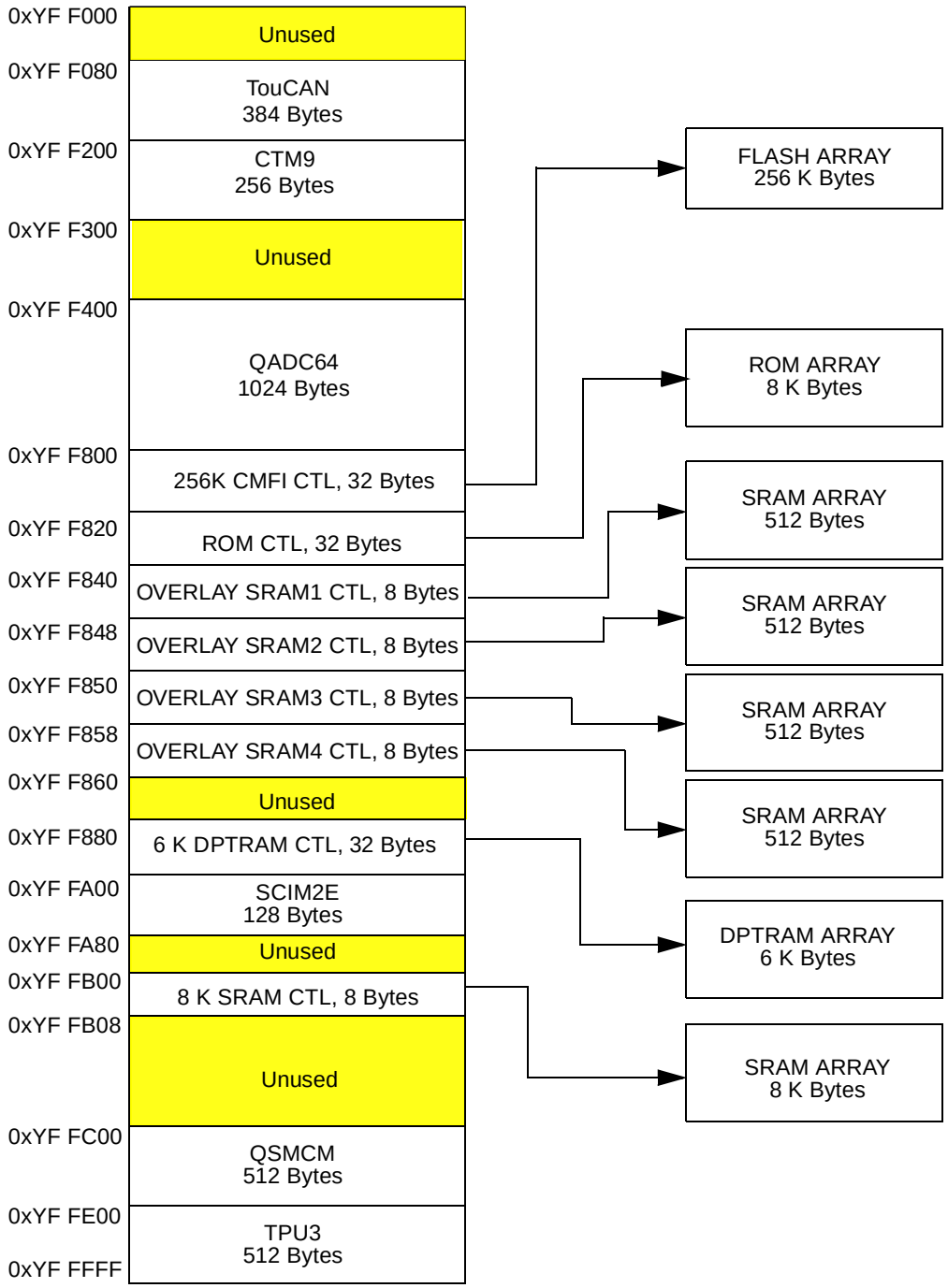
Each MC68300/MC68HC16 derivative MCU has a 4-Kbyte block in the memory map that is assigned to internal module registers. The MSB of the block address is user-programmable via the MM (MODMAP) bit in the SCIMMCR register, see **1.7.1 Address Bus Decoding**. Within this 4-Kbyte block are sub-blocks that are assigned to the individual modules, as shown in the MC68F375 address map in **Figure 1-2**. These sub-blocks vary in size, depending on the register requirements for each module. Positioning of registers within the module sub-blocks is dependent on the individual modules and is specified in the respective module specifications. The RAM, ROM, and CMFI array spaces are not shown with an assigned address because the arrays are not enabled after reset. The arrays are enabled and the base address is set by writing to the control blocks of each array.

Each module that is assigned a sub-block is responsible for responding to any accesses within its assigned address range. Modules do not respond to any register address within the module address space which is not implemented and not reserved. These addresses are mapped outside the MC68F375. An access to a register at an address which is reserved returns zeros, as do accesses to reserved bits within registers.

If the SUPV bit in the module configuration register (MCR) within a module is set, any user mode accesses to the module are mapped externally.

1.7.1 Address Bus Decoding

The internal MODMAP line is compared to internal address line A[23] while A[22:0] are decoded by each module. If a module control block address is decoded, the access will be internal. The value of A[22] down to the module block size is defined in **Figure 1-2** for each module. These bits are fixed for this particular derivative device and are specified by Motorola. The value of MODMAP is specified by the MM control bit in the SCIM2E module configuration register (SCIMMCR), see **4.2.2 Module Mapping**.



Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIM2E configuration register (SCIMMCR).

Figure 1-2 MC68F375 Address Map

SECTION 2
SIGNAL DESCRIPTIONS

2.1 Pin Characteristics

Table 2-1 shows MCU pins and their characteristics. All inputs detect CMOS logic levels. All inputs can be put in a high impedance state, but the method of doing this differs depending upon pin function. Refer to **Table 2-3** for a description of output drivers. An entry in the discrete I/O column of the MC68F375 pin characteristics table indicates that a pin has an alternate I/O function. The port designation is given when it applies. Refer to the MC68F375 block diagram for information about port organization.

Table 2-1 Pin Characteristics

Pin Mnemonic	Module	Output Driver	Input Type			Port Designation
			Sync	Hyst Sync	Hyst/Other	
ADDR[23]/ CS[10]/ECLK	SCIM2E	A	no	yes	no	—
ADDR[22:19]/ CS[9:6]/PC[6:3]	SCIM2E	A	no	yes	no	PC[6:3]
ADDR[18:11]/PA[7:0]	SCIM2E	A	no	yes	no	PA[7:0]
ADDR[10:3]/PB[7:0]	SCIM2E	A	no	yes	no	PB[7:0]
ADDR[2:0]	SCIM2E	A	no	yes	no	—
AN[59:57]/PQA[7:5]	QADC64	Aqa	no	yes[1]	no	PQA[7:5]
AN[56:55]/ETRIG[2:1]/ PQA[4:3]	QADC64	Aqa	no	yes[2]	no	PQA[4:3]
AN[54:52]/MA[2:0]/ PQA[2:0]	QADC64	Aqa	no	yes[3]	no	PQA[2:0]
AN[51:48]/PQB[7:4]	QADC64	—	no	yes	no	PQB[7:4]
AN[3:0]/AN[z,y,x,w]/ PQB[3:0]	QADC64	—	no	yes[1]	no	PQB[3:0]
ANX[15:0]	QADC64	—	no	yes[1]	no	—
AS/PE[5]	SCIM2E	B	no	yes	no	PE[5]
AVEC/PE[2]	SCIM2E	B	yes	no	no	PE[2]
$\overline{\text{BERR}}$	SCIM2E	B	yes	no	no	—
BG/CSM	SCIM2E	B	no	no	no	—
BGACK/CSE	SCIM2E	B	yes	no	no	—
BKPT/DSCLK	CPU	—	no	yes	no	—
BR/CS[0]	SCIM2E	B	yes	no	no	—
CLKOUT	SCIM2E	A	no	no	no	—
CNRX	TOUCAN	—	no	yes	no	—
CNTX	TOUCAN	Bo	no	yes	no	—
CSBOOT	SCIM2E	B	no	no	no	—

Table 2-1 Pin Characteristics (Continued)

Pin Mnemonic	Module	Output Driver	Input Type			Port Designation
			Sync	Hyst Sync	Hyst/Other	
CPWM[8:5]	CTM9	A	no	yes	no	—
CTD[10:9]	CTM9	A	no	yes	no	—
CTM2C	CTM9	A	no	yes	no	—
CTS[20B-14A]	CTM9	A	no	yes	no	—
DATA[15:8]/PG[7:0]	SCIM2E	DB	no	yes[1]	no	PG[7:0]
DATA[7:0]/PH[7:0]	SCIM2E	DB	no	yes	no	PH[7:0]
DS/PE[4]	SCIM2E	B	no	yes	no	PE[4]
DSACK[1:0]/PE[1:0]	SCIM2E	B	yes	no	no	PE[1:0]
ETRIG[2:1]/PQA[4:3]	QADC64	Aqa	no	yes[1]	no	PQA[4:3]
EXTAL[4,5]	PLLWXY	—	no	no	Special	—
FASTREF/PF[0]	SCIM2E	Bp	no	yes[1]	no	PF[0]
FC[2]/CS[5]/PC[2] FC[1]/PC[1] FC[0]/CS[3]/PC[0]	SCIM2E	A	yes	no	no	PC[2:0]
FREEZE /QOUT	SCIM2E	A	no	no	no	—
HALT	SCIM2E	Bop	yes	no	no	—
$\overline{\text{IFETCH}}$/DSI	CPU	A	no	yes	no	—
IPIPE/DSO	CPU	A	no	no	no	—
IRQ[7]/PF[7]	SCIM2E	Bp	no	yes	no	PF[7]
IRQ[6]/PF[6]	SCIM2E	B	no	yes	no	PF[6]
IRQ[5:1]/PF[5:1]	SCIM2E	B	no	yes	no	PF[5:1]
MA[2:0]/PQA[2:0]	QADC64	Aqa	no	yes[1]	no	PQA[2:0]
MISO/PQS[0]	QSMCM	Bo	no	yes[1]	no	PQS[0]
MOSI/PQS[1]	QSMCM	Bo	no	yes[1]	no	PQS[1]
PCS[0]/SS/PQS[3]	QSMCM	Bo	no	yes[1]	no	PQS[3]
PCS[3:1]/PQS[6:4]	QSMCM	Bo	no	yes[1]	no	PQS[6:4]
R/$\overline{\text{W}}$	SCIM2E	A	yes	yes	no	—
RESET	SCIM2E	Bo	no	yes	no	—
RMC(BLOCK)/PE[3]	SCIM2E	B	no	yes	no	PE[3]
RXD[1,2]/PQS[8,10]	QSMCM	—	no	no	yes	PQS[8,10]
SCK/PQS[2]	QSMCM	Bo	no	yes[1]	no	PQS[2]
SIZ[1:0]/PE[7:6]	SCIM2E	B	no	yes	no	PE[7:6]
T2CLK	TPU3	A	no	yes	no	—
TP[15:0]	TPU3	A	no	yes	no	—
$\overline{\text{TSC}}$	SCIM2E	—	no	yes	no	—
TXD[1,2]/PQS[7,9]	QSMCM	Bo	no	yes[1]	no	PQS[7,9]
VDDSYN /MODCLK	SCIM2E	—	no	yes[1]	no	—
EPEB0[3]	CMFI	—	yes	—	—	—

Table 2-1 Pin Characteristics (Continued)

Pin Mnemonic	Module	Output Driver	Input Type			Port Designation
			Sync	Hyst Sync	Hyst/Other	
XFC[4]	PLLWXY	—	no	no	Special	—
XTAL[4,3]	PLLWXY	—	no	no	Special	—

NOTES:
1. DATA[15:0]
2. DATA[15:0]
3. DATA[15:0]
4. EXTAL, XFC and XTAL are clock reference connections.
5. These pins are 3 V level only.

Table 2-2 Power Connections

Pin	Voltage	Description
V_{STBY}	3.3 V[1]	Standby RAM power
$V_{DDDPTRAM}$	3.3 V	DPTRAM power (connect to V_{DDL})
V_{DDSYN} V_{SSSYN}	3.3 V[2] 0 V	Clock synthesizer power
V_{DDA} V_{SSA}	5 V 0 V	QADC64 converter power
V_{RH} V_{RL}	5 V 0 V	QADC64 reference voltage
V_{SS} V_{DDH}	0 V 5 V	Pad output driver power (Source and Drain)
V_{SS} V_{DDL}	0 V 3.3 V	Module power (Source and Drain)
V_{PP}	3.3 V/5 V	CMFI program/erase voltage

NOTES:
1. Throughout this manual 3 V = 3.3 volts ±0.3 volts.
   5 V = 5 volts ±10%, except V_{PP} 5 V = 5 volts ±5%.
2. V_{DDSYN}/MODCLK = 3.3 V for V_{DDSYN} function.
   V_{DDSYN}/MODCLK = 0 V for MODCLK function.

Table 2-3 Output Driver Types

Type	I/O	Description
A	O	Output that is always driven. No external pull-up required.
Aqa	O	Type A output open drain on a QADC64 pin
B	O	Three-state output that includes circuitry to assert output before high impedance is established, to ensure rapid rise time. An external holding resistor is required to maintain logic level while in the high-impedance state.
Bo	O	Type B output that can be operated in an open-drain mode.
Bop	O	Type B output that can be operated in an open-drain mode with weak pull-up.
Bp	O	Type B output with weak pull-up.
DB	O	Data bus driver with weak pull-up which pulls data bus high during reset.

Table 2-4 Signal Characteristics

Signal Name	MCU Module	Signal Type	Active State
ADDR[23:0]	SCIM2E	Bus	—
AN[59:48]/[3:0]	QADC64	Input	—
AN[z, y, x, w]	QADC64	Input	—
ANX[15:0]	Analog MUX	Input	—
AS	SCIM2E	Output	0
AVEC	SCIM2E	Input	0
BERR	SCIM2E	Input	0
BG	SCIM2E	Output	0
BGACK	SCIM2E	Input	0
BKPT	CPU32	Input	0
BR	SCIM2E	Input	0
CLKOUT	SCIM2E	Output	—
CNRX	TOUCAN	Input	---
CNTX	TOUCAN	Output	---
CS[10:5], CS[3], CS[0]	SCIM2E	Output	0
CSBOOT	SCIM2E	Output	0
CSE	SCIM2E	Output	0
CSM	SCIM2E	Output	0
CPWM[8:5]	CTM9	Output	1/0
CTD[10:9]	CTM9	Input/Output	—
CTM2C	CTM9	Input	1/0
CTS[20B - 14A]	CTM9	Input/Output	—
DATA[15:0]	SCIM2E	Bus	—
DS	SCIM2E	Output	0
DSACK[1:0]	SCIM2E	Input	0
DSCLK	CPU32	Input	Serial Clock
DSI	CPU32	Input	Serial Data
DSO	CPU32	Output	Serial Data
ECLK	SCIM2E	Output	—

Table 2-4 Signal Characteristics (Continued)

Signal Name	MCU Module	Signal Type	Active State
ETRIG[2:1]	QADC64	Input	1/0
EXTAL[1]	SCIM2E	Input	—
FASTREF	SCIM2E	Input/Output	1
FC[2]/CS[5]/PC[2] FC[1]/PC[1] FC[0]/CS[3]/PC[0]	SCIM2E	Output	1/0
FREEZE	SCIM2E	Output	1
HALT	SCIM2E	Input/Output	0
IPIPE	CPU32	Output	0
IFETCH	CPU32	Output	0
IRQ[7:1]	SCIM2E	Input	0
MA[2:0]	QADC64	Output	1
MISO	QSMCM	Input/Output	—
MOSI	QSMCM	Input/Output	—
PCS[3:0]	QSMCM	Input/Output	—
PQA[7:0]	QADC64	Input/Output	—
PQB[7:0]	QADC64	Input	—
QUOT	SCIM2E	Output	—
R/$\overline{W}$	SCIM2E	Output	1/0
RESET	SCIM2E	Input/Output	0
RMC	SCIM2E	Output	0
RXD1, RXD2	QSMCM	Input	—
SCK	QSMCM	Input/Output	—
SIZ[1:0]	SCIM2E	Output	1
SS	QSMCM	Input	0
T2CLK	TPU3	Input	—
TP[15:0]	TPU3	Input/Output	—
TSC	SCIM2E	Input	0/1
TXD1, TXD2	QSMCM	Output	—
VDDSYN/MODCLK	SCIM2E	Input	—
EPEB0[1]	CMFI	Input	—
XFC	SCIM2E	Input	—
XTAL[1]	SCIM2E	Output	—

NOTES:
1. These pins are 3 V level only.

Table 2-5 Signal Functions

Signal Name	Mnemonic	Function
Address Bus	ADDR[23:0]	24-bit address bus used by CPU32
QADC64 Analog Input	AN[59:48]/[3:0]	Sixteen channel A/D converter analog input pins
QADC64 Analog Input	AN[z, y, x, w]	Four input channels utilized when operating in multiplexed mode
Analog MUX Inputs	ANX[15:0]	Analog signal inputs multiplexed to the analog converters
Address Strobe	AS	Indicates that a valid address is on the address bus
Autovector	AVEC	Requests an automatic vector during interrupt acknowledge
Bus Error	BERR	Indicates that a bus error has occurred
Bus Grant	BG	Indicates that the MCU has relinquished the bus
Bus Grant Acknowledge	BGACK	Indicates that an external device has assumed bus mastership
Breakpoint	BKPT	Signals a hardware breakpoint to the CPU
Bus Request	BR	Indicates that an external device requires bus mastership
System Clock Out	CLKOUT	Internal system clock
TOUCAN Receive Data	CNRX	CAN 2.0B Serial Data Input
TOUCAN Transmit Data	CNTX	CAN 2.0B Serial Data Output
Chip Selects	CS[10:5], CS[3], CS[0]	Select external devices at programmed addresses
Boot Chip Select	CSBOOT	Chip select for external boot start-up ROM
Emulator Chip Select	CSE	Chip select for external port emulator
Module Chip Select	CSM	Chip select for external ROM emulator
CTM PWM	CPWM[8:5]	PWM channels which can also be used as general-purpose output pins
CTM9 Double Action Channel	CTD[10:9]	Bidirectional CTM9 double action timer channels
CTM9 Modulus Clock	CTM2C	CTM9 modulus counter clock input
CTM9 Single Action Channels	CTS[20B - 14A]	Bidirectional CTM9 single action timer channels
Data Bus	DATA[15:0]	16-bit data bus
Data Strobe	DS	Read cycle — indicates that an external device should place valid data on the data bus. Write cycle — indicates that valid data is on the data bus.
Data and Size Acknowledge	DSACK[1:0]	Provides asynchronous data transfers and dynamic bus sizing
Development Serial In, Out, Clock	DSI, DSO, DSCLK	Serial I/O and clock for background debug mode
QADC64 External Trigger	ETRIG[2:1]	When a scan queue is in external trigger mode, the corresponding ETRIG pin is configured as a digital input and the software programmed I/O direction in the DDR is ignored.
Crystal Oscillator	EXTAL, XTAL	Connections for clock synthesizer circuit reference; a crystal or an external oscillator can be used
VCO Reference Mode Select	FASTREF	Selects between FAST or SLOW reference modes
Function Codes	FC[2:0]	Identify processor state and current address space
Freeze	FREEZE	Indicates that the CPU has acknowledged a breakpoint
Halt	HALT	Suspend external bus activity
Instruction Pipeline	IPIPE	Indicates instruction pipeline activity

Table 2-5 Signal Functions (Continued)

Signal Name	Mnemonic	Function
Instruction Pipeline	IFETCH	Indicates instruction pipeline activity
Interrupt Request Level	IRQ[7:1]	Provides an interrupt priority level to the CPU
QADC64 Multiplexed Address	MA[2:0]	When external multiplexing is used, these pins provide the addresses to the external multiplexer
Master-In Slave-Out	MISO	Serial input to QSPI in master mode; serial output from QSPI in slave mode
Master-Out Slave-In	MOSI	Serial output from QSPI in master mode; serial input to QSPI in slave mode
Peripheral Chip Select	PCS[3:0]	QSPI Peripheral Chip Select
QADC64 Port A	PQA[7:0]	QADC64 port A analog inputs and I/O port PQA[7:0]
QADC64 Port B	PQB[7:0]	QADC64 port B analog inputs and input-only port PQB[7:0]
Quotient Out	QUOT	Provides the quotient bit of the polynomial divider (test mode only)
Read/Write	R/$\overline{W}$	Indicates the direction of data transfer on the bus
Reset	RESET	System reset
Read-Modify-Write Cycle	RMC	Indicates an indivisible read-modify-write instruction
SCI Receive Data	RXD1, RXD2	Serial input to the SCI
QSPI Serial Clock	SCK	Clock output from QSPI in master mode; clock input from QSPI in slave mode
Size	SIZ[1:0]	Indicates the number of bytes remaining to be transferred during a bus cycle
Slave Select	SS	Causes serial transmission when QSPI is in slave mode; chip-select in master mode
TPU3 Clock	T2CLK	TPU3 clock input
TPU3 I/O Channels	TP[15:0]	Bidirectional TPU3 channels
Three-State Control	TSC	Places all output drivers in a high impedance state
SCI Transmit Data	TXD1, TXD2	Serial output from the SCI
Clock Mode Select	VDDSYN/MODCLK	Selects the source of the internal system clock
CMFI Block 0 Program/ Erase Enable	EPEB0	When asserted, allows CMFI block 0 to be programmed or erased.
External Filter Capacitor	XFC	Connection for external phase-locked loop filter capacitor

2.2 Pinout

The production MC68F375 will be bumped flip-chip and PBGA.

2.2.1 Pinout Diagram

The pad numbers for each pad/signal on the die are shown in **Figure 2-1**. Note that the numbers and names correspond to the pad names and order on the die. The pin/bump numbers on the PBGA and Bumped die may be different. The chip layout plan is also shown in **Figure 2-1**.

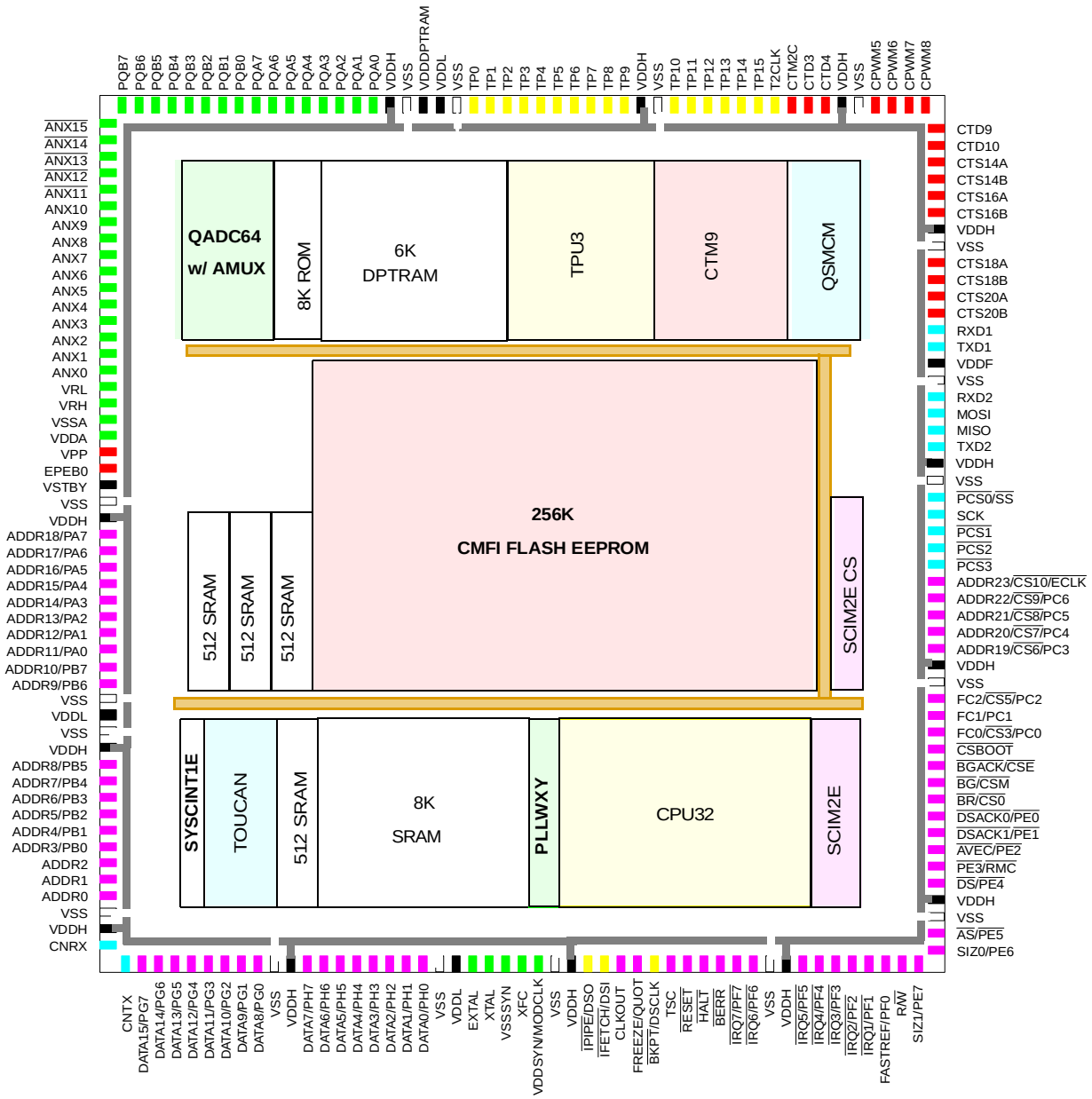
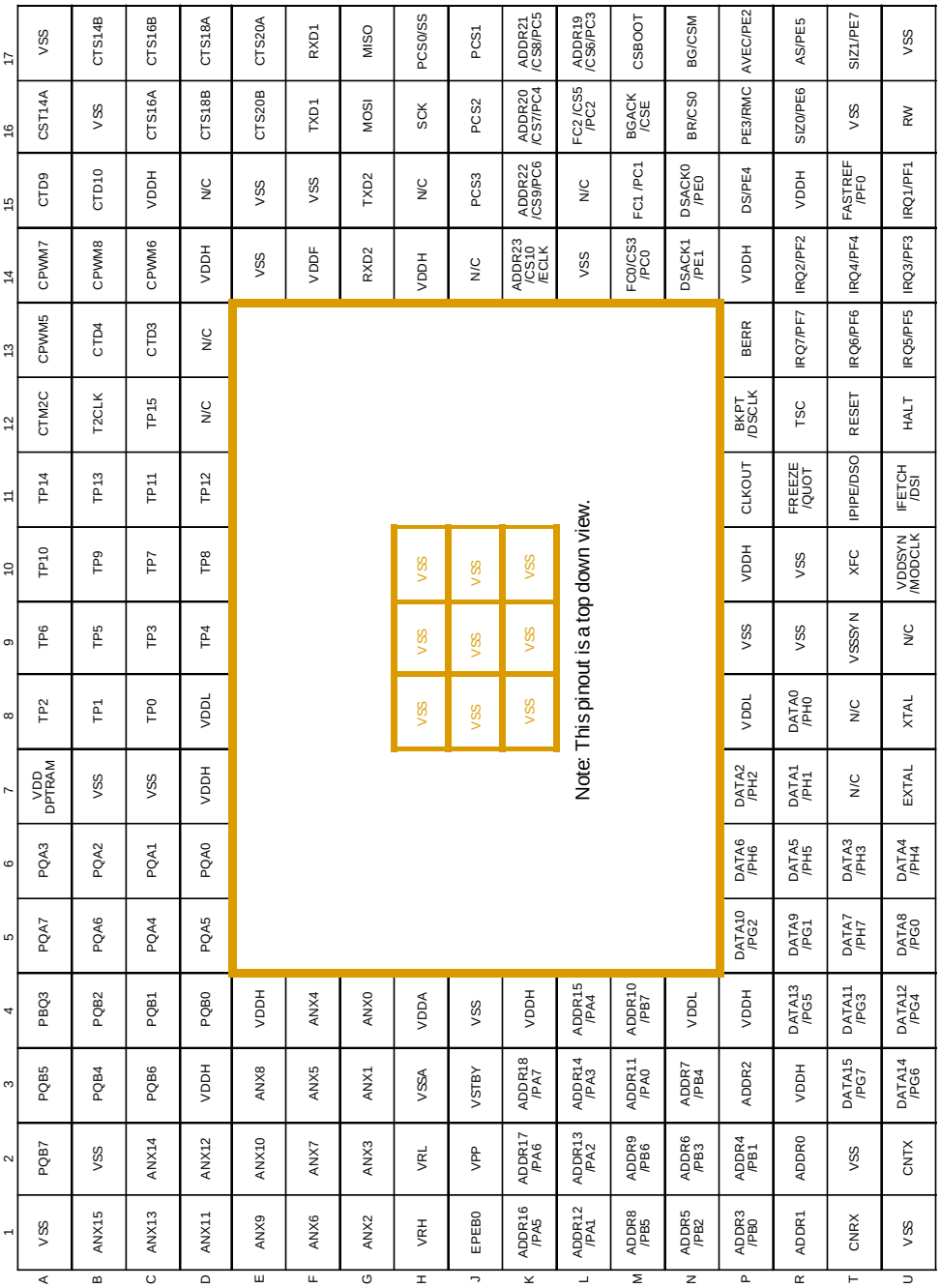


Figure 2-1 MC68F375 Pad Map



Note:
N/C = NO CONNECTION — these balls have no electrical connection inside the package.

Figure 2-2 MC68F375 Ball Map

SECTION 3
CENTRAL PROCESSOR UNIT

The CPU32, the instruction processing module of the M68300 family, is based on the industry-standard MC68000 processor. It has many features of the MC68010 and MC68020, as well as unique features suited for high-performance controller applications. This section is an overview of the CPU32. For detailed information concerning CPU operation, refer to the ***CPU32 Reference Manual*** (CPU32RM/AD).

3.1 General

Ease of programming is an important consideration in using a microcontroller. The CPU32 instruction format reflects a philosophy emphasizing register-memory interaction. There are eight multifunction data registers and seven general-purpose addressing registers.

All data resources are available to all operations requiring those resources. The data registers readily support 8-bit (byte), 16-bit (word), and 32-bit (long-word) operand lengths for all operations. Word and long-word operations support address manipulation. 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 further enhanced by trace and trap capabilities at the instruction level.

A block diagram of the CPU32 is shown in **Figure 3-1**. The major blocks operate in a highly independent fashion that maximizes concurrency 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, managing the internal buses, registers, and functions of the execution unit.

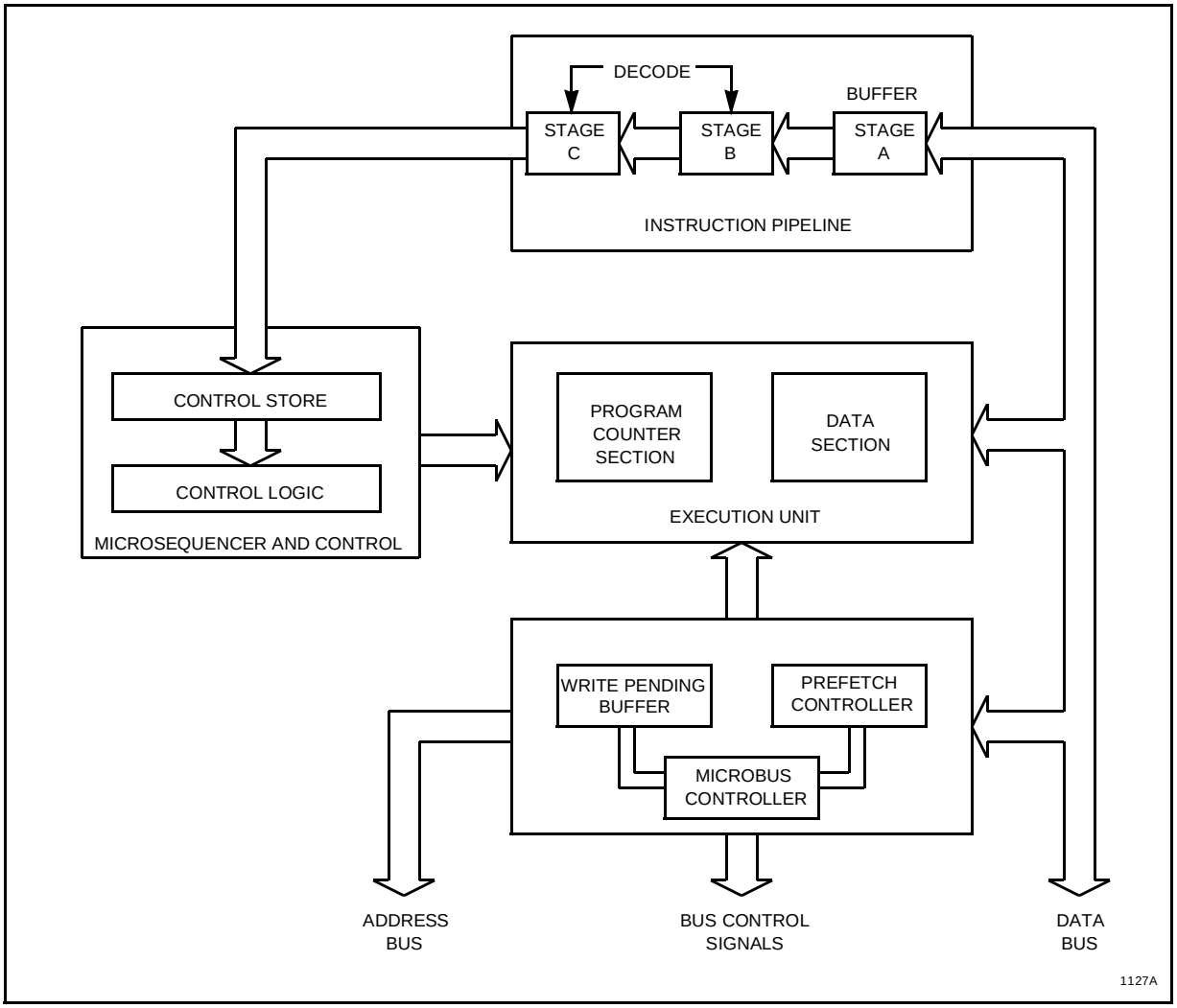


Figure 3-1 CPU32 Block Diagram

3.2 CPU32 Registers

The CPU32 programming model consists of two groups of registers that correspond to the user and supervisor privilege levels. User programs can use only 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 the MC68010 and later processors.

The CPU32 has eight 32-bit data registers, seven 32-bit address registers, a 32-bit program counter, separate 32-bit supervisor and user stack pointers, a 16-bit status register, two alternate function code registers, and a 32-bit vector base register. Refer to **Figure 3-2** and **Figure 3-3**.

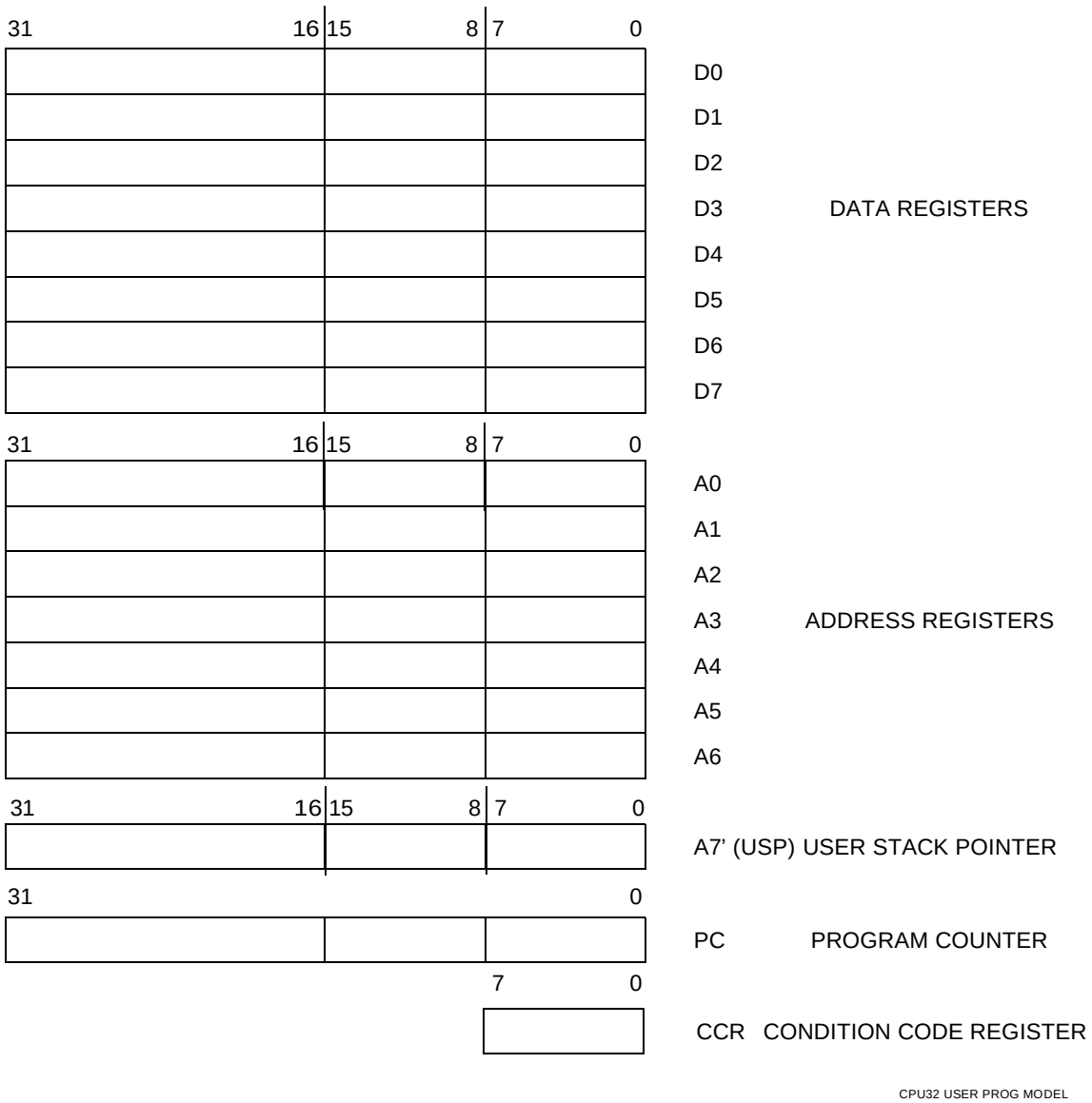


Figure 3-2 User Programming Model

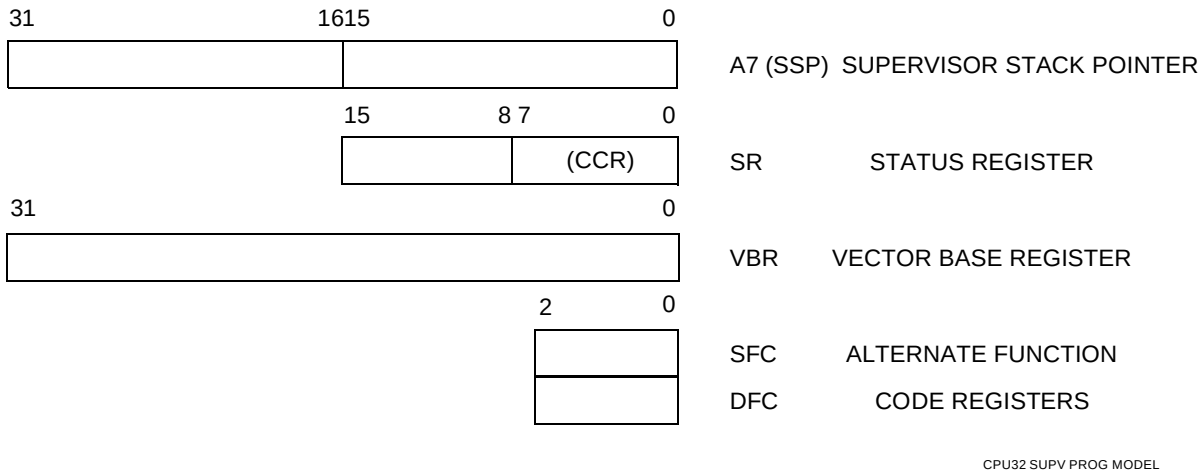


Figure 3-3 Supervisor Programming Model Supplement

3.2.1 Data Registers

The eight data registers can store data operands of 1, 8, 16, 32, and 64 bits and addresses of 16 or 32 bits. The following data types are supported:

- Bits
- Packed Binary-Coded Decimal Digits
- Byte Integers (8 bits)
- Word Integers (16 bits)
- Long-Word Integers (32 bits)
- Quad-Word Integers (64 bits)

Each of data registers D7–D0 is 32 bits wide. Byte operands occupy the low-order 8 bits; word operands, the low-order 16 bits; and long-word operands, the entire 32 bits. When a data register is used as either a source or destination operand, only the appropriate low-order byte or word (in byte or word operations, respectively) is used or changed; the remaining high-order portion is unaffected. The least significant bit (LSB) of a long-word integer is addressed as bit zero, and the most significant bit (MSB) is addressed as bit 31. **Figure 3-4** shows the organization of various types of data in the data registers.

Quad-word data consists of two long words and represents the product of 32-bit multiply or the dividend of 32-bit divide operations (signed and unsigned). Quad-words may be organized in any two data registers without restrictions on order or pairing. There are no explicit instructions for the management of this data type, although the MOVEM instruction can be used to move a quad-word into or out of the registers.

Binary-coded decimal (BCD) data represents decimal numbers in binary form. CPU32 BCD instructions use a format in which a byte contains two digits. The four LSB contain the least significant digit, and the four MSB contain the most significant digit. The ABCD, SBCD, and NBCD instructions operate on two BCD digits packed into a single byte.

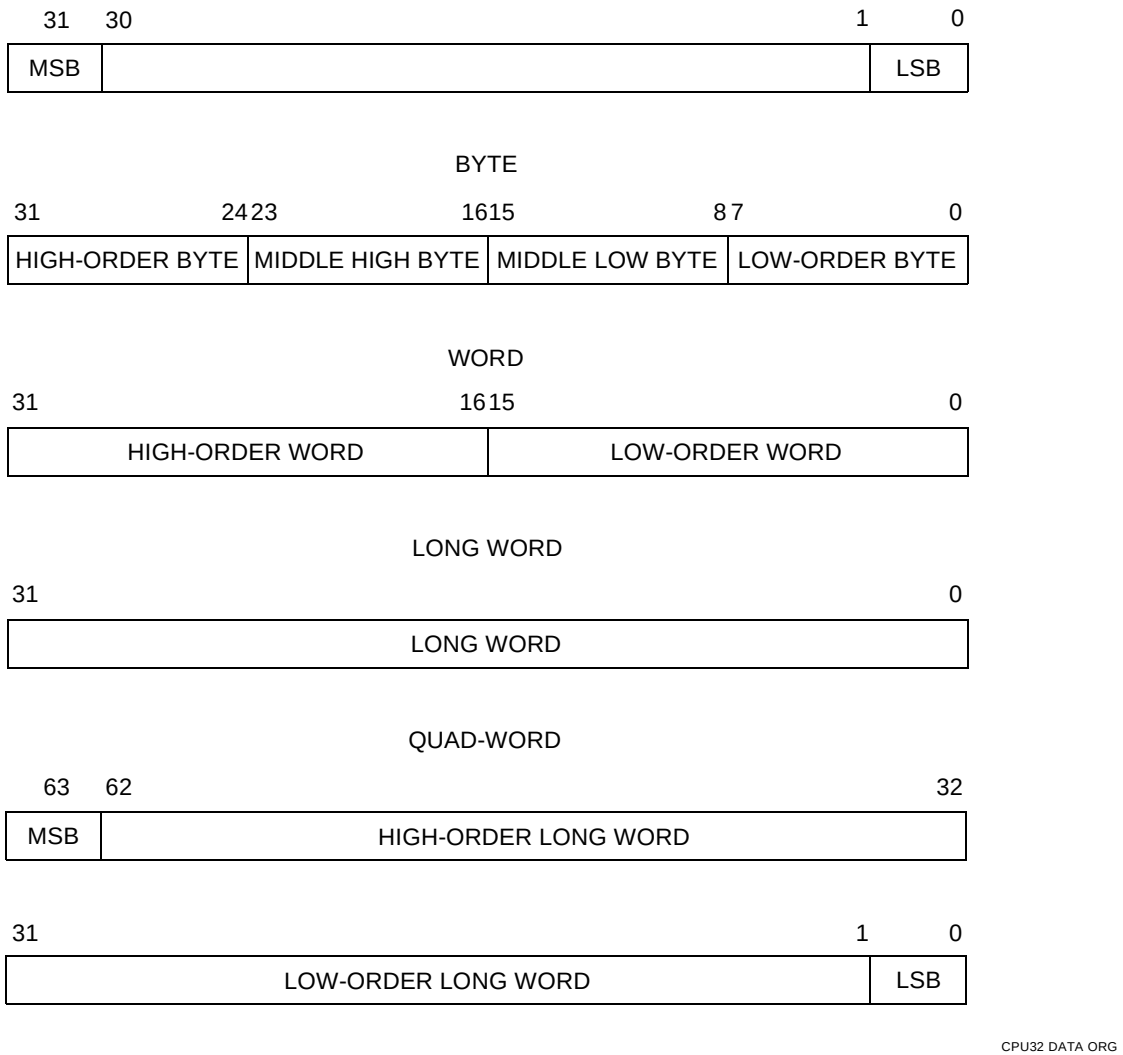


Figure 3-4 Data Organization in Data Registers

3.2.2 Address Registers

Each address register and stack pointer is 32 bits wide and holds a 32-bit address. Address registers cannot be used for byte-sized operands. Therefore, when an address register is used as a source operand, either the low-order word or the entire long-word operand is used, depending upon the operation size. When an address register is used as the destination, the entire register is affected, regardless of the operation size. If the source operand is a word size, the operand is sign-extended to 32 bits. Address registers are used primarily for addresses and to support address computation. The instruction set includes instructions that add to, subtract from, compare, and move the contents of address registers. **Figure 3-5** shows the organization of addresses in address registers.

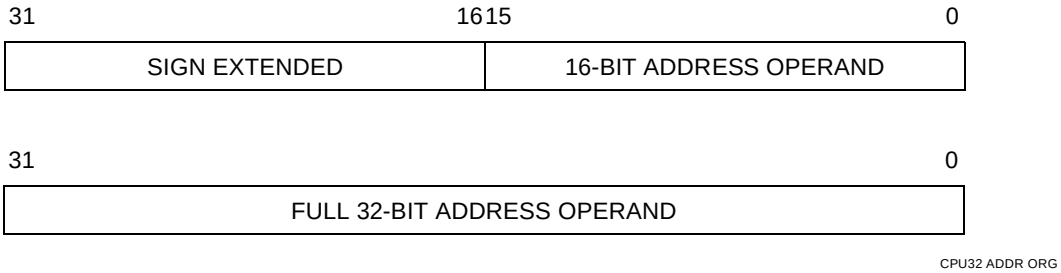


Figure 3-5 Address Organization in Address Registers

3.2.3 Program Counter

The 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.

3.2.4 Control Registers

The control registers described in this section contain control information for supervisor functions and vary in size. With the exception of the condition code register (the user portion of the status register), the control registers are accessed only by instructions at the supervisor privilege level.

3.2.4.1 Status Register

The status register (SR) stores the processor status. It contains the condition codes that reflect the results of a previous operation and can be used for conditional instruction execution in a program. The condition codes are extend (X), negative (N), zero (Z), overflow (V), and carry (C). The user (low-order) byte containing the condition codes is the only portion of the SR information available at the user privilege level; it is referenced as the condition code register (CCR) in user programs.

At the supervisor privilege level, software can access the full status register. The upper byte of this register includes the interrupt priority (IP) mask (three bits), two bits for placing the processor in one of two tracing modes or disabling tracing, and the supervisor/user bit for placing the processor at the desired privilege level.

Undefined bits in the status register are reserved by Motorola for future definition. The undefined bits are read as zeros and should be written as zeros for future compatibility.

All operations to the SR and CCR are word-size operations, but for all CCR operations, the upper byte is read as all zeros and is ignored when written, regardless of privilege level.

Refer to **APPENDIX A INTERNAL MEMORY MAP** for bit/field definitions and a diagram of the status register.

3.2.4.2 Alternate Function Code Registers

Alternate function code registers (SFC and DFC) contain 3-bit function codes. Function codes can be considered extensions of the 24-bit linear address that optionally provide as many as eight 16-Mbyte address spaces. The processor automatically generates function codes to select address spaces for data and programs at the user and supervisor privilege levels and to select a CPU address space used for processor functions (such as breakpoint and interrupt acknowledge cycles).

Registers SFC and DFC are used by the MOVES instruction to specify explicitly the function codes of the memory address. The MOVEC instruction is used to transfer values to and from the alternate function code registers. This is a long-word transfer; the upper 29 bits are read as zeros and are ignored when written.

3.2.5 Vector Base Register (VBR)

The VBR contains the base address of the 1024-byte exception vector table, consisting of 256 exception vectors. Exception vectors contain the memory addresses of routines that begin execution at the completion of exception processing. More information on the VBR and exception processing can be found in **3.9 Exception Processing**.

3.3 Memory Organization

Memory is organized on a byte-addressable basis in which lower addresses correspond to higher order bytes. For example, the address N of a long-word data item corresponds to the address of the most significant byte of the highest order word. The address of the most significant byte of the low-order word is N + 2, and the address of the least significant byte of the long word is N + 3. The CPU32 requires long-word and word data and all instructions to be aligned on word boundaries. Refer to **Figure 3-6**. If this does not happen, an exception will occur when the CPU32 accesses the misaligned instruction or data. Data misalignment is not supported.

BIT DATA

1 BYTE = 8 BITS

7	6	5	4	3	2	1	0

1 BYTE = 8 BITS

15		8 7	0
MSB	BYTE 0	LSB	BYTE 1
	BYTE 2		BYTE 3

WORD = 16 BITS

15		0
MSB	WORD 0	LSB
	WORD 1	
	WORD 2	

LONG WORD = 32 BITS

15		0
MSB	HIGH ORDER	
LONG WORD 0	LOW ORDER	LSB
LONG WORD 1		
LONG WORD 2		

ADDRESS 1

ADDRESS = 32 BITS

15		0
MSB	HIGH ORDER	
ADDRESS 0	LOW ORDER	LSB
ADDRESS 1		
ADDRESS 2		

MSB = Most Significant Bit

LSB = Least Significant Bit

DECIMAL DATA

BCD DIGITS = 1 BYTE

15	12 11	8 7	4 3	0
MSD BCD 0	BCD 1	LSD BCD 2	BCD 3	
BCD 4	BCD 5	BCD 6	BCD 7	

MSD = Most Significant Digit

LSD = Least Significant Digit

1125A

Figure 3-6 Memory Operand Addressing

3.4 Virtual Memory

The full addressing range of the CPU32 on the MC68F375 is 16 Mbytes in each of eight address spaces. Even though most systems implement a smaller physical memory, the system can be made to appear to have a full 16 Mbytes of memory available to each user program by using virtual memory techniques.

A system that supports virtual memory has a limited amount of high-speed physical memory that can be accessed directly by the processor and maintains an image of a much larger virtual memory on a secondary storage device. When the processor attempts to access a location in the virtual memory map that is not resident in physical memory, a page fault occurs. The access to that location is temporarily suspended while the necessary data is fetched from secondary storage and placed in physical memory. The suspended access is then restarted or continued.

The CPU32 uses instruction restart, which requires that only a small portion of the internal machine state be saved. After correcting the fault, the machine state is restored, and the instruction is fetched and started again. This process is completely transparent to the application program.

3.5 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. There is no need for extra instructions to store register contents in memory.

There are seven basic addressing modes:

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

The register indirect addressing modes include postincrement, predecrement, and offset capability. The program counter indirect mode also has index and offset capabilities. In addition to these addressing modes, many instructions implicitly specify the use of the status register, stack pointer, and/or program counter.

3.6 Processing States

The processor is always in one of four processing states: normal, exception, halted, or background. The normal processing state is associated with instruction execution; the bus is used to fetch instructions and operands and to store results.

The exception processing state is associated with interrupts, trap instructions, tracing, and other exception conditions. The exception may be internally generated explicitly by an instruction or by an unusual condition arising during the execution of an instruc-

tion. Exception processing can be forced externally by an interrupt, a bus error, or a reset.

The background processing state is initiated by breakpoints, execution of special instructions, or a double bus fault. Background processing is enabled by pulling $\overline{\text{BKPT}}$ low during $\overline{\text{RESET}}$. Background processing allows interactive debugging of the system via a simple serial interface.

3.7 Privilege Levels

The processor operates at one of two levels of privilege: user or supervisor. Not all instructions are permitted to execute at the user level. All instructions are available at the supervisor level. Effective use of privilege level can protect system resources from uncontrolled access. The state of the S bit in the status register determines the privilege level and whether the user stack pointer (USP) or supervisor stack pointer (SSP) is used for stack operations.

3.8 Instructions

The CPU32 instruction set is summarized in **Table 3-2**. The instruction set of the CPU32 is very similar to that of the MC68020. Two new instructions have been added to facilitate controller applications: low-power stop (LPSTOP) and table lookup and interpolate (TBLS, TBLSN, TBLU, TBLUN).

Table 3-1 shows the MC68020 instructions that are not implemented on the CPU32.

Table 3-1 Unimplemented MC68020 Instructions

Instruction	Description
BFxx	Bit Field Instructions (BFCHG, BFCLR, BFEXTS, BFEXTU, BFFFO, BFINS, BFSET, BFTST)
CALLM, RTM	Call Module, Return Module
CAS, CAS2	Compare and Swap (Read-Modify-Write Instructions)
cpxxx	Coprocessor Instructions (cpBcc, cpDBcc, cpGEN)
PACK, UNPK	Pack, Unpack BCD Instructions
Memory	Memory Indirect Addressing Modes

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 3-2 Instruction Set Summary

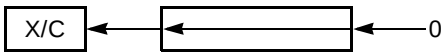
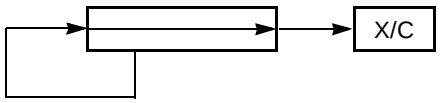
Instruction	Operand Syntax	Operand Size	Description
ABCD	Dn, Dn – (An), – (An)	8 8	$\text{Source}_{10} + \text{Destination}_{10} + X \Rightarrow \text{Destination}$
ADD	Dn, <ea> <ea>, Dn	8, 16, 32 8, 16, 32	Source + Destination ⇒ Destination
ADDA	<ea>, An	16, 32	Source + Destination ⇒ Destination
ADDI	#<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
AND	<ea>, Dn Dn, <ea>	8, 16, 32 8, 16, 32	Source • Destination ⇒ Destination
ANDI	# <data>, <ea>	8, 16, 32	Data • Destination ⇒ Destination
ANDI to CCR	# <data>, CCR	8	Source • CCR ⇒ CCR
ANDI to SR[1]	# <data>, SR	16	Source • SR ⇒ SR
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	
Bcc	label	8, 16, 32	If condition true, then PC + d ⇒ PC
BCHG	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{(\langle\text{bit number}\rangle\text{of destination})} \Rightarrow Z \Rightarrow$ bit of destination
BCLR	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{(\langle\text{bit number}\rangle\text{of destination})} \Rightarrow Z$; 0 ⇒ bit of destination
BGND	none	none	If background mode enabled, then enter background mode, else format/vector ⇒ – (SSP); PC ⇒ – (SSP); SR ⇒ – (SSP); (vector) ⇒ PC
BKPT	# <data>	none	If breakpoint cycle acknowledged, then execute returned operation word, else trap as illegal instruction
BRA	label	8, 16, 32	PC + d ⇒ PC
BSET	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{(\langle\text{bit number}\rangle\text{of destination})} \Rightarrow Z$; 1 ⇒ bit of destination
BSR	label	8, 16, 32	SP – 4 ⇒ SP; PC ⇒ (SP); PC + d ⇒ PC
BTST	Dn, <ea> # <data>, <ea>	8, 32 8, 32	$\overline{(\langle\text{bit number}\rangle\text{of destination})} \Rightarrow Z$
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
CLR	<ea>	8, 16, 32	0 ⇒ Destination

Table 3-2 Instruction Set Summary (Continued)

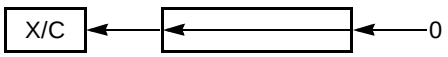
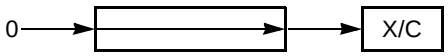
Instruction	Operand Syntax	Operand Size	Description
CMP	<ea>, Dn	8, 16, 32	(Destination – Source), CCR shows results
CMPA	<ea>, An	16, 32	(Destination – Source), CCR shows results
CMPI	# <data>, <ea>	8, 16, 32	(Destination – Data), CCR shows results
CMPM	(An) +, (An) +	8, 16, 32	(Destination – Source), CCR shows results
CMP2	<ea>, Rn	8, 16, 32	Lower bound ≤ Rn ≤ Upper bound, CCR shows result
DBcc	Dn, label	16	If condition false, then Dn – 1 ⇒ PC; if Dn ≠ (– 1), then PC + d ⇒ PC
DIVS/DIVU	<ea>, Dn	32/16 ⇒ 16 : 16	Destination / Source ⇒ Destination (signed or unsigned)
DIVSL/DIVUL	<ea>, Dr : Dq <ea>, Dq <ea>, Dr : Dq	64/32 ⇒ 32 : 32 32/32 ⇒ 32 32/32 ⇒ 32 : 32	Destination / Source ⇒ Destination (signed or unsigned)
EOR	Dn, <ea>	8, 16, 32	Source ⊕ Destination ⇒ Destination
EORI	# <data>, <ea>	8, 16, 32	Data ⊕ Destination ⇒ Destination
EORI to CCR	# <data>, CCR	8	Source ⊕ CCR ⇒ CCR
EORI to SR[1]	# <data>, SR	16	Source ⊕ SR ⇒ SR
EXG	Rn, Rn	32	Rn ⇒ Rn
EXT	Dn Dn	8 ⇒ 16 16 ⇒ 32	Sign extended Destination ⇒ Destination
EXTB	Dn	8 ⇒ 32	Sign extended Destination ⇒ Destination
ILLEGAL	none	none	SSP – 2 ⇒ SSP; vector offset ⇒ (SSP); SSP – 4 ⇒ SSP; PC ⇒ (SSP); SSP – 2 ⇒ SSP; SR ⇒ (SSP); Illegal instruction vector address ⇒ PC
JMP	<ea>	none	Destination ⇒ PC
JSR	<ea>	none	SP – 4 ⇒ SP; PC ⇒ (SP); destination ⇒ PC
LEA	<ea>, An	32	<ea> ⇒ An
LINK	An, # d	16, 32	SP – 4 ⇒ SP, An ⇒ (SP); SP ⇒ An, SP + d ⇒ SP
LPSTOP[1]	# <data>	16	Data ⇒ SR; interrupt mask ⇒ EBI; STOP
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	
MOVE	<ea>, <ea>	8, 16, 32	Source ⇒ Destination
MOVEA	<ea>, An	16, 32 ⇒ 32	Source ⇒ Destination
MOVEA[1]	USP, An An, USP	32 32	USP ⇒ An An ⇒ USP
MOVE from CCR	CCR, <ea>	16	CCR ⇒ Destination
MOVE to CCR	<ea>, CCR	16	Source ⇒ CCR

Table 3-2 Instruction Set Summary (Continued)

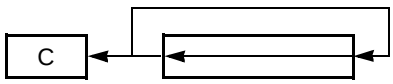
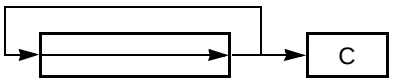
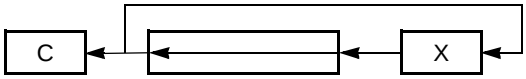
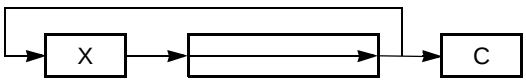
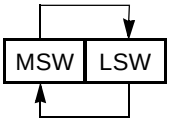
Instruction	Operand Syntax	Operand Size	Description
MOVE from SR[1]	SR, <ea>	16	SR ⇒ Destination
MOVE to SR[1]	<ea>, SR	16	Source ⇒ SR
MOVE USP[1]	USP, An An, USP	32 32	USP ⇒ An An ⇒ USP
MOVEC[1]	Rc, Rn Rn, Rc	32 32	Rc ⇒ Rn Rn ⇒ Rc
MOVEM	list, <ea> <ea>, list	16, 32 16, 32 ⇒ 32	Listed registers ⇒ Destination Source ⇒ Listed registers
MOVEP	Dn, (d16, An) (d16, 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
MOVES[1]	Rn, <ea> <ea>, Rn	8, 16, 32	Rn ⇒ Destination using DFC Source using SFC ⇒ Rn
MULS/MULU	<ea>, Dn <ea>, Dl <ea>, Dh : Dl	16 ∗ 16 ⇒ 32 32 ∗ 32 ⇒ 32 32 ∗ 32 ⇒ 64	Source ∗ Destination ⇒ Destination (signed or unsigned)
NBCD	<ea>	8 8	0 – Destination $_{10}$ – X ⇒ Destination
NEG	<ea>	8, 16, 32	0 – Destination ⇒ Destination
NEGX	<ea>	8, 16, 32	0 – Destination – X ⇒ Destination
NOP	none	none	PC + 2 ⇒ PC
NOT	<ea>	8, 16, 32	$\overline{\text{Destination}}$ ⇒ Destination
OR	<ea>, Dn Dn, <ea>	8, 16, 32 8, 16, 32	Source + Destination ⇒ Destination
ORI	#<data>, <ea>	8, 16, 32	Data + Destination ⇒ Destination
ORI to CCR	#<data>, CCR	16	Source + CCR ⇒ SR
ORI to SR[1]	#<data>, SR	16	Source ; SR ⇒ SR
PEA	<ea>	32	SP – 4 ⇒ SP; <ea> ⇒ SP
RESET[1]	none	none	Assert $\overline{\text{RESET}}$ line
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	
RTD	#d	16	(SP) ⇒ PC; SP + 4 + d ⇒ SP

Table 3-2 Instruction Set Summary (Continued)

Instruction	Operand Syntax	Operand Size	Description
RTE[1]	none	none	(SP) ⇒ SR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP; Restore stack according to format
RTR	none	none	(SP) ⇒ CCR; SP + 2 ⇒ SP; (SP) ⇒ PC; SP + 4 ⇒ SP
RTS	none	none	(SP) ⇒ PC; SP + 4 ⇒ SP
SBCD	Dn, Dn – (An), – (An)	8 8	Destination10 – Source10 – X ⇒ Destination
Scc	<ea>	8	If condition true, then destination bits are set to one; else, destination bits are cleared to zero
STOP[1]	#<data>	16	Data ⇒ SR; STOP
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 – Data ⇒ Destination
SUBQ	#<data>, <ea>	8, 16, 32	Destination – Data ⇒ Destination
SUBX	Dn, Dn – (An), – (An)	8, 16, 32 8, 16, 32	Destination – Source – X ⇒ Destination
SWAP	Dn	16	
TAS	<ea>	8	Destination Tested Condition Codes bit 7 of 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
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
TST	<ea>	8, 16, 32	Source – 0, to set condition codes
UNLK	An	32	An ⇒ SP; (SP) ⇒ An, SP + 4 ⇒ SP

NOTES:
1. Privileged instruction.

3.8.1 M68000 Family Compatibility

It is the philosophy of the M68000 family that all user-mode programs can execute unchanged on future derivatives of the M68000 family. 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. Many of the instruction and addressing mode extensions of the MC68020 are also supported. Refer to the *CPU32 Reference Manual* (CPU32RM/AD) for a detailed comparison of the CPU32 and MC68020 instruction set.

3.8.2 Special Control Instructions

Low-power stop (LPSTOP) and table lookup and interpolate (TBL) instructions have been added to the MC68000 instruction set for use in controller applications.

3.8.2.1 Low-Power Stop (LPSTOP)

In applications where power consumption is a consideration, the CPU32 forces the device into a low-power standby mode when immediate processing is not required. The low-power stop mode is entered by executing the LPSTOP instruction. The processor remains in this mode until an unmasked interrupt or reset occurs.

3.8.2.2 Table Lookup and Interpolate (TBL)

To maximize throughput for real-time applications, reference data is often precalculated and stored in memory for quick access. Storage of many data points can require an inordinate amount of memory. The table lookup instruction requires that only a sample of data points be stored, reducing memory requirements. The TBL instruction recovers intermediate values using linear interpolation. Results can be rounded with a round-to-nearest algorithm.

3.8.2.3 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 3-7** shows the required form of an instruction loop for the processor to enter loop mode.

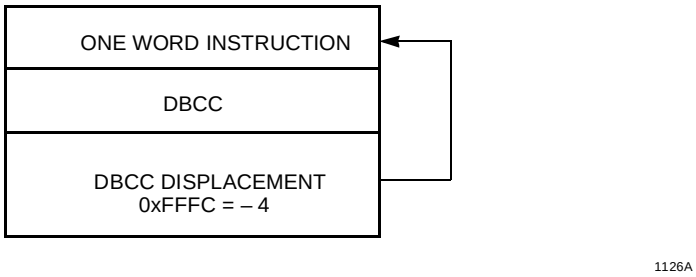


Figure 3-7 Loop Mode Instruction Sequence

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 on interrupts or other exceptions. All single word instructions that do not cause a change of flow can be looped.

3.9 Exception Processing

An exception is a special condition that preempts normal processing. Exception processing is the transition from normal mode program execution to execution of a routine that deals with an exception.

3.9.1 Exception Vectors

An exception vector is the address of a routine that handles an exception. The vector base register (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 each vector in the table is one long word in length. The reset vector is two long words in length. Refer to **Table 3-3** for information on vector assignment.

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.

All exception vectors, except the reset vector and stack pointer, are located in supervisor data space. The reset vector and stack pointer are located in supervisor program space. Only the initial reset vector and stack pointer are 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.

Table 3-3 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, coprocessor protocol violation)
14	56	038	SD	Format error and uninitialized interrupt
15	60	03C	SD	Format error and 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, coprocessor)
59–63	236 252	0EC 0FC	SD	(Unassigned, reserved)
64–255	256 1020	100 3FC	SD	User defined vectors (192)

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 four to calculate vector offset, then adds the offset to the contents of the VBR. The sum is the memory address of the vector.

3.9.2 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.

3.9.3 Exception Processing Sequence

For all exceptions other than a reset exception, exception processing occurs in the following sequence. Refer to **4.7 Reset** for details of reset processing.

As exception processing begins, the processor makes an internal copy of the status register. After the copy is made, the processor state bits in the status register 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 0xF (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 status register and the program counter 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 four, 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 program counter. If no other exception is pending, the processor will resume normal execution at the new address in the PC.

3.10 Development Support

The following features have been implemented on the CPU32 to enhance the instrumentation and development environment:

- M68000 Family Development Support
- Background Debug Mode
- Deterministic Opcode Tracking

• Hardware Breakpoints

3.10.1 M68000 Family Development Support

All M68000 Family members include features to facilitate applications development. These features include the following:

Trace on Instruction Execution — M68000 Family processors include an instruction-by-instruction tracing facility as an aid to program development. The MC68020, MC68030, MC68040, and CPU32 also allow tracing only of those instructions causing a change in program flow. In the trace mode, a trace exception is generated after an instruction is executed, allowing a debugger program to monitor the execution of a program under test.

Breakpoint Instruction — An emulator may insert software breakpoints into the target code to indicate when a breakpoint has occurred. On the MC68010, MC68020, MC68030, and CPU32, this function is provided via illegal instructions, 0x4848–0x484F, to serve as breakpoint instructions.

Unimplemented Instruction Emulation — During instruction execution, 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.

3.10.2 Background Debug Mode

Microcomputer systems generally provide a debugger, implemented in software, for system analysis at the lowest level. The background debug mode (BDM) on the CPU32 is unique in that the debugger has been implemented in CPU microcode.

BDM incorporates a full set of debugging options: registers can be viewed or altered, memory can be read or written to, and test features can be invoked.

A resident debugger simplifies implementation of an in-circuit emulator. In a common setup (refer to **Figure 3-8**), emulator hardware replaces the target system processor. A complex, expensive pod-and-cable interface provides a communication path between the target system and the emulator.

By contrast, an integrated debugger supports use of a bus state analyzer (BSA) for incircuit emulation. The processor remains in the target system (refer to **Figure 3-9**) 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, and many interfacing problems (for example, limitations on high-frequency operation, AC and DC parametric mismatches, and restrictions on cable length) are minimized.

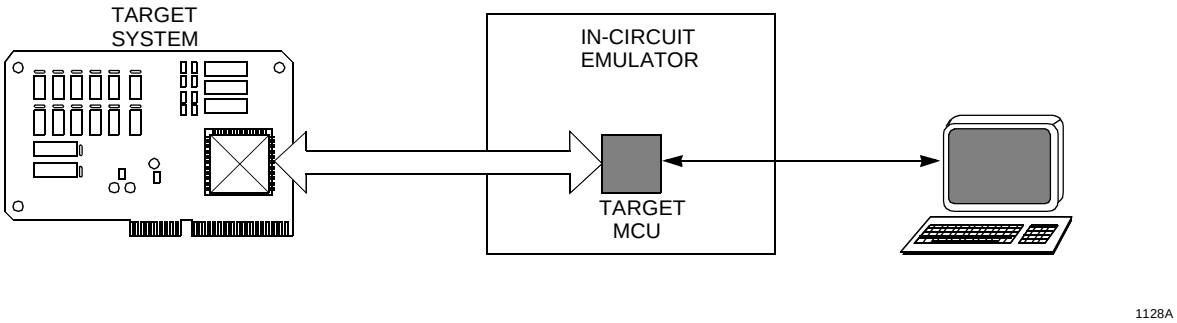


Figure 3-8 Common In-Circuit Emulator Diagram

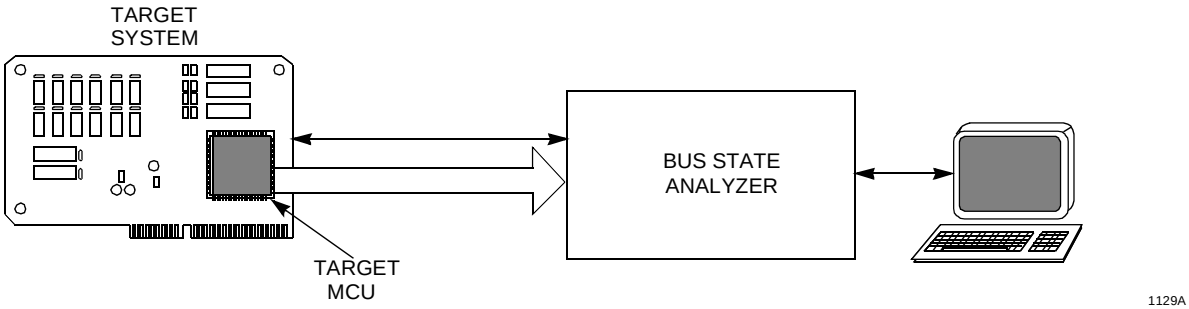


Figure 3-9 Bus State Analyzer Configuration

3.10.3 Enabling BDM

Accidentally entering BDM in a non-development environment can lock up the CPU32 when the serial command interface is not available. For this reason, BDM is enabled during reset via the breakpoint ($\overline{\text{BKPT}}$) 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 $\overline{\text{BKPT}}$ signal on the trailing edge of $\overline{\text{RESET}}$ disables BDM. $\overline{\text{BKPT}}$ is latched again on each rising transition of $\overline{\text{RESET}}$. $\overline{\text{BKPT}}$ is synchronized internally, and must be held low for at least two clock cycles prior to negation of $\overline{\text{RESET}}$.

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, the bus cycle could inadvertently be tagged with a breakpoint. Refer to the *SCIM Reference Manual* (SCIMRM/AD) for timing information.

3.10.4 BDM Sources

Once BDM is enabled, any of several sources can cause the transition from normal mode to BDM. These sources include external breakpoint 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 3-4 summarizes the processing of each source for both enabled and disabled cases. As shown in **Table 3-4**, the $\overline{\text{BKPT}}$ instruction never causes a transition into BDM.

Table 3-4 BDM Source Summary

Source	BDM Enabled	BDM Disabled
$\overline{\text{BKPT}}$	Background	Breakpoint Exception
Double Bus Fault	Background	Halted
BGND Instruction	Background	Illegal Instruction
$\overline{\text{BKPT}}$ Instruction	Opcode Substitution/ Illegal Instruction	Opcode Substitution/ Illegal Instruction

3.10.4.1 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 0x0C) is acknowledged. The $\overline{\text{BKPT}}$ input has the same timing relationship to the data strobe trailing edge as does read cycle data. There is no breakpoint acknowledge bus cycle when BDM is entered.

3.10.4.2 BGND Instruction

An illegal instruction, 0x4AFA, is reserved for use by development tools. The CPU32 defines 0x4AFA (BGND) to be a BDM entry point when BDM is enabled. If BDM is disabled, an illegal instruction trap is acknowledged.

3.10.4.3 Double Bus Fault

The CPU32 normally treats a double bus fault, (an exception that occurs while stacking for another exception) 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 the origin of the fault can be isolated and eliminated.

3.10.4.4 Peripheral Breakpoints

CPU32 peripheral breakpoints are implemented in the same way as external breakpoints — peripherals request breakpoints by asserting the $\overline{\text{BKPT}}$ signal. Consult the appropriate peripheral user's manual for additional details on the generation of peripheral breakpoints.

3.10.5 Entering BDM

When the processor detects a breakpoint or a double bus fault, or decodes a BGND instruction, it suspends instruction execution and asserts the FREEZE output. This 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 (refer to **Table 3-5**) 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 3-5 Polling the BDM Entry Source

Source	ATEMP[31:16]	ATEMP[15:0]
Double Bus Fault	SSW[1]	0xFFFF
BGND Instruction	0x0000	0x0001
Hardware Breakpoint	0x0000	0x0000

NOTES:
1. Special status word (SSW) is described in detail in the ***CPU32 Reference Manual*** (CPU32RM/AD).

A double bus fault during initial stack pointer/program counter (SP/PC) fetch sequence is distinguished by a value of 0xFFFFFFFF in the current instruction PC. At no other time will the processor write an odd value into this register.

3.10.6 BDM Commands

BDM 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. **Table 3-6** is a summary of background mode commands.

Table 3-6 Background Mode Command Summary

Command	Mnemonic	Description
Read D/A Register	RDREG/RAREG	Read the selected address or data register and return the results via the serial interface.
Write D/A Register	WDREG/WAREG	The data operand is written to the specified address or data register.
Read System Register	RSREG	The specified system control register is read. All registers that can be read in supervisor mode can be read in background mode.
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 source function code register (SFC) determines the address space accessed.
Write Memory Location	WRITE	Write the operand data to the memory location specified by the long-word address. The destination function code (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 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 supply the first operand. Subsequent operands are written with the FILL command.
Resume Execution	GO	The pipe is flushed and re-filled before resuming instruction execution at the current PC.
Patch User Code	CALL	Current program counter is stacked at the location of the current stack pointer. Instruction execution begins at user patch code.
Reset Peripherals	RST	Asserts $\overline{\text{RESET}}$ 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.

3.10.7 Background Mode Registers

BDM processing uses three special purpose registers to keep track of program context during development. A description of each follows.

3.10.7.1 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 there was one) is not visible to the user.

3.10.7.2 Return Program Counter (RPC)

The RPC points to the location where fetching will commence after transition from background mode 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.

3.10.7.3 Current Instruction Program Counter (PCC)

The PCC holds a pointer to the first word of the last instruction executed prior to transition into background mode. Due to instruction pipelining, the instruction pointed to may not be the instruction which 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 breakpoint asserted during this cycle will not be acknowledged until the end of the instruction executing at completion of the bus cycle. PCC will contain 0x00000001 if BDM is entered via a double bus fault immediately out of reset.

3.10.8 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 refetch instructions from the location pointed to by the RPC.

The return PC and the memory space referred to by the status register 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 $\overline{\text{IPIPE}}$/$\overline{\text{IFETCH}}$ functionality.

3.10.9 Serial Interface

Communication with the CPU32 during BDM occurs via a dedicated serial interface, which shares pins with other development features. **Figure 3-10** is a block diagram of the interface. The $\overline{\text{BKPT}}$ signal becomes the serial clock (DSCLK); serial input data (DSI) is received on $\overline{\text{IFETCH}}$, and serial output data (DSO) is transmitted on $\overline{\text{IPIPE}}$.

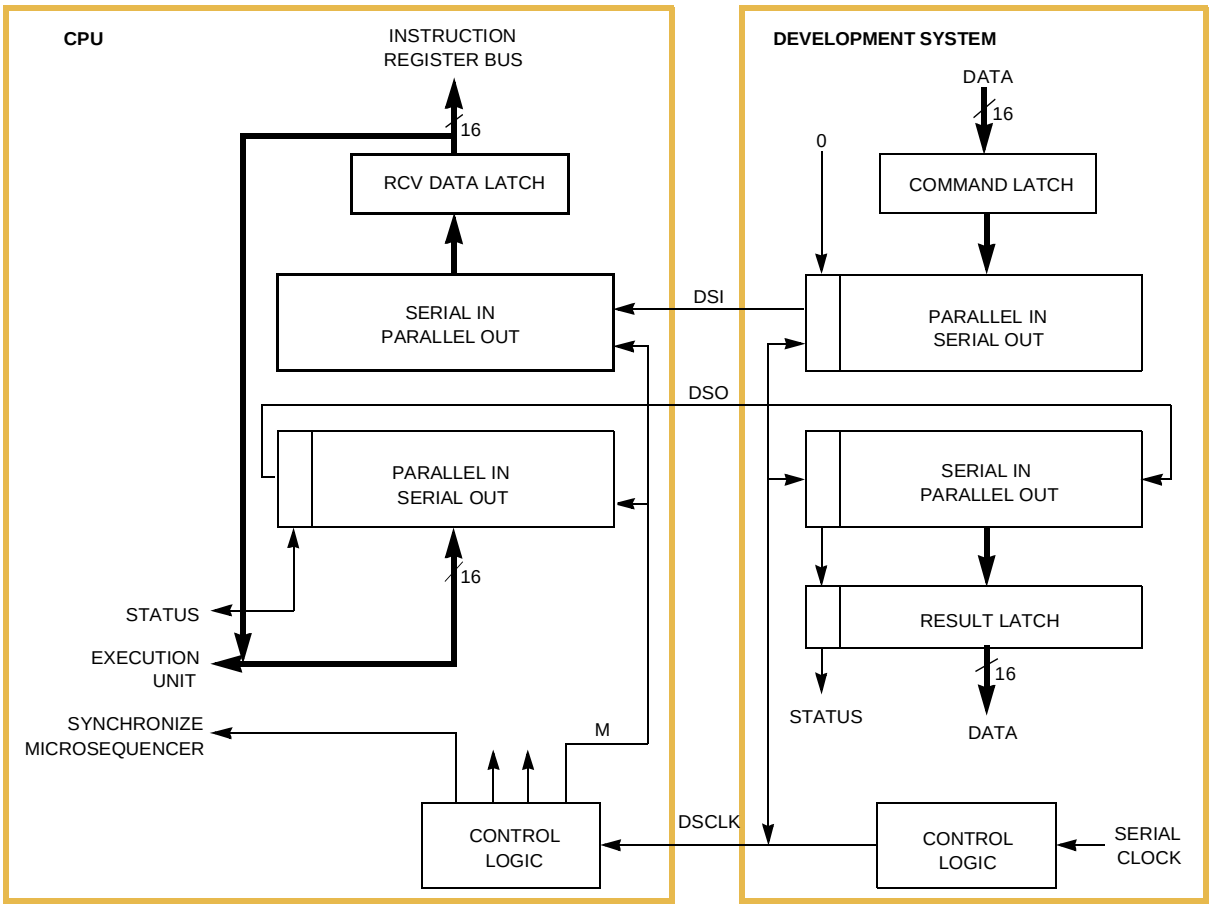


Figure 3-10 Debug Serial I/O Block Diagram

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 — 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.

The serial data word is 17 bits wide, including 16 data bits and a status/control bit (refer to **Figure 3-11**). Bit 16 indicates the status of CPU-generated messages. **Table 3-7** shows the CPU-generated message types.

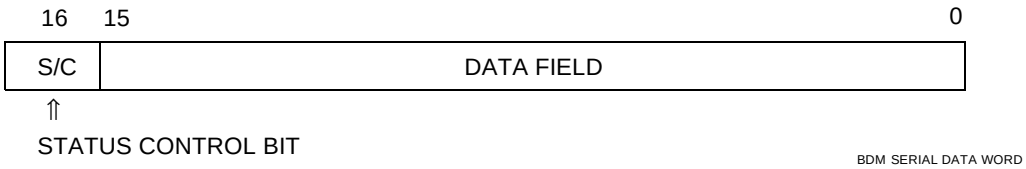


Figure 3-11 BDM Serial Data Word

Table 3-7 CPU Generated Message Encoding

Bit 16	Data	Message Type
0	XXXX	Valid data transfer
0	FFFF	Command complete; Status OK
1	0000	Not ready with response; Come again
1	0001	BERR terminated bus cycle; Data invalid
1	FFFF	Illegal command

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.

3.10.10 Recommended BDM Connection

In order to provide for use of development tools when an MCU is installed in a system, Motorola recommends that appropriate signal lines be routed to a male Berg connector or double-row header installed on the circuit board with the MCU. Refer to **Figure 3-12**.

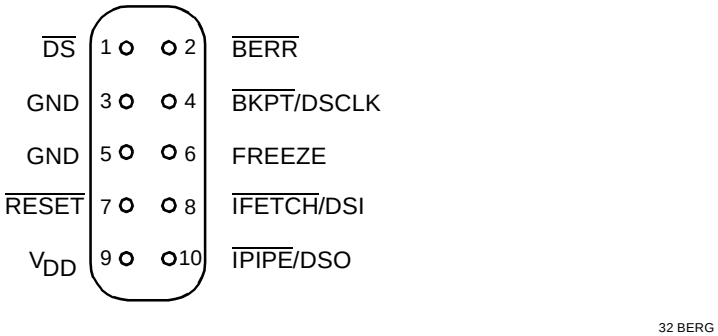


Figure 3-12 BDM Connector Pinout

3.10.11 Deterministic Opcode Tracking

CPU32 function code outputs are augmented by two supplementary signals to monitor the instruction pipeline. The instruction pipe ($\overline{IPIPE}$) output indicates the start of each new instruction and each mid-instruction pipeline advance. The instruction fetch ($\overline{IFETCH}$) output identifies the bus cycles in which the operand is loaded into the

instruction pipeline. Pipeline flushes are also signaled with $\overline{\text{IFETCH}}$. Monitoring these two signals allows a bus state analyzer to synchronize itself to the instruction stream and monitor its activity.

3.10.12 On-Chip Breakpoint Hardware

An external breakpoint input and on-chip breakpoint hardware allow a breakpoint trap on any memory access. Off-chip address comparators preclude breakpoints unless show cycles are enabled. Breakpoints on instruction prefetches that are ultimately flushed from the instruction pipeline are not acknowledged; operand breakpoints are always acknowledged. Acknowledged breakpoints initiate exception processing at the address in exception vector number 12, or alternately enter background mode.

SECTION 4
SINGLE-CHIP INTEGRATION MODULE 2 (SCIM2E)

4.1 Overview

MC68F375 contains the single chip integration module 2 (SCIM2E). The SCIM2E consists of several submodules:

- The system configuration block controls MCU configuration and operating mode.
- The system clock generates clock signals used by the SCIM2E, other IMB modules, and external devices. Circuitry is included to detect loss of the phase-locked loop (PLL) reference frequency and to control clock operation in low power stop mode.
- The system protection block provides bus and software watchdog monitors. It also incorporates a periodic interrupt generator that supports execution of time-critical control routines.
- The external bus interface (EBI) handles the transfer of information between the CPU32 and external address space. Ports A, B, E, F, G, and H comprise the EBI and may be used for discrete I/O subject to the MCU's operating mode.
- The chip-select block provides nine general-purpose chip-select signals and two emulation support chip-select signals. Each general-purpose chip select has associated base address and option registers that control the programmable characteristics of the chip select. Chip-select pins can also be used as general-purpose output port C.

The MC68F375 SCIM2E includes improvements to the regular SCIM. This enhanced SCIM includes improvements to the clock synthesizer. These changes are defined in detail in **4.3.2 Clock Synthesizer Submodule**. Please refer to the ***SCIM Reference Manual* (SCIMRM/AD)** for more details on the characteristics of this module.

The SCIM2E has three basic operating modes:

- 16-bit expanded mode
- 8-bit expanded mode
- Single-chip mode

Operating mode is determined by the logic states of specific MCU pins during reset. Refer to **4.7.8 Operating Configuration Out of Reset** for more detailed information on MCU operating modes.

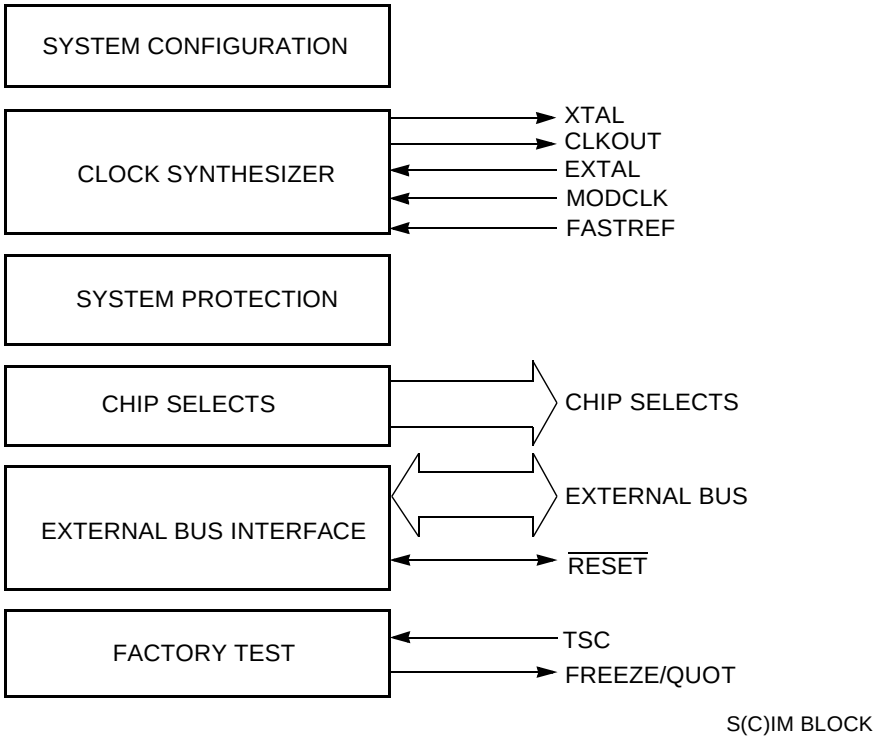


Figure 4-1 SCIM2E Block Diagram

4.2 System Configuration

The MCU can operate as a stand-alone device in single-chip mode, or it can operate with the support of external memory and/or peripheral devices in the 16-bit or 8-bit expanded modes. System configuration is determined by asserting MCU pins during reset and by setting bits in the SCIM2E configuration register (SCIMMCR).

4.2.1 SCIM Module Configuration Register

SCIMMCR — SCIM Module Configuration Register **0xYF FA00**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
EXOFF	FRZSW	FRZBM	CPUD[1]	0[2]	SLOWE	SHEN		SUPV	MM	ABD[1]	RWD[1]	IARB			
RESET:															
0	1	1	*	0	0	0	0	1	1	*	*	1	1	1	1

NOTES:
1. Reset state is mode-dependent. Refer to the following bit descriptions.
2. Ensure that initialization software does not change this value (it should always read zero).

Table 4-1 SCIMMCR Bit Descriptions

Bit(s)	Name	Description
15	EXOFF	External clock off. 0 = The CLKOUT pin is driven during normal operation. 1 = The CLKOUT pin is placed in a high-impedance state.
14	FRZSW	Freeze software enable. Enables or disables the software watchdog and periodic interrupt timer during background debug mode when FREEZE is asserted. 0 = Enables the software watchdog and periodic interrupt timer when FREEZE is asserted. 1 = Disables the software watchdog and periodic interrupt timer when FREEZE is asserted.
13	FRZBM	Freeze bus monitor enable. 0 = When FREEZE is asserted, the bus monitor continues to operate. 1 = When FREEZE is asserted, the bus monitor is disabled.
12	CPUD	CPU development support disable. CPUD is cleared to zero when the MCU is in an expanded mode, and set to one in single-chip mode. 0 = Instruction pipeline signals available on pins $\overline{\text{IPIPE}}$ and $\overline{\text{IFETCH}}$. 1 = Pins $\overline{\text{IPIPE}}$ and $\overline{\text{IFETCH}}$ placed in high-impedance state unless a breakpoint occurs.
11	—	Reserved
10	SLOWE	Slow mode enable. Control bit which forces pins on the chip to operate in fast mode regardless of how they are set up from the controlling module. Slow mode is enabled by setting this bit. 0 = Pins setup by the controlling module to operate in slow mode will operate in fast mode. 1 = Pins will operate at the normal speed controlled by the module.
9:8	SHEN	Show cycle enable. The SHEN field determines how the external bus is driven during internal transfer operations. A show cycle allows internal transfers to be monitored externally. **Table 4-3** indicates whether show cycle data is driven externally, and whether external bus arbitration can occur. To prevent bus conflict, external devices must not be selected during show cycles.
7	SUPV	Supervisor/user data space. The SUPV bit places the SCIM2E global registers in either supervisor or user data space. 0 = Registers access controlled by the SUPV bit accessible in either supervisor or user mode. 1 = Registers access controlled by the SUPV bit restricted to supervisor access only.
6	MM	Module mapping 0 = Internal modules are addressed from 0x7FF000 – 0x7FFFFF. 1 = Internal modules are addressed from 0xFFF000 – 0xFFFFFF.
5	ABD	Address Bus Disable. ABD is cleared to zero when the MCU is in an expanded mode, and set to one in single-chip mode. ABD can be written only once after reset. 0 = Pins ADDR[2:0] operate normally. 1 = Pins ADDR[2:0] are disabled.
4	RWD	Read/write disable. RWD is cleared to zero when the MCU is in an expanded mode, and set to one in single-chip mode. RWD can be written only once after reset. 0 = R/$\overline{\text{W}}$ signal operates normally 1 = R/$\overline{\text{W}}$ signal placed in high-impedance state.
3:0	IARB	Each module that can generate interrupts, including the SCIM2E, has an IARB field. Each IARB field can be assigned a value from 0x0 to 0xF. During an interrupt acknowledge cycle, IARB permits arbitration among simultaneous interrupts of the same priority level. The reset value of the SCIM2 IARB field is 0xF, the highest priority. This prevents SCIM2 interrupts from being discarded during system initialization.

The SCIMMCR register controls the system configuration. SCIMMCR can be read or written at any time, except for the module mapping (MM) bit, which can only be written once after reset, and the reserved bit, which is read-only. Write has no effect.

4.2.2 Module Mapping

Control registers for all the modules in the microcontroller are mapped into a 4-Kbyte block. The state of the module mapping (MM) bit in SCIMMCR determines where the control register block is located in the system memory map. When MM = 0, register addresses range from 0x7FF000 to 0x7FFFFF; when MM = 1, register addresses range from 0xFFF000 to 0xFFFFFF.

4.2.3 Interrupt Arbitration

Each module that can request interrupts has an interrupt arbitration (IARB) field. Arbitration between interrupt requests of the same priority is performed by serial contention between IARB field bit values. Contention will take place whenever an interrupt request is acknowledged, even when there is only a single request pending. For an interrupt to be serviced, the appropriate IARB field must have a non-zero value. If an interrupt request from a module with an IARB field value of 0b0000 is recognized, the CPU32 will start to process the interrupt. The CPU will attempt to run and IACK cycle. Because the IARB values of the interrupting module is 0b0000, the module cannot cause the termination of the IACK cycle. In this case, the IACK cycle can only be terminated by an external DSACK, a software watchdog timeout or a bus error. If the IACK cycle is terminated by BERR, a spurious interrupt exception is taken.

Because the SCIM2E routes external interrupt requests to the CPU32, the SCIM2E IARB field value is used for arbitration between interrupts of the same priority. The reset value of IARB for the SCIM2E is 0b1111, and the reset IARB value for all other modules is 0b0000. This prevents SCIM2E interrupts from being discarded during initialization. Refer to **4.8 Interrupts** for a discussion of interrupt arbitration.

4.2.4 Noise Reduction in Single-Chip Mode

Four bits in SCIMMCR control pins that can be disabled in single-chip mode to reduce MCU noise emissions. The characteristics of these control bits are listed in **Table 4-2**. Except for EXOFF, these bits disable their associated pins when the MCU is configured for single-chip mode ($\overline{BERR}$ = 0 during reset).

Table 4-2 SCIMMCR Noise Control Bits

Bit Mnemonic	Position in SCIMMCR	Function	Reset State
EXOFF	15	Disables CLKOUT when set to one.	0
CPUD	12	Disables $\overline{IPIPE}$/DSO and $\overline{IFETCH}$/DSI pins when set to one.	Inverted state of the $\overline{BERR}$ pin
ABD	5	Disables ADDR[2:0] when set to one.	
RWD	4	Disables R/$\overline{W}$ when set to one.	

EXOFF disables the CLKOUT external clock output pin by placing the pin in a high-impedance state. CLKOUT is enabled at power-up unless explicitly disabled by writing a zero to EXOFF.

CPUD disables the $\overline{\text{IPIPE}}$/DSO and $\overline{\text{IFETCH}}$/DSI instruction tracking pins by placing them in a high-impedance state when the MCU is not in background debug mode (BDM). When the MCU enters BDM and FREEZE is asserted, $\overline{\text{IPIPE}}$/DSO and $\overline{\text{IFETCH}}$/DSI become active and serve as the BDM serial I/O lines.

ABD and RWD disable the ADDR[2:0] and R/$\overline{\text{W}}$ pins by placing them in a high-impedance state. These pins should be disabled because they cannot be used for discrete I/O and have no use in single-chip mode.

4.2.5 Show Internal Cycles

A show cycle allows internal bus transfers to be monitored externally. The SHEN field in SCIMMCR determines what the external data bus interface does during internal transfer operations. **Table 4-3** shows whether data is driven externally, and whether external bus arbitration can occur. The external address bus is always driven. Refer to **4.6.6.1 Show Cycles** for more information.

Table 4-3 Show Cycle Enable Bits

SHEN[1:0]	Effect
00	Show cycles disabled, external arbitration enabled
01	Show cycles enabled, external arbitration disabled
10	Show cycles enabled, external arbitration enabled
11	Show cycles enabled, external arbitration enabled; internal activity is halted by a bus grant

4.2.6 FREEZE Assertion Response

When the CPU32 enters background debug mode, the IMB FREEZE signal is asserted. The FRZ[1:0] bits in SCIMMCR control the behavior of the software watchdog, periodic interrupt timer, and bus monitor in response to FREEZE assertion. By default, these protection mechanisms are disabled in BDM; they can be selectively enabled by the FRZ[1:0] bits as shown in **Table 4-4**.

Table 4-4 Effects of FREEZE Assertion

FRZ[1:0]		Disabled Elements
0	0	None
0	1	Bus monitor
1	0	Software watchdog and periodic interrupt timer
1	1	Both

4.3 System Clock

The system clock in the SCIM2E provides timing signals for IMB modules and the external bus interface. MC68F375 MCUs are fully static MCU designs; register and memory contents are not affected by clock rate changes. System hardware and software support clock rate changes during operation.

CAUTION

The SCIM2E system clock is different from the SCIM. Do not refer to SCIM documentation for SCIM2E clock information.

4.3.1 System Clock Sources

The system clock signal can be generated from one of three sources. An internal phase-locked loop (PLL) can synthesize the clock from either a slow or fast reference. The clock signal can also be input directly from an external 2X frequency source.

The slow reference is typically a 32.768 KHz crystal; the fast reference is typically a 4 to 8 MHz crystal. The slow and fast references may be provided by sources other than a crystal. Keep these clock sources in mind while reading the rest of this section.

The system clock source is determined upon $\overline{\text{RESET}}$ assertion by the state of the V_{DDSYN}/MODCLK and FASTREF/PF0 pins. In addition to selecting the system clock source, these pins govern the functionality of the W, X, and Y bits in the synthesizer control register (SYNCR) and the equation that determines the MCU operating frequency. **Table 4-5** summarizes system clock source information. For more information, see **4.3.6 Clock Synthesizer Control Register**.

Table 4-5 System Clock Sources

V_{DDSYN}/ MODCLK	FASTREF/ PF0	Clock Mode	SYNCR W/X/Y Bit Assignments and Reset Values	MCU Operating Frequency Equation
0	X[1]	External Clock	W : has no effect X=0b1 : Divider Y[2:0] = 0b000 : Divider	$f_{sys} = \frac{\frac{f_{ref}}{2}}{(2-X)(2^{Y})}$, for $Y \le 7$
1	0	Slow Reference	W = 0b0 : Multiplier X = 0b0 : Divider Y[5:0] = 0b11111 : Multiplier	$f_{sys} = 4f_{ref}(Y+1)(2^{(2W+X)})$
1	1	Fast Reference	W[2:0] = 0b011 : Multiplier X = 0b0 : Divider Y[2:0] = 0b111 : Divider	$f_{sys} = \frac{f_{ref}(W+1)}{(2-X)(2^{Y})}$, for $Y \le 7$

NOTES:
1. In external clock mode, the FASTREF/PF0 pin has no effect on clock operation.

The parameter "f_{ref}" refers to the frequency of the clock source connected to the EXTAL pin. The parameter "f_{SYS}" refers to the operating frequency of the MCU and has a defined relationship to f_{ref} that depends on the clock mode selected during reset.

4.3.2 Clock Synthesizer Submodule

The MC68F375 contains an improved version of the clock synthesizer subsystem. The new architecture accommodates both slow or fast crystal references, see **4.3.1 System Clock Sources**. Range of operation and power consumption were taken into

consideration when this new architecture was defined. Compatibility with the previous architecture has been retained when possible.

In general, the improvements fall into four basic categories as follows:

- Configurable PLL for optimization of divider chain based on mode of operation;
- Improved loss-of-clock circuitry based on an independent RC oscillator;
- Improved lock detect circuitry;
- Improved noise immunity by the addition of a V_{SSSYN} pin.

There are three modes for generating the system clock for the MCU. The system clock may be driven directly into the EXTAL pin (external clock mode), it may be generated on-chip by a phase locked loop (PLL) frequency synthesizer using either a slow or fast reference mode. For modes using the PLL, a lock detect circuit detects that the PLL is on frequency and sets a register flag. The PLL mode is determined at reset and behaves according to **Table 4-5**. The source of the system clock is determined at reset by the state of the FASTREF/PF0 and V_{DDSYN}/MODCK pins. To enable external clock mode, the V_{DDSYN}/MODCK pin must be tied directly to V_{SS} at all times

The MC68F375's clock architecture supports the three modes of operation by optionally reconfiguring the number and location of the W bit and Y bit divider stages. In slow reference mode, one W bit and six Y bits are located in the PLL feedback path, enabling frequency multiplication by a factor of up to 2048. The X bit is located in the VCO clock output path to enable dividing the system clock frequency by two without disturbing the VCO and thus requiring re-lock. In fast reference mode, three W bits are located in the PLL feedback path, enabling frequency multiplication by a factor from 1 to 8. Three Y bits and the X bit are located in the VCO clock output path to provide the ability to slow the system clock without disturbing the PLL. In external clock mode, three Y bits and the X bit are located between the EXTAL input and the system clock, to allow slowing the clock for reduced power consumption. Refer to **Figure 4-4**, **Figure 4-3**, and **Figure 4-2** for block diagrams of the architecture in these modes. The reset value of the W, X and Y bits are determined by the clock mode as shown in **Table 4-6**.

NOTE

The crystal oscillator and frequency synthesizer circuits are powered from a separate power pin pair (V_{DDSYN} and V_{SSSYN}) to allow the oscillator to continue to run when the rest of the chip is powered down. This allows avoidance of crystal start-up time. Separate supplies also help improve noise immunity.

The filter for both PLL modes consists of resistor R1 connected in series with capacitor C1. This combination is connected between V_{DDSYN} and XFC. A second capacitor, C2, is also connected between V_{DDSYN} and XFC.

The following sections describe the clock sub-module in detail. One rule applies to all modes of operation — the actual VCO core frequency (before being divided down by X and/or Y bits) must stay at or below the maximum allowable system clock frequency. When changing frequencies, ensure that the values written to the W, X, and Y bits do not select VCO core frequencies above that value. If a system clock frequency is to be

changed with multiple writes to SYNCR, the write sequences should select lower VCO core frequencies first, and the higher VCO core frequencies last unless it is certain the write sequence will not result in VCO core frequencies above the maximum allowable system clock frequency.

4.3.3 Slow Reference Mode

In slow reference mode, the system clock is generated by the PLL typically from a 32.768-KHz reference. The frequency of the system clock is controlled by programming the X, Y, and W bits according to **Table 4-6**.

The W bit is in the feedback path of the VCO. When clear, this bit multiplies the reference frequency by two, and when set, by eight. This bit is clear at reset.

The Y bit divider is a 6-bit modulo counter which can multiply the reference input frequency by up to 256, providing a large number of programmable system clock frequencies. The Y bits are all set to one at reset, providing the highest frequency system clock for a given combination of X and W bits.

The X bit is in the output path of the VCO. It may be used to divide the system clock by two when clear (and pass it without dividing when set). At reset, this bit is cleared to 0.

In slow reference mode, the W and Y bits are both in the feedback path of the PLL. Changing the value of these bits requires the PLL to relock (with some delay) at the new frequency. Changing the X bit, however, will change the system clock frequency without having to wait for the PLL to relock.

Note that for slow reference mode, the crystal is not constrained to be 32.768 KHz but must be in the range of 25 KHz to 50 KHz to ensure that the on-chip crystal oscillator will work.

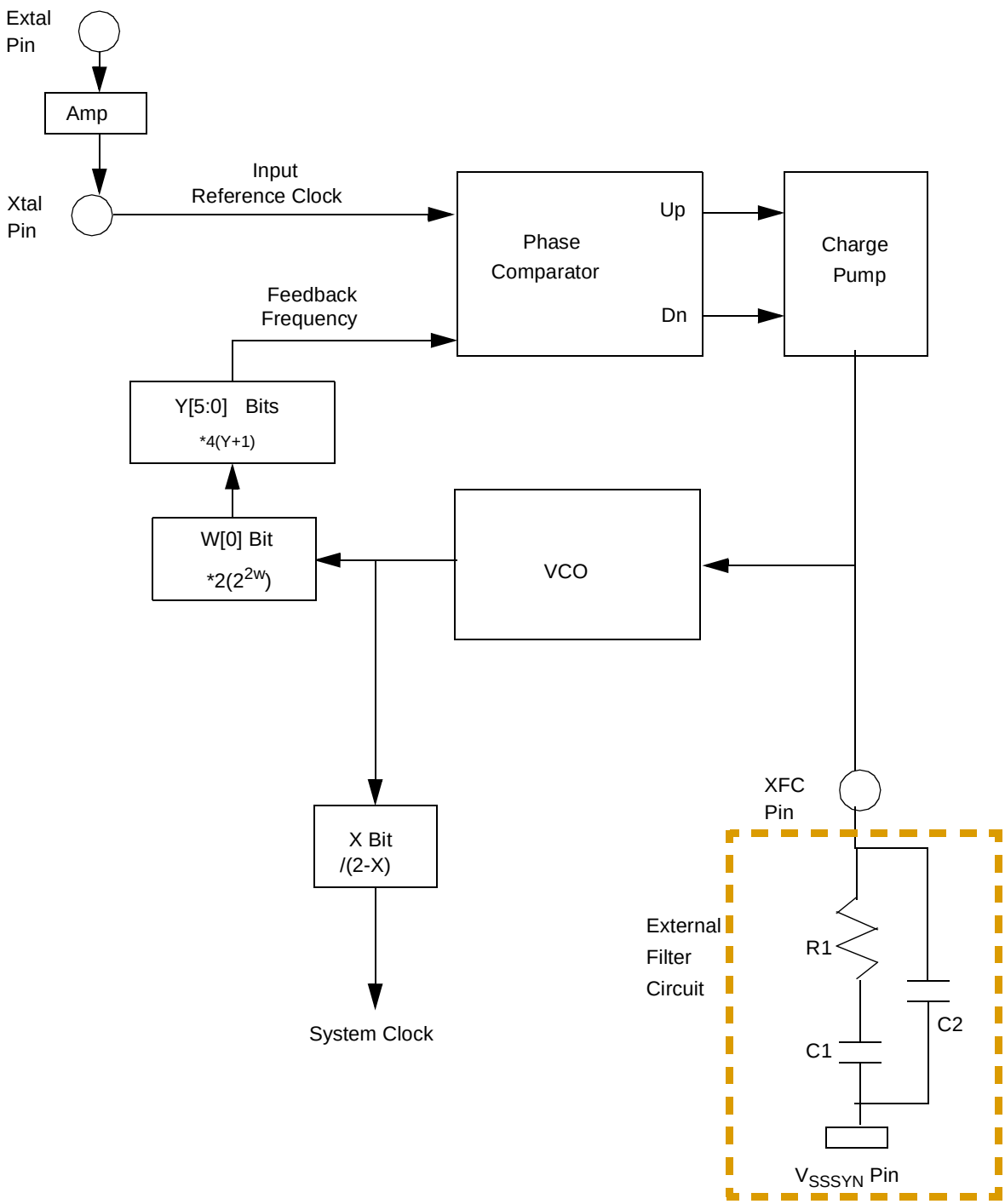


Figure 4-2 Slow Reference Mode

Figure 4-2 depicts the architecture of the system clock generation circuitry when in this mode. **Table 4-6** gives the range of system clock frequencies which may be generated using a 32.768-KHz crystal in this mode. The frequency representing the reset configuration is shaded. Configurations of the PLL in which the system clock or the

VCO core frequency can exceed maximum frequency specification are not supported and are left blank, as a reminder, in the table for frequencies above 25 MHz. Select operating frequencies from the table which do not violate the maximum system clock frequency specification.

Table 4-6 CLKOUT Frequency: Slow Reference; 32.768 KHz Reference

Y (bits)	X=0, W=0[1]	X=1, W=0[2]	X=0, W=1[1]	X=1, W=1[2]	Y (bits)	X=0, W=0[1]	X=1, W=0[2]	X=0, W=1[1]	X=1, W=1[2]
0=000000	131,072	262,144	524,288	1,048,576	32=100000	4,325,376	8,650,752	17,301,504	34,603,008
1=000001	262,144	524,288	1,048,576	2,097,152	33=100001	4,456,448	8,912,896	17,825,792	35,651,584
2=000010	393,216	786,432	1,572,864	3,145,728	34=100010	4,587,520	9,175,040	18,350,080	36,700,160
3=000011	524,288	1,048,576	2,097,152	4,194,304	35=100011	4,718,592	9,437,184	18,874,368	37,748,736
4=000100	655,360	1,310,720	2,621,440	5,242,880	36=100100	4,849,664	9,699,328	19,398,656	38,797,312
5=000101	786,432	1,572,864	3,145,728	6,291,456	37=100101	4,980,736	9,961,472	19,922,944	39,845,888
6=000110	917,504	1,835,008	3,670,016	7,340,032	38=100110	5,111,808	10,223,616	20,447,232	40,894,464
7=000111	1,048,576	2,097,152	4,194,304	8,388,608	39=100111	5,242,880	10,485,760	20,971,520	41,943,040
8=001000	1,179,648	2,359,296	4,718,592	9,437,184	40=101000	5,373,952	10,747,904	21,495,808	42,991,616
9=001001	1,310,720	2,621,440	5,242,880	10,485,760	41=101001	5,505,024	11,010,048	22,020,096	44,040,192
10=001010	1,441,792	2,883,584	5,767,168	11,534,336	42=101010	5,636,096	11,272,192	22,544,384	45,088,768
11=001011	1,572,864	3,145,728	6,291,456	12,582,912	43=101011	5,767,168	11,534,336	23,068,672	46,137,344
12=001100	1,703,936	3,407,872	6,815,744	13,631,488	44=101100	5,898,240	11,796,480	23,592,960	47,185,920
13=001101	1,835,008	3,670,016	7,340,032	14,680,064	45=101101	6,029,312	12,058,624	24,117,248	48,234,496
14=001110	1,966,080	3,932,160	7,864,320	15,728,640	46=101110	6,160,384	12,320,768	24,641,536	49,283,072
15=001111	2,097,152	4,194,304	8,388,608	16,777,216	47=101111	6,291,456	12,582,912	25,165,824	50,331,648
16=010000	2,228,224	4,456,448	8,912,896	17,825,792	48=110000	6,422,528	12,845,056	25,690,112	51,380,224
17=010001	2,359,296	4,718,592	9,437,184	18,874,368	49=110001	6,553,600	13,107,200	26,214,400	52,428,800
18=010010	2,490,368	4,980,736	9,961,472	19,922,944	50=110010	6,684,672	13,369,344	26,738,688	53,477,376
19=010011	2,621,440	5,242,880	10,485,760	20,971,520	51=110011	6,815,744	13,631,488	27,262,976	54,525,952
20=010100	2,752,512	5,505,024	11,010,048	22,020,096	52=110100	6,946,816	13,893,632	27,787,264	55,574,528
21=010101	2,883,584	5,767,168	11,534,336	23,068,672	53=110101	7,077,888	14,155,776	28,311,552	56,623,104
22=010110	3,014,656	6,029,312	12,058,624	24,117,248	54=110110	7,208,960	14,417,920	28,835,840	57,671,680
23=010111	3,145,728	6,291,456	12,582,912	25,165,824	55=110111	7,340,032	14,680,064	29,360,128	58,720,256
24=011000	3,276,800	6,553,600	13,107,200	26,214,400	56=111000	7,471,104	14,942,208	29,884,416	59,768,832
25=011001	3,407,872	6,815,744	13,631,488	27,262,976	57=111001	7,602,176	15,204,352	30,408,704	60,817,408
26=011010	3,538,944	7,077,888	14,155,776	28,311,552	58=111010	7,733,248	15,466,496	30,932,992	61,865,984
27=011011	3,670,016	7,340,032	14,680,064	29,360,128	59=111011	7,864,320	15,728,640	31,457,280	62,914,560
28=011100	3,801,088	7,602,176	15,204,352	30,408,704	60=111100	7,995,392	15,990,784	31,981,568	63,963,136
29=011101	3,932,160	7,864,320	15,728,640	31,457,280	61=111101	8,126,464	16,252,928	32,505,856	65,011,712
30=011110	4,063,232	8,126,464	16,252,928	32,505,856	62=111110	8,257,536	16,515,072	33,030,144	66,060,288
31=011111	4,194,304	8,388,608	16,777,216	33,554,432	63=111111	8,388,608	16,777,216	33,554,432	67,108,864

NOTES:

1. VCO core frequency = 4 times the value in the table.
2. VCO core frequency = 2 times the value in the table. VCO core must not exceed 2x f_{SYS}.

4.3.4 Fast Reference Mode

In fast reference mode, the system clock is generated by the PLL from a reference frequency much higher than that used in slow reference mode (e.g., four MHz). At reset, the system clock frequency will be equal to twice the reference frequency. This is accomplished by configuring the W bits to multiply by four, and setting the X bit to divide by two for a net result of multiplying by two. The frequency may be multiplied using the W bits or divided using the X and Y bits to generate the desired system clock frequency. This mode improves stability over the slow reference mode and, like slow reference mode, provides a 50% duty cycle system clock regardless of the reference duty cycle.

The W bits are in the feedback path of the VCO. They can be programmed to multiply the reference frequency by a factor from one to eight. These bits come out of reset as 0b11, so that the system clock frequency is four times the reference clock frequency. After reset, the W bits can be changed to multiply the reference to the desired system clock frequency. Changing of the W bits will result in PLL unlocking for the PLL lock time specified.

The X bit on the VCO output is used to divide the VCO frequency by two or one. This bit comes out of reset clear (divide by two) so that the system clock frequency is actually one half of what it is multiplied to by the W bits. This bit may be set to increase the system frequency to twice that of the frequency when this bit is clear.

The Y bit divider is a six-stage divider chain in the clock output path, whose output tap is controlled by the three Y register bits. It can divide the output of the VCO down by powers of two to as much as 64, providing a large number of programmable frequencies in this mode.

NOTE

Setting Y to 7 in this mode has the same effect as setting it to 6; the maximum divisor is 2^6. Dividing the PLL output clock with the Y bits reduces the system clock frequency thereby conserving power. Since the X and Y bits are not in the PLL feedback path, the PLL will not have to relock to the new target frequency when they are changed. The Y bits are cleared to all zeros at reset. **Figure 4-3** is a block diagram of the fast reference mode architecture.

Table 4-7 gives example values of the system clock frequency in fast reference mode using a 4.0-MHz reference. The frequency representing the reset configuration is shaded in this table. This frequency ends up being twice the reference frequency when the X bit is cleared, or 8.0 MHz with a 4.0-MHz reference clock. The X bit can then be set to select a system frequency of four times the reference frequency, or 16.0 MHz with a 4.0-MHz reference clock with no PLL re-lock time requirements. Combinations of programmed values for the W, X, and Y bits which would exceed maximum system or VCO core frequencies are not supported and are left blank, as a reminder, in the table for frequencies above 33 MHz.

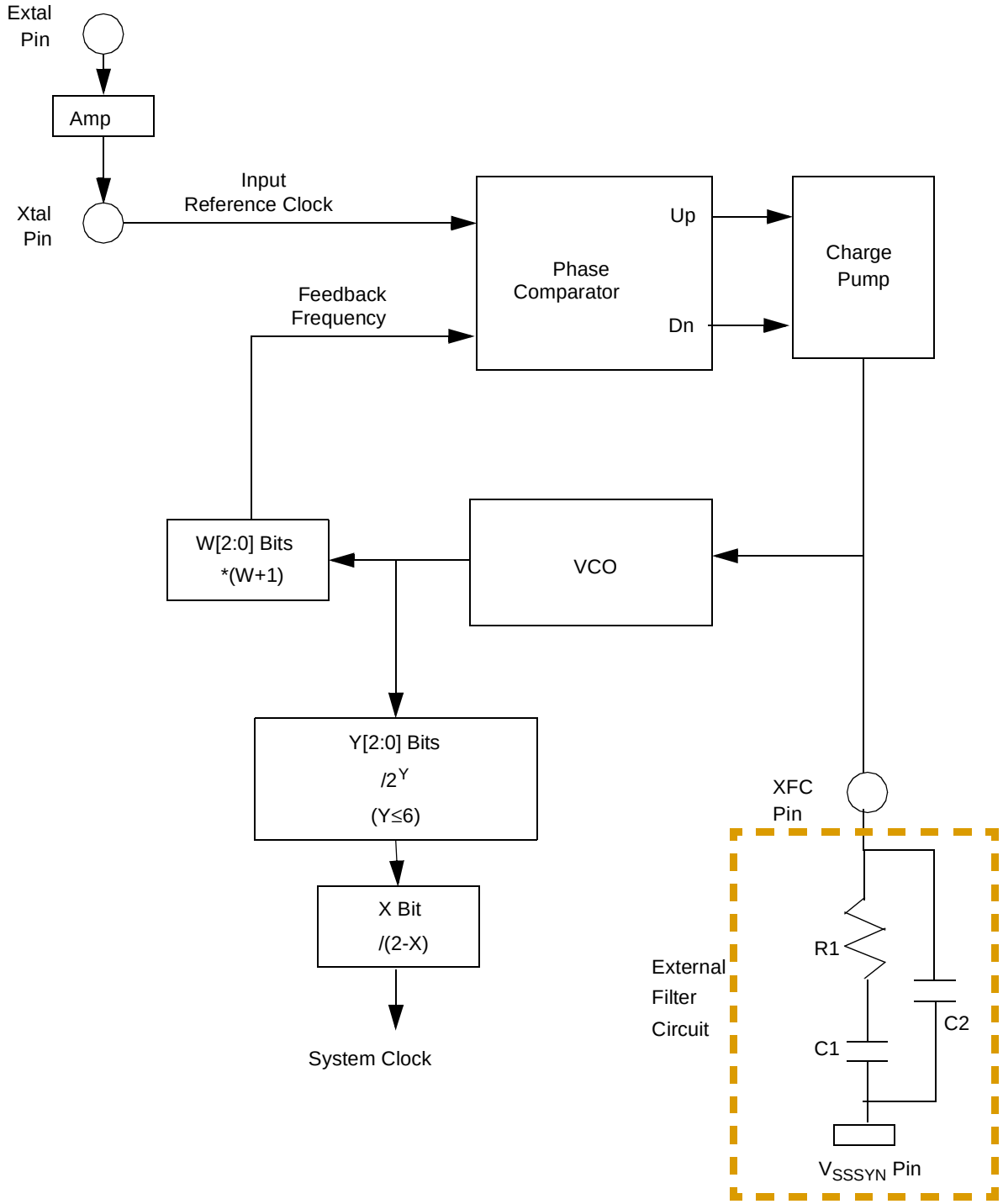


Figure 4-3 Fast Reference Mode

Table 4-7 CLKOUT In Fast Reference Mode with 4.0 MHz Reference

Y X=0	W=000	W=001	W=010	W=011	W=100	W=101	W=110	W=111
0=000	2,000,000	4,000,000	6,000,000	8,000,000	10,000,000	12,000,000	14,000,000	16,000,000
1=001	1,000,000	2,000,000	3,000,000	4,000,000	5,000,000	6,000,000	7,000,000	8,000,000
2=010	500,000	1,000,000	1,500,000	2,000,000	2,500,000	3,000,000	3,500,000	4,000,000
3=011	250,000	500,000	750,000	1,000,000	1,250,000	1,500,000	1,750,000	2,000,000
4=100	125,000	250,000	375,000	500,000	625,000	750,000	875,000	1,000,000
5=101	62,500	125,000	187,500	250,000	312,500	375,000	437,500	500,000
6=110	31,250	62,500	93,750	125,000	156,250	187,500	218,750	250,000
7=111	31,250	62,500	93,750	125,000	156,250	187,500	218,750	250,000

Y X=1	W=000	W=001	W=010	W=011	W=100	W=101	W=110	W=111
0=000	4,000,000	8,000,000	12,000,000	16,000,000	20,000,000	24,000,000	28,000,000	32,000,000
1=001	2,000,000	4,000,000	6,000,000	8,000,000	10,000,000	12,000,000	14,000,000	16,000,000
2=010	1,000,000	2,000,000	3,000,000	4,000,000	5,000,000	6,000,000	7,000,000	8,000,000
3=011	500,000	1,000,000	1,500,000	2,000,000	2,500,000	3,000,000	3,500,000	4,000,000
4=100	250,000	500,000	750,000	1,000,000	1,250,000	1,500,000	1,750,000	2,000,000
5=101	125,000	250,000	375,000	500,000	625,000	750,000	375,000	1,000,000
6=110	62,500	125,000	187,500	250,000	312,500	375,000	187,500	500,000
7=111	62,500	125,000	187,500	250,000	312,500	375,000	187,500	500,000

4.3.5 External Clock Mode

In external clock mode, the clock source, which should be 2x the desired system frequency, must be driven onto the EXTAL pin. This clock is used to generate the system clock directly (the VCO is turned off). At reset, the system clock frequency is one-half the external clock frequency. If this frequency is the the maximum specified system clock frequency, it must not violate strict minimum duty cycle requirements. A block diagram of external clock mode is show in **Figure 4-4**.

In this mode, the six-stage Y divider and the one-stage X divider are placed in the clock output path such that the input clock may be divided down by as much as 128 to produce the system clock. When this is done, it is not necessary to meet the input duty cycle restrictions. The Y bit divider is a six-stage divider chain whose output tap is controlled by the three Y register bits. The X bit divider is a single-stage divider which is bypassed when X is set to 1. X is 1 and Y is 0 after reset, so that the system clock is the same as the external clock.

NOTE

Setting Y to 7 has the same effect as setting it to 6; the maximum divisor is 2^6. The X and Y bit dividers are in the output clock path. Therefore, changing the X or Y bits in this mode causes the frequency to change without a delay.

When the MC68F375 is configured in external clock mode, the VCO will be turned off to save power. Changing the unused W bits will have no effect.

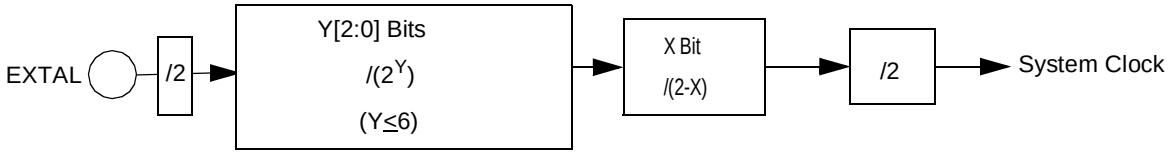


Figure 4-4 External Clock Mode

4.3.6 Clock Synthesizer Control Register

The synthesizer control register (SYNCR) is readable and writable in supervisor mode. The encoding and default value of the upper byte of SYNCR depend on the clock mode selected at reset.

For slow reference mode, the default value is 0x3F which corresponds to an operating frequency of 8.38 MHz for a 32.768-KHz crystal. For fast reference mode, the default value of the upper byte is 0x30, which corresponds to a 2-to-1 match of the reference frequency. For external clock mode, the default value of the upper byte is 0x80, which corresponds to an operating frequency equal to the input frequency on EXTAL. The default value is forced into the SYNCR on reset, along with the default values of the other bits in the register (all zeros).

The encodings and default reset states of the bits in SYNCR in the three clock modes are shown below.

SYNCR — Synthesizer Control Register, Slow Reference Mode **0xYF FA04**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
X	W	Y						EDIV	Re-served	LOSCD	SLIMP	SLOCK	RSTEN	STSCIM	STEXT
RESET:															
0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0

SYNCR — Synthesizer Control Register, Fast Reference Mode **0xYF FA04**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
X	W[2:0]			—	Y			EDIV	Re-served	LOSCD	SLIMP	SLOCK	RSTEN	STSCIM	STEXT
RESET:															
0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0

SYNCR — Synthesizer Control Register, External Clock Mode **0xYF FA04**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
X	Reserved				Y			EDIV	Re-served	LOSCD	SLIMP	SLOCK	RSTEN	STSCIM	STEXT
RESET:															
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

4.3.6.1 Frequency control Bits (X,W,Y)

Bits [15:8] of the SYNCR control the multiplication or division factors of the synthesizer. X bit [15] controls a one-bit divider which drives the system clock in all modes. When X is set, the divider is bypassed; when clear, the system clock is divided by two. The W bits and the Y bits have different field lengths and functions depending on the clock mode. In slow reference mode, bit [14] is the single W bit, and bits [13:8] are the six Y bits, and both fields are used to multiply the reference frequency. In fast reference mode, bits [14:12] are the three W bits, which are used to multiply the reference frequency. Bits [10:8] are the three Y bits, which are used to divide the PLL output frequency. In external clock mode, bits [14:11] are unused, and bits [10:8] are the three Y bits which are used to divide the input clock frequency. Refer to **Table 4-6** and **Table 4-7** for system frequencies available in common configurations.

4.3.6.2 E Clock Divide Rate (EDIV)

The E clock that goes to the chip select section is driven from a divider circuit off of the same clock source that drives the external clock. This allows turning off or leaving on the E clock in LPSTOP mode using the STEXT bit. When EDIV=0, E is the system clock divided by eight. When EDIV=1, E is the system clock divided by 16. EDIV is cleared to zero by reset.

4.3.6.3 Loss of Clock Oscillator Disable (LOSCD)

An internal oscillator is used in the detection of loss of clock. This oscillator can be disabled by setting this bit. See **4.3.7.5 Loss Of Clock Detect Circuit (LOC)** for details of this feature. When LOSCD = 1, the loss of clock oscillator is disabled. When LOSCD = 0, the loss of clock oscillator is enabled. This bit is cleared to 0 on reset.

4.3.6.4 Limp Mode (SLIMP)

This read only status bit indicates whether the loss of crystal detect logic has detected a loss of system clock. If a loss of clock is detected, the synthesizer will use an internal RC oscillator to derive the system clock and enter limp mode, allowing the MCU to continue to run even without an external clock.

SLIMP=0 indicates that the system clock is being provided normally, either by the PLL or by an external clock from the EXTAL input. SLIMP=1 indicates that a loss of system clock has been detected, and the system clock is being provided from the loss of crystal oscillator reference. See **4.3.7.5 Loss Of Clock Detect Circuit (LOC)**. The limp clock is approximately 16 KHz.

4.3.6.5 Synthesizer Lock (SLOCK)

This read only status bit gives an indication of when the PLL is locked in at the specified frequency. Synthesizer lock occurs when the filter circuit switches from the wide bandwidth to the narrow bandwidth mode (see **4.3.7.2 Phase Comparator and Filter**).

SLOCK=0 is an indication that the PLL is enabled and is not yet locked into the narrow bandwidth mode. If SLOCK=1, it indicates that either the PLL is disabled (system clock is driven in directly), or the PLL is locked. This bit is used by the power on reset circuit to determine whether the clock is stable or not.

4.3.6.6 Reset Enable (RSTEN)

This bit dictates what action to take when the loss of clock logic detects a loss of system clock. If RSTEN=1, a loss of clock will cause a system reset. After completing a normal reset sequence, the part will exit reset and run normally in limp mode. If RSTEN is set while the MC68F375 is currently in limp mode, it will enter reset immediately. If RSTEN=0, when a loss of clock occurs, the part will continue to run normally in limp mode. This bit is cleared to 0 by reset.

4.3.6.7 Low Power Stop Mode SCIM2 Clock (STSCIM)

This bit determines what happens to the SCIM2 clock when the CPU executes the LPSTOP instruction. When STSCIM=0, the SCIM2 clock comes from the crystal oscillator circuit and the VCO is turned off to save power. When STSCIM=1, the SCIM2 clock is driven from the VCO. STSCIM is cleared to 0 by reset.

4.3.6.8 Low Power Stop Mode External Clock (STEXT)

This bit determines what happens to the external clock pin when the CPU executes the LPSTOP instruction. When STEXT=0, no external clock will be driven in LPSTOP mode. When STEXT=1, the external clock is driven from the SCIM2 clock as governed by the STSCIM bit. STEXT is cleared to 0 by reset.

4.3.7 Clock Circuits Operation

For the following discussion, refer to **Figure 4-2**, **Figure 4-3**, and **Figure 4-4**.

4.3.7.1 Synthesizer Circuit

The clocks for the MCU can be derived from an external crystal reference frequency which is multiplied using the phase locked loop, frequency synthesizer circuit as described below.

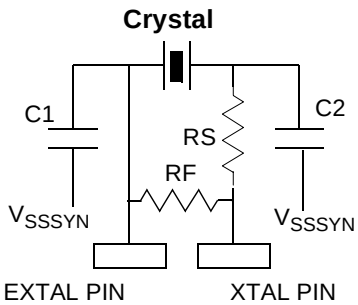
4.3.7.2 Phase Comparator and Filter

The output of the crystal oscillator is compared with the output of the divider chain to determine the frequency relationship of the two signals. The result of this compare is then filtered and used to control the voltage controlled oscillator (VCO).

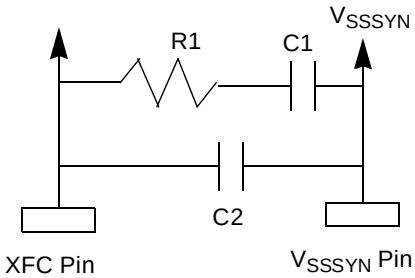
The PLL loop filter has two bandwidths which are automatically selected. When the PLL is first enabled, the wide bandwidth mode is used which enables the PLL frequency to ramp up quickly. Then when the output frequency is near the desired frequency, the filter is switched to the narrow bandwidth to make the final frequency more stable. The PLL requires an external filter network on XFC. **Figure 4-5** shows the suggested values of bypass and PLL external capacitors.

For slow reference mode with a multiplication factor of N=512 (ex: 32.768 KHz reference for a 16.78 MHz f_{SYS}) the values are: R1 = 18 K; C1 = 0.1 µF; and C2 = 3300 pF. For fast reference mode with a multiplication factor of N=4 (ex: 4.194 MHz reference and 16.78 MHz f_{SYS}) the values are: R1 = 20 K; C1 = 1000 pF; and C2 = 100 pF. Reference frequencies and f_{SYS} values very different from those stated may require different filter values (first order affect is multiplication factor N, based on W and Y bits in the SYCNR register). Leakage from the XFC pin must not be greater than that of a 15-mΩ resistor to meet jitter specifications. If the PLL is not enabled, then the XFC filter is not required and the pin may be left unconnected.

The VCO will oscillate at a frequency dictated by the voltage coming out of the filter circuit. To save system power, the VCO is shut off when it is not needed.



Slow Reference Mode (32.768 KHz): RF=1-10 mΩ
RS=330 KΩ, C1=C2=22 pF
Fast Reference Mode (4.194 MHz):RF=1-10 mΩ
RS=1.5 KΩ, C1=C2=27 pF
Fast Reference Mode (5.438 MHz):RF=1-10 mΩ
RS=400 Ω, C1=C2=10 pF
Location of RS can be inboard (as shown here) or outboard of the crystal.
NOTE: Actual circuit should be reviewed with crystal manufacturer.



Slow Reference Mode (32.768 KHz):

$$R1 = R_0 \cdot \sqrt{(Y+1) \cdot 2^{(2W+1)}}; R_0 = 475.2,$$

C1=0.1 µF, C2=3300 pF

Fast Reference Mode (4.194 - 5.438 MHz):

$$R1 = R_0 \cdot \sqrt{(W+1)}; R_0 = 1197,$$

C1=4000 pF, C2=330 pF

Figure 4-5 Crystal Oscillator and External Capacitor Configuration

4.3.7.3 Lock Detect Circuit

The clock generator subsystem on the MC68F375 also includes an improved lock detect circuit. This lock detect does not depend on frequency over-shoot as did the original circuit. It is also more precise than the original circuit. Basic operation is based on two counters and a “frequency match” requirement. When the VCO feedback frequency ramps to within approximately 3.5% of the reference frequency, a count-down time-out is initiated. At the end of that period, the PLL is considered locked and the SLOCK bit is set in the SYNCR register. This same logic will reset the SLOCK bit if the VCO feedback frequency drifts outside of approximately ± 3.5% range. The actual trip points for going into and out of lock depend on the phase relationship of the reference frequency and the VCO feedback frequency.

4.3.7.4 Clock Control Circuit

The clock control circuit generates the following system clock signals:

- EXTCLK is the external system clock which is driven out on the CLKOUT pin. This signal is also used to generate the E clock which is used by the chip selects.
- ICLOCK is the internal module system clock. This clock is used by all of the internal modules of the MCU except for the SCIM2. This clock is stopped in LPSTOP mode.
- SCLOCK is the SCIM2 module’s primary clock. This clock is used by all sections of the SCIM2 except those that must continue to operate in LPSTOP. This clock is stopped in LPSTOP mode.
- SCIMCLK: This is the SCIM2’s secondary system clock which is used by sections of the SCIM2 which continue to operate in LPSTOP mode, such as timers in the system protection block.

4.3.7.5 Loss Of Clock Detect Circuit (LOC)

The loss of clock (LOC) feature is designed to detect the condition in which the SCIM2 SCIMCLK system clock falls in frequency to a range between 20 KHz and 150 Hz. The LOC circuit uses an independent, free-running RC oscillator as a time base to monitor the system clock. The LOC circuit is used as a system protection feature to monitor the operation of the crystal reference for the PLL, or the presence of an external clock signal.

4.3.8 Basic Operation

The SCIM2 internal system clock, SCIMCLK, is monitored by a loss-of-clock subsystem. SCIMCLK remains running in LPSTOP at the PLL frequency if STSCIM=1, or at the crystal frequency if STSCIM=0, or at the external clock frequency if the part is in External Clock mode. The LOC detector should always be triggered if SCIMCLK falls below 150 Hz, and should never be triggered if SCIMCLK is running above 20 KHz. This specified range provides a sufficiently large window of uncertainty to compensate for variations in the RC oscillator frequency due to processing and operating conditions.

The LOC detector operates in either PLL or external clock modes. When it is triggered, limp mode is entered, the SLIMP bit is set, and an alternate clock is provided as the system clock until edges are detected on the crystal/external clock input. The alternate clock is the output of an RC oscillator which is also used as the time-base for the LOC detector. All clock switching is done synchronously, such that no short pulses, or glitches, are caused on the system clock.

4.3.8.1 POR Characteristics

When the power-on-reset logic detects the application of power or the return of power after a power loss situation, the internal RC oscillator is selected as the system clock source. The RC oscillator begins to operate at relatively low levels of VDD and typically several milliseconds before the crystal oscillator/VCO will begin to function. This provides system clocks as soon as possible, allowing I/O pins and modules to reach defined reset states as soon as possible. After the crystal oscillator and VCO start-up is recognized by the LOC logic, a few more RC clock edges followed by a few VCO clock edges will switch the system clock source control logic over from the RC oscillator to the crystal oscillator/VCO subsystem. Exact switching time depends on the RC oscillator frequency and the crystal/VCO clock frequency.

4.3.8.2 External Clock Operating Mode

If SCIMCLK stops in external clock mode, the LOC detect triggers, and the system clock is switched to the alternate clock until the external clock input restarts.

4.3.8.3 PLL Operating Mode

When the PLL is being used to provide the system clock and the crystal reference input stops, the synthesized clock frequency will drop below the LOC threshold. When this occurs, the system clock source is switched to the alternate clock until the crystal reference restarts. If and when the crystal restarts, the clock source will switch back to the PLL, in unlocked mode. The PLL will then relock to the reference frequency.

4.3.8.4 RSTEN Bit Operation

The RSTEN bit is used to put the part in reset if loss of clock is detected. The reset sequence clears the RSTEN bit, and when the sequence finishes, the part exits reset and runs in limp mode. Setting the RSTEN bit while in limp mode will cause reset immediately. This will allow for a continuously reset the part as long as it is in limp mode, by setting the RSTEN bit after exiting reset.

4.3.8.5 Reset Conditions

To save power, the RC oscillator can be disabled by setting the LOSCD bit in the SYNCR register. In this case, the ability to detect loss of clock is disabled. However, if the RESET pin is driven low, the RC oscillator is forced on to provide a system clock, ensuring that external RESET will be recognized even in a DC state. For more information, see **4.7 Reset**.

In order to reset the ports to their post reset state listed in **Table 4-8**, an internal clock and the reset signal must be present. The clocks are generated with the SCIM2E voltage controlled oscillator (VCO). The VCO is biased to operate at approximately eight KHz whenever the crystal oscillator is not detected. This feature causes the VCO to start before the crystal oscillator. (See **APPENDIX E ELECTRICAL CHARACTERISTICS** for the exact frequencies.)

Table 4-8 Port Reset Condition

Port	State of Pins after Reset
A[1]	Input
B	Input
G	Input
H	Input
E	Input
F	Input
C	Output (PC[6:2, 0]) are driven high
PQS0, PQS1, PQS2	Input
PQA, PQB	Input
TPU3	Input

NOTES:
1. Each port requires approximately 4 clocks to assume their post reset state. The VCO startup time is no more than 15 msec after VDD reaches minimum value.

Only single byte or aligned word writes on the IMB to the RAM module will be guaranteed to complete without data corruption for synchronous resets. A long-word write, a misaligned operand write, a write to a peripheral module other than the RAM or a read cycle are not guaranteed. External writes are also guaranteed to complete, provided the external configuration logic on the data bus is conditioned by R/W. Asynchronous reset sources usually indicate a catastrophic failure and require the reset control logic to assert reset to the system immediately.

4.3.8.6 Low Power Operation

Low power operation is initiated by the CPU32. To reduce power consumption selectively, the CPU32 can set the STOP bits in each module configuration register. To minimize overall microcontroller power consumption, the CPU32 can execute the LPSTOP instruction which causes the SCIM2E to turn off the system clock.

A loss of clock will be recognized while the part is in low power stop, unless the RC oscillator is disabled. If it is disabled, external RESET will re-enable it so that RESET will be recognized. If a loss of clock occurs in LPSTOP mode and RSTEN=0, the part will continue to operate normally on the alternate clock. LPSTOP can then be exited normally, either by an interrupt request or by external RESET. If RSTEN=1 in LPSTOP mode, the loss of clock event will cause reset. For more information, see **4.4.9 Low Power Stop Mode**.

When the CPU executes LPSTOP, a special CPU space bus cycle writes a copy of the current interrupt mask into the clock control logic. The SCIM2E brings the MCU out of low power stop mode when one of the following exceptions occur:

- RESET
- Trace
- SCIM2E interrupt of higher priority than the stored interrupt mask

Refer to **4.4.9 Low Power Stop Mode** and **4.6.4.2 LPSTOP Broadcast Cycle** for more information.

During low power stop mode, unless the system clock signal is supplied by an external source and that source is removed, the SCIM2E clock control logic and the SCIM2E clock signal (SCIMCLK) continue to operate. The periodic interrupt timer and input logic for the $\overline{\text{RESET}}$ and $\overline{\text{IRQ}}$ pins are clocked by SCIMCLK. The SCIM2E can also continue to generate the CLKOUT signal while in low power stop mode.

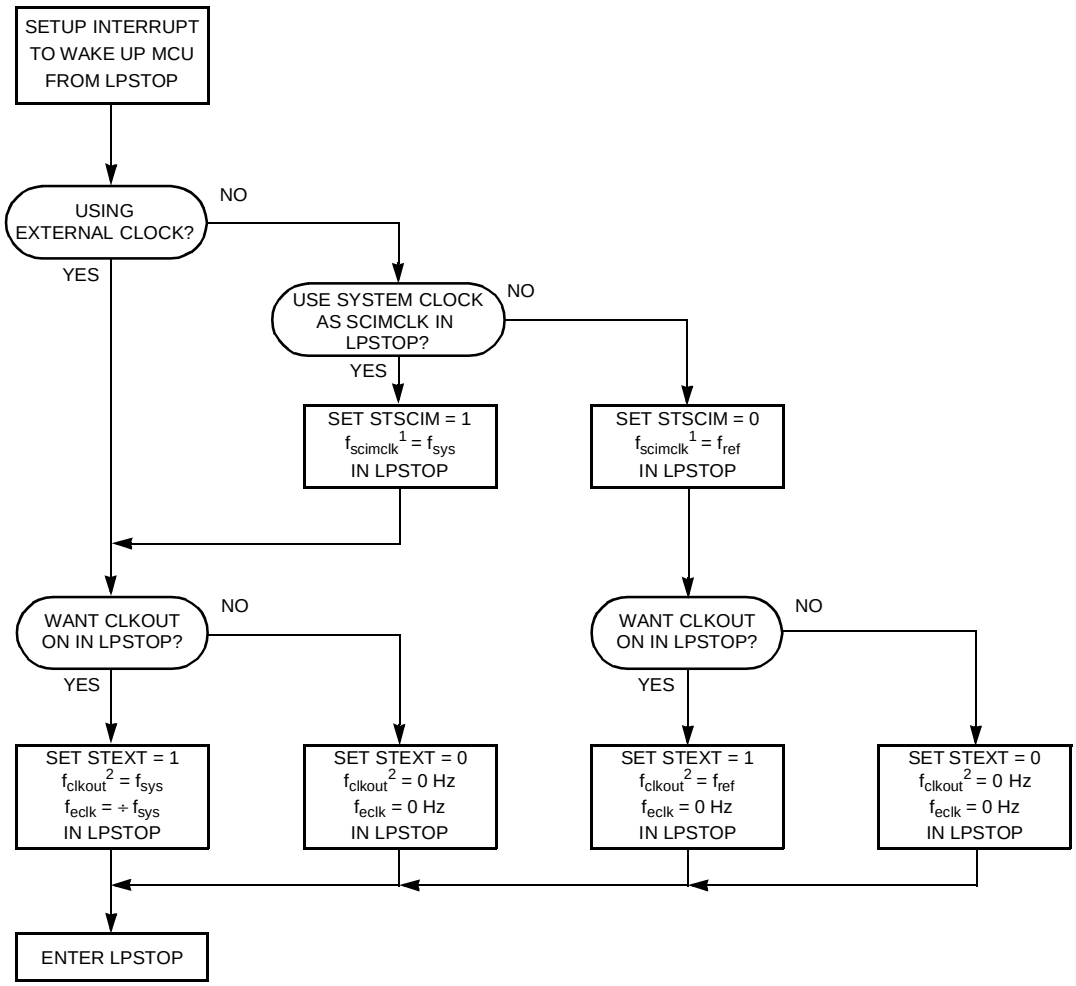
During low power stop mode, the address bus and data bus continue to drive the LPSTOP broadcast cycle, and bus control signals are negated. I/O pins configured as outputs continue to hold their previous state; I/O pins configured as inputs will remain in a high impedance state.

The STSCIM and STEXT bits in SYNCR determine clock operation during low power stop mode.

The flow chart shown in **Figure 4-6** summarizes the effects of the STSCIM and STEXT bits when the MCU enters low power stop mode. Any clock in the off state is held low. If the synthesizer VCO is turned off during low power stop mode, PLL relock delay will occur when the MCU exits LPSTOP mode and the VCO is re-enabled.

NOTE

In LPSTOP mode, the crystal oscillator is not disabled and will continue to run.



1. THE SCIMCLK IS USED BY THE PIT, $\overline{IRQ}$, AND INPUT BLOCKS OF THE SCIM2E.
2. CLKOUT CONTROL DURING LPSTOP IS OVERRIDDEN BY THE EXOFF BIT IN SCIMMCR. IF EXOFF = 1, THE CLKOUT PIN IS ALWAYS IN A HIGH IMPEDANCE STATE AND STEXT HAS NO EFFECT IN LPSTOP. IF EXOFF = 0, CLKOUT IS CONTROLLED BY STEXT IN LPSTOP.

LPSTOPFLOW

Figure 4-6 LPSTOP Flowchart

4.3.8.7 Loss of Reference Signal

The SCIM2E includes circuitry to detect a loss of the synthesizer f_{ref} signal and to force reset or to allow continued operation from an alternate clock source. The LOSCD, SLIMP, and RSTEN bits in SYNCR control and report the behavior of the clock synthesizer when f_{ref} loss is detected.

In the SCIM2E, f_{ref} is compared to the output of an independent free-running RC oscillator to detect failure. The loss of clock detector should always be triggered when f_{ref} falls below 150 Hz and should never be triggered when f_{ref} is above 20 KHz. This range provides a window of uncertainty sufficiently large enough to compensate for

variations in the output of the RC oscillator due to processing and/or operating conditions.

The loss of clock detector can be disabled by writing a one to the loss of clock oscillator bit (LOSCD). This disables the free-running RC oscillator and prevents f_{ref} loss from being detected. The reset state of LOSCD is zero which enables the RC oscillator and loss of clock detector.

The reset enable bit (RSTEN) determines how the MCU will process a loss of clock detection. The default state out of reset for RSTEN is zero. This forces the clock synthesizer into the limp mode operating state. In limp mode, the RC oscillator used by the loss of clock detector provides the system clock. Limp mode frequency varies from device-to-device but does not exceed one-half the maximum system clock frequency.

When set to one, RSTEN allows the clock synthesizer to reset the MCU when the loss of clock detector triggers. After powering-up from a loss of clock reset, the MCU will set the LOC bit in the reset status register (RSR) and begin operation in limp mode.

The limp status bit (SLIMP) in SYNCR indicates that f_{ref} has failed and that the MCU has entered limp mode. SLIMP will remain set until normal f_{ref} operation is restored.

4.4 System Protection

The system protection block reports reset status information, monitors internal bus activity, and provides periodic interrupt generation. **Figure 4-7** is a block diagram of the submodule.

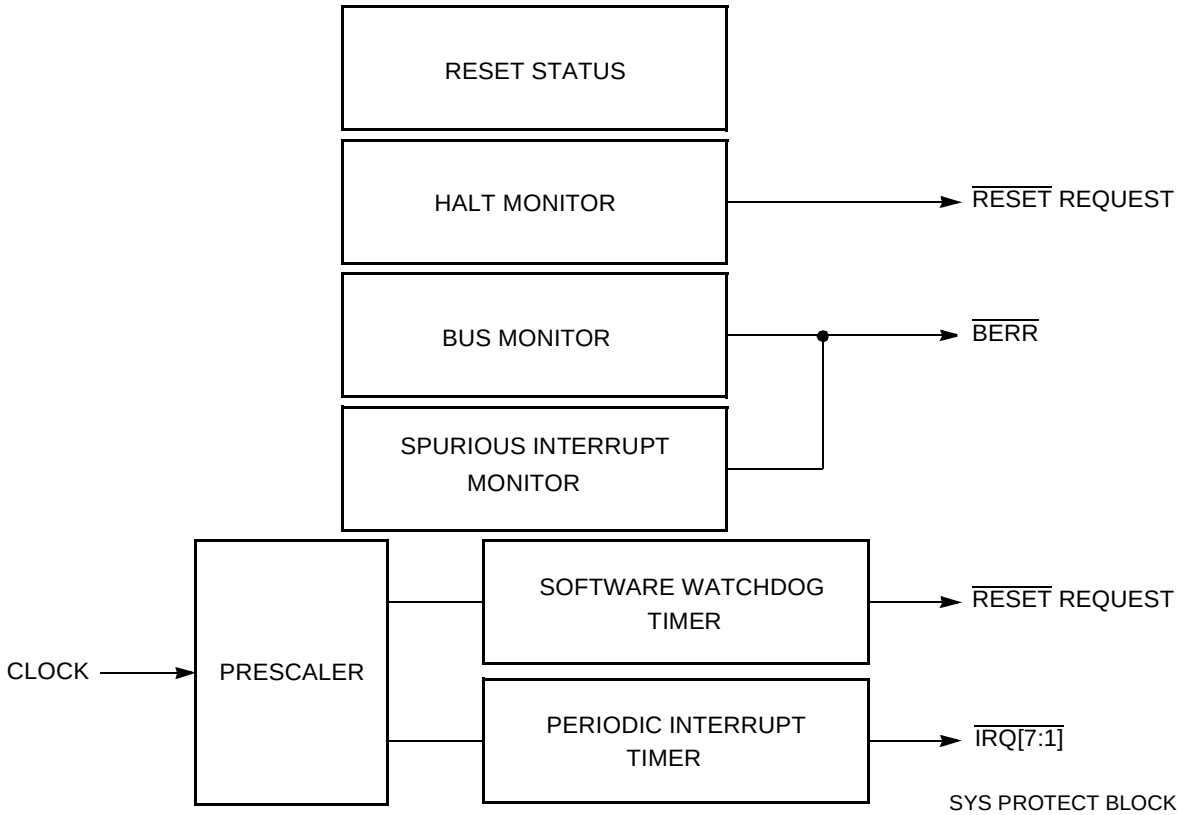


Figure 4-7 System Protection

4.4.1 System Protection Control Register

SYPCR — System Protection Control Register **0xYF FA21**

7	6	5	4	3	2	1	LSB 0
SWE	SWP	SWT[1:0]		HME	BME	BMT[1:0]	
1	$\overline{\text{MODCLK}}$	0	0	0	0	0	0

Table 4-9 SYPCR Bit Descriptions

Bit(s)	Name	Description
7	SWE	Software watchdog enable 0 = Software watchdog is disabled. 1 = Software watchdog is enabled.
6	SWP	Software watchdog prescaler. This bit controls the value of the software watchdog prescaler. The reset value of SWP is the complement of the state of the MODCLK pin during reset. 0 = Software watchdog clock is not prescaled. 1 = Software watchdog clock is prescaled by 512.
5:4	SWT	Software watchdog timing. This field selects the divide ratio used to establish the software watchdog timeout period. Refer to **Table 4-12**.

Table 4-9 SYPCR Bit Descriptions (Continued)

Bit(s)	Name	Description
3	HME	Halt monitor enable 0 = Halt monitor is disabled. 1 = Halt monitor is enabled.
2	BME	Bus monitor external enable 0 = Disable bus monitor for external bus cycles. 1 = Enable bus monitor for external bus cycles.
1:0	BMT	BMT[1:0] — Bus Monitor Timing. This field selects the bus monitor timeout period. Refer to **Table 4-10**.

4.4.2 Reset Status

The reset status register (RSR) latches MCU status during reset. Refer to **4.7.4 Reset Status Register** for more information.

4.4.3 Bus Monitor

The internal bus monitor checks data size acknowledge ($\overline{\text{DSACK}}$) or autovector ($\overline{\text{AVEC}}$) signal response times during normal bus cycles. The monitor asserts the internal bus error ($\overline{\text{BERR}}$) signal when the response time is excessively long.

$\overline{\text{DSACK}}$ and $\overline{\text{AVEC}}$ response times are measured in clock cycles. Maximum allowable response time can be selected by setting the bus monitor timing (BMT[1:0]) field in the system protection control register (SYPCR). **Table 4-10** shows the periods allowed.

Table 4-10 Bus Monitor Period

BMT[1:0]	Bus Monitor Timeout Period
00	64 System Clocks
01	32 System Clocks
10	16 System Clocks
11	8 System Clocks

The monitor does not check $\overline{\text{DSACK}}$ response on the external bus unless the CPU32 initiates a bus cycle. The BME bit in SYPCR enables the internal bus monitor for internal-to-external bus cycles. If a system contains external bus masters, an external bus monitor must be implemented and the internal-to-external bus monitor option must be disabled.

When monitoring transfers to an 8-bit port, the bus monitor does not reset until both byte accesses of a word transfer are completed. Monitor timeout period must be at least twice the number of clocks that a single-byte access requires.

4.4.4 Halt Monitor

The halt monitor responds to an assertion of the $\overline{\text{HALT}}$ signal on the internal bus when a double bus fault occurs. A flag in the reset status register (RSR) will indicate when the last reset was caused by the halt monitor. Halt monitor reset can be inhibited by

the halt monitor (HME) enable bit in SYPCR. Refer to **4.6.5.2 Double Bus Faults** for more information.

4.4.5 Spurious Interrupt Monitor

During interrupt exception processing, the CPU32 normally acknowledges an interrupt request, arbitrates among various sources of interrupt, recognizes the highest priority source, and then acquires a vector or responds to a request for autovectoring. The spurious interrupt monitor asserts the internal bus error signal ($\overline{BERR}$) if no interrupt arbitration occurs during interrupt exception processing. The assertion of $\overline{BERR}$ causes the CPU32 to load the spurious interrupt exception vector into the program counter. The spurious interrupt monitor cannot be disabled.

Refer to **4.8 Interrupts** for further information. For detailed information about interrupt exception processing, refer to **4.8.1 Interrupt Exception Processing**.

4.4.6 Software Watchdog

The software watchdog is controlled by the software watchdog enable (SWE) bit in SYPCR. When enabled, the watchdog requires that a service sequence be written to the software service register (SWSR) on a periodic basis. If servicing does not take place, the watchdog times out and asserts the $\overline{RESET}$ signal.

Each time the service sequence is written, the software watchdog timer restarts. The sequence to restart the software watchdog requires the following steps:

- Write 0x55 to SWSR
- Write 0xAA to SWSR

Both writes must occur before timeout in the order listed. Any number of instructions can be executed between the two writes.

The clock rate of the watchdog timer is affected by clock mode, the software watchdog prescale (SWP) bit, and the software watchdog timing (SWT[1:0]) field in SYPCR. In slow reference mode and external clock mode, f_{ref} or $f_{ref} \div 512$ can be used to clock the watchdog timer. The options in fast reference mode are $f_{ref} \div 128$ or $(f_{ref} \div 128) \div 512$. In all cases, the divide-by-512 option is selected when SWP = 1.

The value of SWP is affected by the state of the V_{DDSYN}/MODCLK pin during reset, as shown in **Table 4-11**. System software can change SWP value.

Table 4-11 SWP Reset States

V_{DDSYN}/MODCLK	SWP
0 (External Clock)	1 (÷ 512)
1 (Synthesized Clock)	0 (÷ 1)

SWT[1:0] selects the divide ratio used to establish the software watchdog timeout period.

The following equation calculates the timeout period in slow reference mode:

$$\text{Timeout Period} = \frac{\text{Divide Ratio Specified by SWP and SWT[1:0]}}{f_{ref}}$$

The following equation calculates the timeout period in fast reference mode:

$$\text{Timeout Period} = \frac{(128)(\text{Divide Ratio Specified by SWP and SWT[1:0]})}{f_{ref}}$$

The following equation calculates the timeout period in external clock mode:

$$\text{Timeout Period} = \frac{\text{Divide Ratio Specified by SWP and SWT[1:0]}}{f_{ref}}$$

Table 4-12 shows the divide ratio for each combination of the SWP and SWT[1:0] bits. When SWT[1:0] are modified, a watchdog service sequence must be performed before the new timeout period can take effect.

Table 4-12 Software Watchdog Divide Ratio

SWP	SWT[1:0]	Divide Ratio
0	00	2^9
0	01	2^{11}
0	10	2^{13}
0	11	2^{15}
1	00	2^{18}
1	01	2^{20}
1	10	2^{22}
1	11	2^{24}

Figure 4-8 is a block diagram of the watchdog timer and the clock control for the periodic interrupt timer.

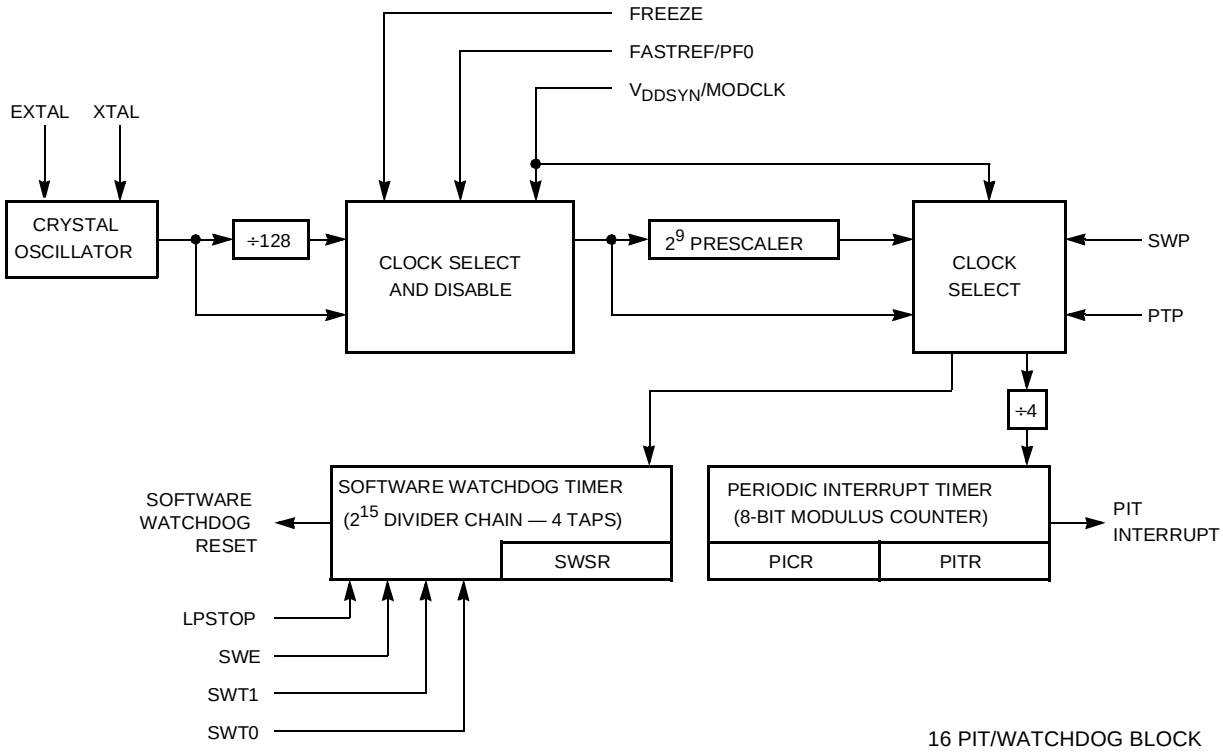


Figure 4-8 Periodic Interrupt Timer and Software Watchdog Timer

4.4.6.1 Software Watchdog Service Register

SWSR — Software Watchdog Service Register[1] **0xYF FA27**

7	6	5	4	3	2	1	LSB 0
SWSR[7:0]							
0	0	0	0	0	0	0	0

NOTES:
1. This register is shown with a read value.

This register can be read or written at any time. Bits [15:8] are reserved and will always read zero.

4.4.7 Periodic Interrupt Timer

The periodic interrupt timer (PIT) allows the generation of interrupts of specific priority at predetermined intervals. This capability is often used to schedule control system tasks that must be performed within time constraints. The PIT consists of a prescaler, a modulus counter, and registers that determine interrupt timing, priority and vector assignment. Refer to **4.8.1 Interrupt Exception Processing** for further information about interrupt exception processing.

The periodic interrupt timer modulus counter is clocked by one of two signals. When the PLL is enabled, f_{ref} is used in slow reference mode and $f_{ref} \div 128$ is used in fast reference mode. When the PLL is disabled, f_{ref} is used. The value of the periodic timer prescaler (PTP) bit in the periodic interrupt timer register (PITR) determines system clock prescaling for the periodic interrupt timer. One of two options, either no prescaling, or prescaling by a factor of 512, can be selected. The value of PTP is affected by the state of the V_{DDSYN}/MODCLK pin during reset, as shown in **Table 4-13**. System software can change PTP value.

Table 4-13 PTP Reset States

V_{DDSYN}/MODCLK	PTP
0 (External Clock)	1 (÷ 512)
1 (Synthesized Clock)	0 (÷ 1)

Either clock signal selected by PTP is divided by four before driving the modulus counter. The modulus counter is initialized by writing a value to the periodic interrupt timer modulus (PITM[7:0]) field in PITR. A zero value turns off the PIT. When the modulus counter reaches zero, an interrupt is generated. The modulus counter is then reloaded with the value in PITM[7:0] and counting repeats. If a new value is written to PITR, it is loaded into the modulus counter when the current count is completed.

The following equation calculates the PIT period in slow reference mode:

$$\text{PIT Period} = \frac{(\text{PITM[7:0]})(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

The following equation calculates the PIT period in fast reference mode:

$$\text{PIT Period} = \frac{(128)(\text{PITM[7:0]})(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

The following equation calculates the PIT period in external clock mode:

$$\text{PIT Period} = \frac{(\text{PITM[7:0]})(1 \text{ if PTP} = 0, 512 \text{ if PTP} = 1)(4)}{f_{ref}}$$

4.4.8 Interrupt Priority and Vectoring for the Periodic Interrupt Timer

Interrupt priority and vectoring for the PIT are determined by the values of the periodic interrupt request level (PIRQL[2:0]) and periodic interrupt vector (PIV[7:0]) fields in the periodic interrupt control register (PICR).

The PIRQL field is compared to the CPU32 interrupt priority mask to determine whether the interrupt is recognized. **Table 4-14** shows PIRQL[2:0] priority values.

Because of SCIM2E hardware prioritization, a PIT interrupt is serviced before an external interrupt request of the same priority. The periodic timer continues to run when the interrupt is disabled.

Table 4-14 Periodic Interrupt Priority

PIRQL[2:0]	Priority Level
000	Periodic Interrupt Disabled
001	Interrupt Priority Level 1
010	Interrupt Priority Level 2
011	Interrupt Priority Level 3
100	Interrupt Priority Level 4
101	Interrupt Priority Level 5
110	Interrupt Priority Level 6
111	Interrupt Priority Level 7

The PIV field contains the periodic interrupt vector. The vector is placed on the IMB when an interrupt request is made. The vector number is used to calculate the address of the appropriate vector in the exception vector table. The reset value of the PIV field is 0x0F, which corresponds to the uninitialized interrupt exception vector.

4.4.9 Low Power Stop Mode

When the CPU32 executes the LPSTOP instruction, the current interrupt priority mask is stored in the clock control logic, internal clocks are disabled according to the state of the STSCIM bit in SYNCR, and the MCU enters low power stop mode. The bus monitor, halt monitor, and spurious interrupt monitor are all inactive during low power stop.

During low power stop mode, the clock input to the software watchdog timer is disabled and the timer stops. The software watchdog begins to run again on the first rising clock edge after the MCU exits low power stop mode. The watchdog is not reset when entering low power stop mode. A service sequence must be performed to reset the timer.

The periodic interrupt timer does not respond to the LPSTOP instruction, but continues to run during LPSTOP. To stop the periodic interrupt timer, PITM[7:0] must be loaded with zero before entering LPSTOP. A PIT interrupt, or an external interrupt request, can bring the MCU out of low power stop mode if it has a higher priority than the interrupt mask value stored in the clock control logic when low power stop mode is entered. LPSTOP can be terminated by a reset.

4.4.9.1 Periodic Interrupt Control Register

PICR sets the interrupt level and vector number for the periodic interrupt timer (PIT). Bits [10:0] can be read or written at any time. Bits [15:11] are reserved and always read zero.

PICR — Periodic Interrupt Control Register **0xYF FA22**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	PIRQL[2:0]			PIV[7:0]							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Table 4-15 PICR Bit Descriptions

Bit(s)	Name	Description
15:11	—	**Reserved**
10:8	PIRQL	Periodic interrupt request level. This field determines the priority of periodic interrupt requests. A value of 0b000 disables PIT interrupts.
7:0	PIV	Periodic interrupt vector. This field specifies the periodic interrupt vector number supplied by the SCIM2 when the CPU32 acknowledges an interrupt request.

4.4.9.2 Periodic Interrupt Timer Register

PITR — Periodic Interrupt Timer Register **0xYF FA24**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	0	0	PTP	PITM[7:0]							
RESET:															
0	0	0	0	0	0	0	$\overline{\text{MOD-CLK}}$	0	0	0	0	0	0	0	0

The PITR contains the count value for the periodic timer. This register can be read or written at any time.

Table 4-16 PITR Bit Descriptions

Bit(s)	Name	Description
15:9	—	Reserved
8	PTP	Periodic timer prescaler 0 = Periodic timer clock not prescaled. 1 = Periodic timer clock prescaled by a value of 512.
7:0	PITM	Periodic interrupt timing modulus. This field determines the periodic interrupt rate.

4.5 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices. **Figure 4-9** shows a basic system with external memory and peripherals.

The external bus has 24 address lines and 16 data lines. The EBI provides dynamic sizing between 8-bit and 16-bit data accesses. It supports byte, word, and long-word transfers. Port width is the maximum number of bits accepted or provided by the external memory system during a bus transfer. Widths of eight and 16 bits are accessed through the use of asynchronous cycles controlled by the size (SIZ1 and SIZ0) and

data size acknowledge ($\overline{DSACK1}$ and $\overline{DSACK0}$) pins. Multiple bus cycles may be required for dynamically sized transfers.

To add flexibility and minimize the necessity for external logic, MCU chip-select logic is synchronized with EBI transfers. Refer to **4.9 Chip Selects** for more information.

4.5.1 Bus Control Signals

The address bus provides addressing information to external devices. The data bus transfers 8-bit and 16-bit data between the MCU and external devices. Strobe signals, one for the address bus and another for the data bus, indicate the validity of an address and provide timing information for data.

Control signals indicate the beginning of each bus cycle, the address space, the size of the transfer, and the type of cycle. External devices can decode these signals and respond to transfer data and terminate the bus cycle. The EBI can operate in an asynchronous mode for any port width.

4.5.1.1 Address Bus

Bus signals ADDR[19:0] define the address of the byte (or the most significant byte) to be transferred during a bus cycle. The MCU places the address on the bus at the beginning of a bus cycle. The address is valid while $\overline{AS}$ is asserted.

4.5.1.2 Address Strobe

Address strobe ($\overline{AS}$) is a timing signal that indicates the validity of an address on the address bus as well as that of many control signals.

4.5.1.3 Data Bus

DATA[15:0] form a bidirectional, non-multiplexed parallel bus that transfers data to or from the MCU. A read or write operation can transfer eight or 16 bits of data in one bus cycle. For a write cycle, all 16 bits of the data bus are driven, regardless of the port width or operand size.

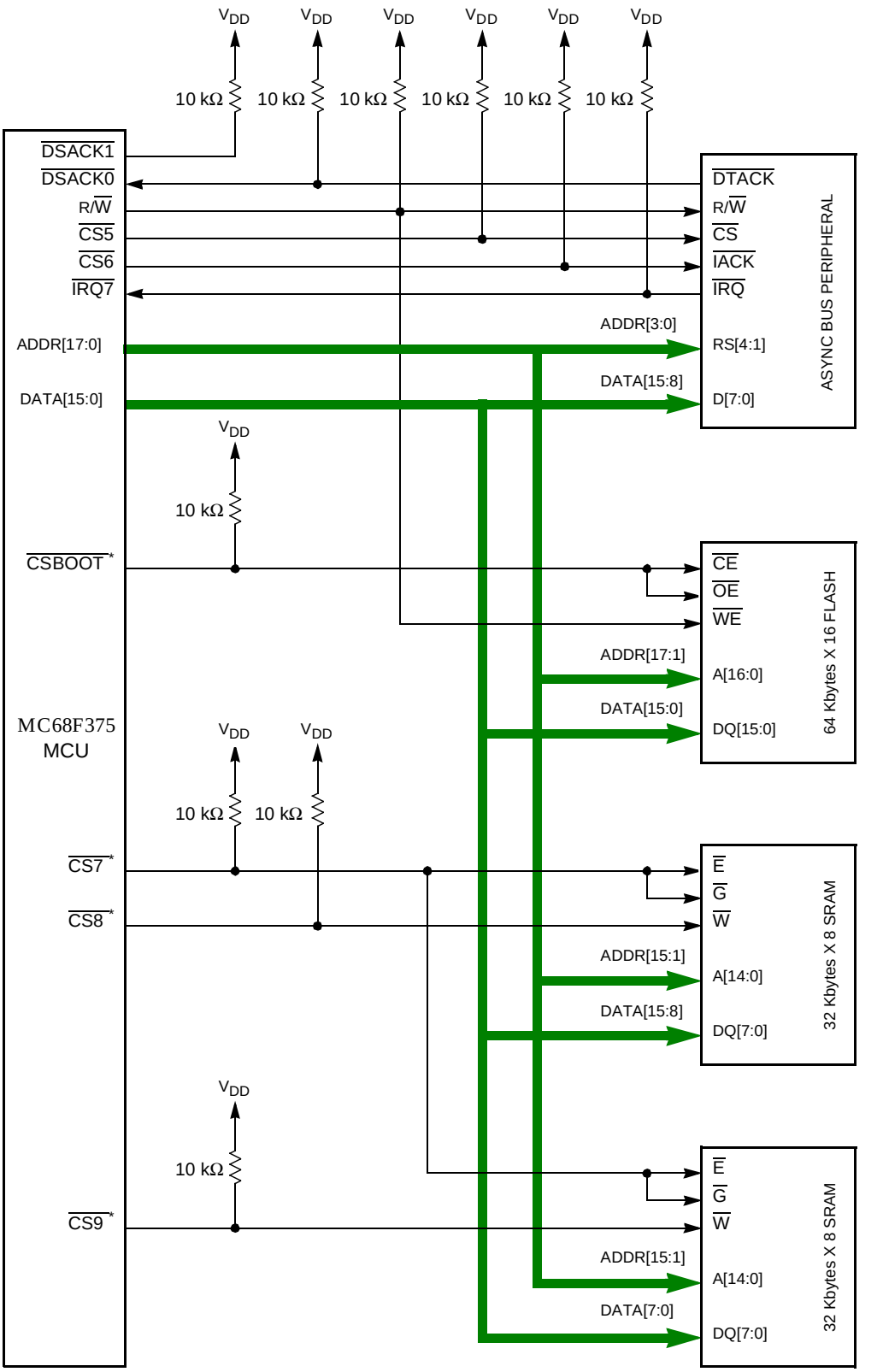


Figure 4-9 MCU Basic System

4.5.1.4 Data Strobe

Data strobe ($\overline{DS}$) is a timing signal. For a read cycle, the MCU asserts $\overline{DS}$ to signal an external device to place data on the bus. $\overline{DS}$ is asserted at the same time as $\overline{AS}$ during a read cycle. For a write cycle, $\overline{DS}$ signals an external device that data on the bus is valid.

4.5.1.5 Read/Write Signal

The read/write signal (R/$\overline{W}$) determines the direction of the transfer during a bus cycle. This signal changes state, when required, at the beginning of a bus cycle, and 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 two consecutive write cycles.

4.5.1.6 Size Signals

Size signals (SIZ[1:0]) indicate the number of bytes remaining to be transferred during an operand cycle. They are valid while $\overline{AS}$ is asserted. **Table 4-17** shows SIZ0 and SIZ1 encoding.

Table 4-17 Size Signal Encoding

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

4.5.1.7 Function Codes

The CPU generates function code signals (FC[2:0]) to indicate the type of activity occurring on the data or address bus. These signals can be considered address extensions that can be externally decoded to determine which of eight external address spaces is accessed during a bus cycle. Only supervisor data, supervisor program, user data, user program, and CPU spaces are defined.

Address space 7 is designated CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Function codes are valid while $\overline{AS}$ is asserted. **Table 4-18** shows address space encoding.

Table 4-18 Address Space Encoding

FC2	FC1	FC0	Address Space
0	0	0	Reserved
0	0	1	User data space
0	1	0	User program space
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Supervisor data space
1	1	0	Supervisor Program space
1	1	1	CPU space

4.5.1.8 Data Size Acknowledge Signals

During normal bus transfers, external devices can assert the data size acknowledge signals ($\overline{\text{DSACK}}$[1:0]) to indicate port width to the MCU. During a read cycle, these signals tell the MCU to terminate the bus cycle and to latch data. During a write cycle, the signals indicate that an external device has successfully stored data and that the cycle can terminate. $\overline{\text{DSACK}}$[1:0] can also be supplied internally by chip-select logic. Refer to **4.9 Chip Selects** for more information.

4.5.1.9 Bus Error Signal

The bus error signal ($\overline{\text{BERR}}$) can be asserted by an external source when a bus cycle is not properly terminated by $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$ assertion. It can also be asserted in conjunction with $\overline{\text{DSACK}}$ to indicate a bus error condition, provided it meets the appropriate timing requirements. Refer to **4.6.5 Bus Exception Control Cycles** for more information.

The internal bus monitor can generate the $\overline{\text{BERR}}$ signal for excessively long internal-to-external transfers. In systems with an external bus master, the SCIM2E bus monitor must be disabled and external logic must be provided to drive the $\overline{\text{BERR}}$ pin, because the internal $\overline{\text{BERR}}$ monitor has no information about transfers initiated by an external bus master. Refer to **4.6.6 External Bus Arbitration** for more information.

4.5.1.10 Halt Signal

The halt signal ($\overline{\text{HALT}}$) can be asserted by an external device for debugging purposes to cause single bus cycle operation or (in combination with $\overline{\text{BERR}}$) a retry of a bus cycle in error. The $\overline{\text{HALT}}$ signal affects external bus cycles only. As a result, a program not requiring use of the external bus may continue executing, unaffected by the $\overline{\text{HALT}}$ signal. When the MCU completes a bus cycle with the $\overline{\text{HALT}}$ signal asserted, DATA[15:0] is placed in a high-impedance state and $\overline{\text{AS}}$ and $\overline{\text{DS}}$ are driven inactive; the address, function code, size, and read/write signals remain in the same state. The MCU does not service interrupt requests while it is halted. Refer to **4.6.5 Bus Exception Control Cycles** for further information.

4.5.1.11 Autovector Signal

The autovector signal ($\overline{AVEC}$) can be used to terminate interrupt acknowledgment cycles for external interrupts only. Assertion of $\overline{AVEC}$ causes the CPU32 to generate vector numbers to locate an interrupt handler routine. If $\overline{AVEC}$ is continuously asserted, autovectors are generated for all external interrupt requests. $\overline{AVEC}$ is ignored during all other bus cycles. Refer to **4.8 Interrupts** for more information. $\overline{AVEC}$ for external interrupt requests can also be supplied internally by chip-select logic. Refer to **4.9 Chip Selects** for more information. The autovector function is disabled when there is an external bus master. Refer to **4.6.6 External Bus Arbitration** for more information.

4.5.2 Dynamic Bus Sizing

The MCU dynamically interprets the port size of an addressed device during each bus cycle, allowing operand transfers to or from 8-bit and 16-bit ports.

During a bus transfer cycle, an external device signals its port size and indicates completion of the bus cycle to the MCU through the use of the $\overline{DSACK}$ inputs, as shown in **Table 4-19**. Chip-select logic can generate data size acknowledge signals for an external device. Refer to **4.9 Chip Selects** for more information.

Table 4-19 Effect of $\overline{DSACK}$ Signals

$\overline{DSACK1}$	$\overline{DSACK0}$	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	Reserved

If the CPU is executing an instruction that reads a long-word operand from a 16-bit port, the MCU latches the first 16 bits of valid data and then 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 $\overline{DSACK}$ signals to indicate the port width. For instance, a 16-bit external device always returns $\overline{DSACK}$ for a 16-bit port (regardless of whether the bus cycle is a byte or 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 16-bit port must reside on data bus bits [15:0], and an 8-bit port must reside on data bus bits [15:8]. This minimizes the number of bus cycles needed to transfer data and ensures that the MCU transfers valid data.

The MCU always attempts to transfer the maximum amount of data on all bus cycles. For a word operation, it is assumed that the port is 16 bits wide when the bus cycle begins.

Operand bytes are designated as shown in **Figure 4-10**. OP[0:3] represent the order of access. For instance, OP0 is the most significant byte of a long-word operand, and is accessed first, while OP3, the least significant byte, is accessed last. The two bytes

of a word-length operand are OP0 (most significant) and OP1. The single byte of a byte-length operand is OP0.

OPERAND	BYTE ORDER			
	31 24	23 16	15 8	7 0
LONG WORD	OP0	OP1	OP2	OP3
THREE BYTE		OP0	OP1	OP2
WORD			OP0	OP1
BYTE				OP0

OPERAND BYTE ORDER

Figure 4-10 Operand Byte Order

4.5.3 Operand Alignment

The EBI data multiplexer establishes the necessary connections for different combinations of address and data sizes. The multiplexer takes the two bytes of the 16-bit bus and routes them to their required positions. Positioning of bytes is determined by the SIZ[1:0] and ADDR0 outputs. SIZ1 and SIZ0 indicate the number of bytes remaining to be transferred during the current bus cycle. The number of bytes transferred is equal to or less than the size indicated by SIZ1 and SIZ0, depending on port width.

ADDR0 also affects the operation of the data multiplexer. During a bus transfer, ADDR[23:1] indicate the word base address of the portion of the operand to be accessed, and ADDR0 indicates the byte offset from the base.

4.5.4 Misaligned Operands

The CPU32 uses a basic operand size of 16 bits. An operand is misaligned when it overlaps a word boundary. This is determined by the value of ADDR0. When ADDR0 = 0 (an even address), the address is on a word and byte boundary. When ADDR0 = 1 (an odd address), the address is on a byte boundary only. A byte operand is aligned at any address; a word or long-word operand is misaligned at an odd address.

The largest amount of data that can be transferred by a single bus cycle is an aligned word. If the MCU transfers a long-word operand through a 16-bit port, the most significant operand word is transferred on the first bus cycle and the least significant operand word is transferred on a following bus cycle.

The CPU32 does not support misaligned word transfers. An attempt to do so will result in an “address error” exception.

4.5.5 Operand Transfer Cases

Table 4-20 shows how operands are aligned for various types of transfers. OPn entries are portions of a requested operand that are read or written during a bus cycle and are defined by SIZ1, SIZ0, and ADDR0 for that bus cycle. **Table 4-20** also shows

to what states $\overline{\text{DSACK[1:0]}}$ must be driven — either by a chip select or by external circuitry — to terminate the given bus cycle.

Table 4-20 Operand Alignment

Current Cycle	Transfer Case	SIZ1	SIZ0	ADDR0	$\overline{\text{DSACK1}}$	$\overline{\text{DSACK0}}$	DATA [15:8]	DATA [7:0]	Next Cycle
1	Byte to 8-bit port (even)	0	1	0	1	0	OP0	(OP0)[1]	—
2	Byte to 8-bit port (odd)	0	1	1	1	0	OP0	(OP0)	—
3	Byte to 16-bit port (even)	0	1	0	0	1	OP0	(OP0)	—
4	Byte to 16-bit port (odd)	0	1	1	0	1	(OP0)	OP0	—
5	Word to 8-bit port (aligned)	1	0	0	1	0	OP0	(OP1)	2
6	Word to 8-bit port (misaligned	1	0	1	1	0	OP0	(OP0)	1
7	Word to 16-bit port (aligned)	1	0	0	0	1	OP0	OP1	—
8	Word to 16-bit port (misaligned)	1	0	1	0	1	(OP0)	OP0	3
9	Long-word to 8-bit port (aligned)	0	0	0	1	0	OP0	(OP1)	13
10	Long-word to 8-bit port (misaligned)[2]	1	0	1	1	0	OP0	(OP0)	1
11	Long-word to 16-bit port (aligned)	0	0	0	0	1	OP0	OP1	7
12	Long-word to 16-bit port (misaligned)[2]	1	0	1	0	1	(OP0)	OP0	3
13	Three byte to 8-bit port[3]	1	1	1	1	0	OP0	(OP0)	5

NOTES:
1. Operands in parentheses are ignored by the CPU32 during read cycles.
2. The CPU32 does not support misaligned operand transfers.
3. Three byte transfer cases occur only as a result of an aligned long word to 8-bit port transfer.

4.6 Bus Operation

Internal microcontroller modules are typically accessed in two system clock cycles. Regular external bus cycles use handshaking between the MCU and external peripherals to manage transfer size and data. These accesses take a minimum of three system clock cycles, with no wait states. During regular cycles, wait states can be inserted as needed by bus control logic. Refer to **4.6.2 Regular Bus Cycle** for more information.

Fast-termination cycles, which are two clock external accesses with no wait states, use chip-select logic to generate handshaking signals internally. Refer to **4.6.3 Fast Termination Cycles** and **4.9 Chip Selects** for more information. Bus control signal timing, as well as chip-select signal timing, is specified in **APPENDIX E ELECTRICAL CHARACTERISTICS**. Refer to the ***SCIM Reference Manual*** **(SCIMRM/AD)** for more information about each type of bus cycle.

4.6.1 Synchronization to CLKOUT

External devices connected to the MCU bus can operate at a clock frequency different from the frequencies of the MCU as long as the external devices satisfy the interface signal timing constraints. Although bus cycles are classified as asynchronous, they are interpreted relative to the MCU system clock output (CLKOUT).

Descriptions are made in terms of individual system clock states, labelled {S0, S1, S2,..., SN}. The designation “state” refers to the logic level of the clock signal, and does not correspond to any implemented machine state. A clock cycle consists of two successive states. Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for more information.

Bus cycles terminated by $\overline{\text{DSACK}}$ assertion normally require a minimum of three CLKOUT cycles. To support systems that use CLKOUT to generate $\overline{\text{DSACK}}$ and other inputs, asynchronous input setup time and asynchronous input hold times are specified. When these specifications are met, the MCU is guaranteed to recognize the appropriate signal on a specific edge of the CLKOUT signal.

4.6.2 Regular Bus Cycle

The following paragraphs contain a discussion of cycles that use external bus control logic. Refer to **4.6.3 Fast Termination Cycles** for more information.

To initiate a transfer, the MCU drives the address bus and the SIZ[1:0] signals. The SIZ signals and ADDR0 are externally decoded to select the active portion of the data bus. Refer to **4.5.2 Dynamic Bus Sizing** for more information. When $\overline{\text{AS}}$, $\overline{\text{DS}}$, and R/$\overline{\text{W}}$ are valid, a peripheral device either places data on the bus (read cycle) or latches data from the bus (write cycle), then asserts a $\overline{\text{DSACK}}$[1:0] combination to indicate the port size.

The $\overline{\text{DSACK}}$[1:0] signals can be asserted before the data from a peripheral device is valid on a read cycle. To ensure valid data is latched by the MCU, a maximum period between MCU assertion of $\overline{\text{DS}}$ and supplied assertion of $\overline{\text{DSACK}}$[1:0] is specified.

There is no specified maximum for the period between MCU assertion of $\overline{\text{AS}}$ and supplied assertion of $\overline{\text{DSACK}}$[1:0]. Although the MCU can transfer data in a minimum of three clock cycles when the cycle is terminated with $\overline{\text{DSACK}}$, the MCU inserts wait cycles in clock period increments until either $\overline{\text{DSACK1}}$ or $\overline{\text{DSACK0}}$ goes low.

If the $\overline{\text{DSACK}}$ bus termination signals remain unasserted, the MCU will continue to insert wait states, and the bus cycle will never end. If no peripheral responds to an access, or if an access is invalid, external logic should assert the $\overline{\text{BERR}}$ or $\overline{\text{HALT}}$ signals to abort the bus cycle (when $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ are asserted simultaneously, the CPU32 acts as though only $\overline{\text{BERR}}$ is asserted). When enabled, the SCIM2E bus monitor asserts $\overline{\text{BERR}}$ when $\overline{\text{DSACK}}$ response time exceeds a predetermined limit. The bus monitor timeout period is determined by the BMT[1:0] field in SYPCR. The maximum bus monitor timeout period is 64 system clock cycles.

4.6.2.1 Read Cycle

During a read cycle, the MCU transfers data from an external memory or peripheral device. If the instruction specifies a long-word or word operation, the MCU attempts to read two bytes at once. For a byte operation, the MCU reads one byte. The portion of the data bus from which each byte is read depends on operand size, peripheral address, and peripheral port size.

Figure 4-11 is a flow chart of a word read cycle. Refer to **4.5.2 Dynamic Bus Sizing**, **4.5.4 Misaligned Operands**, and the ***SCIM Reference Manual* (SCIMRM/AD)** for more information.

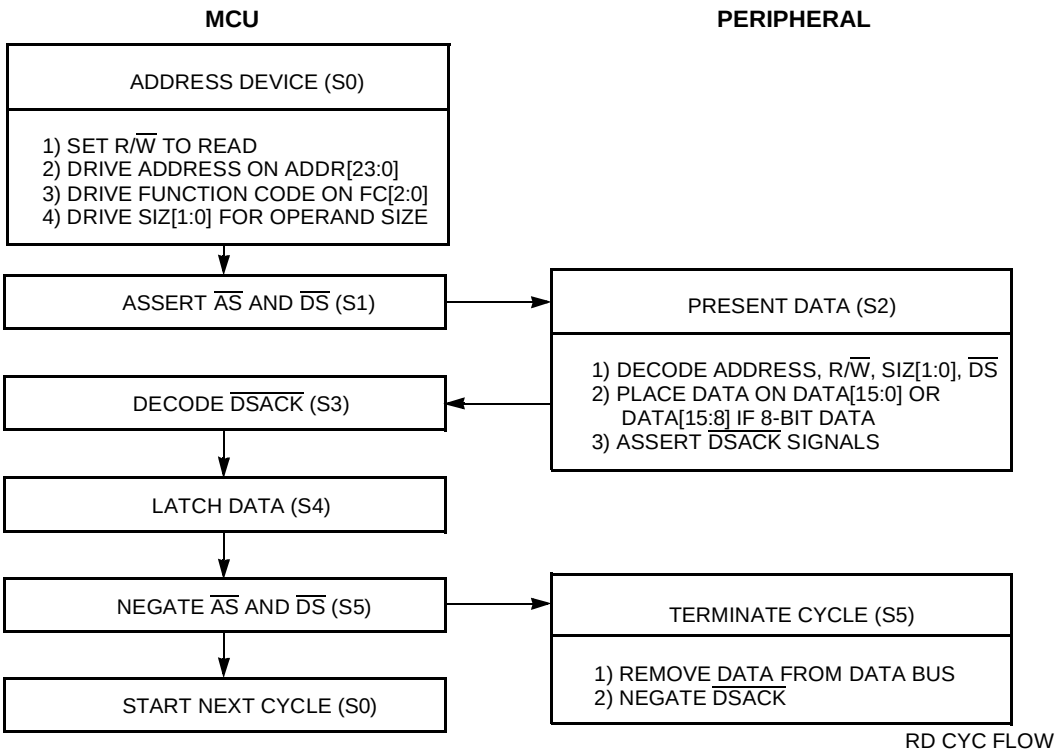


Figure 4-11 Word Read Cycle Flowchart

4.6.2.2 Write Cycle

During a write cycle, the MCU transfers data to an external memory or peripheral device. If the instruction specifies a long-word or word operation, the MCU attempts to write two bytes at once. For a byte operation, the MCU writes one byte. The portion of the data bus upon which each byte is written depends on operand size, peripheral address, and peripheral port size.

Figure 4-12 is a flow chart of a write-cycle. Refer to **4.5.2 Dynamic Bus Sizing**, **4.5.4 Misaligned Operands**, and the ***SCIM Reference Manual* (SCIMRM/AD)** for more information.

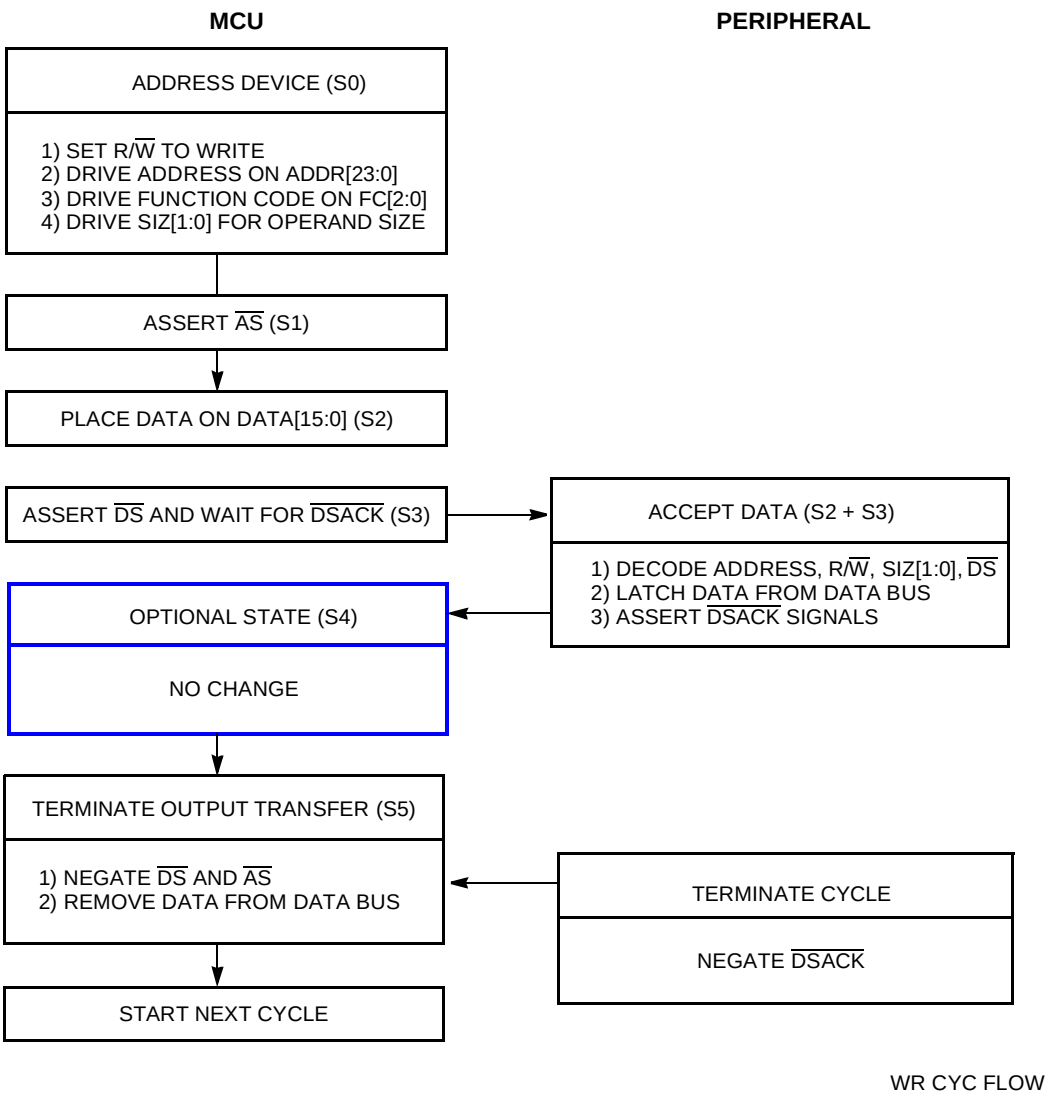


Figure 4-12 Write Cycle Flowchart

4.6.3 Fast Termination Cycles

When an external device can meet fast access timing, the fast termination option of SCIM2E chip selects can provide a two-cycle external bus transfer. Because the chip-select circuits are driven from the system clock, bus cycle termination is inherently synchronized with the system clock.

If multiple chip selects are to be used to provide control signals to a single device and match conditions can occur simultaneously, all MODE, STRB, and associated DSACK fields must be programmed to the same value. This prevents a conflict on the internal bus when the wait states are loaded into the DSACK counter shared by all chip-selects.

Fast termination cycles use internal handshaking signals generated by the chip-select logic. To initiate a transfer, the MCU drives the address bus and the SIZ[1:0] signals. When $\overline{AS}$, $\overline{DS}$, and R/$\overline{W}$ are valid, a peripheral device either places data on the bus (read cycle) or latches data from the bus (write cycle). At the appropriate time, chip-select logic asserts the $\overline{DSACK}$[1:0] signals.

The $\overline{DSACK}$ field in the chip-select option registers determine whether internally generated $\overline{DSACK}$ or externally generated $\overline{DSACK}$ is used. For fast termination cycles, the fast termination encoding (0b1110) must be used. Refer to **4.6.3 Fast Termination Cycles** for information about fast termination setup.

The external $\overline{DSACK}$ lines are always active, regardless of the setting of the $\overline{DSACK}$ field in the chip-select option registers. Thus, an external $\overline{DSACK}$ can always terminate a bus cycle. Holding a $\overline{DSACK}$ line low will cause essentially all external bus cycles to be three-cycle (zero wait states) accesses unless the chip-select option register specifies fast termination accesses.

To use fast termination, an external device must be fast enough to have data ready within the specified setup time (for example, by the falling edge of S4). Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for information about fast termination timing.

When a fast termination cycle is issued, $\overline{DS}$ is asserted for reads but not for writes. The STRB field in the chip-select option register used must be programmed with the address strobe encoding to assert the chip-select signal for a fast termination write.

4.6.4 CPU Space Cycles

Function code signals FC[2:0] designate which of eight external address spaces is accessed during a bus cycle. Address space 7 is designated as CPU space. CPU space is used for control information not normally associated with read or write bus cycles. Function codes are valid only while $\overline{AS}$ is asserted. Refer to **4.5.1.7 Function Codes** for more information on codes and encoding.

During a CPU space access, ADDR[19:16] are encoded to reflect the type of access being made. Three encodings are used by the MCU, as shown in **Figure 4-13**. These encodings represent breakpoint acknowledge (type 0x0) cycles, low power stop broadcast (type 0x3) cycles, and interrupt acknowledge (type 0xF) cycles. Type 0x0 and type 0x3 cycles are discussed in the following paragraphs. Refer to **4.8 Interrupts** for information about interrupt acknowledge bus cycles.

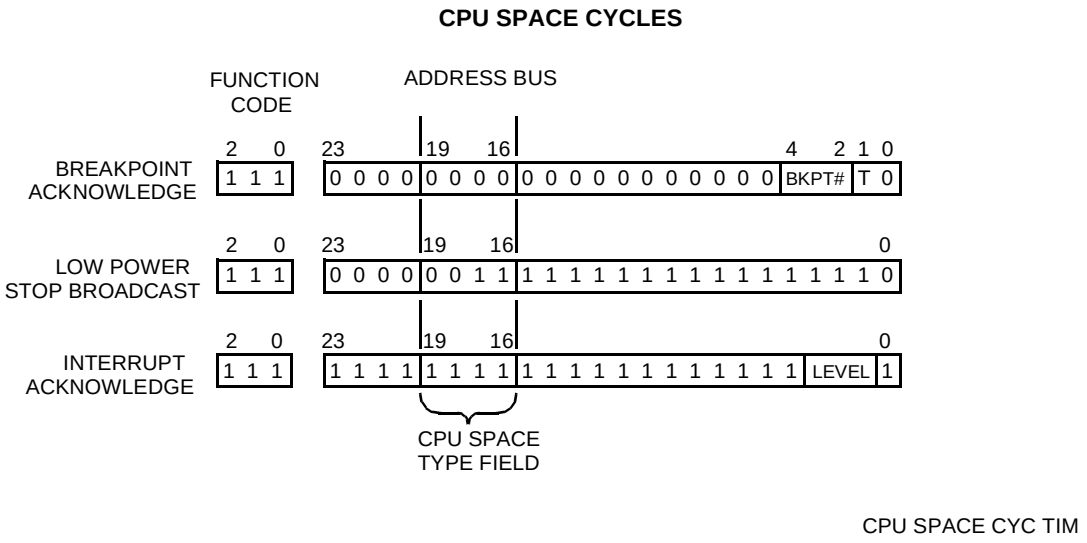


Figure 4-13 CPU Space Address Encoding

4.6.4.1 Breakpoint Acknowledge Cycle

Breakpoints stop program execution at a predefined point during system development. Breakpoints can be used alone or in conjunction with background debug mode. On the MC68F375 microcontroller, both hardware and software can initiate breakpoints.

The CPU32 BKPT instruction allows breakpoints to be inserted through software. The CPU32 responds to this instruction by initiating a breakpoint acknowledge read cycle in CPU space. It places the breakpoint acknowledge (0b0000) code on ADDR[19:16], the breakpoint number (bits [2:0] of the BKPT opcode) on ADDR[4:2], and 0b0 (indicating a software breakpoint) on ADDR1.

External breakpoint circuitry must decode the function code and address lines and responds either by asserting $\overline{\text{BERR}}$ or placing an instruction word on the data bus and asserting $\overline{\text{DSACK}}$. If the bus cycle is terminated by $\overline{\text{DSACK}}$, the CPU32 reads the instruction on the data bus and inserts the instruction into the pipeline. (For 8-bit ports, this instruction fetch may require two read cycles.)

If the bus cycle is terminated by $\overline{\text{BERR}}$, the CPU32 performs illegal instruction exception processing. The CPU32 acquires the number of the illegal instruction exception vector, computes the vector address from this number, loads the content of the vector address into the PC, and jumps to the exception handler routine at that address.

Assertion of the $\overline{\text{BKPT}}$ input initiates a hardware breakpoint. The CPU32 responds by initiating a breakpoint acknowledge read cycle in CPU space. The CPU32 places the breakpoint acknowledge code of 0b0000 on ADDR[19:16], the breakpoint number value of 0b111 on ADDR[4:2], and ADDR1 is set to 0b1, indicating a hardware breakpoint.

External breakpoint circuitry must decode the function code and address lines, place an instruction word on the data bus, and assert $\overline{\text{BERR}}$. The CPU32 then performs hardware breakpoint exception processing: it acquires the number of the hardware breakpoint exception vector, computes the vector address from this number, loads the content of the vector address into the PC, and jumps to the exception handler routine at that address. If the external device asserts $\overline{\text{DSACK}}$ rather than $\overline{\text{BERR}}$, the CPU32 ignores the breakpoint and continues processing.

When $\overline{\text{BKPT}}$ assertion is synchronized with an instruction prefetch, processing of the breakpoint exception occurs at the end of that instruction. The prefetched instruction is “tagged” with the breakpoint when it enters the instruction pipeline. The breakpoint exception occurs after the instruction executes. If the pipeline is flushed before the tagged instruction is executed, no breakpoint occurs. When $\overline{\text{BKPT}}$ assertion is synchronized with an operand fetch, exception processing occurs at the end of the instruction during which $\overline{\text{BKPT}}$ is latched.

Refer to the ***CPU32 Reference Manual* (CPU32RM/AD)** and the ***SCIM Reference Manual* (SCIMRM/AD)** for additional information. Breakpoint operation flow for the CPU32 is shown in **Figure 4-14**.

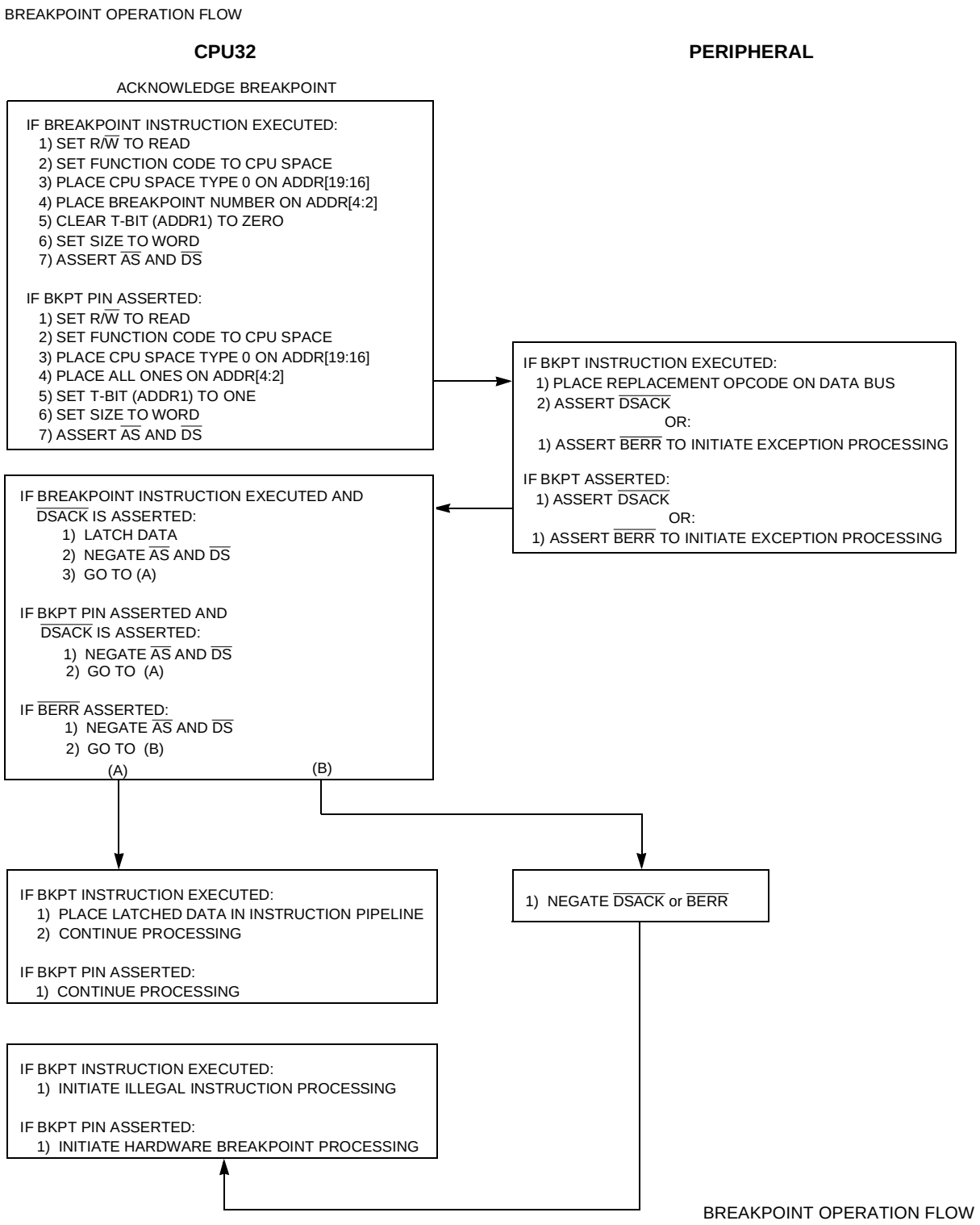


Figure 4-14 Breakpoint Operation Flowchart

4.6.4.2 LPSTOP Broadcast Cycle

Low power stop mode is initiated by the CPU32. Individual modules can be stopped by setting the STOP bits in each module configuration register. The SCIM2E can turn off system clocks after execution of the LPSTOP instruction. When the CPU32 executes LPSTOP, a low power stop broadcast cycle is generated. The SCIM2E brings the MCU out of low power mode when either a reset or an interrupt of higher priority than the interrupt mask level in the CPU32 condition code register occurs.

Refer to **4.3.8.6 Low Power Operation** and **SECTION 3 CENTRAL PROCESSOR UNIT** for more information.

During an LPSTOP broadcast cycle, the CPU32 performs a CPU space write to address 0x3FFFE. This write puts a copy of the interrupt mask value in the clock control logic. The mask is encoded on the data bus as shown in **Figure 4-15**.

The LPSTOP CPU space cycle is shown externally (if the bus is available) as an indication to external devices that the MCU is going into low power stop mode. The SCIM2E provides an internally generated $\overline{\text{DSACK}}$ response to this cycle. The timing of this bus cycle is the same as for a fast termination write cycle. If the bus is not available (arbitrated away), the LPSTOP broadcast cycle is not shown externally.

NOTE

$\overline{\text{BERR}}$ assertion during the LPSTOP broadcast cycle is ignored.

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	IP MASK		

LPSTOP MASK LEVEL

Figure 4-15 LPSTOP Interrupt Mask Encoding on DATA[15:0]

4.6.5 Bus Exception Control Cycles

An external device or a chip-select circuit must assert at least one of the $\overline{\text{DSACK}}$[1:0] signals or the $\overline{\text{AVEC}}$ signal to terminate a bus cycle normally. Bus exception control cycles are used when bus cycles are not terminated in the expected manner.

Acceptable bus cycle termination sequences are summarized as follows. The case numbers refer to **Table 4-21**, which indicates the results of each type of bus cycle termination.

- Normal Termination
  — $\overline{\text{DSACK}}$ is asserted; $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ remain negated (case 1).
- Halt Termination
  — $\overline{\text{HALT}}$ is asserted at the same time or before $\overline{\text{DSACK}}$, and $\overline{\text{BERR}}$ remains negated (case 2).
- Bus Error Termination
  — $\overline{\text{BERR}}$ is asserted in lieu of, at the same time as, or before $\overline{\text{DSACK}}$ (case 3),

or after $\overline{DSACK}$ (case 4), and $\overline{HALT}$ remains negated; $\overline{BERR}$ is negated at the same time or after $\overline{DSACK}$.

- Retry Termination
  - — $\overline{HALT}$ and $\overline{BERR}$ are asserted in lieu of, at the same time as, or before $\overline{DSACK}$ (case 5) or after $\overline{DSACK}$ (case 6); $\overline{BERR}$ is negated at the same time or after $\overline{DSACK}$; $\overline{HALT}$ may be negated at the same time or after $\overline{BERR}$.

Table 4-21 shows various combinations of control signal sequences and the resulting bus cycle terminations.

Table 4-21 $\overline{DSACK}$, $\overline{BERR}$, and $\overline{HALT}$ Assertion Results

Case Number	Control Signal	Asserted on Rising Edge of State		Result
		N[1]	N + 2	
1	$\overline{DSACK}$ $\overline{BERR}$ $\overline{HALT}$	A[2] NA[3] NA	S[4] NA X[5]	Normal termination.
2	$\overline{DSACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA A/S	S NA S	Halt termination: normal cycle terminate and halt. Continue when $\overline{HALT}$ is negated.
3	$\overline{DSACK}$ $\overline{BERR}$ $\overline{HALT}$	NA/A A NA	X S X	Bus error termination: terminate and take bus error exception, possibly deferred.
4	$\overline{DSACK}$ $\overline{BERR}$ $\overline{HALT}$	A A NA	X S NA	Bus error termination: terminate and take bus error exception, possibly deferred.
5	$\overline{DSACK}$ $\overline{BERR}$ $\overline{HALT}$	NA/A A A/S	X S S	Retry termination: terminate and retry when $\overline{HALT}$ is negated.
6	$\overline{DSACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA NA	X A A	Retry termination: terminate and retry when $\overline{HALT}$ is negated.

NOTES:
1. N = The number of current even bus state (S2, S4, etc.).
2. A = Signal is asserted in this bus state.
3. NA = Signal is not asserted in this state.
4. X = Don't care.
5. S = Signal was asserted in previous state and remains asserted in this state.

To control termination of a bus cycle for a retry or a bus error condition properly, $\overline{DSACK}$, $\overline{BERR}$, and $\overline{HALT}$ must be asserted and negated with the rising edge of CLKOUT. This ensures that when two signals are asserted simultaneously, the required setup time and hold time for both of them are met for the same falling edge of the MCU clock. Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for timing requirements. External circuitry that provides these signals must be designed with these constraints in mind, or else the internal bus monitor must be used.

$\overline{DSACK}$, $\overline{BERR}$, and $\overline{HALT}$ may be negated after $\overline{AS}$ is negated.

WARNING

If $\overline{\text{DSACK}}$ or $\overline{\text{BERR}}$ remain asserted into S2 of the next bus cycle, that cycle may be terminated prematurely.

4.6.5.1 Bus Errors

The CPU32 treats bus errors as a type of exception. Bus error exception processing begins when the CPU32 detects assertion of the IMB $\overline{\text{BERR}}$ signal (by the internal bus monitor or an external source) while the $\overline{\text{HALT}}$ signal remains negated.

$\overline{\text{BERR}}$ assertions do not force immediate exception processing. The signal is synchronized with normal bus cycles and is latched into the CPU32 at the end of the bus cycle in which it was asserted. Because bus cycles can overlap instruction boundaries, bus error exception processing may not occur at the end of the instruction in which the bus cycle begins. Timing of $\overline{\text{BERR}}$ detection/acknowledge is dependent upon several factors:

- Which bus cycle of an instruction is terminated by assertion of $\overline{\text{BERR}}$.
- The number of bus cycles in the instruction during which $\overline{\text{BERR}}$ is asserted.
- The number of bus cycles in the instruction following the instruction in which $\overline{\text{BERR}}$ is asserted.
- Whether $\overline{\text{BERR}}$ is asserted during a program space access or a data space access.

Because of these factors, it is impossible to predict precisely how long after occurrence of a bus error the bus error exception is processed.

CAUTION

The external bus interface does not latch data when an external bus cycle is terminated by a bus error. When this occurs during an instruction prefetch, the IMB precharge state (bus pulled high, or 0xFFFF) is latched into the CPU32 instruction register, with indeterminate results.

4.6.5.2 Double Bus Faults

Exception processing for bus error exceptions follows the standard exception processing sequence. Refer to **3.9 Exception Processing** for more information. However, a special case of bus error, called double bus fault, can abort exception processing.

$\overline{\text{BERR}}$ assertion is not detected until an instruction is complete. The $\overline{\text{BERR}}$ latch is cleared by the first instruction of the $\overline{\text{BERR}}$ exception handler. Double bus fault occurs in three ways:

1. When bus error exception processing begins and a second $\overline{\text{BERR}}$ is detected before the first instruction of the exception handler is executed.
2. When one or more bus errors occur before the first instruction after a reset exception is executed.
3. A bus error occurs while the CPU32 is loading information from a bus error

stack frame during a return from exception (RTE) instruction.

Multiple bus errors within a single instruction that can generate multiple bus cycles cause a single bus error exception after the instruction has been executed.

Immediately after assertion of a second $\overline{\text{BERR}}$, the MCU halts and drives the $\overline{\text{HALT}}$ line low. Only a reset can restart a halted MCU. However, bus arbitration can still occur. Refer to **4.6.6 External Bus Arbitration** for more information. A bus error or address error that occurs after exception processing has been completed (during the execution of the exception handler routine, or later) does not cause a double bus fault. The MCU continues to retry the same bus cycle as long as the external hardware requests it.

4.6.5.3 Retry Operation

When an external device asserts $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ during a bus cycle, the MCU enters the retry sequence. A delayed retry can also occur. The MCU terminates the bus cycle, places the $\overline{\text{AS}}$ and $\overline{\text{DS}}$ signals in their inactive state, and does not begin another bus cycle until the $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ signals are negated by external logic. After a synchronization delay, the MCU retries the previous cycle using the same address, function codes, data (for a write), and control signals. The $\overline{\text{BERR}}$ signal should be negated before S2 of the read cycle to ensure correct operation of the retried cycle.

If $\overline{\text{BR}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ are all asserted on the same cycle, the EBI will enter the rerun sequence but first relinquishes the bus to an external master. Once the external master returns the bus and negates $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$, the EBI runs the previous bus cycle. This feature allows an external device to correct the problem that caused the bus error and then try the bus cycle again.

The MCU retries any read or write cycle of an indivisible read-modify-write operation separately. $\overline{\text{RMC}}$ remains asserted during the entire retry sequence. The MCU will not relinquish the bus while $\overline{\text{RMC}}$ is asserted. Any device that requires the MCU to give up the bus and retry a bus cycle during a read-modify-write cycle must assert $\overline{\text{BERR}}$ and $\overline{\text{BR}}$ only ($\overline{\text{HALT}}$ must remain negated). The bus error handler software should examine the read-modify-write bit in the special status word and take the appropriate action to resolve this type of fault when it occurs. Refer to the ***SCIM Reference Manual* (SCIMRM/AD)** for additional information on read-modify-write and retry operations.

4.6.5.4 Halt Operation

When $\overline{\text{HALT}}$ is asserted while $\overline{\text{BERR}}$ is not asserted, the MCU halts external bus activity after negation of $\overline{\text{DSACK}}$. The MCU may complete the current word transfer in progress. For a long-word to byte transfer, this could be after S2 or S4. For a word to byte transfer, activity ceases after S2.

Negating and reasserting $\overline{\text{HALT}}$ according to timing requirements provides single-step (bus cycle to bus cycle) operation. The $\overline{\text{HALT}}$ signal affects external bus cycles only, so that a program that does not use the external bus can continue executing.

During dynamically-sized 8-bit transfers, external bus activity may not stop at the next cycle boundary. Occurrence of a bus error while $\overline{\text{HALT}}$ is asserted causes the CPU32 to initiate a retry sequence.

When the MCU completes a bus cycle while the $\overline{\text{HALT}}$ signal is asserted, the data bus goes into a high-impedance state and the $\overline{\text{AS}}$ and $\overline{\text{DS}}$ signals are driven to their inactive states. Address, function code, size, and read/write signals remain in the same state.

The halt operation has no effect on bus arbitration. However, when external bus arbitration occurs while the MCU is halted, address and control signals go into a high-impedance state. If $\overline{\text{HALT}}$ is still asserted when the MCU regains control of the bus, address, function code, size, and read/write signals revert to the previous driven states. The MCU cannot service interrupt requests while halted.

4.6.6 External Bus Arbitration

The MCU bus design provides for a single bus master at any one time. Either the MCU or an external device can be master. Bus arbitration protocols determine when an external device can become bus master. Bus arbitration requests are recognized during normal processing, $\overline{\text{HALT}}$ assertion, and when the CPU32 has halted due to a double bus fault.

The MCU bus controller manages bus arbitration signals so that the MCU has the lowest priority. External devices that need to obtain the bus must assert 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 protocol sequence for assuming bus mastership from the MCU is:

1. An external device asserts the bus request signal ($\overline{\text{BR}}$).
2. The MCU asserts the bus grant signal ($\overline{\text{BG}}$) to indicate that the bus is available.
3. An external device asserts the bus grant acknowledge ($\overline{\text{BGACK}}$) signal to indicate that it has assumed bus mastership.

$\overline{\text{BR}}$ can be asserted 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 operand transfer.

If more than one external device can be bus master, required external arbitration must begin when a requesting device receives $\overline{\text{BG}}$. An external device must assert $\overline{\text{BGACK}}$ when it assumes mastership, and must maintain $\overline{\text{BGACK}}$ assertion as long as it is bus master.

Two conditions must be met for an external device to assume bus mastership. The device must receive $\overline{\text{BG}}$ through the arbitration process, and $\overline{\text{BGACK}}$ must be inactive,

indicating that no other bus master is active. This technique allows the processing of bus requests during data transfer cycles.

$\overline{BG}$ is negated a few clock cycles after $\overline{BGACK}$ transition. However, if bus requests are still pending after $\overline{BG}$ is negated, the MCU asserts $\overline{BG}$ again within a few clock cycles. This additional $\overline{BG}$ assertion allows external arbitration circuitry to select the next bus master before the current master has released the bus.

Refer to **Figure 4-16** which shows bus arbitration for a single device. The flow chart shows $\overline{BR}$ negated at the same time $\overline{BGACK}$ is asserted. Refer to the ***SCIM Reference Manual* (SCIMRM/AD)** for more information on bus arbitration.

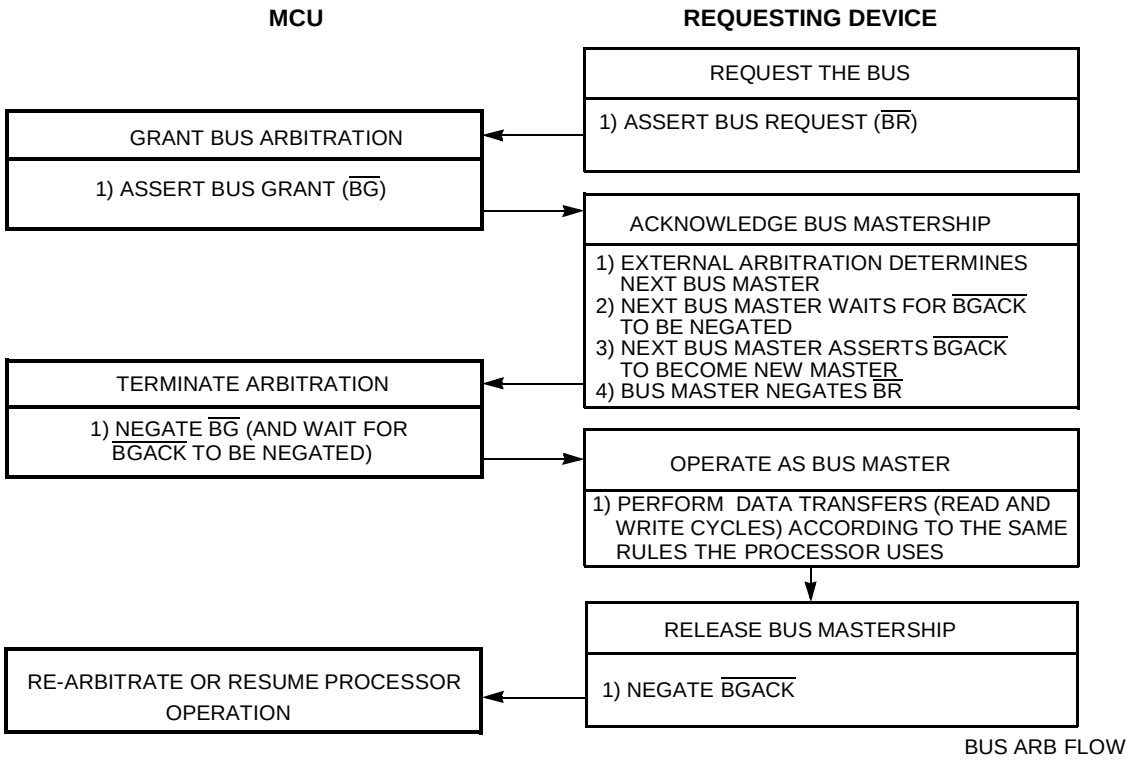


Figure 4-16 Bus Arbitration Flowchart for Single Request

4.6.6.1 Show Cycles

The MCU normally performs internal data transfers without affecting the external data bus, but it is possible to show these transfers during debugging. $\overline{AS}$ is not asserted externally during show cycles.

Show cycles are controlled by SHEN[1:0] in SCIMMCR. This field set to 0b00 by reset. When show cycles are disabled, the address bus, function codes, size, and read/write signals reflect internal bus activity, but $\overline{AS}$ and $\overline{DS}$ are not asserted externally and external data bus pins are in a high-impedance state during internal accesses. Refer to **4.2.5 Show Internal Cycles** and the ***SCIM Reference Manual* (SCIMRM/AD)** for more information.

When show cycles are enabled, $\overline{DS}$ is asserted externally during internal cycles, and internal data is driven out on the external data bus. Because internal cycles normally continue to run when the external bus is granted, one SHEN[1:0] encoding halts internal bus activity while there is an external master.

The SIZ[1:0] signals reflect bus allocation during show cycles. Only the appropriate portion of the data bus is valid during the cycle. During a byte write to an internal address, the portion of the bus that represents the byte that is not written reflects internal bus conditions, and is indeterminate. During a byte write to an external address, the data multiplexer in the SCIM2E drives the value of the byte that is written to both halves of the data bus.

4.7 Reset

Reset occurs when an active low logic level on the $\overline{RESET}$ pin is clocked into the SCIM2E. The $\overline{RESET}$ input is synchronized to the system clock. If there is no clock when $\overline{RESET}$ is asserted, reset does not occur until the clock starts. Resets are clocked to allow completion of write cycles in progress at the time $\overline{RESET}$ is asserted.

Reset procedures handle system initialization and recovery from catastrophic failure. The MCU performs resets with a combination of hardware and software. The SCIM2E determines whether a reset is valid, asserts control signals, performs basic system configuration and boot memory selection based on hardware mode-select inputs, then passes control to the CPU32.

4.7.1 SCIM2E Reset Control Logic

The SCIM2E reset control logic differs somewhat from the SCIM. Do not use information in SCIM documentation for the SCIM2E. As with the SCIM, the asserted state of the external reset pin is ‘0’. The released state is ‘1’. The external circuit must make some provision to pull the reset pin high (usually, just a pullup resistor and a capacitor to ground) when the SCIM2’s reset controller releases reset. The specified maximum time in which external circuit must release the reset pin (the input signal must be at or above VIH) has been extended by approximately 180 clock cycles (22.5 µs at eight MHz for slow reference clock mode, 45 µs at four MHz for fast reference clock mode, for more information about clock modes, see **4.3 System Clock**.

For the SCIM, reset must rise to VIH within 10 clocks (1.25 µs at eight MHz). For applications that meet the current requirements of the SCIM’s reset timing, no apparent change in reset timing will occur on the SCIM2E. Reset vector fetch and code execution for the SCIM2E will begin after the 10th clock as it does on the SCIM. Applications that do not pull reset high by the end of the first 10 clocks are effectively given one more chance to have properly released reset by the 190th clock (180 clocks after the end of the initial 10-clock period). If reset has not been released by this time, reset is re-asserted by the SCIM2’s reset controller for 512 clocks, reset configuration is again latched from the data bus, and the sequence is repeated.

4.7.1.1 SCIM2E Reset Control Flow

The reset control flow for SCIM2 is depicted in **Figure 4-17**. This is the same as the original flow with one exception: After the external system is given the 10 clocks to pull reset high, the reset pin is sampled. If it is not high, the external system is given one more opportunity to pull it high. This time period is 180 clocks long. At the end of this second period, if the external system has negated reset, code execution begins. Otherwise, the reset controller loops back into the 512-clock hard reset sequence after waiting for the reset pin to go high. The boxes that have been added to the original reset control flow are shaded for easy identification.

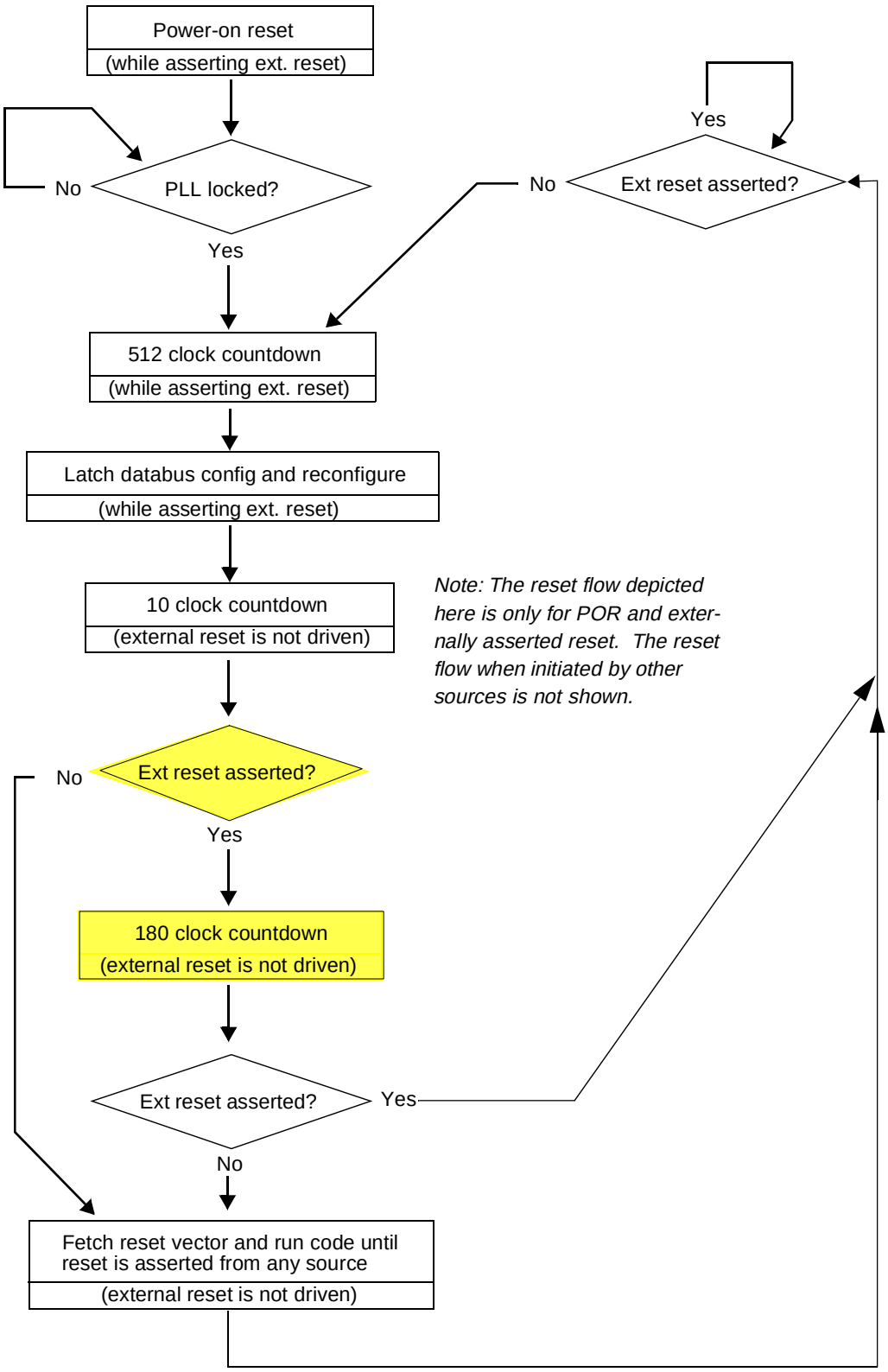


Figure 4-17 SCIM2 Reset Control Flow

4.7.2 Reset Exception Processing

The CPU32 processes resets as a type of asynchronous exception. An exception is an event that preempts normal processing and can be caused by internal or external events. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with an exception. Each exception has an assigned vector that points to an associated handler routine. These vectors are stored in the exception vector table. The exception vector table consists of 256 four-byte vectors and occupies 1024 bytes of address space. The CPU32 uses vector numbers to calculate displacement into the table. Refer to **3.9 Exception Processing** for more information.

Reset is the highest-priority CPU32 exception. Unlike all other exceptions, a reset occurs at the end of a bus cycle and not at an instruction boundary. Handling resets in this way prevents write cycles in progress at the time the reset signal is asserted from being corrupted. However, any processing in progress is aborted by the reset exception and cannot be restarted. Only essential reset tasks are performed during exception processing. Other initialization tasks must be accomplished by the exception handler routine.

NOTE

External circuitry is required to disable external bus configuration logic until $\overline{DS}$ and R/$\overline{W}$ are negated to ensure that bus cycles in progress at the time $\overline{RESET}$ is asserted complete correctly.

4.7.3 Reset Source Summary

SCIM2E reset control logic determines the cause of a reset, synchronizes request signals to CLKOUT, and asserts reset control logic. All resets are gated by CLKOUT. Asynchronous resets can occur on any clock edge and are assumed to be catastrophic. Synchronous resets are timed to occur at the end of bus cycles. When a synchronous reset is detected, the SCIM2E bus monitor is automatically enabled. If the bus cycle during which a synchronous reset is detected does not terminate normally, the bus monitor will terminate the cycle and allow the reset to proceed. **Table 4-22** is a summary of reset sources.

Table 4-22 Reset Source Summary

Type	Source	Timing
External	Assertion of $\overline{RESET}$ pin	Synchronous
Power on	Rising voltage on V_{DD}	Asynchronous
Software watchdog	Timeout of software watchdog	Asynchronous
Halt	Halt monitor (e.g. double bus fault)	Asynchronous
Loss of clock	Reference failure caught by loss of clock detector	Synchronous
Test	Test submodule	Synchronous

Internal byte and aligned word write cycles are guaranteed valid for synchronous resets. External writes will also complete uncorrupted, provided the data bus is conditioned with a circuit that incorporates $\overline{\text{RESET}}$, such as that shown in **Figure 4-19**.

4.7.4 Reset Status Register

The reset status register (RSR) contains a bit for each reset source in the MCU. When a reset occurs, a bit corresponding to the reset type is set. When multiple causes of reset occur at the same time, more than one bit in RSR may be set. The reset status register is updated by the reset control logic when $\overline{\text{RESET}}$ is released.

RSR — Reset Status Register **0xYF FA06**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
Reserved								EXT	POW	SW	HLT	0	Re-served	SYS	TST

Table 4-23 RSR Bit Descriptions

Bit(s)	Name	Description
15:8	—	Reserved
7	EXT	External reset. Reset caused by the $\overline{\text{RESET}}$ pin.
6	POW	Power-up reset. Reset caused by the power-up reset circuit.
5	SW	Software watchdog reset. Reset caused by the software watchdog circuit.
4	HLT	Halt monitor reset. Reset caused by the halt monitor.
3:2	—	Reserved
1	SYS	System reset. Reset caused by a RESET instruction.
0	TST	Test submodule reset. Reset caused by the test submodule. Used during factory test reserved operating mode only.

This register can be read at any time; a write has no effect. Bits [15:8] are reserved and always read zero.

4.7.5 Reset Timing

When an external device asserts the $\overline{\text{RESET}}$ pin for at least four clock cycles, the signal will be latched and held internally until completion of the current bus cycle. Any further processing of the reset exception is then delayed until the SCIM2E reset control logic detects that the $\overline{\text{RESET}}$ pin is no longer being externally driven. Two clock cycles will elapse (during which time the pullup resistor on $\overline{\text{RESET}}$ will pull the pin high) while the reset control logic switches the $\overline{\text{RESET}}$ pin from an input to an output. $\overline{\text{RESET}}$ will then be driven low for 512 clock cycles.

If a synchronous internal reset is detected (e.g., from the loss of clock detector or the test submodule), the reset control logic will wait for bus cycle completion and then drive $\overline{\text{RESET}}$ low for 512 clock cycles. An asynchronous internal reset (e.g., from the halt monitor or the software watchdog) will immediately drive $\overline{\text{RESET}}$ low for 512 clock cycles without waiting for the current bus cycle to complete.

After the 512-clock cycle assertion of the $\overline{\text{RESET}}$ pin, the processing flow for both internal and external resets is the same. The SCIM2E reset control logic will release the $\overline{\text{RESET}}$ pin and read configuration information from $\overline{\text{BERR}}$, $\overline{\text{BKPT}}$, and DATA[15:0]. Refer to **4.7.8 Operating Configuration Out of Reset** for more information.

Ten clock cycles will elapse to allow the pull-up resistor on $\overline{\text{RESET}}$ to pull the pin high. The reset control logic will then sample the $\overline{\text{RESET}}$ pin. If the pin is high, the reset control logic will release the external bus interface (EBI) and allow the reset vector to be fetched. If $\overline{\text{RESET}}$ is still low, 180 clock cycles will elapse, and the reset control logic will sample the pin again. As above, if $\overline{\text{RESET}}$ is high, processing will resume and the reset vector will be fetched.

If $\overline{\text{RESET}}$ still has not risen to logic one, the reset control logic will begin the external reset sequence as described at the beginning of this section. Further reset exception processing will not proceed until $\overline{\text{RESET}}$ is sampled at logic one after the 10-clock cycle or 180-clock cycle delays described above. **Figure 4-18** depicts the reset sequence for the SCIM2E.

4.7.6 Power-On Reset

Power-on reset (POR) operation involves special circumstances related to the application of system power and, if the PLL is used, clock synthesizer power. V_{DD} ramp time affects pin state during reset. Slow V_{DD} ramp times can leave MCU pins in an indeterminate state longer than is desired or is tolerable in some applications.

When the PLL is used to generate the MCU system clock, oscillator start up time also determines how long MCU pins remain in an indeterminate state. Immediate application of V_{DDSYN}/MODCLK power and careful attention to crystal specifications and oscillator circuit design play an important role in minimizing start up time.

Grounding V_{DDSYN}/MODCLK places the MCU in external clock mode, initially operating at the frequency input on the EXTAL pin. In this case, any events requiring clock cycles during POR will occur as quickly as those clock cycles are input on the EXTAL pin.

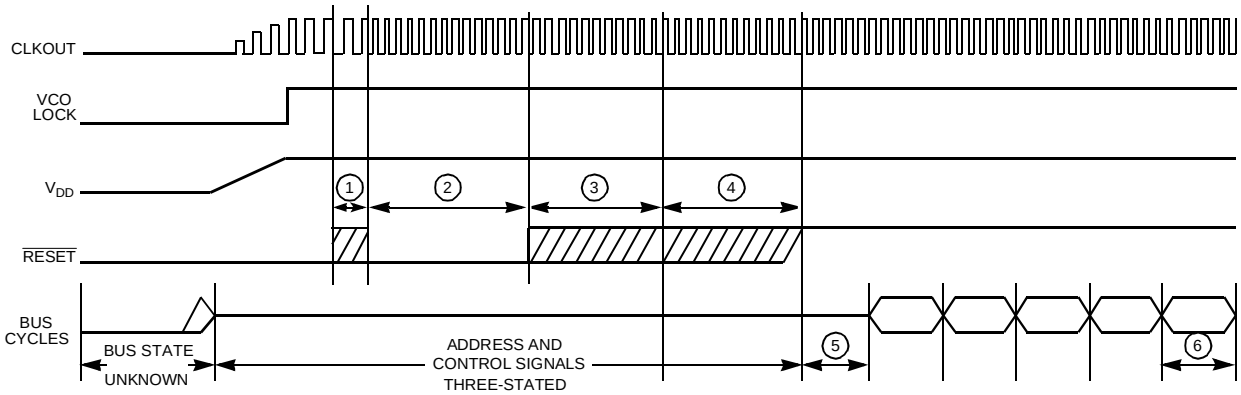
Power-on reset activates a circuit in the SCIM2E that asserts the internal and external $\overline{\text{RESET}}$ lines. As V_{DD} ramps up to the minimum operating voltage, the PLL (if enabled) begins to generate the system clock and the internal $\overline{\text{RESET}}$ line is negated. This initializes SCIM2E pins the values shown in **Table 4-24**.

At this point, POR will proceed no further until the PLL locks at two or 256 times f_{ref} in fast or slow reference modes, respectively. Reset exception processing will then continue as outlined in **4.7.5 Reset Timing.** In external clock mode, the PLL is disabled which permits normal reset exception processing as soon as the internal $\overline{\text{RESET}}$ line is negated.

The SCIM2E propagates $\overline{\text{RESET}}$ and the system clock to all other MCU modules. Once the clock is running and internal $\overline{\text{RESET}}$ is asserted for at least four clock cycles, these modules reset. V_{DD} and PLL ramp up times determine how long these four clock

cycles take. Worst case occurs in slow reference mode and is approximately 15 milliseconds. During this period, MCU pins may be in an indeterminate state. While pull-up resistors may be used on input only pins, active logic will be required to condition input/output or output only pins.

Figure 4-18 depicts the timing of the power-on reset sequence.



NOTES:
1. TWO CLOCK CYCLE DELAY REQUIRED BY RESET CONTROL LOGIC TO SWITCH $\overline{\text{RESET}}$ PIN FROM INPUT TO OUTPUT.
2. RESET CONTROL LOGIC DRIVES THE $\overline{\text{RESET}}$ PIN LOW FOR 512 CLOCK CYCLES TO GUARANTEE THIS LENGTH OF RESET TO THE ENTIRE SYSTEM.
3. TEN CLOCK CYCLE DELAY DURING WHICH THE $\overline{\text{RESET}}$ PIN IS NO LONGER DRIVEN AND IS ALLOWED TO RISE TO A LOGIC ONE. OPERATING CONFIGURATION IS LATCHED FROM DATA[15:0] AND $\overline{\text{BERR}}$ WHEN THIS DELAY BEGINS.
4. ADDITIONAL 180 CLOCK DELAY PROVIDED BY RESET CONTROL LOGIC TO ALLOW THE RESET PIN TO RISE TO A LOGIC ONE IF IT DID NOT DO SO DURING THE PRIOR 10 CLOCK CYCLES.
5. INTERNAL START-UP TIME.
6. FIRST INSTRUCTION FETCHED.

SCIM2E POR TIM

Figure 4-18 Power-On Reset

4.7.7 Pin State During Reset

It is important to keep the distinction between pin function and pin electrical state clear. Although control register values and mode select inputs determine pin function, a pin driver can be active, inactive, or in a high impedance state when reset occurs. During power-on reset, pin state is subject to the constraints discussed in **4.7.6 Power-On Reset**.

NOTE

Pins that are not used should either be configured as outputs (if possible) or as inputs and be pulled to an appropriate inactive state. This decreases unnecessary current consumption caused by digital inputs floating near mid-supply level.

4.7.7.1 Reset States of SCIM2E Pins

Generally, while $\overline{\text{RESET}}$ is asserted, SCIM2E pins either go into a high-impedance state or are driven to their inactive states. Operating mode selection occurs when

$\overline{\text{RESET}}$ is released. Mode select inputs are driven to the appropriate states at this time. Use an active circuit, such as that shown in **Figure 4-19**, for this purpose. **Table 4-24** shows the state of SCIM2E pins during reset.

Table 4-24 SCIM2E Pin States During Reset

Pin(s)	Pin State During $\overline{\text{RESET}}$
ADDR[2:0]	High-Z
ADDR[10:3]/PB[7:0]	High-Z
ADDR[18:11]/PA]7:0]	High-Z
ADDR[22:19]/$\overline{\text{CS}}$[9:6]/PC[6:3]	V_{DD}
ADDR23/$\overline{\text{CS10}}$/ECLK	V_{DD}
$\overline{\text{AS}}$/PE5	High-Z
$\overline{\text{AVEC}}$/PE2	High-Z
$\overline{\text{BERR}}$	Mode Select Input
$\overline{\text{BG}}$/$\overline{\text{CSM}}$	V_{DD}
$\overline{\text{BGACK}}$/$\overline{\text{CSE}}$	V_{DD}
$\overline{\text{BR}}$/$\overline{\text{CS0}}$	V_{DD}
CLKOUT	Output
$\overline{\text{CSBOOT}}$	V_{DD}
DATA[7:0]/PH[7:0]	Mode Select Inputs
DATA[15:8]/PG[7:0]	Mode Select Inputs
$\overline{\text{DS}}$/PE4	High-Z
$\overline{\text{DSACK0}}$/PE0	High-Z
$\overline{\text{DSACK1}}$/PE1	High-Z
FASTREF/PF0	Mode Select Input
FC0/$\overline{\text{CS3}}$/PC0	V_{DD}
FC1/PC1	V_{DD}
FC2/$\overline{\text{CS3}}$/PC2	V_{DD}
$\overline{\text{HALT}}$	High-Z
$\overline{\text{IRQ}}$[7:1]/PF[7:1]	High-Z
PE3	High-Z
R/$\overline{\text{W}}$	High-Z
$\overline{\text{RESET}}$	Asserted
SIZ[1:0]/PE[7:6]	High-Z
TSC	Three State Enable Input

4.7.7.2 Reset States of Pins Assigned to Other MCU Modules

As a rule, module pins that can be configured for general purpose I/O go into a high-impedance state during reset. However, during power-on reset, module port pins may be in an indeterminate state for a short period of time. Refer to **4.7.6 Power-On Reset** for more information.

4.7.8 Operating Configuration Out of Reset

When $\overline{\text{RESET}}$ is released, the SCIM2E acquires setup information from several MCU pins. Individually or in groups, these pins control the four basic areas of MCU configuration outlined in **Table 4-25**.

Table 4-25 Pins Associated with Basic Configuration Options

Option	Controlling Pins
Background Debug Mode Disable/Enable	$\overline{\text{BKPT}}$
Flash/ROM Module Disable/Enable	DATA[15:12]
Operating Mode Selection	$\overline{\text{BERR}}$, DATA1
SCIM2E I/O Port Configuration	DATA[11:2], DATA0

Clock mode is not listed in **Table 4-25** because it is not selected when $\overline{\text{RESET}}$ is released. Instead, it is latched immediately from V_{DDSYN}/MODCLK and FASTREF/PF0 upon assertion of $\overline{\text{RESET}}$. Refer to **4.3.1 System Clock Sources** for more information.

4.7.8.1 Operating Mode Selection

The logic states of $\overline{\text{BERR}}$ and DATA1 determine MCU operating mode when $\overline{\text{RESET}}$ is released. Care should be taken to guarantee that $\overline{\text{BERR}}$ is driven to a known state during reset. Unlike DATA1 which has a weak pull-up resistor, no conditioning circuitry is present on the $\overline{\text{BERR}}$ pin. If $\overline{\text{BERR}}$ is allowed to float during reset, improper mode determination may occur. Operating mode selection is summarized in **Table 4-26**.

The configuration of the SCIM2E EBI is dependent on operating mode. ADDR[18:3] serve as general purpose I/O ports A and B when the MCU is running in single-chip mode. DATA[7:0] serve as general purpose I/O port H in the single-chip and 8-bit expanded modes, and DATA[15:8] serve as general purpose I/O port G in single-chip mode.

Table 4-26 Mode Configuration During Reset

Pin(s) Affected	Mode Select Pin	Effect of Mode Pin High During RESET	Effect of Mode Pin Low During RESET	Default for 8-Bit Data Bus Mode 11	Default Single Chip Mode 01, 00
A[18:3] D[15:0]	BERR	Expanded	Single Chip	—	A[18:3] = PortA, B D[15:0] = Port G, H
—	MODCK	VCO= System Clock	EXTAL= System Clock	Decode MODCK	Decode MODCK
—	BKPT	BGND Mode Disabled	BGND Mode Enabled	Decode BKPT	Decode BKPT
CSBOOT	D0	16-bit	8-bit	8-bit	8-bit
D[7:0]	D1	8-Bit Data Bus	16-Bit Data Bus	D[7:0] = Port H	Ignored
BR/CS0 FC0/CS3/PC0 FC1/PC1 FC2/CS5/PC2	D2	CS0 CS3 FC1 CS5	BR FC0 FC1 FC2	CS0 CS3 FC1 CS5	CS0 PC0 PC1 PC2
A19/CS6/PC3 A20/CS7/PC4 A21/CS8/PC5 A22/CS9/PC6 A23/CS10/E	D3-D7 D4-D7 D5-D7 D6–D7 D7	CS6 CS7 CS8 CS9 CS10	A19 A20 A21 A22 A23	CS6 CS7 CS8 CS9 CS10	PC3 PC4 PC5 PC6 CS10
DSACK0/PE0 DSACK1/PE1 AVEC/PE2 RMC/PE3 DS/PE4 AS/PE5 SIZ0/PE6 SIZ1/PE7	D8	DSACK0 DSACK1 AVEC RMC DS AS SIZ0 SIZ1	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7	— (Decode D8) — — — — — — —	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
FASTREF/PF0 IRQ1/PF1 IRQ2/PF2 IRQ3/PF3 IRQ4/PF4 IRQ5/PF5 IRQ6/PF6 IRQ7/PF7	D9	MODCK IRQ1 IRQ2 IRQ3 IRQ4 IRQ5 IRQ6 IRQ7	PF0 PF1 PF2 PF3 PF4 PF5 PF6 PF7	— (Decode D9) — — — — — — —	PF0 PF1 PF2 PF3 PF4 PF5 PF6 PF7
BGACK/CSE BG/CSM	D10 D11 D12-15				
	D10	BGACK/BG Emulator mode disabled	CSE/CSM Emulator mode enabled	BGACK/BG Emulator mode disabled	BGACK/BG Emulator mode disabled
	D11	Slave mode disabled	Slave mode enabled	Slave mode disabled	Slave mode disabled
	D12	Not used			
	D13	CMFI and ROM EMUL enable (high = enabled, low = disabled)			
	D14	ROM STOP (high = enabled, low = disabled)			
	D15	CMFI STOP (high = enabled, low = disabled)			

In single-chip mode, the default setting of the address bus disable (ABD) bit places ADDR[2:0] in a high-impedance state to reduce noise emissions. ADDR[2:0] function

as normal address bus pins in expanded operating modes. Refer to **4.2.4 Noise Reduction in Single-Chip Mode** and **APPENDIX A INTERNAL MEMORY MAP** for information on the address bus disable bit (ABD).

The ADDR[23:19] pins have multiple functions as high-order address lines, chip selects, or discrete outputs and are configured differently depending on operating mode selection and data bus conditioning when $\overline{\text{RESET}}$ is released. The following paragraphs contain a summary of pin configuration options for each external bus configuration.

4.7.8.2 Data Bus Mode Selection

DATA[15:0] have weak internal pull-up devices. When pins are held high by the internal pull-ups, the MCU uses a default operating configuration. Specific lines can be held low externally during reset to achieve alternate configurations.

NOTE

External bus loading can overcome the weak internal pull-up devices on data bus lines and hold pins low during reset.

Use an active device to properly configure data bus lines while $\overline{\text{RESET}}$ is low. Data bus configuration logic must release the bus before the first bus cycle after reset to prevent conflict with external memory devices. The first bus cycle occurs ten CLKOUT cycles after $\overline{\text{RESET}}$ is released. If external mode selection logic causes a conflict of this type, an isolation resistor on the driven lines may be required. **Figure 4-19** shows a recommended method for conditioning the data bus mode select signals.

The mode configuration drivers are conditioned with R/$\overline{\text{W}}$ and $\overline{\text{DS}}$ to prevent conflicts between external devices and the MCU when $\overline{\text{RESET}}$ is asserted. If $\overline{\text{RESET}}$ is asserted during an external write cycle, R/$\overline{\text{W}}$ conditioning (as shown in **Figure 4-19**) prevents corruption of the data during the write. Similarly, $\overline{\text{DS}}$ conditions the mode configuration drivers so that external reads are not corrupted when $\overline{\text{RESET}}$ is asserted during an external read cycle.

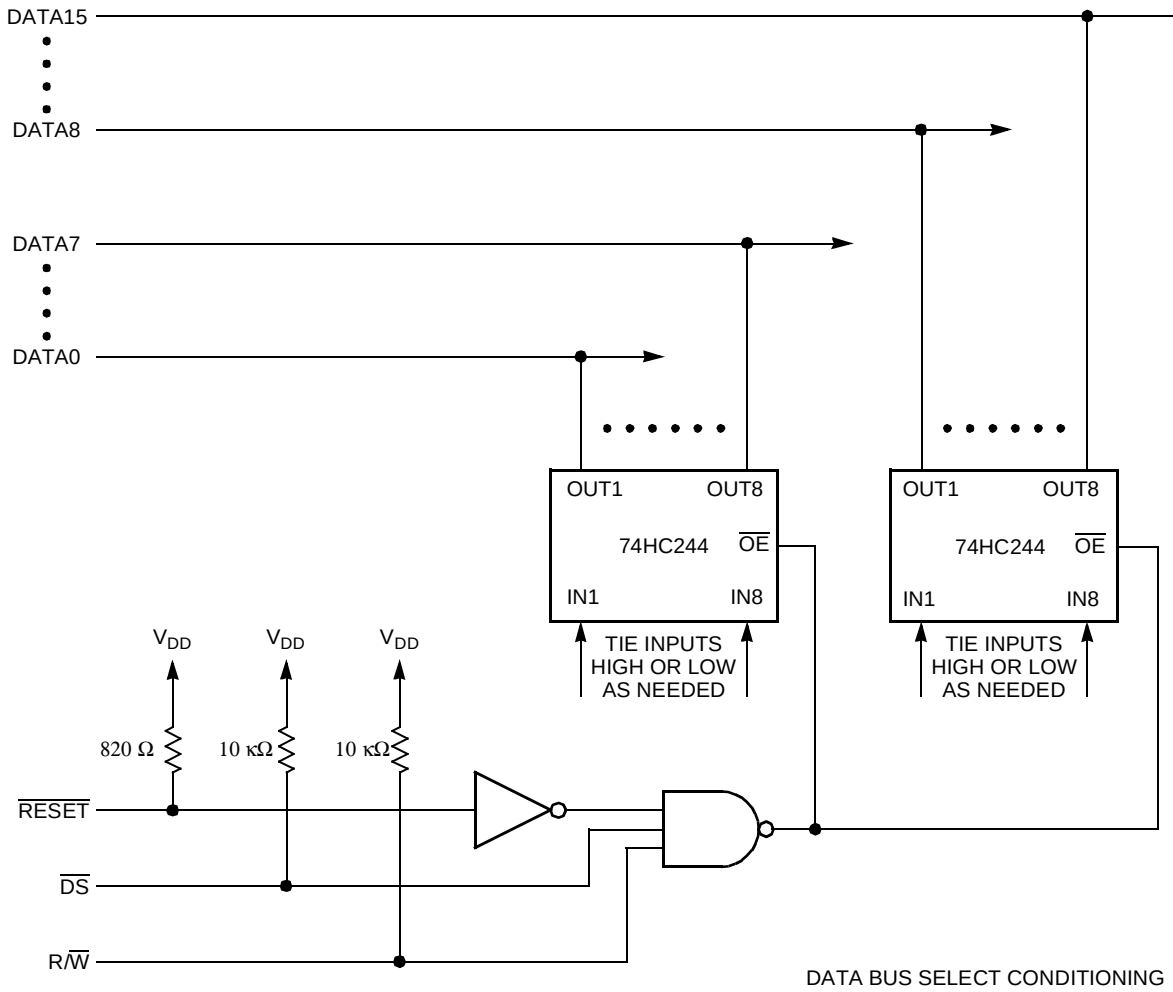


Figure 4-19 Preferred Circuit for Data Bus Mode Select Conditioning

Alternate methods can be used for driving data bus pins low during reset. **Figure 4-20** shows two of these options.

NOTE

These simpler circuits do not offer the protection from potential memory corruption during $\overline{\text{RESET}}$ assertion as does the circuit shown in **Figure 4-19**.

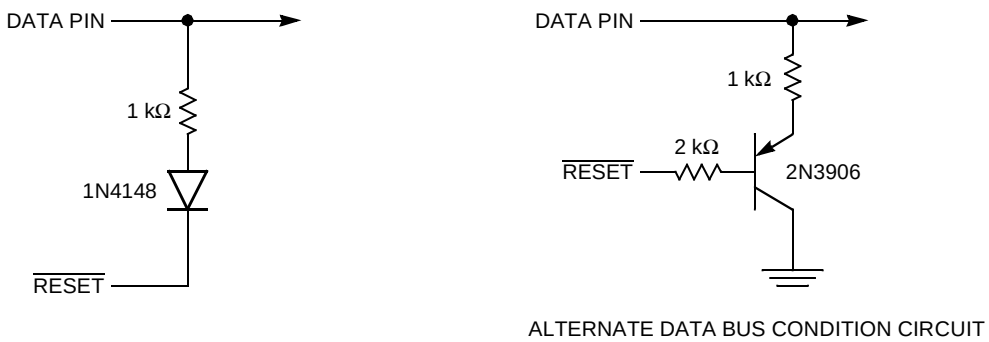


Figure 4-20 Alternate Circuit for Data Bus Mode Select Conditioning

In the simpler of these two circuits, a resistor is connected in series with a diode from the data bus pin to the RESET line. A bipolar transistor can be used for the same purpose, but an additional current limiting resistor must be connected between the base of the transistor and the RESET pin. If a MOSFET is substituted for the bipolar transistor, only the 1 KΩ isolation resistor is required.

4.7.8.3 Single-Chip Mode

When BERR = 0 at the release of RESET, single-chip operation is selected. BERR must return to a logic 1 before the first bus cycle after reset to insure proper device operation. The external bus interface is essentially disabled in single-chip mode, and SCIM2E pins generally serve as discrete inputs and outputs. The behavior of specific pin groups is discussed in the following paragraphs.

NOTE

The paragraphs that follow describe the behavior of SCIM2E pins in single-chip mode only. Sections that follow cover 16-bit and 8-bit expanded modes.

ADDR[2:0] have no discrete I/O function in single-chip mode. These pins are placed in a high-impedance state at power-on but can be enabled by clearing the ABD bit in SCIMMCR.

ADDR[18:11] become port A input/output pins PA[7:0], and ADDR[10:3] become port B input/output pins PB[7:0]. Each port is configurable entirely as inputs or outputs on a per port basis by the DDA and DDB bits in the port A/B data direction register (DDRAB).

Special attention should be paid to chip-select pins in single-chip mode. While each chip-select base address register and option register is active and may be programmed as desired, a match condition will not assert the corresponding pin. For this reason, chip selects should be used expressly to provide autovector termination of interrupt acknowledge cycles generated in response to assertion of the IRQ[7:1] pins.

Because match conditions do not result in chip-select assertion, the 0b10 (8-bit port) and 0b11 (16-bit port) encodings of the pin assignment fields in CSPAR0 and CSPAR1 serve only to drive pins so configured high at all times. Consequently, any chip select may provide autovector termination, even if its pin assignment field in CSPAR0 or CSPAR1 is programmed with the 0b00 (discrete output) or 0b01 (alternate function) encoding.

The first chip select that should be used for autovector termination in single-chip mode is $\overline{\text{CSBOOT}}$; it has no discrete output or alternate function capability. Although typically not needed in single-chip mode, the SCIM2E bus arbitration feature may be used when the $\overline{\text{BR}}$/$\overline{\text{CS0}}$, $\overline{\text{BG}}$/$\overline{\text{CSM}}$, and $\overline{\text{BGACK}}$/$\overline{\text{CSE}}$ pins are configured for their alternate functions. Of these three pins, only $\overline{\text{BR}}$/$\overline{\text{CS0}}$ can provide autovector termination. The $\overline{\text{CSM}}$ and $\overline{\text{CSE}}$ chip selects function in emulation mode only and are not user programmable.

CSPAR0 initially configures the SCIM2E function code pins FC[2:0] as port C discrete outputs PC[2:0]. Each pin may, however, still operate as a function code output, and when configured as such, will be driven during appropriate bus cycles. The chip-select functions of FC0/$\overline{\text{CS3}}$/PC0 and FC2/$\overline{\text{CS5}}$/PC2 may also be used for autovector termination in the fashion described above.

ADDR[22:19]/$\overline{\text{CS[9:6]}}$/PC[6:3] are initially configured as port C discrete outputs PC[6:3] by chip-select pin assignment register 1 (CSPAR1). Each pin may, however, still operate as an address line, and when configured as such, will be driven during appropriate bus cycles.

CSPAR1 initially configures ADDR23/$\overline{\text{CS10}}$/ECLK as a 16-bit chip select (0b11 pin assignment field encoding) to drive the pin to its inactive state. ADDR23/ $\overline{\text{CS10}}$/ECLK has no discrete output function. When 0b00 is programmed into its pin assignment field in CSPAR1, ADDR23/$\overline{\text{CS10}}$/ECLK will drive the M6800 bus E clock signal.

Just as the chip-select, function code, and bus arbitration signals associated with port C can be made active in single-chip mode, so too can the bus control signals associated with port E. While initially configured as discrete I/O by the port E pin assignment register (PEPAR), any port E bus control signal can be made active and will be driven or accept input during appropriate bus cycles.

Port F pins will initially be configured for discrete I/O in single-chip mode but can otherwise serve as interrupt request lines or edge-detect I/O pins with optional interrupt capability. Because no external bus is available in single-chip mode, interrupt requests from port F pins configured as interrupts (as opposed to interrupt requests from the port F edge-detect logic) must have their interrupt acknowledge cycles terminated by autovector.

In single-chip mode, the data bus is disabled at all times. DATA[15:8] become port G input/output pins PG[7:0], and DATA[7:0] become port H input/output pins PH[7:0]. Port G and H pins are configurable as inputs or outputs on a per pin basis.

Like ADDR[2:0] (which can be disabled by setting the ABD bit in SCIMMCR), the R/W line and instruction tracking pins (IPIPE/DSO and IFETCH/DSI) can be disabled by setting RWD and CPUD bits in SCIMMCR, respectively.

4.7.8.4 Fully (16-bit) Expanded Mode

Operation in 16-bit expanded mode is selected when BERR = 1 and DATA1 = 0 at the release of RESET. In this configuration, ADDR[18:11]/PA[7:0] and ADDR[10:3]/PB[7:0] become part of the address bus. Likewise, DATA[15:8]/PG[7:0] and DATA[7:0]/PH[7:0] become the data bus. The ABD, RWD, and CPUD bits in SCIMMCR are clear, enabling ADDR[2:0], R/W, and the instruction tracking pins (IPIPE/DSO and IFETCH/DSI), respectively. Ports A, B, G, and H are unavailable in 16-bit expanded mode. The initial configuration of all other SCIM2E pins is controlled by DATA[11:2] and DATA0 and is outlined in **Table 4-27** below.

Table 4-27 Fully (16-bit) Expanded Mode Reset Configuration

Select Pin	Affected Pin(s)	Default Function (Pin Held High)	Alternate Function (Pin Held Low)
DATA0	CSBOOT	16-bit CSBOOT	8-bit CSBOOT
DATA2	BR/CS0 FC0/CS3/PC0 FC1/PC1 FC2/CS5/PC2	CS0 CS3 FC1 CS5	BR FC0 FC1 FC2
DATA[7:3]	ADDR23/CS10/ECLK ADDR[22:19]/CS[9:6]/PC[6:3]	See **Table 4-28**	
DATA8	DSACK0/PE0 DSACK1/PE1 AVEC/PE2 RMC/PE3 DS/PE4 AS/PE5 SIZ0/PE6 SIZ1/PE7	DSACK0 DSACK1 AVEC RMC DS AS SIZ0 SIZ1	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
DATA9	FASTREF/PF0 IRQ[7:1]/PF[7:1]	FASTREF[1] IRQ[7:1]	PF0 PF[7:1]
DATA10[2]	BGACK/CSE BG/CSM	BGACK BG	CSE CSM[3]
DATA11[4]	External Bus Interface	Normal Operation	Factory Test

NOTES:

1. The FASTREF function is used only at reset and serves no purpose during normal operation.
2. If DATA1 and DATA10 are low at the rising edge of RESET, the SCIM2E will operate in emulation mode, a special variation of 16-bit expanded mode.
3. For CSM to be active, the SCIM2E must be configured for emulation mode, as described above, and any on-chip masked ROM modules must be disabled by driving their associated data bus pins low at the rising edge of RESET. At present, only masked ROM modules support memory emulation by means of the CSM chip select. CSM will not assert on MCUs with flash EEPROM modules.
4. DATA11 must be high at the rising edge of RESET for normal MCU operation.

DATA[7:3] select in a contiguous fashion whether ADDR[23:19]/$\overline{\text{CS}}$[10:6] serve as high-order address lines or chip selects. **Table 4-28** shows the reset correspondence between these pins.

Table 4-28 Reset Pin Function of $\overline{\text{CS}}$[10:6]

Data Bus Pins at Reset					Chip-Select/Address Bus Pin Function				
DATA7	DATA6	DATA5	DATA4	DATA3	$\overline{\text{CS10}}$/ ADDR23	$\overline{\text{CS9}}$/ ADDR22	$\overline{\text{CS8}}$/ ADDR21	$\overline{\text{CS7}}$/ ADDR20	$\overline{\text{CS6}}$/ ADDR19
1	1	1	1	1	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	$\overline{\text{CS8}}$	$\overline{\text{CS7}}$	$\overline{\text{CS6}}$
1	1	1	1	0	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	$\overline{\text{CS8}}$	$\overline{\text{CS7}}$	ADDR19
1	1	1	0	X	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	$\overline{\text{CS8}}$	ADDR20	ADDR19
1	1	0	X	X	$\overline{\text{CS10}}$	$\overline{\text{CS9}}$	ADDR21	ADDR20	ADDR19
1	0	X	X	X	$\overline{\text{CS10}}$	ADDR22	ADDR21	ADDR20	ADDR19
0	X	X	X	X	ADDR23	ADDR22	ADDR21	ADDR20	ADDR19

DATA[15:12] allow implementation dependent disabling of on-chip ROM and/or flash EEPROM modules. **Table 4-29** shows which modules on the MC68F375 are affected by these pins.

Table 4-29 Reset Configuration for MC68F375 Memory Modules

Select Pin	State of Select Pin at Rising Edge of $\overline{\text{RESET}}$	Memory Modules Affected
DATA10, DATA13	0	CMFI/ROM emulation mode
	1	CMFI/ROM normal mode
DATA14[1]	0	8-Kbyte Masked ROM array disabled All four 32-Kbyte FLASH arrays disabled
	1	8-Kbyte Masked ROM array enabled All four 32-Kbyte FLASH arrays enabled
DATA15[2]	0	All four 32-Kbyte CMFI FLASH blocks disabled
	1	All four 32-Kbyte CMFI FLASH blocks enabled

NOTES:
1. The ROM can be disabled if the STOP shadow bit is programmed to one or if DATA14 is low at the rising edge of $\overline{\text{RESET}}$.
2. The CMFI array is disabled if its STOP shadow bit is programmed to one or if DATA15 is low at the rising edge of $\overline{\text{RESET}}$.

Operation in 8-bit expanded mode is selected when $\overline{\text{BERR}}$ = 1 and DATA1 = 1 at the release of $\overline{\text{RESET}}$. In this configuration, ADDR[18:11]/PA[7:0] and ADDR[10:3]/PB[7:0] become part of the address bus, and only DATA[15:8]/PG[7:0] are used for the data bus. The ABD, RWD, and CPUD bits in SCIMMCR are clear, enabling ADDR[2:0], R/$\overline{\text{W}}$, and the instruction tracking pins ($\overline{\text{IPIPE}}$/DSO and $\overline{\text{IFETCH}}$/DSI), respectively. Ports A, B, and G are unavailable in 8-bit expanded mode, and DATA[7:0]/PH[7:0] serve as port H discrete I/O pins only. All remaining SCIM2E pins are configured as shown in **Table 4-30.**

Table 4-30 Partially (8-bit) Expanded Mode Reset Configuration

Select Pin	Affected Pin(s) or Module(s)	Default Function (Pin Held High)	Alternate Function (Pin Held Low)
NA[1]	$\overline{CSBOOT}$	8-bit $\overline{CSBOOT}$	
NA[1]	$\overline{BR}/\overline{CS0}$ FC0/$\overline{CS3}$/PC0 FC1/PC1 FC2/$\overline{CS5}$/PC2	$\overline{CS0}$ $\overline{CS3}$ FC1 $\overline{CS5}$	
NA[1]	ADDR23/$\overline{CS10}$/ECLK ADDR[22:19]/$\overline{CS[9:6]}$/PC[6:3]	$\overline{CS[10:6]}$	
DATA8	$\overline{DSACK0}$/PE0 $\overline{DSACK1}$/PE1 $\overline{AVEC}$/PE2 $\overline{RMC}$/PE3 $\overline{DS}$/PE4 $\overline{AS}$/PE5 SIZ0/PE6 SIZ1/PE7	$\overline{DSACK0}$ $\overline{DSACK1}$ $\overline{AVEC}$ $\overline{RMC}$ $\overline{DS}$ $\overline{AS}$ SIZ0 SIZ1	PE0 PE1 PE2 PE3 PE4 PE5 PE6 PE7
DATA9	FASTREF/PF0 $\overline{IRQ[7:1]}$/PF[7:1]	FASTREF[2] $\overline{IRQ[7:1]}$	PF0 PF[7:1]
DATA10	$\overline{BGACK}/\overline{CSE}$[3] $\overline{BG}/\overline{CSM}$[3]	$\overline{BGACK}$ $\overline{BG}$	

NOTES:
1. Because DATA[7:0] are unavailable in 8-bit expanded mode, these pins default to the reset configurations noted.
2. The FASTREF function is used only at reset and serves no purpose during normal operation.
3. The $\overline{CSE}$ and $\overline{CSM}$ emulation chip selects do not function in 8-bit expanded mode.

Just as in 16-bit expanded mode, DATA[15:12] allow implementation dependent disabling of on-chip ROM and/or flash EEPROM modules. Refer to **Table 4-29** above for which modules on the MC68F375 are affected by these pins.

4.7.8.5 Breakpoint Mode Selection

Background debug mode (BDM) is enabled when the breakpoint ($\overline{BKPT}$) pin is sampled at logic zero at the release of $\overline{RESET}$. Subsequent assertion of the $\overline{BKPT}$ pin or the internal breakpoint signal (for instance, execution of the CPU32 BGND instruction) will place the CPU32 in BDM.

If $\overline{BKPT}$ is sampled at logic one at the rising edge of $\overline{RESET}$, BDM is disabled. Assertion of the $\overline{BKPT}$ pin or execution of the BKPT instruction will result in normal breakpoint exception processing. Execution of the BGND instruction will cause an illegal instruction exception to be taken.

BDM remains enabled until the next system reset. $\overline{BKPT}$ is relatched and synchronized on each rising transition of $\overline{RESET}$ and must be held low for at least two clock cycles prior to $\overline{RESET}$ negation for BDM to be enabled. $\overline{BKPT}$ assertion logic must be designed with special care. If $\overline{BKPT}$ assertion extends into the first bus cycle following the release of $\overline{RESET}$, the bus cycle could inadvertently be tagged with a breakpoint.

Refer to **3.10.2 Background Debug Mode** and the ***CPU32 Reference Manual* (CPU32RM/AD)** for more information on background debug mode. Refer to the ***SCIM Reference Manual* (SCIMRM/AD)** and **APPENDIX E ELECTRICAL CHARACTERISTICS** for more information concerning $\overline{\text{BKPT}}$ signal timing.

4.7.8.6 Emulation Mode Selection

The SCIM2E contains logic that can be used to replace on-chip ports externally. The SCIM2E also contains special support logic that allows external emulation of internal ROM. These emulation support features enable the development of a single-chip application in expanded mode.

Emulation mode is a special type of 16-bit expanded operation. It is entered by holding DATA10 low, $\overline{\text{BERR}}$ high, and DATA1 low during reset. In emulation mode, all port A, B, E, G, and H data and data direction registers and the port E pin assignment register are mapped externally. The port C data, port F data and data direction registers, and port F pin assignment register are accessible normally in emulation mode.

The port emulation chip select ($\overline{\text{CSE}}$) is asserted whenever any of the externally mapped registers are addressed. The signal is asserted on the falling edge of $\overline{\text{AS}}$. The SCIM2E does not respond to these accesses, allowing external logic, such as a port replacement unit (PRU) to respond. Accesses to externally mapped registers require three clock cycles.

CMFI and ROM emulation is enabled by holding $\overline{\text{BERR}}$ high and by holding low DATA1, DATA10, and DATA13 when $\overline{\text{RESET}}$ is released. While CMFI and ROM emulation mode is enabled, the emulation memory chip-select signal ($\overline{\text{CSM}}$) is asserted whenever an access to an address assigned to the masked ROM module or CMFI module is made.

CMFI and ROM modules do not acknowledge IMB accesses while in emulation mode. This causes the SCIM2E to run an external bus cycle for each access. The bus cycle is terminated by the module, however, ensuring consistent timing.

4.7.9 Use of the Three-State Control Pin

Asserting the three-state control (TSC) input to a logic 0 causes the MCU to place all output drivers in a disabled, high-impedance state. TSC must remain asserted for approximately ten clock cycles in order for drivers to change state.

When the SCIM2E clock synthesizer is used, PLL ramp-up time affects how long the ten cycles take. Worst case is approximately 20 ms from TSC assertion.

When an external clock signal is applied, pins go high-impedance as soon after TSC assertion as approximately ten clock pulses have been applied to the EXTAL pin.

NOTE

When TSC assertion takes effect, internal signals are forced to values that can cause inadvertent mode selection. Once the output drivers change state, the MCU must be powered down and restarted before normal operation can resume.

4.8 Interrupts

Interrupt recognition and servicing involve complex interaction between the SCIM2E, the CPU32, and a device or module requesting interrupt service. This discussion provides an overview of the entire interrupt process. Chip-select logic can also be used to respond to interrupt requests. Refer to **4.9 Chip Selects** for more information.

4.8.1 Interrupt Exception Processing

The CPU32 handles interrupts as a type of asynchronous exception. An exception is an event that preempts normal processing. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with an exception. Each exception has an assigned vector that points to an associated handler routine. These vectors are stored in a vector table located from 0x00000 to 0x001FF. The CPU32 uses vector numbers to calculate displacement into the table. Refer to **3.9 Exception Processing** for more information.

4.8.2 Interrupt Priority and Recognition

The CPU32 provides seven levels of interrupt priority (1–7), seven automatic interrupt vectors, and 200 assignable interrupt vectors. All interrupts with priorities less than seven can be masked by the interrupt priority (IP) field in the condition code register (CCR).

The IP field consists of CCR bits [7:5]. Binary values 0b000 to 0b111 provide eight priority masks. Each mask prevents an interrupt request of a priority less than or equal to the mask value (except for $\overline{\text{IRQ7}}$) from being recognized. When the IP field contains 0b000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

There are seven interrupt request signals ($\overline{\text{IRQ[7:1]}}$) with corresponding external pins that can be asserted by microcontroller modules or external devices. Simultaneous requests of different priorities can be made. Internal assertion of an interrupt request line does not affect the state of the corresponding MCU pin.

External interrupt requests are routed to the CPU32 via the EBI and SCIM2E interrupt control logic. All requests for interrupt service are treated as if they come from internal modules. The CPU32 treats external interrupt requests as if they come from the SCIM2E.

The $\overline{\text{IRQ[6:1]}}$ pins are active-low level-sensitive inputs. The $\overline{\text{IRQ7}}$ pin is an active-low transition-sensitive input; it requires both an edge and a voltage level to be valid.

$\overline{IRQ[6:1]}$ are maskable. $\overline{IRQ7}$ is non-maskable. The $\overline{IRQ7}$ input is transition sensitive to prevent redundant servicing and stack overflow. A non-maskable interrupt is generated each time $\overline{IRQ7}$ is asserted and each time the CCR is written while $\overline{IRQ7}$ is asserted. A write to the CCR re-arms the $\overline{IRQ7}$ detection circuitry; consequently, any write to the CCR while $\overline{IRQ7}$ is asserted, even one that sets the IP field to 0b111, will generate a new $\overline{IRQ7}$ interrupt.

Interrupt requests are sampled on consecutive falling edges of the system clock. Interrupt request input circuitry has hysteresis. To be valid, a request signal must be asserted for at least two consecutive clock cycles. Valid requests do not cause immediate exception processing, but are left pending. Pending requests are processed at instruction boundaries or when processing of higher priority exceptions is complete.

The CPU32 does not latch the priority of pending interrupt requests. If an interrupt source of higher priority makes a request while a lower priority request is pending, the higher priority request will be serviced. If an interrupt request with a priority less than or equal to the current IP mask value is made, the CPU32 will not recognize the request. If simultaneous interrupt requests of different priorities are made, and both have a priority greater than the mask value, the CPU32 will recognize the higher priority request.

4.8.3 Interrupt Acknowledge and Arbitration

When the CPU32 detects one or more interrupt requests of a priority higher than the IP mask value, a read cycle from address 0b11111111111111111111: [IP]:1 in CPU space is executed. Refer to **4.5.1.7 Function Codes** and **4.6.4 CPU Space Cycles** for more information.

The CPU space read cycle performs two functions. It places a mask value corresponding to the highest priority interrupt request on the address bus, and it acquires an exception vector number from the interrupt source. The mask value is decoded by modules or external devices that have requested interrupt service to determine whether the current interrupt acknowledge (IACK) cycle pertains to them. It is also latched into the IP field to mask lower priority interrupts during exception processing.

Modules that have requested interrupt service decode the IP value placed on the address bus at the beginning of the IACK cycle, and if their requests are at the specified IP level, respond to the cycle. Arbitration between simultaneous requests of the same priority is performed by serial contention between module interrupt arbitration (IARB) field bit values.

Each module that can request interrupt service, including the SCIM2E, has an IARB field in its module configuration register. To implement an arbitration scheme, each module that can request interrupt service must be assigned a unique, non-zero IARB field value during system initialization. Arbitration priorities can range from 0b0001 (lowest) to 0b1111 (highest). If the CPU32 recognizes an interrupt request from a module that has an IARB field value of 0b0000, a spurious interrupt exception will be processed.

Because the EBI manages external interrupt requests, the SCIM2E IARB field value is used for arbitration between internal and external interrupt requests of the same priority. The reset value of IARB for the SCIM2E is 0b1111, and the reset value of IARB for all other modules is 0b0000. As noted above, initialization software must assign different values to each IARB field to implement an arbitration scheme.

NOTE

Do not assign the same arbitration priority to more than one module. When two or more IARB fields have the same non-zero value, the CPU32 will interpret multiple vector numbers simultaneously with unpredictable consequences.

Although arbitration is intended to deal with simultaneous interrupt requests of the same priority level, it always take place, even when a single source is requesting service. This is important for two reasons: the EBI does not transfer the IACK cycle to the external bus unless the SCIM2E wins contention, and failure to contend causes the IACK cycle to be terminated early by bus error.

When arbitration is complete, the winning module must place a vector number on the data bus and terminate the IACK cycle with $\overline{\text{DSACK}}$. In the case of external interrupt requests, the IACK cycle is transferred to the external bus. The device requesting interrupt service must decode the mask value then respond with a vector number and generate data and size acknowledge ($\overline{\text{DSACK}}$) termination signals, or it must assert $\overline{\text{AVEC}}$ to request an autovector. If the device does not respond in time, the SCIM2E bus monitor, if enabled, will assert the internal $\overline{\text{BERR}}$ signal and a spurious interrupt exception will be taken.

Chip-select logic can also be used to generate internal $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$ signals in response to external interrupt requests. Chip-select address match logic functions only after the SCIM2E has won arbitration, and the resulting IACK cycle is transferred to the external bus. For this reason, interrupt requests from modules other than the SCIM2E will never have their IACK cycles terminated by chip-select generated $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$. Refer to **4.9.4.1 Using Chip-Select Signals for Interrupt Acknowledge Cycle Termination** for more information.

As stated above, all interrupt requests from internal modules have their associated IACK cycles terminated by $\overline{\text{DSACK}}$. For this reason, user vectors (instead of autovectors) must always be used for interrupts generated by internal modules.

For periodic timer interrupts, the PIRQL[2:0] field in the periodic interrupt control register (PICR) determines PIT priority level. A PIRQL[2:0] value of 0b000 disables PIT interrupts. By hardware convention, PIT interrupts are serviced before external interrupt service requests or port F edge-detect interrupts requests of the same priority. External interrupt requests are serviced before requests from the port F edge-detect logic, as well. Refer to **4.4.7 Periodic Interrupt Timer** and **4.10.4 Port F** for more information.

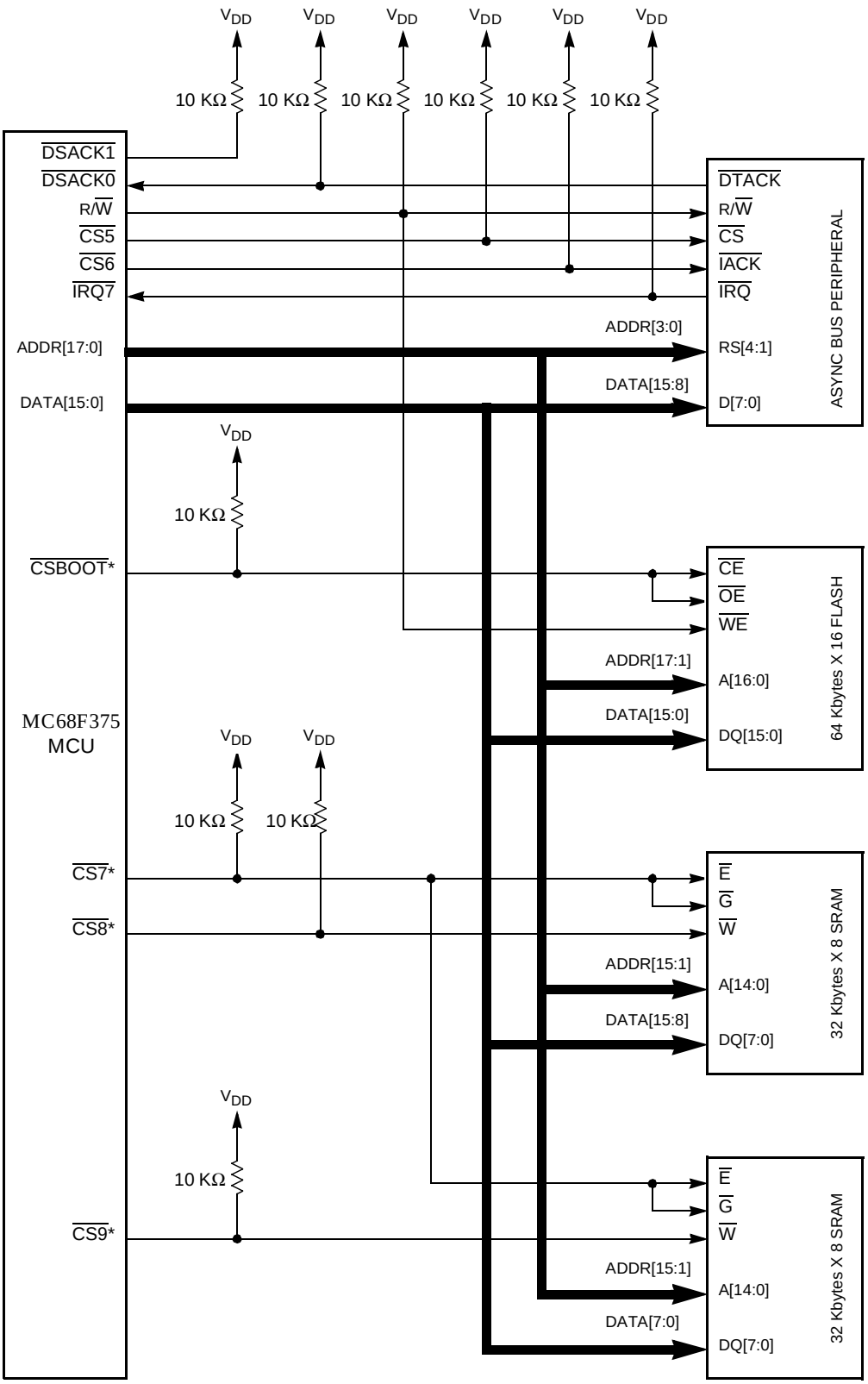
4.8.4 Interrupt Processing Summary

A summary of the entire interrupt processing sequence follows. When the sequence begins, a valid interrupt service request has been detected and is pending.

1. The CPU32 finishes higher priority exception processing or reaches an instruction boundary.
2. Processor state is stacked.
3. The interrupt acknowledge cycle begins:
   a. FC[2:0] are driven to 0b111 (CPU space) encoding.
   b. The address bus is driven as follows. ADDR[23:20] = 0b1111 ADDR[19:16] = 0b1111, which indicates that the cycle is an interrupt acknowledge CPU space cycle; ADDR[15:4] = 0b111111111111; ADDR[3:1] = the priority of the interrupt request being acknowledged; and ADDR0 = 0b1.
   c. Request priority is latched into the CCR IP field from the address bus.
4. Modules or external peripherals that have requested interrupt service decode the priority value in ADDR[3:1]. Each module or device with a request level equal to the value in ADDR[3:1] enters interrupt arbitration.
5. After arbitration, the interrupt acknowledge cycle is completed in one of the following ways:
   a. When there is no contention (responding modules have IARB = 0b0000), the internal bus monitor, if enabled, asserts $\overline{\text{BERR}}$, and the CPU32 generates the spurious interrupt vector number.
   b. The interrupt source that wins arbitration supplies a vector number and $\overline{\text{DSACK}}$ signals appropriate to the access. The CPU32 acquires the vector number.
   c. The $\overline{\text{AVEC}}$ signal is asserted either by the external device requesting interrupt service ($\overline{\text{AVEC}}$ can be tied low if all external interrupts are to use autovectors) or by an appropriately programmed SCIM2E chip select, and the CPU32 generates an autovector number corresponding to the interrupt priority.
   d. The bus monitor or external device asserts $\overline{\text{BERR}}$ and the CPU32 generates the spurious interrupt vector number.
6. The vector number is converted to a vector address.
7. The content of the vector address is loaded into the PC and the processor transfers control to the exception handler routine.

4.9 Chip Selects

Typical microcontrollers require additional hardware to provide chip-select signals for external devices. The SCIM2E includes nine programmable chip-select circuits that can provide from 2- to 16-clock cycle access to external memory and peripherals. Address block sizes of two Kbytes to one Mbyte can be selected. **Figure 4-21** is a diagram of a basic system that uses chip selects.



1. THESE CHIP-SELECT LINES ARE CONFIGURED FOR 16-BIT PORT OPERATION IN THIS EXAMPLE.

F396 SCIM2E B

Figure 4-21 Basic MCU System

Chip-select assertion can be synchronized with bus control signals to provide output enable, read/write strobe, or interrupt acknowledge signals. Chip-select logic can also generate $\overline{\text{DSACK}}$ and $\overline{\text{AVEC}}$ signals internally. Each signal can also be synchronized with the ECLK signal available on ADDR23.

When a memory access occurs, chip-select logic compares address space type, address, type of access, transfer size, and interrupt priority (in the case of interrupt acknowledge) to parameters stored in chip-select registers. If all parameters match, the appropriate chip-select signal is asserted. Chip-select signals are active low. If a chip-select function is given the same address as a microcontroller module or an internal memory array, an access to that address goes to the module or array, and the chip-select signal is not asserted. The external address and data buses do not reflect the internal access.

All chip-select signals except $\overline{\text{CSBOOT}}$ are disabled after the release of $\overline{\text{RESET}}$, and cannot be asserted until the R/$\overline{\text{W}}$[1:0] and BYTE[1:0] fields in the corresponding option register are programmed to non-zero values. $\overline{\text{CSBOOT}}$ is automatically enabled out of reset in 8-bit and 16-bit expanded modes. Alternate functions for chip-select pins are enabled if appropriate data bus pins are held low at the release of $\overline{\text{RESET}}$. Refer to **4.7.8.2 Data Bus Mode Selection** for more information. **Figure 4-22** is a functional diagram of a single chip-select circuit.

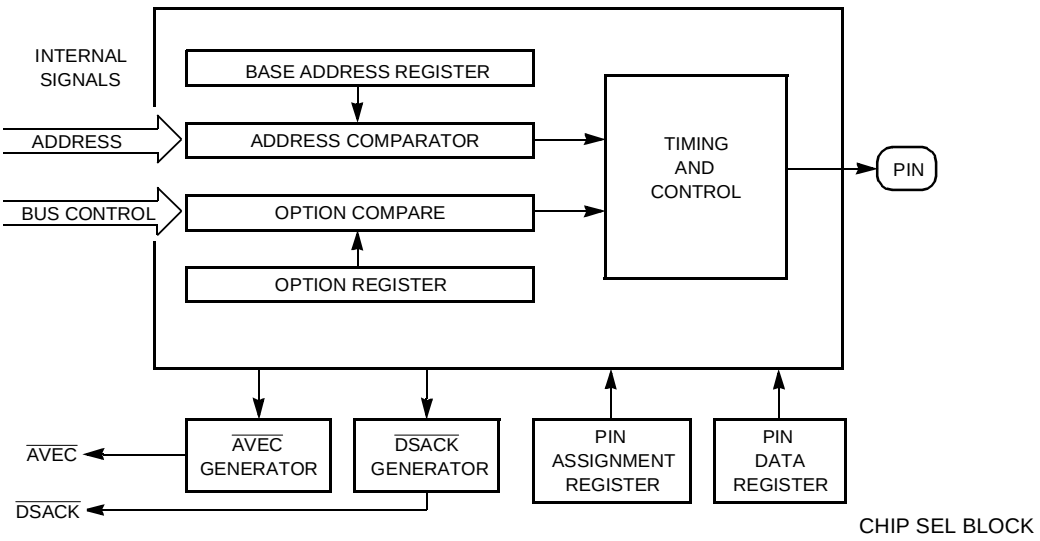


Figure 4-22 Chip-Select Circuit Block Diagram

4.9.1 Chip-Select Pin Assignment Register

The pin assignment registers contain twelve 2-bit fields that determine the functions of the chip-select pins. Each pin has two or three possible functions, as shown in **Table 4-31** and **Table 4-32**.

CSPAR0 — Chip-Select Pin Assignment Register 0 **0xYF FA44**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	CS5PA[1:0]		CS4PA[1:0]		CS3PA[1:0]		CSEPA[1:0]		CSMPA[1:0]		CS0PA[1:0]		CSBTPA[1:0]	
RESET:															
0	0	DATA2[1]	1	DATA2[1]	1	DATA2[1]	1	DATA10[1]	1	DATA10[1]	1	DATA2[1]	1	1	DATA0[1]

NOTES:
1. The default state of this bit is taken from the listed bit of the data bus during reset.

This register contains seven 2-bit fields that determine the function of corresponding chip-select pins. Bits [15:14] are not used. These bits always read zero; writes have no effect. CSPAR0 bit 1 always reads one; writes to CSPAR0 bit 1 have no effect. The alternate functions can be enabled by data bus mode selection during reset. This register may be read or written at any time. After reset, software may enable one or more pins as discrete outputs.

Table 4-31 shows CSPAR0 pin assignments.

Table 4-31 CSPAR0 Pin Assignments

CSPAR0 Field[1]	Chip-Select Signal	Alternate Signal	Discrete Output
CS5PA[1:0]	CS5	FC2	PC2
CS4PA[1:0]	—	FC1	PC1
CS3PA[1:0]	CS3	FC0	PC0
CSEPA[1:0]	CSE	$\overline{\text{BGACK}}$	—
CSMPA[1:0]	CSM	$\overline{\text{BG}}$	—
CS0PA[1:0]	CS0	$\overline{\text{BR}}$	—
CSBTPA[1:0]	$\overline{\text{CSBOOT}}$	—	—

NOTES:
1. See **Table 4-34** for pin assignment field encoding.

CSPAR1 — Chip-Select Pin Assignment Register 1 **0xYF FA46**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	0	CS10PA[1:0]		CS9PA[1:0]		CS8PA[1:0]		CS7PA[1:0]		CS6PA[1:0]	
RESET:															
0	0	0	0	0	0	DATA 7[1]	1	DATA 6[1]	1	DATA 5[1]	1	DATA 4[1]	1	DATA 3[1]	1

NOTES:
1. Refer to **Table 4-28** for CSPAR1 reset state information.

CSPAR1 contains five 2-bit fields that determine the functions of corresponding chip-select pins. Bits [15:10] are reserved. These bits always read zero; writes have no effect. **Table 4-32** shows CSPAR1 pin assignments, including alternate functions that can be enabled by data bus mode selection during reset.

Table 4-32 CSPAR1 Pin Assignments

CSPAR1 Field	Chip-Select Signal	Alternate Signal	Discrete Output
CS10PA[1:0]	$\overline{CS10}$	ADDR23	ECLK
CS9PA[1:0]	$\overline{CS9}$	ADDR22	PC6
CS8PA[1:0]	$\overline{CS8}$	ADDR21	PC5
CS7PA[1:0]	$\overline{CS7}$	ADDR20	PC4
CS6PA[1:0]	$\overline{CS6}$	ADDR19	PC3

The reset state of DATA[7:3] determines whether pins controlled by CSPAR1 are initially configured as high-order address lines or chip-selects. **Table 4-28** shows the correspondence between DATA[7:3] and the reset configuration of $\overline{CS}$[10:6]/ ADDR[23:19]. This register may be read or written at any time. After reset, software may enable one or more pins as discrete outputs.

Table 4-33 Reset Pin Function of $\overline{CS}$[10:6]

Data Bus Pins at Reset					Chip-Select/Address Bus Pin Function				
DATA7	DATA6	DATA5	DATA4	DATA3	$\overline{CS10}$/ ADDR23	$\overline{CS9}$/ ADDR22	$\overline{CS8}$/ ADDR21	$\overline{CS7}$/ ADDR20	$\overline{CS6}$/ ADDR19
1	1	1	1	1	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	$\overline{CS7}$	$\overline{CS6}$
1	1	1	1	0	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	$\overline{CS7}$	ADDR19
1	1	1	0	X	$\overline{CS10}$	$\overline{CS9}$	$\overline{CS8}$	ADDR20	ADDR19
1	1	0	X	X	$\overline{CS10}$	$\overline{CS9}$	ADDR21	ADDR20	ADDR19
1	0	X	X	X	$\overline{CS10}$	ADDR22	ADDR21	ADDR20	ADDR19
0	X	X	X	X	ADDR23	ADDR22	ADDR21	ADDR20	ADDR19

Table 4-34 shows pin assignment field encoding. Pins that have no discrete output function must not be programmed with the 0b00 encoding as this will configure the pin for its alternate function. For instance, programming CS0PA[1:0] to 0b00 will configure $\overline{CS0}$/$\overline{BR}$ for the bus request ($\overline{BR}$) function.

Table 4-34 Pin Assignment Field Encoding

CSxPA[1:0]	Description
00	Discrete output
01	Alternate function
10	Chip-select (8-bit port)
11	Chip-select (16-bit port)

Port size determines the way in which bus transfers to an external address are allocated. A port size of eight bits or sixteen bits can be selected when a pin is assigned as a chip select. Port size and transfer size affect how the chip-select signal is asserted. Refer to **4.9.3 Chip-Select Option Registers** for more information.

From the release of reset, chip-select pin functions are determined by logic levels on certain data bus pins. The data bus pins have weak internal pull-up devices but can be held low by external logic. This allows a pin's 16-bit chip-select function (data bus pin(s) held high) or its alternate function (data bus pin(s) held low) to be selected at the release of RESET. Refer to **4.7.8.2 Data Bus Mode Selection** for more information.

The CSBOOT signal is enabled out of reset. The state of DATA0 during reset determines what port width CSBOOT uses. If DATA0 is held high, 16-bit port size is selected. If DATA0 is held low, 8-bit port size is selected. In 8-bit expanded mode, the state of DATA0 is ignored, and CSBOOT is configured for 8-bit operation.

A pin programmed as a discrete output will drive the value specified in the port C data register. No discrete output function is available for the CSBOOT, CS0/BR, CSM/BG,and CSE/BGACK pins. ADDR23 provides the ECLK output rather than a discrete output signal.

When a pin is programmed for discrete output or alternate function, internal chip-select logic still functions and can be used to generate DSACK or AVEC (to terminate IACK cycles generated in response to external interrupt requests) internally on an address and control signal match.

4.9.1.1 Port C Data Register

The port C data register (PORTC) latches data for port C pins programmed as discrete outputs. When a pin is assigned as a discrete output, the value in this register appears at the output. Port C bit 7 is not used. Writing to this bit has no effect, and it always reads zero.

PORTC — Port C Data Register **0xYF FA41**

7	6	5	4	3	2	1	LSB 0
0	PC6	PC5	PC4	PC3	PC2	PC1	PC0
RESET:							
0	1	1	1	1	1	1	1

PORTC latches data for chip-select pins configured as discrete outputs.

4.9.2 Chip-Select Base Address Registers

Each chip select has an associated base address register, CSBAR[0], [3] and [5:10]. A base address is the lowest address in the block of addresses enabled by a chip select. Block size is the extent of the address block above the base address. Block size is determined by the value contained in BLKSZ[2:0]. Multiple chip selects assigned to the same block of addresses must have the same number of wait states.

BLKSZ[2:0] determines which bits in the base address field are compared to corresponding bits on the address bus during an access. Provided other constraints determined by option register fields are also satisfied, when a match occurs, the associated chip-select signal is asserted. **Table 4-35** shows BLKSZ[2:0] encoding.

Table 4-35 Block Size Encoding

BLKSZ[2:0]	Block Size	Address Lines Compared
000	2 Kbytes	ADDR[23:11]
001	8 Kbytes	ADDR[23:13]
010	16 Kbytes	ADDR[23:14]
011	64 Kbytes	ADDR[23:16]
100	128 Kbytes	ADDR[23:17]
101	256 Kbytes	ADDR[23:18]
110	512 Kbytes	ADDR[23:19]
111	1 Mbyte	ADDR[23:20]

The chip-select address compare logic uses only the most significant bits to match an address within a block. For this reason, the value of the base address must be an integer multiple of the block size.

After reset, the CPU32 fetches initialization values from addresses 0x00000 to 0x00007 in program space. To support bootstrap operation from reset, the chip-select boot base address register (CSBARBT) defaults to 0x0007 which specifies a base address of 0x000000 and a block size of 512 Kbytes. This allows a memory device containing the reset vector and startup routine to automatically be selected by $\overline{\text{CSBOOT}}$. Refer to **4.9.4.2 Chip-Select Reset Operation** for more information.

Bit and field definitions for CSBARBT and CSBAR[0], [3] and [5:10] are the same, but reset block sizes differ. For the bit definitions of CSBARBT and CSBAR[0], [3] and [5:10] registers, see **Table 4-36**.

CSBARBT — Chip-Select Base Address Register Boot **0xYF FA48**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ[2:0]		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

CSBAR0 — Chip-Select Base Address Registers **0xYF FA4C**
CSBAR3 **0xYF FA58**
CSBAR5 **0xYF FA60**
CSBAR6 **0xYF FA64**
CSBAR7 **0xYF FA68**
CSBAR8 **0xYF FA6C**
CSBAR9 **0xYF FA70**
CSBAR10 **0xYF FA74**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
ADDR 23	ADDR 22	ADDR 21	ADDR 20	ADDR 19	ADDR 18	ADDR 17	ADDR 16	ADDR 15	ADDR 14	ADDR 13	ADDR 12	ADDR 11	BLKSZ[2:0]		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4-36 CSBARBT/CSBAR Bit Descriptions

Bit(s)	Name	Description
15:3	ADDR[23:11]	Chip-select base address. This field sets the starting address of a particular chip select's address space. The address compare logic uses only the most significant bits to match an address within a block. The value of the base address must be an integer multiple of the block size.
2:0	BLKSZ[2:0]	Block size field. This field determines the size of the block that is enabled by the chip select. **Table 4-35** shows bit encoding for the base address registers block size field.

4.9.3 Chip-Select Option Registers

Fields in the chip-select option registers determine the timing of and conditions for assertion of chip-select signals. For a chip select to assert and to provide $\overline{\text{DSACK}}$ or autovector termination, other constraints set by fields in its option register and base address register must also be satisfied. The following paragraphs summarize option register functions.

CSORBT — Chip-Select Option Register Boot **0xYF FA4A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MODE	BYTE[1:0]		R/$\overline{\text{W}}$[1:0]		STRB	$\overline{\text{DSACK}}$[3:0]				SPACE[1:0]		IPL[2:0]			$\overline{\text{AVEC}}$
RESET:															
0	1	1	1	1	0	1	1	0	1	1	1	0	0	0	0

CSOR0 — Chip-Select Option Registers **0xYF FA4E**
CSOR3 **0xYF FA5A**
CSOR5 **0xYF FA62**
CSOR6 **0xYF FA66**
CSOR7 **0xYF FA6A**
CSOR8 **0xYF FA6E**
CSOR9 **0xYF FA72**
CSOR10 **0xYF FA76**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MODE	BYTE[1:0]		R/$\overline{W}$[1:0]		STRB	$\overline{DSACK}$[3:0]				SPACE[1:0]		IPL[2:0]			$\overline{AVEC}$
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSORBT and CSOR[0], [3] and [5:10] contain parameters that support operations from external memory devices. Bit and field definitions for CSORBT and CSOR[0], [3] and [5:10] are the same.

Table 4-37 CSOR Bit Descriptions

Bit(s)	Name	Description
15	MODE	Asynchronous/Synchronous Mode. In asynchronous mode, chip-select assertion is synchronized with $\overline{AS}$ and $\overline{DS}$. 0 = Asynchronous mode is selected. 1 = Synchronous mode is selected, and used with ECLK peripherals.
14:13	BYTE	Upper/lower byte option. This field is used only when the chip-select 16-bit port option is selected in the pin assignment register. This allows the usage of two external 8-bit memory devices to be concatenated to form a 16-bit memory. 00 = Disable 01 = Lower byte 10 = Upper byte 11 = Both bytes
12:11	R/$\overline{W}$	Read/write. This field causes a chip select to be asserted only for a read, only for a write, or for both reads and writes. 00 = Disable 01 = Read only 10 = Write only 11 = Read/Write
10	STRB	Address strobe/data strobe. This bit controls the timing for assertion of a chip select in asynchronous mode only. Selecting address strobe causes the chip select to be asserted synchronized with address strobe. Selecting data strobe causes the chip select to be asserted synchronized with data strobe. Data strobe timing is used to create a write strobe when needed. 0 = Address strobe 1 = Data strobe
9:6	$\overline{DSACK}$	Data strobe acknowledge. This field specifies the source of $\overline{DSACK}$ in asynchronous mode as internally generated or externally supplied. It also allows adjust bus timing adjustment with internal $\overline{DSACK}$ generation by controlling the number of wait states that are inserted to optimize bus speed in a particular application. **Table 4-38** shows the $\overline{DSACK}$[3:0] field encoding. The fast termination encoding (0b1110) effectively corresponds to –1 wait states.

Table 4-37 CSOR Bit Descriptions (Continued)

Bit(s)	Name	Description
5:4	SPACE	Address space select. Use this option field to select an address space for the chip-select logic. The CPU32 normally operates in supervisor or user space, but interrupt acknowledge cycles must take place in CPU space 00 = CPU Space 01 = User Space 10 = Supervisor Space 11 = Supervisor/User Space
3:1	IPL	Interrupt priority level. When SPACE[1:0] is set for CPU space (0b00), chip-select logic can be used as an interrupt acknowledge strobe for an external device. During an interrupt acknowledge cycle, the interrupt priority level is driven on address lines ADDR[3:1] is then compared to the value in IPL[2:0]. If the values match, an interrupt acknowledge strobe will be generated on the particular chip-select pin, provided other option register conditions are met. **Table 4-39** shows IPL[2:0] field encoding.
0	$\overline{\text{AVEC}}$	Autovector enable. This field selects one of two methods of acquiring an interrupt vector during an interrupt acknowledge cycle. This field is not applicable when SPACE[1:0] = 0b00. 0 = External interrupt vector enabled 1 = Autovector enabled

Table 4-38 $\overline{\text{DSACK}}$ Field Encoding

$\overline{\text{DSACK}}$[3:0]	Clock Cycles Required Per Access	Wait States Inserted Per Access
0000	3	0
0001	4	1
0010	5	2
0011	6	3
0100	7	4
0101	8	5
0110	9	6
0111	10	7
1000	11	8
1001	12	9
1010	13	10
1011	14	11
1100	15	12
1101	16	13
1110	2	–1 (Fast termination)
1111	—	External $\overline{\text{DSACK}}$

Table 4-39 Interrupt Priority Level Field Encoding

IPL[2:0]	Interrupt Priority Level
000	Any Level[1]
001	1
010	2
011	3
100	4
101	5
110	6
111	7

NOTES:
1. Any level means that chip select is asserted regardless of the level of the interrupt acknowledge cycle.

If the chip select is configured to trigger on an interrupt acknowledge cycle (SPACE[1:0] = 0b00) and the $\overline{\text{AVEC}}$ field is set to one, the chip select automatically generates $\overline{\text{AVEC}}$ and completes the interrupt acknowledge cycle. If the $\overline{\text{AVEC}}$ bit = 0, then the vector must be supplied by the requesting external device to complete the IACK read cycle.

The BYTE field controls the data placement conditions under which a particular chip select asserts. This is a different function from that of the chip-select pin assignment registers which determine if transfers controlled by a particular chip select are fundamentally eight or sixteen bits in length. Instead, BYTE[1:0] specifies whether the chip select will assert for data placed on the lower half, upper half, or both halves of the data bus.

When a chip select is configured for 8-bit port operation, only DATA[15:8] are used. Consequently, any BYTE field value other than 0b00 will permit signal assertion when all other match conditions are met.

When a chip select is configured for 16-bit port operation, BYTE[1:0] determines which combinations of ADDR0 and SIZ0 will result in chip-select assertion. A chip select configured for both bytes (0b11) will assert (assuming all other conditions are met) regardless of the states of ADDR0 and SIZ0. A chip select configured for upper byte (0b10) will assert only when ADDR0 = 0 (even addresses). A chip select configured for lower byte (0b01) must assert on all accesses to odd addresses (ADDR0 = 1) and on word accesses to even addresses (ADDR0 = 0 and SIZ0 = 1). When the boolean expression $\overline{\text{ADDR0}} \cdot \text{SIZ0}$ is false, lower byte chip-select assertion will occur. In all cases, the routing of information onto the data bus by the EBI data multiplexer is controlled by ADDR0 and SIZ0.

When SPACE[1:0] = 0b00 (CPU space), IPL[2:0] specifies the interrupt priority that must be matched when chip-select logic is used to terminate IACK cycles generated in response to external requests for interrupt service. When SPACE[1:0] is set to 0b00 (CPU space), ADDR[3:1] is compared to the IPL field at the beginning of an IACK cycle. If these values are the same (and other option register constraints are satisfied),

the specified chip select will be asserted. This field only affects the response of chip-select logic to IACK cycles and does not affect interrupt recognition by the CPU32. Setting IPL[2:0] to 0b000 when SPACE[1:0] = 0b00 will cause chip-select assertion regardless of the IACK cycle priority, provided other option register conditions are met. When SPACE[1:0] = 0b01, 0b10, or 0b11, the IPL field specifies whether chip-select assertion should occur during accesses to data space, program space, or both.

4.9.4 Chip-Select Operation

When the MCU makes an access, each enabled chip-select circuit compares the following items:

- Function code signals FC[2:0] to the SPACE field, and to the IP field if the SPACE field is not programmed for CPU space.
- Appropriate address bus bits to base address field.
- The R/$\overline{W}$ signal to the R/$\overline{W}$ field.
- ADDR0 and/or SIZ to the BYTE field (only chip selects configured for 16-bit operation).
- Priority of the interrupt being acknowledged (ADDR[3:1]) to the IPL field when the access is an interrupt acknowledge cycle and the SPACE field is programmed for CPU space.

When a match occurs, the chip-select signal is asserted. Assertion occurs at the same time as $\overline{AS}$ or $\overline{DS}$ assertion in asynchronous mode. Assertion is synchronized with ECLK in synchronous mode. In asynchronous mode, the $\overline{DSACK}$ field specifies internal or external $\overline{DSACK}$ assertion and the number of wait states inserted if internal $\overline{DSACK}$ assertion is selected.

The number of wait states needed by an external device is determined by its access time. Normally, wait states are inserted into the bus cycle during state S3 until a peripheral asserts $\overline{DSACK}$. If a peripheral does not generate $\overline{DSACK}$, internal $\overline{DSACK}$ generation must be selected and a predetermined number of wait states can be programmed into the chip-select option register.

4.9.4.1 Using Chip-Select Signals for Interrupt Acknowledge Cycle Termination

Ordinary bus cycles are issued in supervisor or user space. Interrupt acknowledge bus cycles are issued in CPU space. Refer to **4.6.4 CPU Space Cycles** and **4.8 Interrupts** for more information. The SCIM2E chip selects operate identically in each type of space, but base address and option registers must be properly programmed for each type of external bus cycle.

During a CPU space cycle, bits [15:3] of the appropriate base register must be configured to match ADDR[23:11], as the address is compared to an address generated by the CPU.

Figure 4-23 shows CPU space encoding for an interrupt acknowledge cycle. FC[2:0] drive 0b111, designating a CPU space access. ADDR[3:1] denote interrupt priority, and the space type field (ADDR[19:16]) is set to 0b1111, the interrupt acknowledge code. The rest of the address lines are set to one.

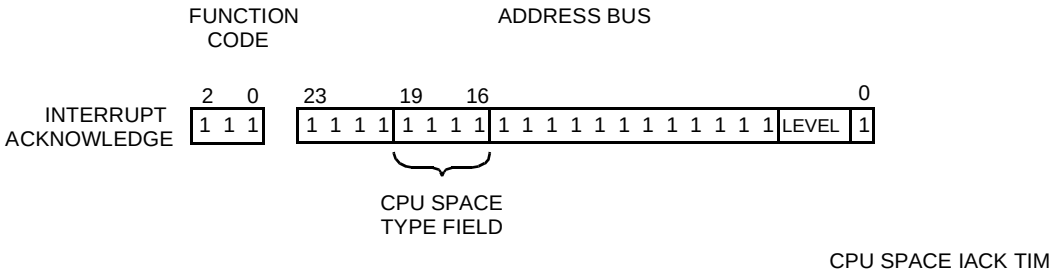


Figure 4-23 CPU Space Encoding for Interrupt Acknowledge

Chip-select address match logic functions only after the SCIM2E has won arbitration, and the resulting IACK cycle is transferred to the external bus. For this reason, interrupt requests from modules other than the SCIM2E will never have their IACK cycles terminated by chip-select generated $\overline{\text{AVEC}}$ or $\overline{\text{DSACK}}$.

Use the procedure that follows to configure a chip select to provide IACK cycle termination.

1. Program the base address field to all ones.
2. Program block size to no more than 64 Kbytes, so that the address comparator checks ADDR[19:16] against the corresponding bits in the base address register. (The CPU space bus cycle type is placed on ADDR[19:16]).
3. Set the R/$\overline{\text{W}}$ field to read only. An interrupt acknowledge cycle is performed as a read in CPU space.
4. Set the BYTE field to lower byte when using a 16-bit port, as the external vector for a 16-bit port is fetched from the lower byte. Set the BYTE field to upper byte when using an 8-bit port.

If an interrupting device does not provide a vector number, an autovector must be generated, either by asserting the $\overline{\text{AVEC}}$ pin or by having the chip select assert $\overline{\text{AVEC}}$ internally. The latter is accomplished by setting the chip-select option register $\overline{\text{AVEC}}$ bit. This terminates the bus cycle.

4.9.4.2 Chip-Select Reset Operation

The LSB of each of the 2-bit pin assignment fields in CSPAR0 and CSPAR1 has a reset value of one. The reset values of the MSBs of each field are determined by the states of DATA[7:1] during reset. Weak internal pull-up devices condition each of the data lines so that chip-select operation is selected by default out of reset. Excessive bus loading can overcome the internal pull-up devices, resulting in inadvertent configuration out of reset. Use external pull-up resistors or active devices to avoid this.

The base address fields in chip-select base address registers CSBAR[0,3, 5:10], CSBAR3, and CSBAR0 and chip-select option registers CSOR[10:5], CSOR3, and CSOR0 have the reset values shown in **Table 4-40**. The BYTE and R/$\overline{\text{W}}$ fields of each option register have a reset value of “disable”, so that a chip-select signal cannot be asserted until the base and option registers are initialized.

Table 4-40 Chip-Select Base and Option Register Reset Values

Fields	Reset Values
Base address	0x000000
Block size	2 Kbytes
Async/sync Mode	Asynchronous mode
Upper/lower byte	Disabled
Read/write	Disabled
$\overline{AS}/\overline{DS}$	$\overline{AS}$
$\overline{DSACK}$	No wait states
Address space	CPU space
IPL	Any level
Autovector	External interrupt vector

Following reset, the MCU fetches initialization values from the reset vector, beginning at 0x000000 in supervisor program space. The $\overline{CSBOOT}$ chip-select signal is enabled and can select an external boot device mapped to a base address of 0x000000.

The MSB of the CSBTPA field in CSPAR0 has a reset value of one, so that chip-select function is selected by default out of reset. The BYTE field in chip-select option register CSORBT has a reset value of “both bytes” so that the select signal is enabled out of reset. The LSB of the $\overline{CSBOOT}$ field, determined by the logic level of DATA0 during reset, selects the boot ROM port size. When DATA0 is held low during reset, a port size of eight bits is selected. When DATA0 is held high during reset, a port size of 16 bits is selected. DATA0 has a weak internal pull-up device, so that a 16-bit port is selected by default out of reset. As mentioned above, the internal pull-up device can be overcome by bus loading effects. To ensure a particular configuration out of reset, use a pull-up resistor or an active device to place DATA0 in a known state during reset.

The base address field in the boot chip-select base address register CSBARBT has a reset value of all zeros, so that when the initial access to address 0x000000 is made, an address match occurs, and the $\overline{CSBOOT}$ signal is asserted. The block size field in CSBARBT has a reset value of 0b111 (one Mbyte on CPU32-based MCUs and 512 Kbytes on CPU16-based MCUs). **Table 4-41** shows $\overline{CSBOOT}$ reset values.

Table 4-41 $\overline{\text{CSBOOT}}$ Base and Option Register Reset Values

Fields	Reset Values
Base address	0x000000
Block size	1 Mbyte
Async/sync mode	Asynchronous mode
Upper/lower byte	Both bytes
Read/write	Read/write
$\overline{\text{AS}}$/$\overline{\text{DS}}$	$\overline{\text{AS}}$
$\overline{\text{DSACK}}$	13 wait states
Address space	Supervisor space
IPL	Any level
Autovector	External vector externally

4.10 General-Purpose Input/Output

The SCIM2E has six general-purpose input/output ports: A, B, E, F, G, and H. (Port C, an output-only port, is included under the discussion of chip-selects). Ports A, B, and G are available in single-chip mode only and port H is available in single-chip and 8-bit expanded modes only. Ports E, F, G, and H have associated data direction registers to configure each port pin as an input or output. Ports A and B share a data direction register that configures each port entirely as inputs or outputs. Ports E and F have associated pin assignment registers that allow the digital I/O or alternate function of each port pin to be selected. Port F has an edge-detect flag register that indicates whether a transition has occurred on any of its pins.

Table 4-42 shows the shared functions of the general-purpose I/O ports and the modes in which they are available.

Table 4-42 General-Purpose I/O Ports

Port	Shared Function	Modes
A	ADDR[18:11]	Single-chip
B	ADDR[10:3]	Single-chip
E	Bus control signals	All
F	$\overline{\text{IRQ}}$[7:1]/FASTREF	All
G	DATA[15:8]	Single-chip
H	DATA[7:0]	Single-chip, 8-bit expanded

Access to the port A, B, E, G, and H data and data direction registers, and the port E pin assignment register require three clock cycles to ensure timing compatibility with external port replacement logic. Accesses to the port F registers require two clock cycles. Port registers are byte-addressable and are grouped to allow coherent word access to port data register pairs A-B and G-H, as well as word aligned long-word coherency of the port A-B-G-H data registers.

If emulation mode is enabled, accesses to the port A, B, F, G, and H data and data direction registers and the port E pin assignment register are mapped externally, and cause the $\overline{CSE}$ port emulation chip select to be asserted. The SCIM2E does not respond to these accesses, but allows external logic, such as the Motorola MC68HC33 port replacement unit (PRU), to respond. Accesses to the port F registers will still be handled by the SCIM2E.

A write to the port A, B, E, F, G, or H data register is stored in the port's internal data latch. If any port pin is configured as an output, the value stored for that bit is driven on the pin. A read of the port data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the data latch.

4.10.1 Ports A and B

Ports A and B are available in single-chip mode only. One data direction register controls the data direction for both ports. The SCIM2E will respond to port A and B registers at any time the MCU is not in emulation mode.

The port A/B data direction bits (DDA and DDB) control the direction of the pin drivers for ports A and B, respectively. Setting DDA or DDB to one configures all corresponding port pins as outputs. Clearing DDA or DDB to zero configures all corresponding port pins as inputs.

4.10.2 Port A and B Data Registers

PORTA — Port A Data Register **0xYF FA0A**
PORTB — Port B Data Register **0xYF FA0B**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Ports A and B are available in single-chip mode only. PORTA and PORTB can be read or written any time the MCU is not in emulator mode.

4.10.3 Port E

Port E can be made available in all operating modes. The state of $\overline{BERR}$ and DATA8 at the release of $\overline{RESET}$ controls whether the port E pins are initially configured as bus control signals or discrete I/O lines.

If the MCU is in emulation mode, accesses to the port E data, data direction, and pin assignment registers (PORTE, DDRE, and PEPAR) are mapped externally. This allows port replacement logic to be supplied externally, giving an emulator access to the bus control signals.

4.10.3.1 Port E Data Register

PORTE0 — Port E0 Data Register **0xYF FA11**
PORTE1 — Port E1 Data Register **0xYF FA13**

MSB 7	6	5	4	3	2	1	LSB 0
PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
RESET:							
U	U	U	U	U	U	U	U

This register can be accessed in two locations and can be read or written at any time. A write to this register is stored in an internal data latch, and if any pin in the corresponding port is configured as an output, the value stored for that bit is driven out on the pin. A read of this data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register. Bits [15:8] are reserved and will always read zero.

4.10.3.2 Port E Data Direction Register

DDRAB — Port A/B Data Direction Register **0xYF FA14**
DDRE — Port E Data Direction Register **0xYF FA15**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	0	DDA	DDB	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	DDE1	DDE0
RESET:															
								0	0	0	0	0	0	0	0

The port E data direction register controls the direction of the port E pin drivers when pins are configured for I/O. Setting a bit configures the corresponding pin as an output; clearing a bit configures the corresponding pin as an input. This register can be read or written at any time.

The port A/B data direction register controls the direction of the pin drivers for ports A and B, respectively, when the pins are configured for I/O. Setting DDA or DDB to one configures all pins in the corresponding port as outputs. Clearing DDA or DDB to zero configures all pins in the corresponding port as inputs. Bits [15:10] are reserved and will always read zero.

4.10.3.3 Port E Pin Assignment Register

PEPAR — Port E Pin Assignment **0xYF FA17**

MSB 7	6	5	4	3	2	1	LSB 0
PEPA7	PEPA6	PEPA5	PEPA4	PEPA3	PEPA2	PEPA1	PEPA0
RESET:							
DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8	DATA8

This register determines the function of port E pins. Setting a bit assigns the corresponding pin to a bus control signal; clearing a bit assigns the pin to I/O port E. Bits

[15:8] are reserved and will always read zero. **Table 4-43** displays port E pin assignments.

Table 4-43 Port E Pin Assignments

PEPAR Bit	Port E Signal	Bus Control Signal
PEPA7	PE7	SIZ1
PEPA6	PE6	SIZ0
PEPA5	PE5	$\overline{AS}$
PEPA4	PE4	$\overline{DS}$
PEPA3	PE3	$\overline{RMC}$
PEPA2	PE2	$\overline{AVEC}$
PEPA1	PE1	$\overline{DSACK1}$
PEPA0	PE0	$\overline{DSACK0}$

4.10.4 Port F

Port F consists of eight I/O pins, a data register, a data direction register, a pin assignment register, an edge-detect flag register, an edge-detect interrupt vector register, an edge-detect interrupt level register, and associated control logic. **Figure 4-24** is a block diagram of port F pins, registers, and control logic.

Port F pins can be configured as interrupt request inputs, edge-detect input/outputs, or discrete input/outputs. When port F pins are configured for edge detection, and a priority level is specified in the port F edge-detect interrupt level register (PFLVR), the port F control logic will generate an interrupt request when the specified edge is detected. Interrupt vector assignment is made by writing a value to the port F edge-detect interrupt vector register (PFIVR). Edge-detect interrupts have the lowest arbitration priority in the SCIM2E.

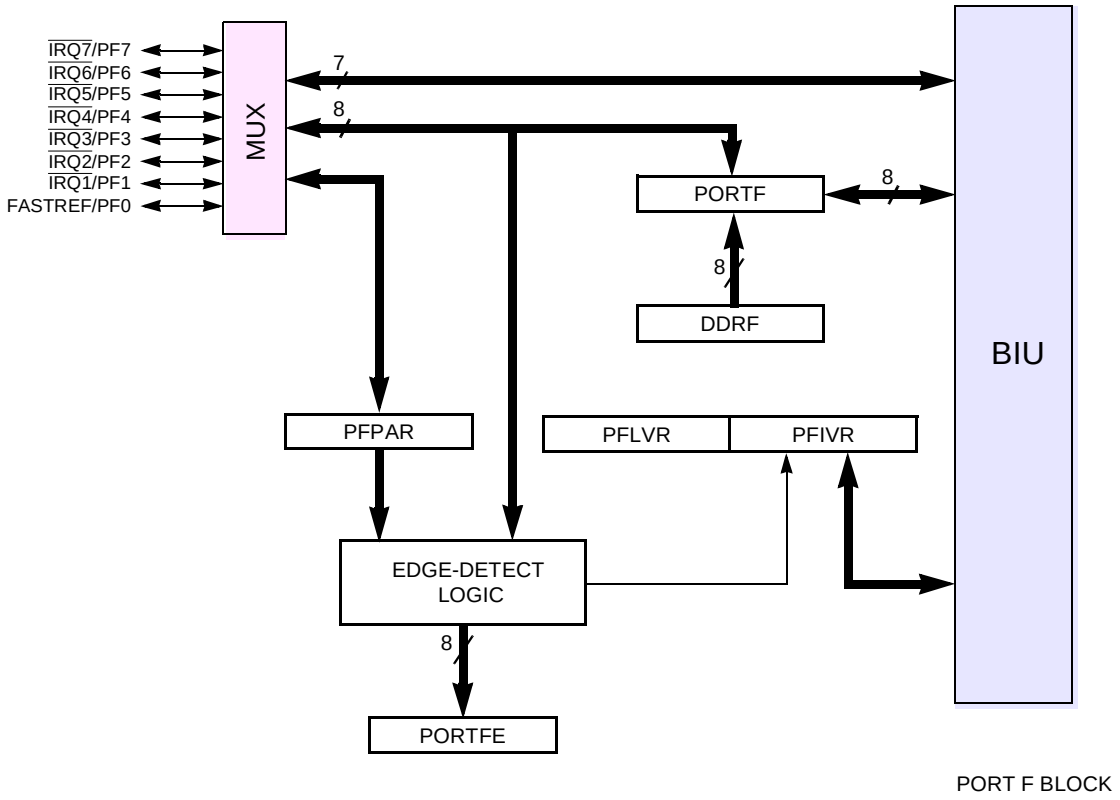


Figure 4-24 Port F Block Diagram

4.10.4.1 Port F Data Register

PORTF0 — Port F Data Register 0 **0xYF FA19**
PORTF1 — Port F Data Register 1 **0xYF FA1B**

MSB 7	6	5	4	3	2	1	LSB 0
PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
RESET:							
U	U	U	U	U	U	U	U

This register can be accessed in two locations and can be read or written at any time. A write to this register is stored in an internal data latch, and if any pin in the corresponding port is configured as an output, the value stored for that bit is driven out on the pin. A read of this data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register. Bits [15:8] are reserved and will always read zero.

4.10.4.2 Port F Data Direction Register

DDRF — Port F Data Direction Register **0xYF FA1D**

MSB 7	6	5	4	3	2	1	LSB 0
DDF7	DDF6	DDF5	DDF4	DDF3	DDF2	DDF1	DDF0
RESET:							
0	0	0	0	0	0	0	0

This register controls the direction of the port F pin drivers when pins are configured for I/O. Setting a bit configures the corresponding pin as an output; clearing a bit configures the corresponding pin as an input. This register can be read or written at any time. Bits [15:8] are reserved and will always read zero.

4.10.4.3 Port F Pin Assignment Register

PFPAR — Port F Pin Assignment Register **0xYF FA1F**

MSB 7	6	5	4	3	2	1	LSB 0
PFPA3		PFPA2		PFPA1		PFPA0	
RESET							
8- AND 16-BIT EXPANDED MODES							
DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9	DATA9
SINGLE-CHIP MODE							
0	0	0	0	0	0	0	0

This register determines the function of port F pins. Setting a bit assigns the corresponding pin to a control signal; clearing a bit assigns the pin to port F. Bits [15:8] are reserved and will always read zero. **Table 4-44** shows port F pin assignments. **Table 4-45** shows PFPAR pin functions.

Table 4-44 Port F Pin Assignments

PFPAR Field	Port F Signal	Alternate Signal
PFPA3	PF[7:6]	$\overline{\text{IRQ[7:6]}}$
PFPA2	PF[5:4]	$\overline{\text{IRQ[5:4]}}$
PFPA1	PF[3:2]	$\overline{\text{IRQ[3:2]}}$
PFPA0	PF[1:0]	$\overline{\text{IRQ1}}$, FASTREF

Table 4-45 PFPAR Pin Functions

PFPAx[1:0]	Port F Signal
00	I/O pin without edge detect
01	Rising edge detect
10	Falling edge detect
11	Interrupt request

4.10.4.4 Port F Edge-Detect Flag Register

PORTFE — Port F Edge-Detect Flag Register **0xYF FA29**

MSB 7	6	5	4	3	2	1	LSB 0
PEF7	PEF6	PEF5	PEF4	PEF3	PEF2	PEF1	PEF0
RESET:							
0	0	0	0	0	0	0	0

When the corresponding pin is configured for edge detection, a PORTFE bit is set if an edge is detected. PORTFE bits remain set, regardless of the subsequent state of the corresponding pin, until cleared. To clear a bit, first read PORTFE, then write the bit to zero. When a pin is configured for general-purpose I/O or for use as an interrupt request input, PORTFE bits do not change state. Bits [15:8] are reserved and will always read zero.

4.10.4.5 Port F Edge-Detect Interrupt Vector

PFIVR — Port F Edge-Detect Interrupt Vector Register **0xYF FA2B**

MSB 7	6	5	4	3	2	1	LSB 0
PFIVR[7:0]							
RESET:							
0	0	0	0	0	0	0	0

This register determines which vector in the exception vector table is used for interrupts generated by the port F edge-detect logic. Program PFIVR[7:0] to the value pointing to the appropriate interrupt vector. Bits [15:8] are reserved and will always read zero.

4.10.4.6 Port F Edge-Detect Interrupt Level

PFLVR — Port F Edge-Detect Interrupt Level Register **0xYF FA2D**

MSB 7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	PFLV[2:0]		
RESET:							
0	0	0	0	0	0	0	0

This register determines the priority level of the port F edge-detect interrupt. The reset value is 0x00, indicating that the interrupt is disabled. When several sources of interrupts from the SCIM are arbitrating for the same level, the port F edge-detect interrupt has the lowest arbitration priority. Bits [15:8] are reserved and will always read zero.

4.10.5 Port G

Port G is available in single-chip mode only. These pins are always configured for use as general-purpose I/O in single-chip mode.

The SCIM2E will respond to port G data register (PORTG) accesses at any time the MCU is not in emulation mode. Reset has no effect on this register.

4.10.5.1 Port G and H Data Registers

PORTG — Port G Data Register **0xYF FA0C**
PORTH — Port H Data Register **0xYF FA0D**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0	PH7	PH6	PH5	PH4	PH3	PH2	PH1	PH0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

Port G is available in single-chip mode only. These pins are always configured for use as general-purpose I/O in single-chip mode.

Port H is available in single-chip and 8-bit expanded modes only. The function of these pins is determined by the operating mode. There is no pin assignment register associated with this port.

These port data registers can be read or written any time the MCU is not in emulation mode. Reset has no effect.

4.10.5.2 Port G and H Data Direction Registers

DDRG — Port G Data Direction Register **0xYF FA0E**
DDRH— Port H Data Direction Register **0xYF FA0F**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
DDG7	DDG6	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0	DDH7	DDH6	DDH5	DDH4	DDH3	DDH2	DDH1	DDH0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The bits in this register control the direction of the port pin drivers when pins are configured as I/O. Setting a bit configures the corresponding pin as an output. Clearing a bit configures the corresponding pin as an input.

4.10.6 Port H

Port H is available in single-chip and 8-bit expanded modes only. The function of these pins is determined by the operating mode. There is no pin assignment register associated with this port.

The SCIM2E will respond to port H data register (PORTH) accesses at any time the MCU is not in emulation mode. Reset has no effect on this register.

The port H data direction register (DDRH) controls the direction of the pin drivers when port H pins are configured for I/O. Setting a bit configures the corresponding pin as an output. Clearing a bit configures the corresponding pin as an input.

SECTION 5
QUEUED ANALOG-TO-DIGITAL CONVERTER MODULE-64

The MC68F375 includes one independent queued analog-to-digital converter (QADC64) module. For details of QADC64 operation not included in this section, refer to the ***QADC Reference Manual*** (QADCRM/AD).

5.1 Overview

The QADC64 modules consist of an analog front-end and a digital control subsystem, which includes an intermodule bus (IMB3) interface block. Refer to **Figure 5-1**.

The analog section includes input pins, channel selection logic, an analog multiplexer, and one sample and hold analog circuit. The analog conversion is performed by the digital-to-analog converter (DAC) resistor-capacitor array, a high-gain comparator, and a successive approximation register (SAR).

The digital control section contains the conversion sequencing logic. Also included are the periodic/interval timer, control and status registers, the conversion command word (CCW) table RAM, and the result word table RAM.

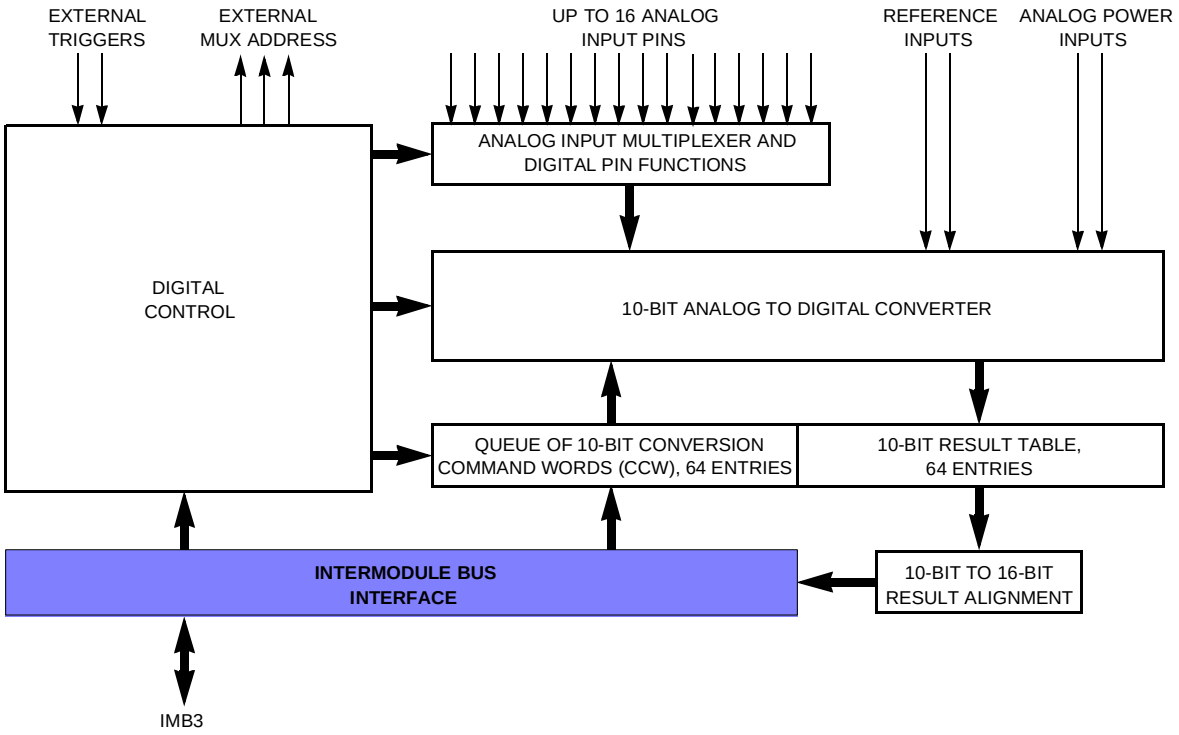


Figure 5-1 QADC64 Block Diagram

5.2 Features

The QADC64 module offers the following features:

- Internal sample and hold
- Up to 16 analog input channels using internal multiplexing
- Directly supports up to four external multiplexers (for example, the MC14051)
- Up to 41 total input channels with internal and external multiplexing
- Programmable input sample time for various source impedances
- Two conversion command queues with a total of 64 entries
- Sub-queues possible using pause mechanism
- Queue complete and pause software interrupts available on both queues
- Queue pointers indicate current location for each queue
- Automated queue modes initiated by:
  - — External edge trigger [queues 1 and 2] and gated mode [queue 1 only]
  - — Periodic/interval timer, within QADC64 module [queues 1 and 2]
  - — Software command [queues 1 and 2]
- Single-scan or continuous-scan of queues
- 64 result registers
- Output data readable in three formats:
  - — Right-justified unsigned
  - — Left-justified signed
  - — Left-justified unsigned
- Unused analog channels can be used as digital ports

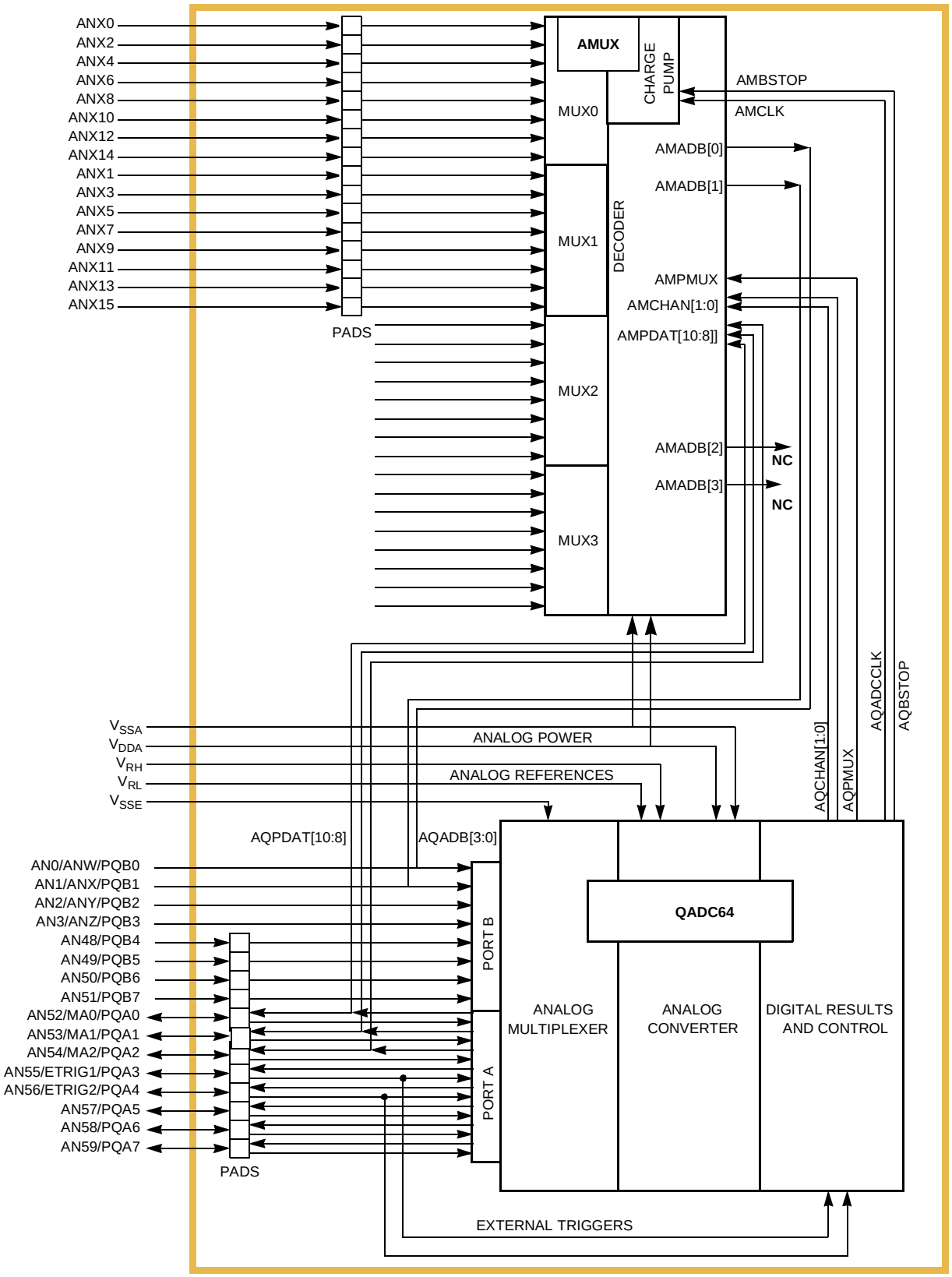
5.3 QADC64 Pin Functions

The QADC64 module uses the following 36 pins:

- 2 analog reference pins, to which all analog input voltages are scaled
- 16 analog input pins with 3 analog inputs multiplexed with multiplex address signals and 2 multiplexed with the on-chip analog multiplexer inputs
- 16 analog input pins through the on-chip AMUX circuit
- 2 analog power pins (V_{DDA}, V_{SSA})
- 2 trigger pins shared with AN[55:56]/PQA[3:4]

The 16 channel/port pins can support up to 41 channels when external multiplexing is used (including internal channels). All of the channel pins can also be used as general-purpose digital port pins.

The following paragraphs describe QADC64 pin functions. **Figure 5-2** shows the QADC64 module pins.



*** Not bonded out on this chip.**

Figure 5-2 QADC64 Input and Output Signals

5.3.1 Port A Pin Functions

The eight port A pins can be used as analog inputs, or as a bidirectional 8-bit digital input/output port.

5.3.1.1 Port A Analog Input Pins

When used as analog inputs, the eight port A pins are referred to as AN[59:52]. Due to the digital output drivers associated with port A, the analog characteristics of port A are different from those of port B. All of the analog signal input pins may be used for at least one other purpose.

5.3.1.2 Port A Digital Input/Output Pins

Port A pins are referred to as PQA when used as a bidirectional 8-bit digital input/output port. These eight pins may be used for general-purpose digital input signals or digital output signals.

Port A pins are connected to a digital input synchronizer during reads and may be used as general purpose digital inputs. Since port A read captures the data on all pins, including those used for digital outputs or analog inputs, the user should employ a “masking” operation to filter the inappropriate bits from the input byte.

Each port A pin is configured as input or output by programming the port data direction register (DDRQA). Digital input signal states are read into the PORTQA data register when DDRQA specifies that the pins are inputs. Digital data in PORTQA is driven onto the port A pins when the corresponding bits in DDRQA specify outputs.

5.3.2 Port B Pin Functions

The eight port B pins can be used as analog inputs, or as an 8-bit digital input only port. Refer to the following paragraphs for more information.

5.3.2.1 Port B Analog Input Pins

When used as analog inputs, the eight port B pins are referred to as AN[51:48]/AN[3:0]. Since port B functions as analog and digital input only, the analog characteristics are different from those of port A. All of the analog signal input pins may be used for at least one other purpose.

5.3.2.2 Port B Digital Input Pins

Port B pins are referred to as PQB[7:0] when used as an 8-bit digital input only port. In addition to functioning as analog input pins, the port B pins are also connected to the input of a synchronizer during reads and may be used as general-purpose digital inputs.

Since port B pins are input only, there is no associated data direction register. Digital input signal states are read from the PORTQB data register. Since a port B read captures the data on all pins, including those used for analog inputs, the user should employ a “masking” operation to filter the inappropriate bits from the input byte.

5.3.3 External Trigger Input Pins

The QADC64 has two external trigger pins (ETRIG[2:1]). Each of the two external trigger pins is associated with one of the scan queues. When a queue is in external trigger mode, the corresponding external trigger pin is configured as a digital input.

5.3.4 Multiplexed Address Output Pins

In non-multiplexed mode, the 16 channel pins are connected to an internal multiplexer which routes the analog signals into the A/D converter.

In externally multiplexed mode, the QADC64 allows automatic channel selection through up to four external 1-of-8 multiplexer chips. The QADC64 provides a 3-bit multiplexed address output to the external multiplexer chips to allow selection of one of eight inputs. The multiplexed address output signals MA[2:0] can be used as multiplex address output bits or as general-purpose I/O.

When externally multiplexed mode is enabled, MA[2:0] are used as the address inputs for up to four 1-of-8 multiplexer chips (for example, the MC14051 and the MC74HC4051). Since MA[2:0] are digital outputs in multiplexed mode, the software programmed input/output direction and data for these pins in DDQA[2:0], DDRQA, and PQA[2:0] is ignored, and the value for MA[2:0] is taken from the currently executing CCW.

5.3.5 Multiplexed Analog Input Pins

In externally multiplexed mode, four of the port B pins are redefined to each represent a group of eight input channels. Refer to **Table 5-1**.

The analog output of each external multiplexer chip is connected to one of the AN[w, x, y, z] inputs in order to convert a channel selected by the MA[2:0] multiplexed address outputs.

Table 5-1 Multiplexed Analog Input Channels

Multiplexed Analog Input	Channels
ANw[1]	Even numbered channels from 0 to 14
ANx[1]	Odd numbered channels from 1 to 15
ANy[2]	Even numbered channels from 16 to 30
ANz[2]	Odd numbered channels from 17 to 31

NOTES:
1. If the on-chip multiplexer is enabled, ANw and ANx are used as inputs for the AMUX outputs.
2. If the on-chip AMUX is enabled, then AN2 and AN3 should be read as channels AN16 and AN17.

5.3.6 Voltage Reference Pins

V_{RH} and V_{RL} are the dedicated input pins for the high and low reference voltages. Separating the reference inputs from the power supply pins allows for additional external

filtering, which increases reference voltage precision and stability, and subsequently contributes to a higher degree of conversion accuracy.

5.3.7 Dedicated Analog Supply Pins

V_{DDA} and V_{SSA} pins supply power to the analog subsystems of the QADC64 module. Dedicated power is required to isolate the sensitive analog circuitry from the normal levels of noise present on the digital power supply.

5.3.8 External Digital Supply Pin

Each port A pin includes a digital output driver, an analog input signal path, and a digital input synchronizer. The V_{SSE} pin provides the ground level for the drivers on the port A pins. V_{DDH} provides the supply level for the drivers on port A pins.

5.3.9 Digital Supply Pins

V_{DD} and V_{SS} provide the power for the digital portions of the QADC64, and for all other digital MCU modules.

5.4 QADC64 Bus Interface

The QADC64 supports 8-bit, 16-bit, and 32-bit data transfers, at even and odd addresses. Coherency of results read, (ensuring that all results read were taken consecutively in one scan) is not guaranteed. For example, if two consecutive 16-bit locations in a result area are read, the QADC64 could change one 16-bit location in the result area between bus cycles. There is no holding register for the second 16-bit location. All read and write accesses that require more than one 16-bit access to complete occur as two or more independent bus cycles. Depending on bus master protocol, these accesses could include misaligned and 32-bit accesses.

Normal reads from and writes to the QADC64 require two clock cycles. However, if the CPU tries to access locations that are also accessible to the QADC64 while the QADC64 is accessing them, the bus cycle will require additional clock cycles. The QADC64 may insert from one-to-four wait states in the process of a CPU read from, or write to, such a location.

5.5 Module Configuration

The QADC64 module configuration register (QADC64MCR) defines freeze and stop mode operation, supervisor space access, and interrupt arbitration priority. Unimplemented bits read zero and writes have no effect. QADC64MCR is typically written once when software initializes the QADC64, and not changed thereafter. Refer to **5.12.1 QADC64 Module Configuration Register** for register and bit descriptions.

5.5.1 Low-Power Stop Mode

When the STOP bit in QADC64MCR is set, the clock signal to the A/D converter is disabled, effectively turning off the analog circuitry. This results in a static, low power consumption, idle condition. Low-power stop mode aborts any conversion sequence in progress. Because the bias currents to the analog circuits are turned off in low-

power stop mode, the QADC64 requires some recovery time (t_{SR}, see **Table E-12** in **APPENDIX E ELECTRICAL CHARACTERISTICS**) to stabilize the analog circuits after the STOP bit is cleared.

In low-power stop mode, the BIU state machine and logic do not shut down: the QADC64MCR and the interrupt register (QADC64INT) are fully accessible and are not reset. The data direction register (DDRQA), port data register (PORTQA/PORTQB), and control register 0 (QACR0) are not reset and are read-only accessible. The RAM is not reset and is not accessible. Control register 1 (QACR1), control register 2 (QACR2), and the status registers (QASR0 and QASR1) are reset and are read-only accessible. In addition, the periodic/interval timer is held in reset during stop mode.

If the STOP bit is clear, low-power stop mode is disabled. The STOP bit must be clear to program CCW's into RAM or read results from RAM.

5.5.2 Freeze Mode

The QADC64 enters freeze mode when background debug mode is enabled and a breakpoint is processed. This is indicated by assertion of the FREEZE line on the IMB3. The FRZ bit in QADC64MCR determines whether or not the QADC64 responds to an IMB FREEZE assertion. Freeze mode is useful when debugging an application.

When the IMB FREEZE line is asserted and the FRZ bit is set, the QADC64 finishes any conversion in progress and then freezes. Depending on when the FREEZE is asserted, there are three possible queue freeze scenarios:

- When a queue is not executing, the QADC64 freezes immediately.
- When a queue is executing, the QADC64 completes the current conversion and then freezes.
- If during the execution of the current conversion, the queue operating mode for the active queue is changed, or a queue 2 abort occurs, the QADC64 freezes immediately.

When the QADC64 enters the freeze mode while a queue is active, the current CCW location of the queue pointer is saved.

During freeze, the analog clock, QCLK, is held in reset and the periodic/interval timer is held in reset. External trigger events that occur during the freeze mode are not captured. The BIU remains active to allow IMB access to all QADC64 registers and RAM. Although the QADC64 saves a pointer to the next CCW in the current queue, the software can force the QADC64 to execute a different CCW by writing new queue operating modes for normal operation. The QADC64 looks at the queue operating modes, the current queue pointer, and any pending trigger events to decide which CCW to execute.

If the FRZ bit is clear, assertion of the IMB FREEZE line is ignored.

5.5.3 Supervisor/Unrestricted Address Space

The QADC64 memory map is divided into two segments: supervisor-only data space and assignable data space. Access to supervisor-only data space is permitted only

when the CPU is operating in supervisor mode. Assignable data space can have either restricted to supervisor-only data space access or unrestricted supervisor and user data space accesses. The SUPV bit in QADC64MCR designates the assignable space as supervisor or unrestricted.

Attempts to read or write supervisor-only data space when the CPU is not in supervisor mode cause the bus master to assert the internal transfer error acknowledge ($\overline{TEA}$) signal.

The supervisor-only data space segment contains the QADC64 global registers, which include QADC64MCR and QADC64INT. The supervisor/unrestricted space designation for the CCW table, the result word table, and the remaining QADC64 registers is programmable.

5.6 General-Purpose I/O Port Operation

QADC64 port pins, when used as general-purpose input, are conditioned by a synchronizer with an enable feature. The synchronizer is not enabled until the QADC64 decodes an IMB bus cycle which addresses the port data register to minimize the high-current effect of mid-level signals on the inputs used for analog signals. Digital input signals must meet the input low voltage (V_{IL}) or input high voltage (V_{IH}) requirements, see **Table E-3** in **APPENDIX E ELECTRICAL CHARACTERISTICS**. If an analog input pin does not meet the digital input pin specifications when a digital port read operation occurs, an indeterminate state is read. To avoid reading inappropriate values on analog inputs, the user software should employ a “masking” operation.

During a port data register read, the actual value of the pin is reported when its corresponding bit in the data direction register defines the pin to be an input (port A only). When the data direction bit specifies the pin to be an output, the content of the port data register is read. By reading the latch which drives the output pin, software instructions (like bit manipulation instructions) that read data, modify it, and write the result work correctly.

There is one special case to consider for digital I/O port operation. When the MUX (externally multiplexed) bit is set in QACR0, the data direction register settings are ignored for the bits corresponding to PQA[2:0] and the three multiplexed address MA[2:0] output pins. The MA[2:0] pins are forced to be digital outputs, regardless of the data direction setting, and the multiplexed address outputs are driven. The data returned during a port data register read is the value of the multiplexed address latches which drive MA[2:0], regardless of the data direction setting.

5.6.1 Port Data Register

QADC64 ports A and B are accessed through two 8-bit port data registers (PORTQA and PORTQB). Port A pins are referred to as PQA when used as an 8-bit input/output port. Port A can also be used for analog inputs AN[59:52] and external multiplexer address outputs MA[2:0].

Port B pins are referred to as PQB when used as an 8-bit input-only digital port. Port B can also be used for non-multiplexed AN[51:48]/AN[3:0] and multiplexed ANz, ANy, ANx, ANw analog inputs.

PORTQA and PORTQB are unaffected by reset. Refer to **5.12.3 Port A/B Data Register** for register and bit descriptions.

5.6.2 Port Data Direction Register

The port data direction register (DDRQA) is associated with the port A digital I/O pins. These bidirectional pins may have somewhat higher leakage and capacitance specifications. Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for more information.

Any bit in this register set to one configures the corresponding pin as an output. Any bit in this register cleared to zero configures the corresponding pin as an input. Software is responsible for ensuring that DDRQA bits are not set to one on pins used for analog inputs. When a DDRQA bit is set to one and the pin is selected for analog conversion, the voltage sampled is that of the output digital driver as influenced by the load.

NOTE

Caution should be exercised when mixing digital and analog inputs. This should be minimized as much as possible. Input pin rise and fall times should be as large as possible to minimize AC coupling effects.

Since port B is input-only, a data direction register is not needed. Read operations on the reserved bits in DDRQA return zeros, and writes have no effect. Refer to **5.12.4 Port Data Direction Register** for register and bit descriptions.

5.7 External Multiplexing Operation

External multiplexers concentrate a number of analog signals onto a few inputs to the analog converter. This is helpful in applications that need to convert more analog signals than the A/D converter can normally provide. External multiplexing also puts the multiplexer closer to the signal source. This minimizes the number of analog signals that need to be shielded due to the close proximity of noisy, high speed digital signals near the MCU.

NOTE

The main QADC64 treats the AMX as an external multiplexer. It is recommended that full external multiplexing not be used on the MC68F375. Mixed internal and external multiplexing should be used. See **5.13.2 Mixed AMUX/External Multiplexing** and **Figure 5-13**.

The QADC64 can use from one-to-four external multiplexers to expand the number of analog signals that may be converted. Up to 32 analog channels can be converted through external multiplexer selection. The externally multiplexed channels are auto-

matically selected from the channel field of the conversion command word (CCW) table, the same as internally multiplexed channels.

All of the automatic queue features are available for externally and internally multiplexed channels. The software selects externally multiplexed mode by setting the MUX bit in QACR0.

Figure 5-3 shows the maximum configuration of four external multiplexers connected to the QADC64. The external multiplexers select one of eight analog inputs and connect it to one analog output, which becomes an input to the QADC64. The QADC64 provides three multiplexed address signals (MA[2:0]), to select one of eight inputs. These outputs are connected to all four multiplexers. The analog output of each multiplexer is each connected to one of four separate QADC64 inputs — ANw, ANx, ANy, and ANz.

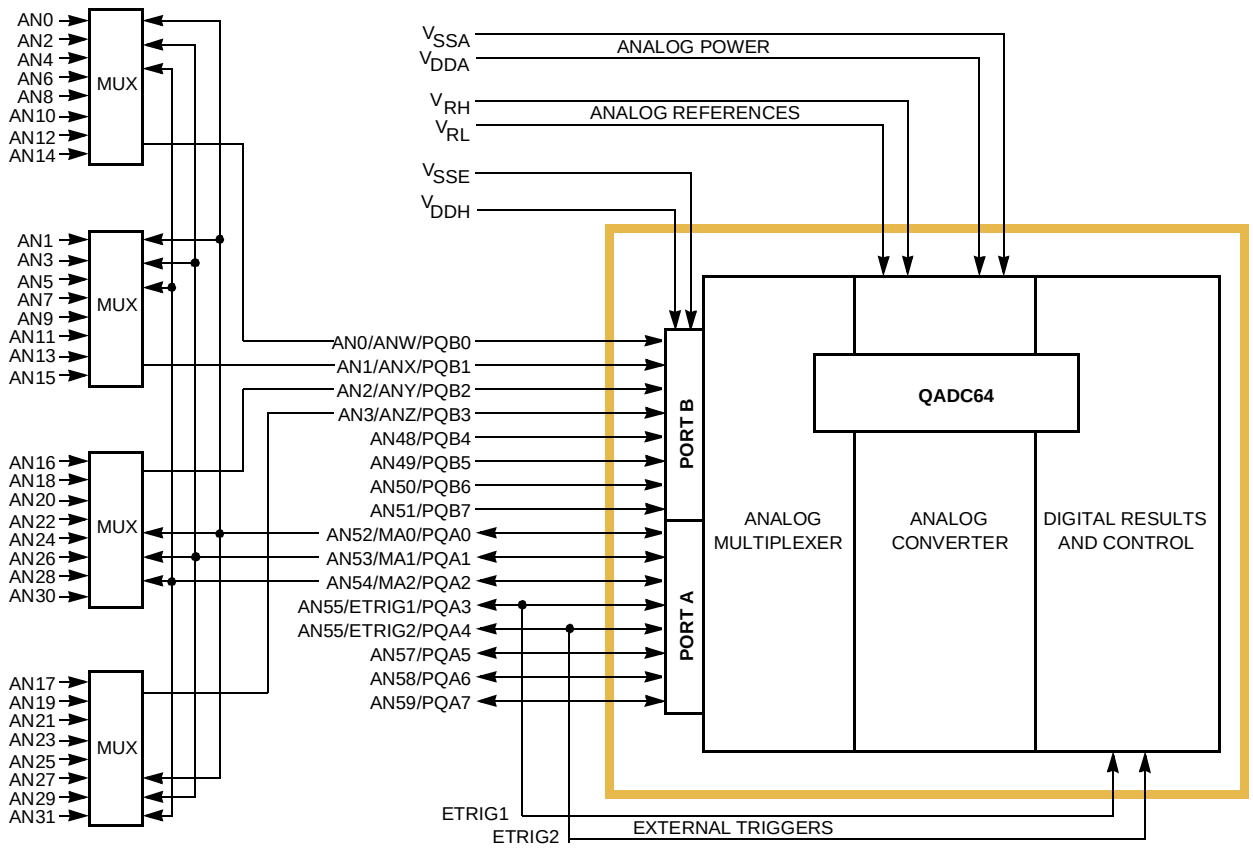


Figure 5-3 Example of Full External Multiplexing

When the external multiplexed mode is selected, the QADC64 automatically creates the MA[2:0] output signals from the channel number in each CCW. The QADC64 also converts the proper input channel (ANw, ANx, ANy, and ANz) by interpreting the CCW channel number. As a result, up to 32 externally multiplexed channels appear to the

conversion queues as directly connected signals. Software simply puts the channel number of an externally multiplexed channel into a CCW.

Figure 5-3 shows that MA[2:0] may also be analog or digital input pins. When external multiplexing is selected, none of the MA[2:0] pins can be used for analog or digital inputs. They become multiplexed address outputs.

5.8 Analog Input Channels

The number of available analog channels varies, depending on whether or not external multiplexing is used. A maximum of 16 analog channels are supported by the internal multiplexing circuitry of the converter. **Table 5-2** shows the total number of analog input channels supported with zero-to-four external multiplexers.

Table 5-2 Analog Input Channels

Number of Analog Input Channels Available Directly Connected + External Multiplexed = Total Channels[1]				
No External Mux Chips	**One External Mux Chip**	**Two External Mux Chips**	**Three External Mux Chips**	**Four External Mux Chips**
16	12 + 8 = 20	11 + 16 = 27	10 + 24 = 34	9 + 32 = 41

NOTES:

1. When external multiplexing is used, three input channels become multiplexed address outputs, and for each external multiplexer chip, one input channel becomes a multiplexed analog input.

5.9 Analog Subsystem

The QADC64 analog subsystem includes a front-end analog multiplexer, a digital-to-analog converter (DAC) array, a comparator, and a successive approximation register (SAR).

The analog subsystem path runs from the input pins through the input multiplexing circuitry, into the DAC array, and through the analog comparator. The output of the comparator feeds into the SAR.

Figure 5-4 shows a block diagram of the QADC64 analog submodule.

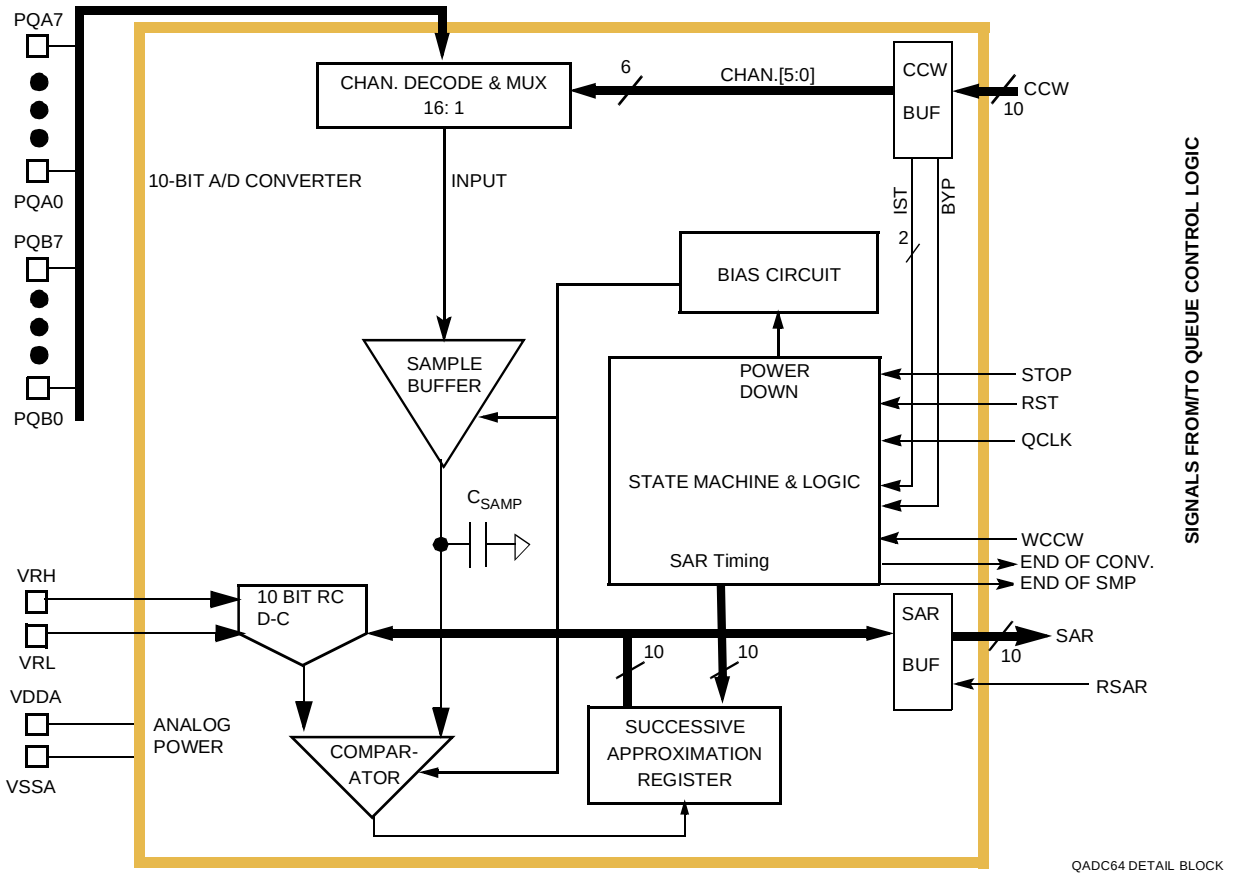


Figure 5-4 QADC64 Module Block Diagram

5.9.1 Conversion Cycle Times

Total conversion time is made up of initial sample time, final sample time, and resolution time. Initial sample time refers to the time during which the selected input channel is driven by the buffer amplifier onto the sample capacitor. The buffer amplifier can be disabled by means of the BYP bit in the CCW. During the final sampling period, amplifier is bypassed, and the multiplexer input charges the RC DAC array directly. During the resolution period, the voltage in the RC DAC array is converted to a digital value and stored in the SAR.

Initial sample time is fixed at two QCLK cycles. Final sample time can be 2, 4, 8, or 16 QCLK cycles, depending on the value of the IST field in the CCW. Resolution time is ten QCLK cycles.

Sample and resolution require a minimum of 14 QCLK clocks (7 µs with a 2 MHz QCLK). If the maximum final sample time period of 16 QCLKs is selected, the total conversion time is 13.0 µs with a 2 MHz QCLK.

Figure 5-5 illustrates the timing for conversions. This diagram assumes a final sampling period of two QCLK cycles.

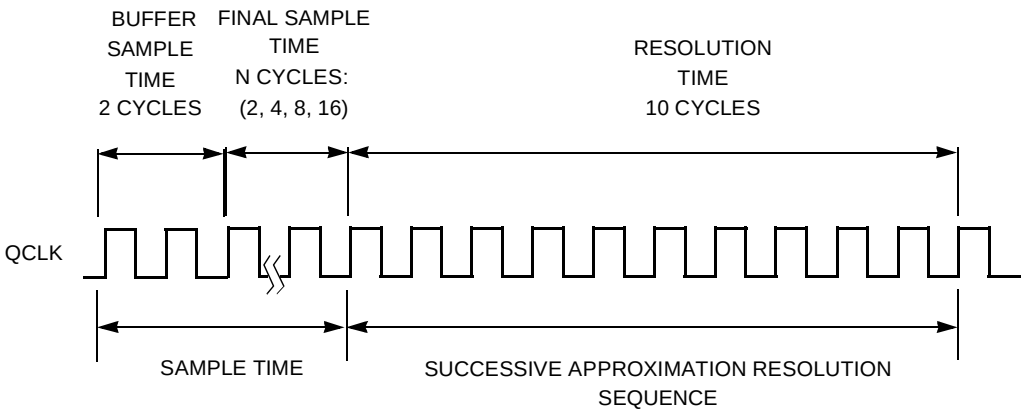


Figure 5-5 Conversion Timing

5.9.1.1 Amplifier Bypass Mode Conversion Timing

If the amplifier bypass mode is enabled for a conversion by setting the amplifier bypass (BYP) bit in the CCW, the timing changes to that shown in **Figure 5-6**. The buffered sample time is eliminated, reducing the potential conversion time by two QCLKs. However, due to internal RC effects, a minimum final sample time of four QCLKs must be allowed. This results in no savings of QCLKs. When using the bypass mode, the external circuit should be of low source impedance, typically less than 10 kΩ. Also, the loading effects of the external circuitry by the QADC64 need to be considered, since the benefits of the sample amplifier are not present.

NOTE

Because of internal RC time constants, a sample time of 2 QCKLs in bypass mode for high frequency operation is not recommended.

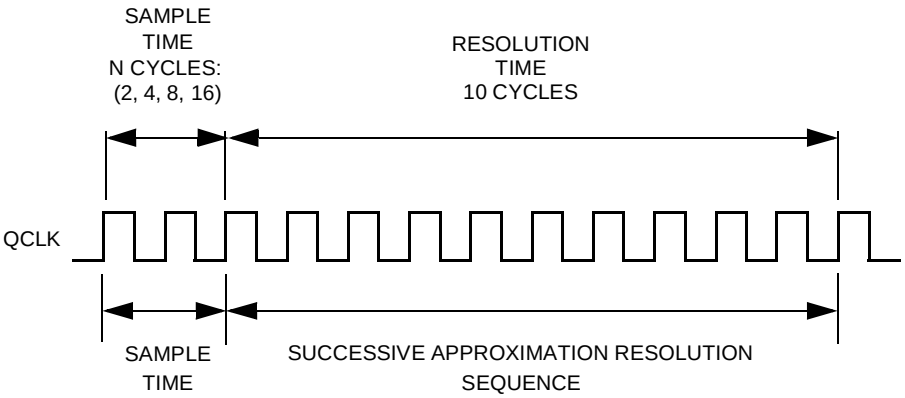


Figure 5-6 Bypass Mode Conversion Timing

5.9.2 Front-End Analog Multiplexer

The internal multiplexer selects one of the 16 analog input pins or one of three special internal reference channels for conversion. The following are the three special channels:

- V_{RH} — Reference voltage high
- V_{RL} — Reference voltage low
- $(V_{RH} – V_{RL})/2$ — Mid-reference voltage

The selected input is connected to one side of the DAC capacitor array. The other side of the DAC array is connected to the comparator input. The multiplexer also includes positive and negative stress protection circuitry, which prevents other channels from affecting the present conversion when excessive voltage levels are applied to the other channels. Refer to **APPENDIX E ELECTRICAL CHARACTERISTICS** for specific voltage level limits.

5.9.3 Digital-to-Analog Converter Array

The digital-to-analog converter (DAC) array consists of binary-weighted capacitors and a resistor-divider chain. The array serves two purposes:

- The array holds the sampled input voltage during conversion.
- The resistor-capacitor array provides the mechanism for the successive approximation A/D conversion.

Resolution begins with the MSB and works down to the LSB. The switching sequence is controlled by the SAR logic.

5.9.4 Comparator

The comparator is used during the approximation process to sense whether the digitally selected arrangement of the DAC array produces a voltage level higher or lower than the sampled input. The comparator output feeds into the SAR which accumulates the A/D conversion result sequentially, starting with the MSB.

5.9.5 Successive Approximation Register

The input of the successive approximation register (SAR) is connected to the comparator output. The SAR sequentially receives the conversion value one bit at a time, starting with the MSB. After accumulating the ten bits of the conversion result, the SAR data is transferred by the queue control logic in the digital section to the appropriate result location, where it may be read by user software.

5.10 Digital Control Subsystem

The digital control subsystem includes the clock and periodic/interval timer, control and status registers, the conversion command word table RAM, and the result word table RAM.

The central element for control of QADC64 conversions is the 64-entry conversion command word (CCW) table. Each CCW specifies the conversion of one input channel. Depending on the application, one or two queues can be established in the CCW

table. A queue is a scan sequence of one or more input channels. By using a pause mechanism, subqueues can be created within the two queues. Each queue can be operated using several different scan modes. The scan modes for queue 1 and queue 2 are programmed in QACR1 and QACR2. Once a queue has been started by a trigger event (any of the ways to cause the QADC64 to begin executing the CCWs in a queue or subqueue), the QADC64 performs a sequence of conversions and places the results in the result word table.

5.10.1 Queue Priority

Queue 1 has execution priority over queue 2 execution. **Table 5-3** shows the conditions under which queue 1 asserts its priority:

Table 5-3 Queue 1 Priority Assertion

Queue State	Result
Inactive	A trigger event for queue 1 or queue 2 causes the corresponding queue execution to begin.
Queue 1 active/trigger event occurs for queue 2	Queue 2 cannot begin execution until queue 1 reaches completion or the paused state. The status register records the trigger event by reporting the queue 2 status as trigger pending. Additional trigger events for queue 2, which occur before execution can begin, are recorded as trigger overruns.
Queue 2 active/trigger event occurs for queue 1	The current queue 2 conversion is aborted. The status register reports the queue 2 status as suspended. Any trigger events occurring for queue 2 while queue 2 is suspended are recorded as trigger overruns. Once queue 1 reaches the completion or the paused state, queue 2 begins executing again. The programming of the resume bit in QACR2 determines which CCW is executed in queue 2.
Simultaneous trigger events occur for queue 1 and queue 2	Queue 1 begins execution and the queue 2 status is changed to trigger pending.
Subqueues paused	The pause feature can be used to divide queue 1 and/or queue 2 into multiple subqueues. A subqueue is defined by setting the pause bit in the last CCW of the subqueue.

Figure 5-7 shows the CCW format and an example of using pause to create subqueues. Queue 1 is shown with four CCWs in each subqueue and queue 2 has two CCWs in each subqueue.

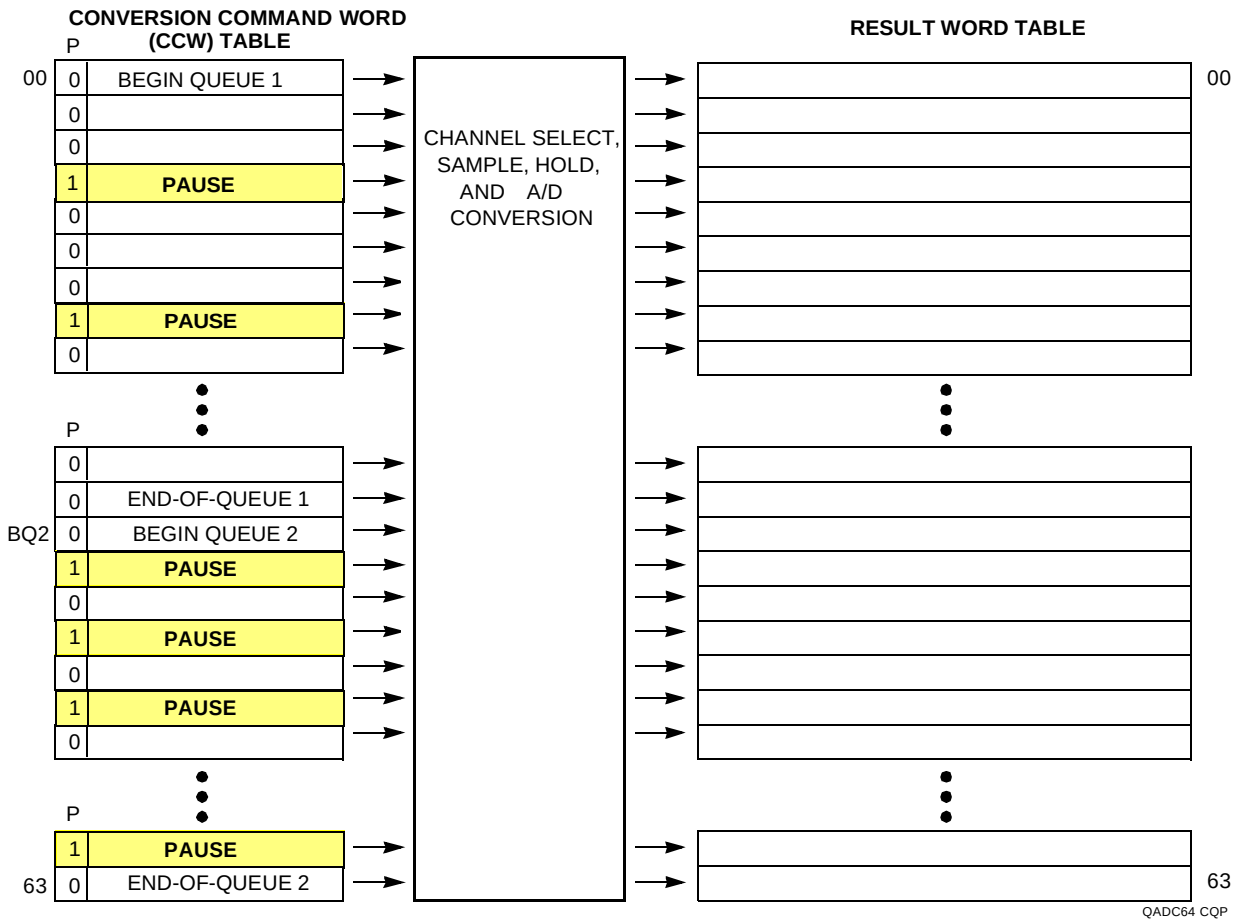


Figure 5-7 QADC64 Queue Operation with Pause

The queue operating mode selected for queue 1 determines what type of trigger event causes the execution of each of the subqueues within queue 1. Similarly, the queue operating mode for queue 2 determines the type of trigger event required to execute each of the subqueues within queue 2.

The choice of single-scan or continuous-scan applies to the full queue, and is not applied to each subqueue. Once a subqueue is initiated, each CCW is executed sequentially until the last CCW in the subqueue is executed and the pause state is entered. Execution can only continue with the next CCW, which is the beginning of the next subqueue. A subqueue cannot be executed a second time before the overall queue execution has been completed.

Trigger events which occur during the execution of a subqueue are ignored, except that the trigger overrun flag is set. When continuous-scan mode is selected, a trigger event occurring after the completion of the last subqueue (after the queue completion flag is set), causes execution to continue with the first subqueue, starting with the first CCW in the queue.

When the QADC64 encounters a CCW with the pause bit set, the queue enters the paused state after completing the conversion specified in the CCW with the pause bit. The pause flag is set and a pause software interrupt may optionally be issued. The status of the queue is shown to be paused, indicating completion of a subqueue. The QADC64 then waits for another trigger event to again begin execution of the next subqueue.

5.10.2 Queue Boundary Conditions

The following are queue operation boundary conditions:

- The first CCW in a queue contains channel 63, the end-of-queue (EOQ) code. The queue becomes active and the first CCW is read. The end-of-queue is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.
- BQ2 (beginning of queue 2) is set at the end of the CCW table (63) and a trigger event occurs on queue 2. **5.12.5 QADC64 Control Register 0 (QACR0)** on BQ2. The end-of-queue condition is recognized, a conversion is performed, the completion flag is set, and the queue becomes idle.
- BQ2 is set to CCW0 and a trigger event occurs on queue 1. After reading CCW0, the end-of-queue condition is recognized, the completion flag is set, and the queue becomes idle. A conversion is not performed.
- BQ2 (beginning of queue 2) is set beyond the end of the CCW table (64–127) and a trigger event occurs on queue 2. Refer to **5.12.7 QADC64 Control Register 2 (QACR2)** for information on BQ2. The end-of-queue condition is recognized immediately, the completion flag is set, and the queue becomes idle. A conversion is not performed.

NOTE

Multiple end-of-queue conditions may be recognized simultaneously, although there is no change in the QADC64 behavior. For example, if BQ2 is set to CCW0, CCW0 contains the EOQ code, and a trigger event occurs on queue 1, the QADC64 reads CCW0 and detects both end-of-queue conditions. The completion flag is set for queue 1 only and it becomes idle.

Boundary conditions also exist for combinations of pause and end-of-queue. One case is when a pause bit is in one CCW and an end-of-queue condition is in the next CCW. The conversion specified by the CCW with the pause bit set completes normally. The pause flag is set. However, since the end-of-queue condition is recognized, the completion flag is also set and the queue status becomes idle, not paused. Examples of this situation include:

- The pause bit is set in CCW5 and the channel 63 (EOQ) code is in CCW6.
- The pause bit is set in CCW63.
- During queue 1 operation, the pause bit is set in CCW14 and BQ2 points to CCW15.

Another pause and end-of-queue boundary condition occurs when the pause and an end-of-queue condition occur in the same CCW. Both the pause and end-of-queue conditions are recognized simultaneously. The end-of-queue condition has precedence so a conversion is not performed for the CCW and the pause flag is not set. The QADC64 sets the completion flag and the queue status becomes idle. Examples of this situation are:

- The pause bit is set in CCW0 and EOQ is programmed into CCW0.
- During queue 1 operation, the pause bit is set in CCW20, which is also BQ2.

5.10.3 Scan Modes

The QADC64 queuing mechanism provides several methods for automatically scanning input channels. In single-scan mode, a single pass through a sequence of conversions defined by a queue is performed. In continuous-scan mode, multiple passes through a sequence of conversions defined by a queue are executed. The possible modes are:

- Disabled and reserved mode
- Software initiated single-scan mode
- External trigger single-scan mode
- External gated single-scan mode (queue 1 only)
- Interval timer single-scan mode
- Software initiated continuous-scan mode
- External trigger continuous-scan mode
- External gated continuous-scan mode (queue 1 only)
- Interval timer continuous-scan mode

The following paragraphs describe the disabled/reserved, single-scan, and continuous-scan operations.

5.10.3.1 Disabled Mode

When the disabled mode is selected, the queue is not active. Trigger events cannot initiate queue execution. When both queue 1 and queue 2 are disabled, wait states are not encountered for IMB accesses of the RAM. When both queues are disabled, it is safe to change the QCLK prescaler values.

5.10.3.2 Reserved Mode

Reserved mode allows for future mode definitions. When the reserved mode is selected, the queue is not active.

CAUTION

Do not use a reserved mode. Unspecified operations may result.

5.10.3.3 Single-Scan Modes

When the application software wants to execute a single pass through a sequence of conversions defined by a queue, a single-scan queue operating mode is selected. By programming the MQ field in QACR1 or QACR2, the following modes can be selected:

- Software initiated single-scan mode
- External trigger single-scan mode
- External gated single-scan mode (queue 1 only)
- Interval timer single-scan mode

NOTE

Queue 2 can not be programmed for external gated single-scan mode.

In all single-scan queue operating modes, the software must also enable the queue to begin execution by writing the single-scan enable bit to a one in the queue's control register. The single-scan enable bits, SSE1 and SSE2, are provided for queue 1 and queue 2 respectively.

Until the single-scan enable bit is set, any trigger events for that queue are ignored. The single-scan enable bit may be set to a one during the write cycle, which selects the single-scan queue operating mode. The single-scan enable bit can be written as a one or a zero, but is always read as a zero. The completion flag, completion interrupt, or queue status are used to determine when the queue has completed.

After the single-scan enable bit is set, a trigger event causes the QADC64 to begin execution with the first CCW in the queue. The single-scan enable bit remains set until the queue is completed. After the queue reaches completion, the QADC64 resets the single-scan enable bit to zero. If the single-scan enable bit is written to a one or a zero by the software before the queue scan is complete, the queue is not affected. However, if the software changes the queue operating mode, the new queue operating mode and the value of the single-scan enable bit are recognized immediately. The conversion in progress is aborted and the new queue operating mode takes effect.

In the software initiated single-scan mode, the writing of a 1 to the single-scan enable bit causes the QADC64 to internally generate a trigger event and the queue execution begins immediately. In the other single-scan queue operating modes, once the single-scan enable bit is written, the selected trigger event must occur before the queue can start. The single-scan enable bit allows the entire queue to be scanned once. A trigger overrun is captured if a trigger event occurs during queue execution in the external trigger single-scan mode and the interval timer single-scan mode.

In the interval timer single-scan mode, the next expiration of the timer is the trigger event for the queue. After the queue execution is complete, the queue status is shown as idle. The software can restart the queue by setting the single-scan enable bit to a 1. Queue execution begins with the first CCW in the queue.

Software Initiated Single-Scan Mode. Software can initiate the execution of a scan sequence for queue 1 or 2 by selecting the software initiated single-scan mode, and writing the single-scan enable bit in QACR1 or QACR2. A trigger event is generated internally and the QADC64 immediately begins execution of the first CCW in the queue. If a pause occurs, another trigger event is generated internally, and then execution continues without pausing.

The QADC64 automatically performs the conversions in the queue until an end-of-queue condition is encountered. The queue remains idle until the software again sets the single-scan enable bit. While the time to internally generate and act on a trigger event is very short, software can momentarily read the status conditions, indicating that the queue is paused. The trigger overrun flag is never set while in the software initiated single-scan mode.

The software initiated single-scan mode is useful in the following applications:

- Allows software complete control of the queue execution.
- Allows the software to easily alternate between several queue sequences.

External Trigger Single-Scan Mode. The external trigger single-scan mode is a variation of the external trigger continuous-scan mode, and is also available with both queue 1 and queue 2. The software programs the polarity of the external trigger edge that is to be detected, either a rising or a falling edge. The software must enable the scan to occur by setting the single-scan enable bit for the queue.

The first external trigger edge causes the queue to be executed one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. After the queue is completed, the QADC64 clears the single-scan enable bit. Software may set the single-scan enable bit again to allow another scan of the queue to be initiated by the next external trigger edge.

The external trigger single-scan mode is useful when the input trigger rate can exceed the queue execution rate. Analog samples can be taken in sync with an external event, even though the software is not interested in data taken from every edge. The software can start the external trigger single-scan mode and get one set of data, and at a later time, start the queue again for the next set of samples.

When a pause bit is encountered during external trigger single-scan mode, another trigger event is required for queue execution to continue. Software involvement is not needed to enable queue execution to continue from the paused state.

The external trigger single-scan mode is also useful when the software needs to change the polarity of the external trigger so that both the rising and falling edges cause queue execution.

External Gated Single-Scan Mode. The QADC64 provides external gating for queue 1 only. When external gated single-scan mode is selected, a transition on the associated external trigger pin initiates queue execution. The polarity of the external gated signal is fixed so only a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. Software must enable the scan to occur by setting the single-scan enable bit for queue 1. If a pause in a CCW is encountered, the pause flag **will not set**, and execution continues without pausing.

While the gate is open, queue 1 executes one time. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When queue 1 completes, the QADC64 sets the completion flag (CF1) and clears the

single-scan enable bit. Software may set the single-scan enable bit again to allow another scan of queue 1 to be initiated during the next open gate.

If the gate closes before queue 1 completes execution, the current CCW completes, execution of queue 1 stops, the single-scan enable bit is cleared, and the PF1 bit is set. Software can read the CWPQ1 to determine the last valid conversion in the queue. Software must set the single-scan enable bit again and should clear the PF1 bit before another scan of queue 1 is initiated during the next open gate. The start of queue 1 is always the first CCW in the CCW table.

Interval Timer Single-Scan Mode. Both queues can use the periodic/interval timer in a single-scan queue operating mode. The timer interval can range from 128 to 128K QCLK cycles in binary multiples. When the interval timer single-scan mode is selected and the software sets the single-scan enable bit in QACR1(2), the timer begins counting. When the time interval elapses, an internal trigger event is created to start the queue and the QADC64 begins execution with the first CCW.

The QADC64 automatically performs the conversions in the queue until a pause or an end-of-queue condition is encountered. When a pause occurs, queue execution stops until the timer interval elapses again, and queue execution continues. When the queue execution reaches an end-of-queue situation the single-scan enable bit is cleared. Software may set the single-scan enable bit again, allowing another scan of the queue to be initiated by the interval timer.

The interval timer generates a trigger event whenever the time interval elapses. The trigger event may cause the queue execution to continue following a pause, or may be considered a trigger overrun. Once the queue execution is completed, the single-scan enable bit must be set again to enable the timer to count again.

Normally, only one queue will be enabled for interval timer single-scan mode and the timer will reset at the end-of-queue. However, if both queues are enabled for either single-scan or continuous interval timer mode, the end-of-queue condition will not reset the timer while the other queue is active. In this case, the timer will reset when both queues have reached end-of-queue.

The interval timer single-scan mode can be used in applications which need coherent results, for example:

- When it is necessary that all samples are guaranteed to be taken during the same scan of the analog pins.
- When the interrupt rate in the periodic timer continuous-scan mode would be too high.
- In sensitive battery applications, where the single-scan mode uses less power than the software initiated continuous-scan mode.

5.10.3.4 Continuous-Scan Modes

When the application software wants to execute multiple passes through a sequence of conversions defined by a queue, a continuous-scan queue operating mode is

selected. By programming the MQ1(2) field in QACR1(2), the following software initiated modes can be selected:

- Software initiated continuous-scan mode
- External trigger continuous-scan mode
- External gated continuous-scan mode (queue 1 only)
- Interval timer continuous-scan mode

When a queue is programmed for a continuous-scan mode, the single-scan enable bit in the queue control register does not have any meaning or effect. As soon as the queue operating mode is programmed, the selected trigger event can initiate queue execution.

In the case of the software initiated continuous-scan mode, the trigger event is generated internally and queue execution begins immediately. In the other continuous-scan queue operating modes, the selected trigger event must occur before the queue can start. A trigger overrun is captured if a trigger event occurs during queue execution in the external trigger continuous-scan mode and the periodic timer continuous-scan mode.

After the queue execution is complete, the queue status is shown as idle. Since the continuous-scan queue operating modes allow the entire queue to be scanned multiple times, software involvement is not needed to enable queue execution to continue from the idle state. The next trigger event causes queue execution to begin again, starting with the first CCW in the queue.

NOTE

In this version of QADC64, coherent samples can be guaranteed. The time between consecutive conversions has been designed to be consistent, provided the sample time bits in both the CCW and IST are identical. However, there is one exception. For queues that end with a CCW containing EOQ code (channel 63), the last queue conversion to the first queue conversion requires 1 additional CCW fetch cycle. Therefore continuous samples are not coherent at this boundary.

In addition, the time from trigger to first conversion can not be guaranteed since it is a function of clock synchronization, programmable trigger events, queue priorities, and other factors.

Software Initiated Continuous-Scan Mode. When the software initiated continuous-scan mode is programmed, the trigger event is generated automatically by the QADC64. Queue execution begins immediately. If a pause is encountered, another trigger event is generated internally, and then execution continues without pausing. When the end-of-queue is reached, another internal trigger event is generated, and queue execution begins again from the beginning of the queue.

While the time to internally generate and act on a trigger event is very short, software can momentarily read the status conditions, indicating that the queue is idle. The trigger overrun flag is never set while in the software initiated continuous-scan mode.

The software initiated continuous-scan mode keeps the result registers updated more frequently than any of the other queue operating modes. The software can always read the result table to get the latest converted value for each channel. The channels scanned are kept up to date by the QADC64 without software involvement. Software can read a result value at any time.

The software initiated continuous-scan mode may be chosen for either queue, but is normally used only with queue 2. When the software initiated continuous-scan mode is chosen for queue 1, that queue operates continuously and queue 2, being lower in priority, never gets executed. The short interval of time between a queue 1 completion and the subsequent trigger event is not sufficient to allow queue 2 execution to begin.

The software initiated continuous-scan mode is a useful choice with queue 2 for converting channels that do not need to be synchronized to anything, or for the slow-to-change analog channels. Interrupts are normally not used with the software initiated continuous-scan mode. Rather, the software reads the latest conversion result from the result table at any time. Once initiated, software action is not needed to sustain conversions of channel. Data read at different locations, however, may or may not be coherent (that is, from the same queue scan sequence).

External Trigger Continuous-Scan Mode. The QADC64 provides external trigger pins for both queues. When the external trigger software initiated continuous-scan mode is selected, a transition on the associated external trigger pin initiates queue execution. The polarity of the external trigger signal is programmable, so that the software can choose to begin queue execution on the rising or falling edge. Each CCW is read and the indicated conversions are performed until an end-of-queue condition is encountered. When the next external trigger edge is detected, the queue execution begins again automatically. Software initialization is not needed between trigger events.

When a pause bit is encountered in external trigger continuous-scan mode, another trigger event is required for queue execution to continue. Software involvement is not needed to enable queue execution to continue from the paused state.

Some applications need to synchronize the sampling of analog channels to external events. There are cases when it is not possible to use software initiation of the queue scan sequence, since interrupt response times vary.

External Gated Continuous-Scan Mode . The QADC64 provides external gating for queue 1 only. When external gated continuous-scan mode is selected, a transition on the associated external trigger pin initiates queue execution. The polarity of the external gated signal is fixed so a high level opens the gate and a low level closes the gate. Once the gate is open, each CCW is read and the indicated conversions are performed until the gate is closed. When the gate opens again, the queue execution automatically begins again from the beginning of the queue. Software initialization is not needed between trigger events. If a pause in a CCW is encountered, the pause flag **will not set**, and execution continues without pausing.

The purpose of external gated continuous-scan mode is to continuously collect digitized samples while the gate is open and to have the most recent samples available. To ensure consistent sample times in waveform digitizing, for example, the programmer must ensure that all CCW's have identical sample time settings in IST.

It is up to the programmer to ensure that the queue is large enough so that a maximum gate open time will not reach an end-of-queue. However it is useful to take advantage of a smaller queue in the manner described below.

In the event that the queue completes before the gate closes, a completion flag will be set and the queue will roll over to the beginning and continue conversions until the gate closes. If the gate remains open and the queue completes a second time, the trigger overrun flag will be set and the queue will roll over again. The queue will continue to execute until the gate closes or the mode is disabled.

If the gate closes before queue 1 completes execution, the current CCW completes and execution of queue 1 stops and QADC64 sets the PF1 bit to indicate an incomplete queue. Software can read the CWPQ1 to determine the last valid conversion in the queue. In this mode, if the gate opens again, execution of queue 1 begins again. The start of queue 1 is always the first CCW in the CCW table.

Interval Timer Continuous-Scan Mode. The QADC64 includes a dedicated periodic/interval timer for initiating a scan sequence on queue 1 and/or queue 2. Software selects a programmable timer interval ranging from 128 to 128K times the QCLK period in binary multiples. The QCLK period is prescaled down from the intermodule bus (IMB) MCU clock.

When a periodic timer continuous-scan mode is selected for queue 1 and/or queue 2, the timer begins counting. After the programmed interval elapses, the timer generated trigger event starts the appropriate queue. Meanwhile, the QADC64 automatically performs the conversions in the queue until an end-of-queue condition or a pause is encountered. When a pause occurs, the QADC64 waits for the periodic interval to expire again, then continues with the queue. Once end-of-queue has been detected, the next trigger event causes queue execution to begin again with the first CCW in the queue.

The periodic timer generates a trigger event whenever the time interval elapses. The trigger event may cause the queue execution to continue following a pause or queue completion, or may be considered a trigger overrun. As with all continuous-scan queue operating modes, software action is not needed between trigger events.

Software enables the completion interrupt when using the periodic timer continuous-scan mode. When the interrupt occurs, the software knows that the periodically collected analog results have just been taken. The software can use the periodic interrupt to obtain non-analog inputs as well, such as contact closures, as part of a periodic look at all inputs.

5.10.4 QADC64 Clock (QCLK) Generation

Figure 5-8 is a block diagram of the clock subsystem. The QCLK provides the timing for the A/D converter state machine, which controls the timing of the conversion. The QCLK is also the input to a 17-stage binary divider which implements the periodic/interval timer. To retain the specified analog conversion accuracy, the QCLK frequency (F_{QCLK}) must be within the tolerance specified in **APPENDIX E ELECTRICAL CHARACTERISTICS**.

Before using the QADC64, the software must initialize the prescaler with values that put the QCLK within the specified range. Though most software applications initialize the prescaler once and do not change it, write operations to the prescaler fields are permitted.

NOTE

For software compatibility with earlier versions of QADC64, the definition of PSL, PSH, and PSA have been maintained. However, the requirements on minimum time and minimum low time no longer exist.

CAUTION

A change in the prescaler value while a conversion is in progress is likely to corrupt the result from any conversion in progress. Therefore, any prescaler write operation should be done only when both queues are in the disabled modes.

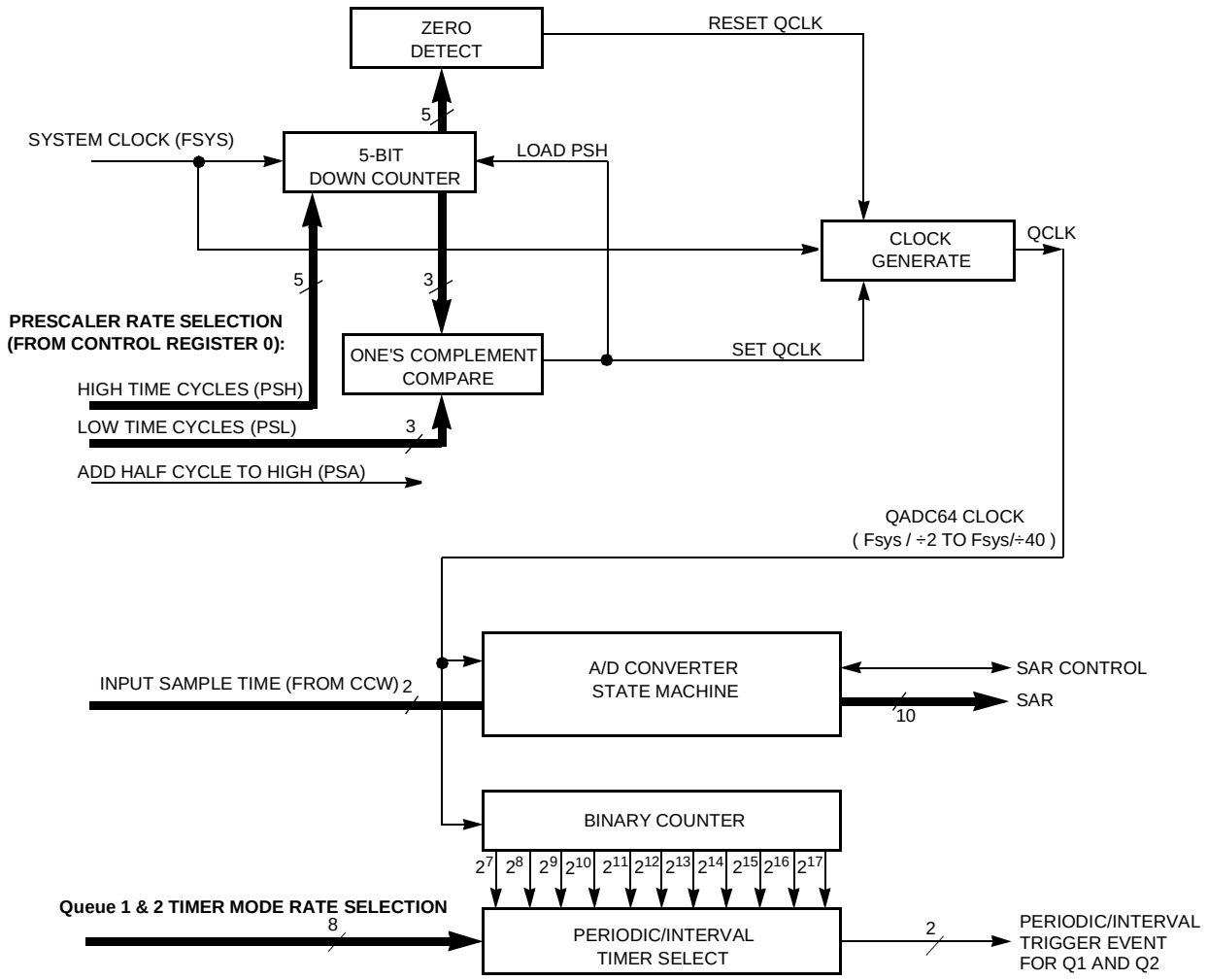


Figure 5-8 QADC64 Clock Subsystem Functions

To accommodate wide variations of the main MCU clock frequency (IMB system clock – F_{SYS}), QCLK is generated by a programmable prescaler which divides the MCU system clock to a frequency within the specified QCLK tolerance range. To allow the A/D conversion time to be maximized across the spectrum of system clock frequencies, the QADC64 prescaler permits the frequency of QCLK to be software selectable. It also allows the duty cycle of the QCLK waveform to be programmable.

The software establishes the basic high phase of the QCLK waveform with the PSH (prescaler clock high time) field in QACR0, and selects the basic low phase of QCLK with the PSL (prescaler clock low time) field. The combination of the PSH and PSL parameters establishes the frequency of the QCLK.

NOTE

The guideline for selecting PSH and PSL is select is to maintain approximately 50% duty cycle. So for prescaler values less then 16, or PSH ≈ PSL. For prescaler values greater than 16 keep PSL as large as possible.

Figure 5-8 shows that the prescaler is essentially a variable pulse width signal generator. A 5-bit down counter, clocked at the system clock rate, is used to create both the high phase and the low phase of the QCLK signal. At the beginning of the high phase, the 5-bit counter is loaded with the 5-bit PSH value. When the zero detector finds that the high phase is finished, the QCLK is reset. A 3-bit comparator looks for a one's complement match with the 3-bit PSL value, which is the end of the low phase of the QCLK. The PSA bit was maintained for software compatibility, but has no effect on QADC64.

The following equations define Qclk frequency:

$$\text{High QCLK Time} = (\text{PSH} + 1) \div F_{SYS}$$

$$\text{Low QCLK Time} = (\text{PSL} + 1) \div F_{SYS}$$

$$F_{QCLK} = 1 \div (\text{High QCLK Time} + \text{Low QCLK Time})$$

Where:

- PSH = 0 to 31, the prescaler QCLK high cycles in QACR0
- PSL = 0 to 7, the prescaler QCLK low cycles in QACR0
- F_{SYS} = System clock frequency
- F_{QCLK} = QCLK frequency

The following are equations for calculating the QCLK high and low phases in example 1 shown in **Figure 5-9**:

$$\text{High QCLK Time} = (11 + 1) \div 40 \times 10^6 = 300 \text{ ns}$$

$$\text{Low QCLK Time} = (7 + 1) \div 40 \times 10^6 = 200 \text{ ns}$$

$$F_{QCLK} = 1/(300 + 200) = 2 \text{ Mhz}$$

The following are equations for calculating the QCLK high and low phases in example 2 shown in **Figure 5-9**:

$$\text{High QCLK Time} = (7 + 1) \div 32 \times 10^6 = 250 \text{ ns}$$

$$\text{Low QCLK Time} = (7 + 1) \div 32 \times 10^6 = 250 \text{ ns}$$

$$F_{QCLK} = 1/(250 + 250) = 2 \text{ Mhz}$$

Figure 5-9 and **Table 5-4** show examples of QCLK programmability. The examples include conversion times based on the following assumption:

- Input sample time is as fast as possible (IST = 0, 2 QCLK cycles).

Figure 5-9 and **Table 5-4** also show the conversion time calculated for a single conversion in a queue. For other MCU system clock frequencies and other input sample times, the same calculations can be made.

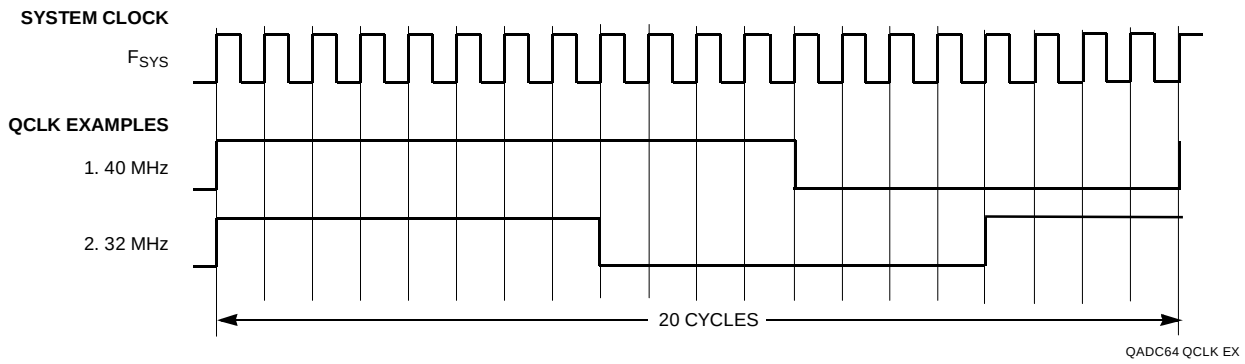


Figure 5-9 QADC64 Clock Programmability Examples

Table 5-4 QADC64 Clock Programmability

Control Register 0 Information					Input Sample Time (IST) = 0b00	
Example Number	Frequency	PSH	PSA	PSL	QCLK (MHz)	Conversion Time (µs)
1	40 Mhz	11	0	7	2.0	7.0
2	32 Mhz	7	0	7	2.0	7.0

NOTE

PSA is maintained for software compatibility but has no functional benefit to this version of the module.

The MCU system clock frequency is the basis of the QADC64 timing. The QADC64 requires that the system clock frequency be at least twice the QCLK frequency. The QCLK frequency is established by the combination of the PSH and PSL parameters in QACR0. The 5-bit PSH field selects the number of system clock cycles in the high phase of the QCLK wave. The 3-bit PSL field selects the number of system clock cycles in the low phase of the QCLK wave.

Example 1 in **Figure 5-9** shows that when PSH = 11, the QCLK remains high for twelve cycles of the system clock. It also shows that when PSL = 7, the QCLK remains low for eight system clock cycles. In example 2, PSH = 7, the QCLK remains high for eight cycles of the system clock. It also shows that when PSL = 7, the QCLK remains low for eight system clock cycles.

5.10.5 Periodic/Interval Timer

The on-chip periodic/interval timer is enabled to generate trigger events at a programmable interval, initiating execution of queue 1 and/or 2. The periodic/interval timer stays reset under the following conditions:

- Queue 1 and queue 2 are programmed to any queue operating mode which does not use the periodic/interval timer
- Interval timer single-scan mode is selected, but the single-scan enable bit is set to zero
- IMB system reset or the master reset is asserted
- Stop mode is selected
- Freeze mode is selected

Two other conditions which cause a pulsed reset of the timer are:

- Roll over of the timer counter
- A queue operating mode change from one periodic/interval timer mode to another periodic/interval timer mode, depending on which queues are active in timer mode.

NOTE

The periodic/interval timer will not reset for a queue 2 operating mode change from one periodic/interval timer mode to another periodic/interval timer mode while queue 1 is in an active periodic/interval timer mode.

During the low power stop mode, the periodic/interval timer is held in reset. Since low power stop mode causes QACR1 and QACR2 to be reset to zero, a valid periodic or interval timer mode must be written after stop mode is exited to release the timer from reset.

When the IMB internal FREEZE line is asserted and a periodic or interval timer mode is selected, the timer counter is reset after the conversion in progress completes. When the periodic or interval timer mode has been enabled (the timer is counting), but a trigger event has not been issued, the freeze mode takes effect immediately, and the timer is held in reset. When the internal FREEZE line is negated, the timer counter starts counting from the beginning.

5.11 Interrupts

Interrupt recognition and servicing involve interaction between the integration module, the CPU, and the module requesting interrupt service. This section provides an overview of the QADC interrupt process. Polled operation, an alternative to using interrupts, is discussed along with the different aspects of interrupt operation.

An interrupt is a special form of exception processing. Interrupt requests can be generated on-chip, or can come from external sources. However, the CPU services all interrupt requests as though originated by an on-chip module; to the CPU, an external interrupt request appears to come from the integration module. There are schemes to prioritize all interrupt requests and to arbitrate between simultaneous requests of the

same priority. The QADC is configured to support interrupt acknowledge (IACK) cycles and vector generation.

5.11.1 Interrupt Operation

Figure 5-10 displays the QADC64 interrupt flow.

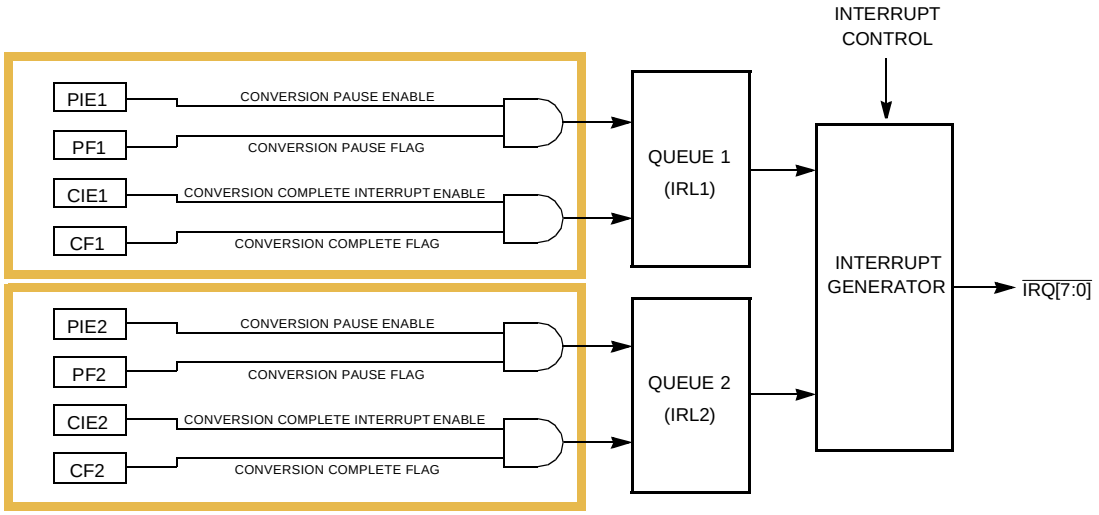


Figure 5-10 QADC64 Interrupt Flow Diagram

5.11.1.1 Polled and Interrupt-Driven Operation

QADC inputs can be monitored by polling or by using interrupts. When interrupts are not needed, software can disable the pause and completion interrupts and monitor the completion flag and the pause flag for each queue in the status register (QASR). In other words, flag bits can be polled to determine when new results are available.

Table 5-5 displays the status flag and interrupt enable bits which correspond to queue 1 and queue 2 activity. If interrupts are enabled for an event, the QADC requests interrupt service when the event occurs. Using interrupts does not require continuously polling the status flags to see if an event has taken place. However, status flags must be cleared after an interrupt is serviced, in order to disable the interrupt request. In both polled and interrupt-driven operating modes, status flags must be re-enabled after an event occurs. Flags are re-enabled by clearing appropriate QASR bits in a particular sequence. The register must first be read, then zeros must be written to the flags that are to be cleared. If a new event occurs betwe en the time that the register is read and the time that it is written, the associate d flag is not cleared.

5.11.2 Interrupt Sources

The QADC includes four sources of interrupt service requests, each of which is separately enabled. Each time the result is written for the last conversion command word (CCW) in a queue, the completion flag for the corresponding queue is set, and when

enabled, an interrupt request is generated. In the same way, each time the result is written for a CCW with the pause bit set, the queue pause flag is set, and when enabled, an interrupt request is generated.

Table 5-5 displays the status flag and interrupt enable bits which correspond to queue 1 and queue 2 activity. The pause and complete interrupts for queue 1 and queue 2 have separate interrupt vector levels, so that each source can be separately serviced.

Table 5-5 QADC64 Status Flags and Interrupt Sources

Queue	Queue Activity	Status Flag	Interrupt Enable Bit
Queue 1	Result written for the last CCW in queue 1	CF1	CIE1
	Result written for a CCW with pause bit set in queue 1	PF1	PIE1
Queue 2	Result written for the last CCW in queue 2	CF2	CIE2
	Result written for a CCW with pause bit set in queue 2	PF2	PIE2

5.11.3 Interrupt Priority

Interrupt priority is determined with a three-bit interrupt priority mask that is located in the bus master condition code register or status register. The interrupt priority mask can have eight possible values, from 0b000 to 0b111.

There are seven levels of interrupt priority, one to seven, each corresponding to a particular interrupt request signal. The bus master compares the priority of each interrupt service request to the mask value. Interrupt request levels greater than the mask value are accepted; interrupt request levels less than or equal to the mask value are ignored, except for the nonmaskable level seven interrupt request, which is serviced even if the bus master interrupt mask value is seven.

The values contained in the IRL1 and IRL2 fields in the interrupt register (QADC64INT) determine the priority of QADC interrupt service requests. A value of 0b000 in either field disables the interrupts associated with that field. IRL1 determines the priority of both queue 1 interrupt sources. IRL2 determines the priority of both queue 2 interrupt sources. As a result, queue 1 and queue 2 can have different priorities in the overall interrupt hierarchy of the MCU. The QADC also has an internal interrupt request prioritization. Queue1 interrupt requests are higher in priority than queue 2 requests, and completion flag requests ar e higher in priority than pause requests.

5.11.4 Interrupt Arbitration

After queue 1 or queue 2 issues an interrupt service request, the bus master performs an interrupt acknowledge cycle. During the interrupt acknowledge cycle, the bus master identifies the interrupt request level being acknowledged by placing it on the address bus. The QADC compares the acknowledged interrupt level with IRL1 and IRL2 values, and responds if the values match.

The same interrupt priority level can be assigned to more than one module. For example, the QADC and the queued serial module (QSM) can both be assigned priority five. If the QADC and the QSM request interrupt service simultaneously, then the interrupt arbitration (IARB) fields in the respective module configuration registers are used to determine which module is serviced first.

The IARB field is essentially a second-level priority in case of a tie. Each module that can request interrupt service has an IARB field. Arbitration is per formed by means of serial contention of IARB field bit values. Arbitration always takes place, even when a single source requests service.

IARB fields contain four bits. An IARB value of 0b1111 has the highest arbitration priority, and 0b0001 has the lowest. If a module with an IARB field value of 0b0000 requests interrupt service, the bus master processes a spurious interrupt exception because the module requesting the interrupt service cannot confirm that it made the request. Initialization software must assign each IARB field a unique non-zero value in order to implement the arbitration scheme. If two or more modules are assigned the same non-zero IARB field value, operation is undefined when interrupts of the same priority level are recognized.

5.11.5 Interrupt Vectors

When the QADC is the only module with an interrupt request pending at the level being acknowledged, or when the QADC IARB value is higher than that of other modules with requests pending at the acknowledged level, the QADC responds to the interrupt acknowledge cycle with an 8-bit interrupt vector number. The CPU uses the vector number to calculate a displacement into the exception vector table, then uses the vector at that location to jump to an interrupt service routine.

The interrupt vector base (IVB) field establishes the six high-order bits of the 8-bit interrupt vector number, and the QADC provides two low-order bits which correspond to one of the four internal QADC interrupt sources. **Figure 5-11** shows the format of the interrupt vector, and lists the binary coding of the two low-order bits for the four QADC interrupt sources.

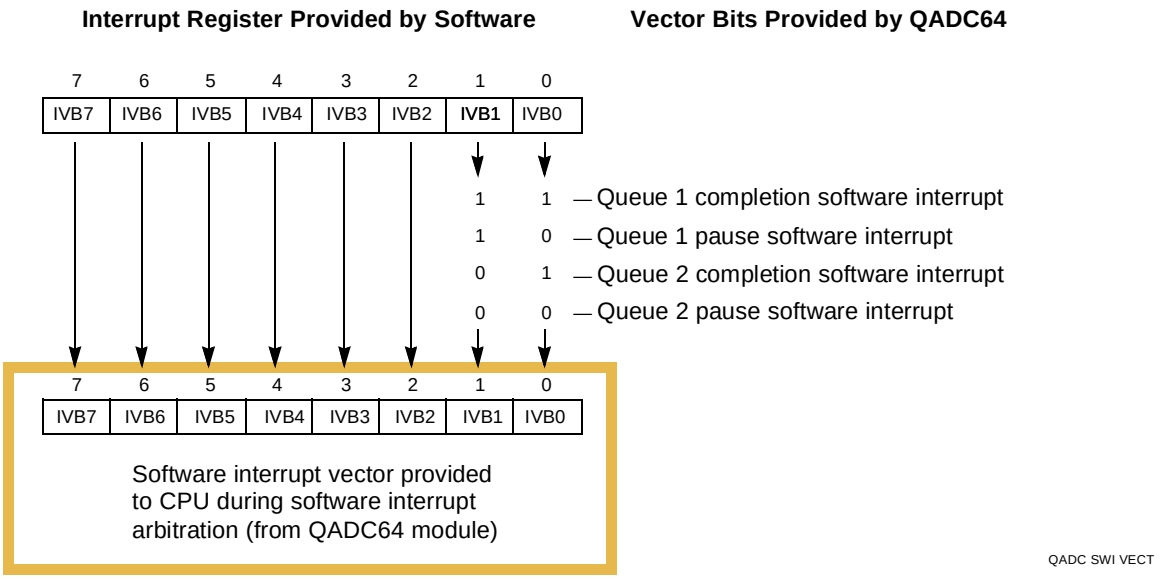


Figure 5-11 QADC64 Interrupt Vector Format

5.12 Programming Model

Each QADC64 occupies 1 Kbyte (512 16-bit entries) of address space. The address space consists of ten 16-bit control, status, and port registers; 64 16-bit entries in the CCW table; and 64 16-bit entries in the result table. The result table occupies 192 16-bit address locations because the result data is readable in three data alignment formats.

Table 5-6 shows the QADC64 memory map. The lowercase “x” appended to each register name represents “A” or “B” for the QADC64_A or QADC64_B module, respectively. The address is the base address of the module. Refer to **APPENDIX A INTERNAL MEMORY MAP** to locate each QADC64 module in the MC68F375 memory map.

Table 5-6 QADC64 Address Map

Access	Address	MSB	LSB
S[1]	0xYF F400	QADC64 Module Configuration Register (QADC64MCR) See **Table 5-7** for bit descriptions.	
T[2]	0xYF F402	QADC64 Test Register (QADC64TEST)	
S	0xYF F404	Interrupt Register (QADC64INT) See **Table 5-8** for bit descriptions.	
S/U[3]	0xYF F406	Port A Data (PORTQA) See **Table 5-10** for bit descriptions.	Port B Data (PORTQB)
S/U	0xYF F408	Port A Data Direction Register (DDRQA) See **Table 5-10** for bit descriptions.	
S/U	0xYF F40A	QADC64 Control Register 0 (QACR0) See **Table 5-11** for bit descriptions.	
S/U	0xYF F40C	QADC64 Control Register 1 (QACR1) See **Table 5-12** for bit descriptions.	
S/U	0xYF F40E	QADC64 Control Register 2 (QACR2) See **Table 5-14** for bit descriptions.	
S/U	0xYF F410	QADC64 Status Register 0 (QASR0) See **Table 5-16** for bit descriptions.	
S/U	0xYF F412	QADC64 Status Register 1 (QASR1) See **Table 5-16** for bit descriptions.	
—	0xYF F414 – 0xYF F5FE	Reserved	
S/U	0xYF F600 — 0xYF F67F	Conversion Command Word (CCW) Table See **Table 5-19** for bit descriptions.	
S/U	0xYF F680 – 0xYF F6FE	Result Word Table Right-Justified, Unsigned Result Register (RJURR) See **5.12.11 Result Word Table** for bit descriptions.	
S/U	0xYF F700 – 0xYF F77E	Result Word Table Left-Justified, Signed Result Register (LJSRR) See **5.12.11 Result Word Table** for bit descriptions.	
S/U	0xYF F780 – 0xYF F7FE	Result Word Table Left-Justified, Unsigned Result Register (LJURR) See **5.12.11 Result Word Table** for bit descriptions.	

NOTES:
1. S = Supervisor only
2. Access is restricted to supervisor only and factory test mode only.
3. S/U = Unrestricted or supervisor depending on the state of the SUPV bit in the QADC64MCR.

The QADC64 has two global registers for configuring module operation: the module configuration register (QADC64MCR) and the interrupt register (QADC64INT). The global registers are always defined to be in supervisor data space. The CPU allows software to establish the global registers in supervisor data space and the remaining registers and tables in user space.

All QADC64 analog channel/port pins that are not used for analog input channels can be used as digital port pins. Port values are read/written by accessing the port A and B data registers (PORTQA and PORTQB). Port A pins are specified as inputs or outputs by programming the port data direction register (DDRQA). Port B is an input only port.

5.12.1 QADC64 Module Configuration Register

QADC64MCR — QADC64 Module Configuration Register **0xYF F400**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	FRZ	RESERVED						SUPV	RESERVED			IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 5-7 QADC64MCR Bit Settings

Bit(s)	Name	Description
15	STOP	Low-power stop mode enable. When the STOP bit is set, the clock signal to the QADC64 is disabled, effectively turning off the analog circuitry. 0 = Enable QADC64 clock. 1 = Disable QADC64 clock.
14	FRZ	FREEZE assertion response. The FRZ bit determines whether or not the QADC64 responds to assertion of the IMB3 FREEZE signal. 0 = QADC64 ignores the IMB3 FREEZE signal. 1 = QADC64 finishes any current conversion, then freezes.
13:8	—	Reserved
7	SUPV	Supervisor/unrestricted data space. The SUPV bit designates the assignable space as supervisor or unrestricted. 0 = Only the module configuration register, test register, and interrupt register are designated as supervisor-only data space. Access to all other locations is unrestricted. 1 = All QADC64 registers and tables are designated as supervisor-only data space.
6:4	—	Reserved
3:0	IARB	Interrupt arbitration number. IARB determines QADC64 interrupt arbitration priority. An IARB field can be assigned a value from 0b0001 (lowest priority) to 0b1111 (highest value). Note that the logic associated with the IARB field is implemented for bus masters with interrupt acknowledge cycles (IACK).

5.12.2 QADC64 Interrupt Register

QADC64INT specifies the priority level of QADC64 interrupt requests and the vector provided. The interrupt level for queue 1 and queue 2 may be different. The interrupt register is read/write accessible in supervisor data space only. The implemented interrupt register fields can be read and written, reserved bits read zero and writes have no effect. They are typically written once when the software initializes the QADC, and not changed afterwards.

QADC64INT — QADC64 Interrupt Register **0xYF F404**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
Re-served	IRL1			Re-served	IRL2			IVB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-8 QADC64INT Bit Settings

Bit(s)	Name	Description
15	—	Reserved
14:12	IRL1	Interrupt level for queue 1. A value of 00000 provides an interrupt level of 0; 0b111 provides a level interrupt. All interrupts are presented on the IMB3. Interrupt level priority software determines which level has the highest priority request.
11	—	Reserved
10:8	IRL2	Interrupt level for queue 2. A value of 00000 provides an interrupt level of 0; 0b111 provides a level interrupt. All interrupts are presented on the IMB3. Interrupt level priority software determines which level has the highest priority request.
7:0	IVB	Interrupt vector base. Initialization software inputs the upper six IVB bits in the interrupt register. During interrupt arbitration, the vector provided to the bus master by the QADC is made up of the upper six IVB bits , plus two low-order bits provided to the QADC to identify one of four QADC interrupt requests. The interrupt vector number is independent of the interrupt level and the interrupt arbitration number. A 0x0F vector number corresponds to the uninitialized interrupt vector. After reset, the lower byte of the interrupt register reads as 0x0F. Once the IVB field is written, the two least significant bits always read as zeros.

5.12.3 Port A/B Data Register

QADC64 ports A and B are accessed through two 8-bit port data registers (PORTQA and PORTQB).

PORTQA — Port QA Data Register **0xYF F406**
PORTQB — Port QB Data Register

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PQA7	PQA6	PQA5	PQA4	PQA3	PQA2	PQA1	PQA0	PQB7	PQB6	PQB5	PQB4	PQB3	PQB2	PQB1	PQB0
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
ANALOG CHANNEL:															
AN59	AN58	AN57	AN56	AN55	AN54	AN53	AN52	AN51	AN50	AN49	AN48	AN3	AN2	AN1	AN0
MULTIPLEXED ADDRESS OUTPUTS:															
					MA2	MA1	MA0								
MULTIPLEXED ANALOG INPUTS:															
												ANz	ANy	ANx	ANw

Table 5-9 PORTQA, PORTQB Bit Settings

Bit(s)	Name	Description
15:8	PQA[0:7]	Port A pins are referred to as PQA when used as an 8-bit input/output port. Port A can also be used for analog inputs (AN[59:52]), and external multiplexer address outputs (MA[2:0]).
7:0	PQB[0:7]	Port B pins are referred to as PQB when used as an 8-bit input only port. Port B can also be used for non-multiplexed (AN[51:48])/AN[3:0]) and multiplexed (ANz, ANy, ANx, ANw) analog inputs.

5.12.4 Port Data Direction Register

DDRQA — Port QA Data Direction Register **0xYF F408**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
DDQA7	DDQA6	DDQA5	DDQA4	DDQA3	DDQA2	DDQA1	DDQA0	RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-10 DDRQA Bit Settings

Bit(s)	Name	Description
15:8	DDQA[7:0]	Bits in this register control the direction of the port QA pin drivers when pins are configured for I/O. Setting a bit configures the corresponding pin as an output; clearing a bit configures the corresponding pin as an input. This register can be read or written at any time.
7:0	—	Reserved

5.12.5 QADC64 Control Register 0 (QACR0)

Control register 0 establishes the QCLK with prescaler parameter fields and defines whether external multiplexing is enabled. All of the implemented control register fields can be read or written, reserved fields read zero and writes have no effect. They are typically written once when the software initializes the QADC64, and not changed afterwards.

QACR0 — QADC64 Control Register 0 **0xYF F40A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MUX	RESERVED		TRG	RESERVED			PSH					PSA	PSL		
RESET:															
0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	1

Table 5-11 QACR0 Bit Settings

Bit(s)	Name	Description
15	MUX	Externally multiplexed mode. The MUX bit configures the QADC64 for externally multiplexed mode, which affects the interpretation of the channel numbers and forces the MA[2:0] pins to be outputs. 0 = Internally multiplexed, 16 possible channels. AMUX is disabled. 1 = Externally multiplexed, 41 possible channels. This enables the on-chip AMUX.
14:13	—	Reserved
12	TRG	Trigger assignment. TRG allows the software to assign the ETRIG[2:1] pins to queue 1 and queue 2. 0 = ETRIG1 triggers queue 1; ETRIG2 triggers queue 2 1 = ETRIG1 triggers queue 2; ETRIG2 triggers queue 1
11:9	—	Reserved
8:4	PSH	Prescaler clock high time. The PSH field selects the QCLK high time in the prescaler. PSH value plus 1 represents the high time in system clocks
3	PSA	Note that this bit location is maintained for software compatibility with previous versions of the QADC64. It serves no functional benefit in the MC68F375 and is not operational.
2:0	PSL	Prescaler clock low time. The PSL field selects the QCLK low time in the prescaler. PSL value plus 1 represents the low time in system clocks

5.12.6 QADC64 Control Register 1 (QACR1)

Control register 1 is the mode control register for the operation of queue 1. The applications software defines the queue operating mode for the queue, and may enable a completion and/or pause interrupt. All of the control register fields are read/write data. However, the SSE1 bit always reads as zero unless the test mode is enabled. Most of the bits are typically written once when the software initializes the QADC64, and not changed afterwards.

QACR1 — Control Register 1 **0xYF F40C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CIE1	PIE1	SSE1	MQ1					RESERVED							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-12 QACR1 Bit Settings

Bit(s)	Name	Description
15	CIE1	Queue 1 completion interrupt enable. CIE1 enables completion interrupts for queue 1. The interrupt request is generated when the conversion is complete for the last CCW in queue 1. 0 = Queue 1 completion interrupts disabled. 1 = Generate an interrupt request after completing the last CCW in queue 1.
14	PIE1	Queue 1 pause interrupt enable. PIE1 enables pause interrupts for queue 1. The interrupt request is generated when the conversion is complete for a CCW that has the pause bit set. 0 = Queue 1 pause interrupts disabled. 1 = Generate an interrupt request after completing a CCW in queue 1 which has the pause bit set.
13	SSE1	Queue 1 single-scan enable. SSE1 enables a single-scan of queue 1 after a trigger event occurs. The SSE1 bit may be set to a one during the same write cycle that sets the MQ1 bits for the single-scan queue operating mode. The single-scan enable bit can be written as a one or a zero, but is always read as a zero. The SSE1 bit allows a trigger event to initiate queue execution for any single-scan operation on queue 1. The QADC64 clears SSE1 when the single-scan is complete.
12:8	MQ1	Queue 1 operating mode. The MQ1 field selects the queue operating mode for queue 1. **Table 5-13** shows the different queue 1 operating modes.
7:0	—	Reserved

Table 5-13 Queue 1 Operating Modes

MQ1	Operating Modes
00000	Disabled mode, conversions do not occur
00001	Software triggered single-scan mode (started with SSE1)
00010	External trigger rising edge single-scan mode
00011	External trigger falling edge single-scan mode
00100	Interval timer single-scan mode: time = QCLK period x 2^7
00101	Interval timer single-scan mode: time = QCLK period x 2^8
00110	Interval timer single-scan mode: time = QCLK period x 2^9
00111	Interval timer single-scan mode: time = QCLK period x 2^{10}
01000	Interval timer single-scan mode: time = QCLK period x 2^{11}
01001	Interval timer single-scan mode: time = QCLK period x 2^{12}
01010	Interval timer single-scan mode: time = QCLK period x 2^{13}
01011	Interval timer single-scan mode: time = QCLK period x 2^{14}
01100	Interval timer single-scan mode: time = QCLK period x 2^{15}
01101	Interval timer single-scan mode: time = QCLK period x 2^{16}
01110	Interval timer single-scan mode: time = QCLK period x 2^{17}
01111	External gated single-scan mode (started with SSE1)
10000	Reserved mode
10001	Software triggered continuous-scan mode
10010	External trigger rising edge continuous-scan mode
10011	External trigger falling edge continuous-scan mode
10100	Periodic timer continuous-scan mode: time = QCLK period x 2^7

Table 5-13 Queue 1 Operating Modes (Continued)

MQ1	Operating Modes
10101	Periodic timer continuous-scan mode: time = QCLK period x 2^8
10110	Periodic timer continuous-scan mode: time = QCLK period x 2^9
10111	Periodic timer continuous-scan mode: time = QCLK period x 2^{10}
11000	Periodic timer continuous-scan mode: time = QCLK period x 2^{11}
11001	Periodic timer continuous-scan mode: time = QCLK period x 2^{12}
11010	Periodic timer continuous-scan mode: time = QCLK period x 2^{13}
11011	Periodic timer continuous-scan mode: time = QCLK period x 2^{14}
11100	Periodic timer continuous-scan mode: time = QCLK period x 2^{15}
11101	Periodic timer continuous-scan mode: time = QCLK period x 2^{16}
11110	Periodic timer continuous-scan mode: time = QCLK period x 2^{17}
11111	External gated continuous-scan mode

5.12.7 QADC64 Control Register 2 (QACR2)

Control register 2 is the mode control register for the operation of queue 2. Software specifies the queue operating mode of queue 2, and may enable a completion and/or a pause interrupt. All control register fields are read/write data, except the SSE2 bit, which is readable only when the test mode is enabled. Most of the bits are typically written once when the software initializes the QADC64, and not changed afterwards.

QACR2 — Control Register 2 **0xYF F40E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CIE2	PIE2	SSE2	MQ2					RE-SUME	BQ2						
RESET:															
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Table 5-14 QACR2 Bit Settings

Bit(s)	Name	Description
15	CIE2	Queue 2 completion interrupt enable. CIE2 enables completion interrupts for queue 2. The interrupt request is generated when the conversion is complete for the last CCW in queue 2. 0 = Queue 2 completion interrupts disabled. 1 = Generate an interrupt request after completing the last CCW in queue 2.
14	PIE2	Queue 2 pause interrupt enable. PIE2 enables pause interrupts for queue 2. The interrupt request is generated when the conversion is complete for a CCW that has the pause bit set. 0 = Queue 2 pause interrupts disabled. 1 = Generate an interrupt request after completing a CCW in queue 2 which has the pause bit set.
13	SSE2	Queue 2 single-scan enable bit. SSE2 enables a single-scan of queue 2 after a trigger event occurs. The SSE2 bit may be set to a one during the same write cycle that sets the MQ2 bits for the single-scan queue operating mode. The single-scan enable bit can be written as a one or a zero, but is always read as a zero. The SSE2 bit allows a trigger event to initiate queue execution for any single-scan operation on queue 2. The QADC64 clears SSE2 when the single-scan is complete.
12:8	MQ2	Queue 2 operating mode. The MQ2 field selects the queue operating mode for queue 2.**Table 5-15** shows the bits in the MQ2 field which enable different queue 2 operating modes.
7	RESUME	Queue 2 resume. RESUME selects the resumption point after queue 2 is suspended by queue 1. If RESUME is changed during execution of queue 2, the change is not recognized until an end-of-queue condition is reached, or the queue operating mode of queue 2 is changed. 0 = After suspension, begin execution with the first CCW in queue 2 or the current subqueue. 1 = After suspension, begin execution with the aborted CCW in queue 2.
6:0	BQ2	Beginning of queue 2. The BQ2 field indicates the location in the CCW table where queue 2 begins. The BQ2 field also indicates the end-of-queue 1 and thus creates an end-of-queue condition for queue 1. Setting BQ2 to any value ≥ 64 (1000000) allows the entire RAM space for queue 1 CCW's.

Table 5-15 Queue 2 Operating Modes

MQ2	Operating Modes
00000	Disabled mode, conversions do not occur
00001	Software triggered single-scan mode (started with SSE2)
00010	External trigger rising edge single-scan mode
00011	External trigger falling edge single-scan mode
00100	Interval timer single-scan mode: interval = QCLK period x 2^7
00101	Interval timer single-scan mode: interval = QCLK period x 2^8
00110	Interval timer single-scan mode: interval = QCLK period x 2^9
00111	Interval timer single-scan mode: interval = QCLK period x 2^{10}
01000	Interval timer single-scan mode: interval = QCLK period x 2^{11}
01001	Interval timer single-scan mode: interval = QCLK period x 2^{12}
01010	Interval timer single-scan mode: interval = QCLK period x 2^{13}
01011	Interval timer single-scan mode: interval = QCLK period x 2^{14}
01100	Interval timer single-scan mode: interval = QCLK period x 2^{15}
01101	Interval timer single-scan mode: interval = QCLK period x 2^{16}
01110	Interval timer single-scan mode: interval = QCLK period x 2^{17}
01111	Reserved mode
10000	Reserved mode
10001	Software triggered continuous-scan mode (started with SSE2)
10010	External trigger rising edge continuous-scan mode
10011	External trigger falling edge continuous-scan mode
10100	Periodic timer continuous-scan mode: period = QCLK period x 2^7
10101	Periodic timer continuous-scan mode: period = QCLK period x 2^8
10110	Periodic timer continuous-scan mode: period = QCLK period x 2^9
10111	Periodic timer continuous-scan mode: period = QCLK period x 2^{10}
11000	Periodic timer continuous-scan mode: period = QCLK period x 2^{11}
11001	Periodic timer continuous-scan mode: period = QCLK period x 2^{12}
11010	Periodic timer continuous-scan mode: period = QCLK period x 2^{13}
11011	Periodic timer continuous-scan mode: period = QCLK period x 2^{14}
11100	Periodic timer continuous-scan mode: period = QCLK period x 2^{15}
11101	Periodic timer continuous-scan mode: period = QCLK period x 2^{16}
11110	Periodic timer continuous-scan mode: period = QCLK period x 2^{17}
11111	Reserved mode

5.12.8 QADC64 Status Register 0 (QASR0)

QASR0 contains information about the state of each queue and the current A/D conversion. Except for the four flag bits (CF1, PF1, CF2, and PF2) and the two trigger overrun bits (TOR1 and TOR2), all of the status register fields contain read-only data.

The four flag bits and the two trigger overrun bits are cleared by writing a zero to the bit after the bit was previously read as a one.

QASR0 — QADC64 Status Register **0xYF F410**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CF1	PF1	CF2	PF2	TOR1	TOR2	QS				CWP					
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5-16 QASR0 Bit Settings

Bit(s)	Name	Description
15	CF1	Queue 1 completion flag. CF1 indicates that a queue 1 scan has been completed. CF1 is set by the QADC64 when the conversion is complete for the last CCW in queue 1, and the result is stored in the result table. 0 = Queue 1 scan is not complete. 1 = Queue 1 scan is complete.
14	PF1	Queue 1 pause flag. PF1 indicates that a queue 1 scan has reached a pause. PF1 is set by the QADC64 when the current queue 1 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table. 0 = Queue 1 has not reached a pause. 1 = Queue 1 has reached a pause.
13	CF2	Queue 2 completion flag. CF2 indicates that a queue 2 scan has been completed. CF2 is set by the QADC64 when the conversion is complete for the last CCW in queue 2, and the result is stored in the result table. 0 = Queue 2 scan is not complete. 1 = Queue 2 scan is complete.
12	PF2	Queue 2 pause flag. PF2 indicates that a queue 2 scan has reached a pause. PF2 is set by the QADC64 when the current queue 2 CCW has the pause bit set, the selected input channel has been converted, and the result has been stored in the result table. 0 = Queue 2 has not reached a pause. 1 = Queue 2 has reached a pause.
11	TOR1	Queue 1 trigger overrun. TOR1 indicates that an unexpected queue 1 trigger event has occurred. TOR1 can be set only while queue 1 is active. A trigger event generated by a transition on ETRIG1/ETRIG2 may be recorded as a trigger overrun. TOR1 can only be set when using an external trigger mode. TOR1 cannot occur when the software initiated single-scan mode or the software initiated continuous-scan mode is selected. 0 = No unexpected queue 1 trigger events have occurred. 1 = At least one unexpected queue 1 trigger event has occurred.
10	TOR2	Queue 2 trigger overrun. TOR2 indicates that an unexpected queue 2 trigger event has occurred. TOR2 can be set when queue 2 is in the active, suspended, and trigger pending states. A trigger event generated by a transition depending on the value of TRG in QACR or ETRIG1/ETRIG2 or by the periodic/interval timer may be recorded as a trigger overrun. TOR2 can only be set when using an external trigger mode or a periodic/interval timer mode. Trigger overruns cannot occur when the software initiated single-scan mode and the software initiated continuous-scan mode are selected. 0 = No unexpected queue 2 trigger events have occurred. 1 = At least one unexpected queue 2 trigger event has occurred.

Table 5-16 QASR0 Bit Settings (Continued)

Bit(s)	Name	Description
9:6	QS	Queue status. This 4-bit read-only field indicates the current condition of queue 1 and queue 2. QS[0:1] are associated with queue 1, and QS[2:3] are associated with queue 2. Since the queue priority scheme interlinks the operation of queue 1 and queue 2, the status bits should be considered as one 4-bit field. **Table 5-17** shows the bit encodings of the QS field.
5:0	CWP	Command word pointer. CWP indicates which CCW is executing at present, or was last completed. The CWP is a read-only field; writes to it have no effect.

Table 5-17 Queue Status

QS	Description
0000	Queue 1 idle, queue 2 idle
0001	Queue 1 idle, queue 2 paused
0010	Queue 1 idle, queue 2 active
0011	Queue 1 idle, queue 2 trigger pending
0100	Queue 1 paused, queue 2 idle
0101	Queue 1 paused, queue 2 paused
0110	Queue 1 paused, queue 2 active
0111	Queue 1 paused, queue 2 trigger pending
1000	Queue 1 active, queue 2 idle
1001	Queue 1 active, queue 2 paused
1010	Queue 1 active, queue 2 suspended
1011	Queue 1 active, queue 2 trigger pending
1100	Reserved
1101	Reserved
1110	Reserved
1111	Reserved

5.12.9 QADC64 Status Register 1 (QASR1)

The QASR1 contains two fields: command word pointers for queue 1 and queue 2.

QASR1 — Status Register 1 **0xYF F412**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED		CWPQ1						RESERVED		CWPQ2					
RESET:															
0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1

Table 5-18 QASR1 Bit Settings

Bit(s)	Name	Description
15:14	—	Reserved
13:8	CWPQ1	Command word pointer for queue 1. This field is a software read-only field, and write operations have no effect. CWPQ1 allows software to read the last executed CCW in queue 1, regardless which queue is active. The CWPQ1 field is a CCW word pointer with a valid range of 0 to 63. In contrast to CWP, CPWQ1 is updated when the conversion result is written. When the QADC64 finishes a conversion in queue 1, both the result register is written and the CWPQ1 are updated. Finally, when queue 1 operation is terminated after a CCW is read that is defined as BQ2, CWP points to BQ2 while CWPQ1 points to the last CCW queue 1. During the stop mode, the CWPQ1 is reset to 63, since the control registers and the analog logic are reset. When the freeze mode is entered, the CWPQ1 is unchanged; it points to the last executed CCW in queue 1.
7:6	—	Reserved
5:0	CWPQ2	Command word pointer for queue 2. This field is a software read-only field, and write operations have no effect. CWPQ2 allows software to read the last executed CCW in queue 2, regardless which queue is active. The CWPQ2 field is a CCW word pointer with a valid range of 0 to 63. In contrast to CWP, CPWQ2 is updated when the conversion result is written. When the QADC64 finishes a conversion in queue 2, both the result register is written and the CWPQ2 are updated. During the stop mode, the CWPQ2 is reset to 63, since the control registers and the analog logic are reset. When the freeze mode is entered, the CWP is unchanged; it points to the last executed CCW in queue 2.

5.12.10 Conversion Command Word Table

The CCW table is a RAM, 64 words long and 10 bits wide, which can be programmed by the software to request conversions of one or more analog input channels. The entries in the CCW table are 10-bit conversion command words. The CCW table is written by software and is not modified by the QADC64. Each CCW requests the conversion of an analog channel to a digital result. The CCW specifies the analog channel number, the input sample time, and whether the queue is to pause after the current CCW.

The ten implemented bits of the CCW word are read/write data. They may be written when the software initializes the QADC64. Unimplemented bits are read as zeros, and write operations have no effect. Each location in the CCW table corresponds to a location in the result word table. When a conversion is completed for a CCW entry, the 10-bit result is written in the corresponding result word entry. The QADC64 provides 64 CCW table entries.

The beginning of queue 1 is always the first location in the CCW table. The first location of queue 2 is specified by the beginning of queue 2 pointer (BQ2) in QACR2. To dedicate the entire CCW table to queue 1, software must do the following:

- Program queue 2 to be in the disabled mode, and
- Program the beginning of BQ2 to ≥ 64.

To dedicate the entire CCW table to queue 2, software must do the following:

- Program queue 1 to be in the disabled mode
- Program BQ2 to be the first location in the CCW table.

Figure 5-12 illustrates the operation of the queue structure.

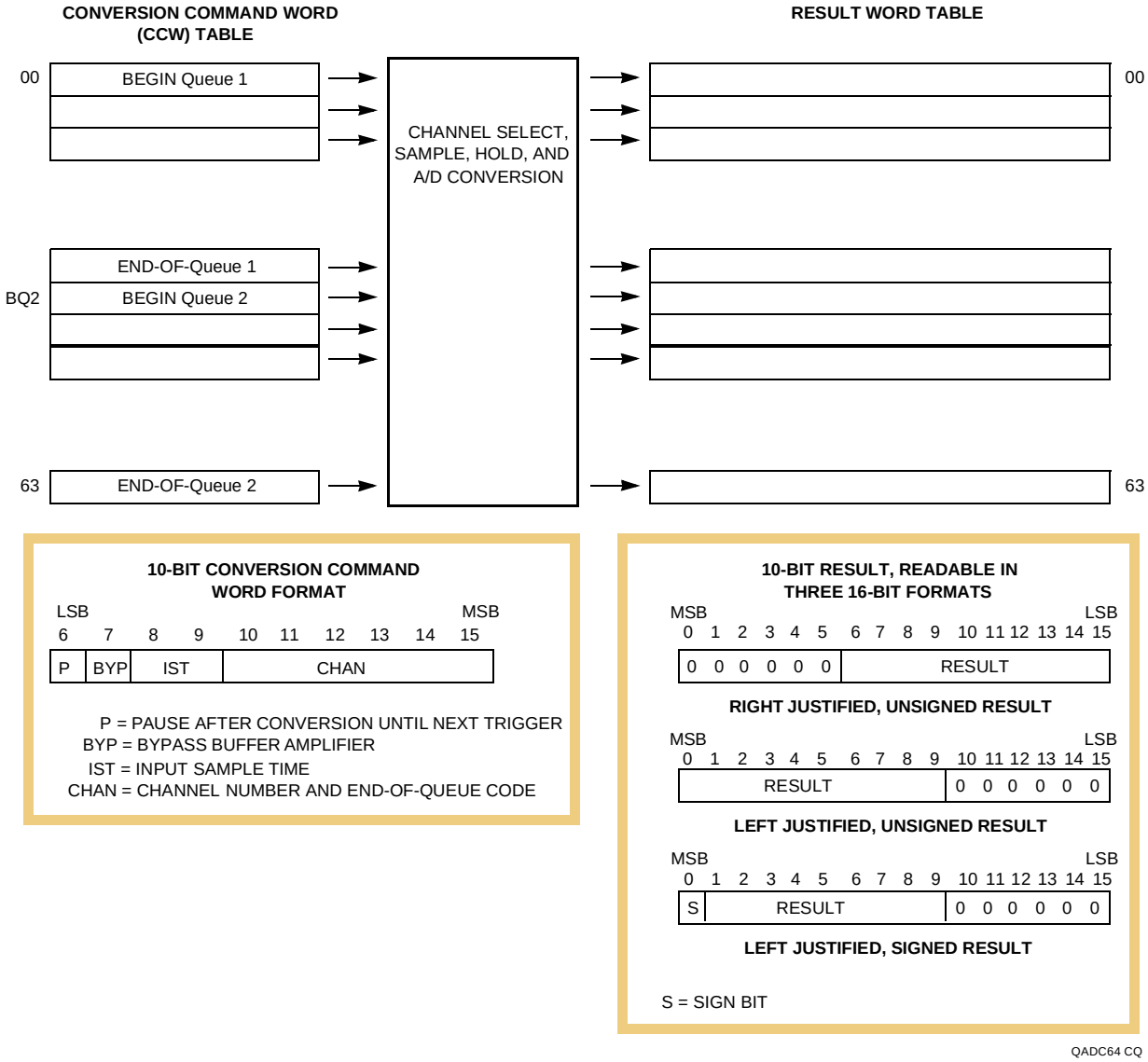


Figure 5-12 QADC64 Conversion Queue Operation

To prepare the QADC64 for a scan sequence, the software writes to the CCW table to specify the desired channel conversions. The software also establishes the criteria for initiating the queue execution by programming the queue operating mode. The queue operating mode determines what type of trigger event causes queue execution to begin. “Trigger event” refers to any of the ways to cause the QADC64 to begin executing the CCWs in a queue or subqueue. An external trigger is only one of the possible trigger events.

A scan sequence may be initiated by the following:

- A software command

- Expiration of the periodic/interval timer
- External trigger signal
- External gated signal (queue 1 only)

The software also specifies whether the QADC64 is to perform a single pass through the queue or is to scan continuously. When a single-scan mode is selected, the software selects the queue operating mode and sets the single-scan enable bit. When a continuous-scan mode is selected, the queue remains active in the selected queue operating mode after the QADC64 completes each queue scan sequence.

During queue execution, the QADC64 reads each CCW from the active queue and executes conversions in three stages:

- Initial sample
- Final sample
- Resolution

During initial sample, a buffered version of the selected input channel is connected to the sample capacitor at the output of the sample buffer amplifier.

During the final sample period, the sample buffer amplifier is bypassed, and the multiplexer input charges the sample capacitor directly. Each CCW specifies a final input sample time of 2, 4, 8, or 16 QCLK cycles. When an analog-to-digital conversion is complete, the result is written to the corresponding location in the result word table. The QADC64 continues to sequentially execute each CCW in the queue until the end of the queue is detected or a pause bit is found in a CCW.

When the pause bit is set in the current CCW, the QADC64 stops execution of the queue until a new trigger event occurs. The pause status flag bit is set, which may cause an interrupt to notify the software that the queue has reached the pause state. After the trigger event occurs, the paused state ends and the QADC64 continues to execute each CCW in the queue until another pause is encountered or the end of the queue is detected.

The following indicate the end-of-queue condition:

- The CCW channel field is programmed with 63 (0x3F) to specify the end of the queue.
- The end-of-queue 1 is implied by the beginning of queue 2, which is specified in the BQ2 field in QACR2.
- The physical end of the queue RAM space defines the end of either queue.

When any of the end-of-queue conditions is recognized, a queue completion flag is set, and if enabled, an interrupt is issued to the software. The following situations prematurely terminate queue execution:

- Since queue 1 is higher in priority than queue 2, when a trigger event occurs on queue 1 during queue 2 execution, the execution of queue 2 is suspended by aborting the execution of the CCW in progress, and the queue 1 execution begins. When queue 1 execution is completed, queue 2 conversions restart with the first CCW entry in queue 2 or the first CCW of the queue 2 subqueue being exe-

cuted when queue 2 was suspended. Alternately, conversions can restart with the aborted queue 2 CCW entry. The RESUME bit in QACR2 allows the software to select where queue 2 begins after suspension. By choosing to re-execute all of the suspended queue 2 queue and subqueue CCWs, all of the samples are guaranteed to have been taken during the same scan pass. However, a high trigger event rate for queue 1 can prohibit the completion of queue 2. If this occurs, the software may choose to begin execution of queue 2 with the aborted CCW entry.

- Software can change the queue operating mode to disabled mode. Any conversion in progress for that queue is aborted. Putting a queue into the disabled mode does not power down the converter.
- Software can change the queue operating mode to another valid mode. Any conversion in progress for that queue is aborted. The queue restarts at the beginning of the queue, once an appropriate trigger event occurs.
- For low power operation, software can set the stop mode bit to prepare the module for a loss of clocks. The QADC64 aborts any conversion in progress when the stop mode is entered.
- When the freeze enable bit is set by software and the IMB internal FREEZE line is asserted, the QADC64 freezes at the end of the conversion in progress. When internal FREEZE is negated, the QADC64 resumes queue execution beginning with the next CCW entry.

CCW — Conversion Command Word Table **0xYF F600 – 0xYF F67F**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED						P	BYP	IST		CHAN					
RESET:															
0	0	0	0	0	0	U	U	U	U	U	U	U	U	U	U

Table 5-19 CCW Bit Settings

Bit(s)	Name	Description
15:10	—	Reserved
9	P	Pause. The pause bit allows the creation of sub-queues within queue 1 and queue 2. The QADC64 performs the conversion specified by the CCW with the pause bit set, and then the queue enters the pause state. Another trigger event causes execution to continue from the pause to the next CCW. 0 = Do not enter the pause state after execution of the current CCW. 1 = Enter the pause state after execution of the current CCW.
8	BYP	Sample amplifier bypass. Setting BYP enables the amplifier bypass mode for a conversion, and subsequently changes the timing. Refer to **5.9.1.1 Amplifier Bypass Mode Conversion Timing** for more information. 0 = Amplifier bypass mode disabled. 1 = Amplifier bypass mode enabled.
7:6	IST	Input sample time. The IST field specifies the length of the sample window. Longer sample times permit more accurate A/D conversions of signals with higher source impedances, especially if BYP=1. 00 = QCKL period x 2 01 = QCKL period x 4 10 = QCKL period x 8 11 = QCKL period x 16
5:0	CHAN	Channel number. The CHAN field selects the input channel number corresponding to the analog input pin to be sampled and converted. The analog input pin channel number assignments and the pin definitions vary depending on whether the QADC64 is operating in multiplexed or non-multiplexed mode. The queue scan mechanism sees no distinction between an internally or externally multiplexed analog input. If CHAN specifies a reserved channel number (channels 32 to 47) or an invalid channel number (channels 4 to 31 in non-multiplexed mode), the low reference level (V_{RL}) is converted. Programming the channel field to channel 63 indicates the end of the queue. Channels 60 to 62 are special internal channels. When one of these channels is selected, the sample amplifier is not used. The value of V_{RL}, V_{RH}, or $(V_{RH} – V_{RL})/2$ is placed directly into the converter. Programming the input sample time to any value other than two for one of the internal channels has no benefit except to lengthen the overall conversion time. **Table 5-20** shows the channel number assignments for the non-multiplexed mode. **Table 5-21** shows the channel number assignments for the multiplexed mode.

Table 5-20 Non-Multiplexed Channel Assignments and Pin Designations

Non-multiplexed Input Pins				Channel Number in CHAN	
Port Pin Name	**Analog Pin Name**	**Other Functions**	**Pin Type**	**Binary**	**Decimal**
PQB0	AN0	—	Input	000000	0
PQB1	AN1	—	Input	000001	1
PQB2	AN2	—	Input	000010	2
PQB3	AN3	—	Input	000011	3
—	—	Invalid	—	000100 to 011111	4 to 31
—	—	Reserved	—	10XXXX	32 to 47
PQB4	AN48	—	Input	110000	48
PQB5	AN49	—	Input	110001	49
PQB6	AN50	—	Input	110010	50
PQB7	AN51	—	Input	110011	51
PQA0	AN52	—	Input/Output	110100	52
PQA1	AN53	—	Input/Output	110101	53
PQA2	AN54	—	Input/Output	110110	54
PQA3	AN55	—	Input/Output	110111	55
PQA4	AN56	—	Input/Output	111000	56
PQA5	AN57	—	Input/Output	111001	57
PQA6	AN58	—	Input/Output	111010	58
PQA7	AN59	—	Input/Output	111011	59
—	V_{RL}	—	Input	111100	60
—	V_{RH}	—	Input	111101	61
—	—	$(V_{RH} – V_{RL})/2$	—	111110	62
—	—	End-of-Queue Code	—	111111	63

Table 5-21 Multiplexed Channel Assignments and Pin Designations

Multiplexed Input Pins				Channel Number in CHAN	
Port Pin Name	**Analog Pin Name**	**Other Functions**	**Pin Type**	**Binary**	**Decimal**
PQB0	ANw	—	Input	00xxx0	0 to 14 even
PQB1	ANx	—	Input	00xxx1	1 to 15 odd
PQB2	ANy	—	Input	01xxx0	16 to 30 even
PQB3	ANz	—	Input	01xxx1	17 to 31 odd
—	—	Reserved	—	10xxxx	32 to 47
PQB4	AN48	—	Input	110000	48
PQB5	AN49	—	Input	110001	49
PQB6	AN50	—	Input	110010	50
PQB7	AN51	—	Input	110011	51
PQA0	—	MA0	Input/Output	110100	52
PQA1	—	MA1	Input/Output	110101	53
PQA2	—	MA2	Input/Output	110110	54
PQA3	AN55	—	Input/Output	110111	55
PQA4	AN56	—	Input/Output	111000	56
PQA5	AN57	—	Input/Output	111001	57
PQA6	AN58	—	Input/Output	111010	58
PQA7	AN59	—	Input/Output	111011	59
—	V_{RL}	—	Input	111100	60
—	V_{RH}	—	Input	111101	61
—	—	$(V_{RH} -V_{RL})/2$	—	111110	62
—	—	End-of-Queue Code	—	111111	63

5.12.11 Result Word Table

The result word table is a 64-word long, 10-bit wide RAM. The QADC64 writes a result word after completing an analog conversion specified by the corresponding CCW. The result word table can be read or written, but in normal operation, software reads the result word table to obtain analog conversions from the QADC64. Unimplemented bits are read as zeros, and write operations have no effect.

While there is only one result word table, the data can be accessed in three different alignment formats:

- Right justified, with zeros in the higher order unused bits.
- Left justified, with the most significant bit inverted to form a sign bit, and zeros in the unused lower order bits.
- Left justified, with zeros in the unused lower order bits.

The left justified, signed format corresponds to a half-scale, offset binary, two's complement data format. The data is routed onto the IMB according to the selected format. The address used to access the table determines the data alignment format. All write operations to the result word table are right justified.

RJURR — Right Justified, Unsigned Result Register **0xYF F680 – 0xYF F6FE**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED						RESULT									
RESET:															
0	0	0	0	0	0										

The conversion result is unsigned, right justified data. Unused bits return zero when read.

LJSRR — Left Justified, Signed Result Register **0xYF F700 – 0xYF F77E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
S[1]	RESULT									RESERVED					
RESET:															
										0	0	0	0	0	0

NOTES:
1. S = Sign bit.

The conversion result is signed, left justified data. Unused bits return zero when read.

LJURR — Left Justified, Unsigned Result Register **0xYF F780 – 0xYF F7FE**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESULT										RESERVED					
RESET:															
										0	0	0	0	0	0

The conversion result is unsigned, left justified data. Unused bits return zero when read.

5.13 Analog Multiplexer Submodule

The analog multiplexer (AMUX) submodule expands the channel capacity of the QADC64 analog-to-digital converter inputs to a maximum of 27 analog channels (using only the on-chip multiplexer, 41 with the addition of off-chip multiplexers). The AMUX does not have an inter-module bus (IMB3) interface; control is through the QADC64.

The AMUX is a “high voltage” device requiring 5 V nominal. There is no on-board charge pump as with the previous UDR implementation of the AMUX. Performance of the AMUX should be superior to that of an external multiplexer because of the greatly reduced parasitic capacitances. In addition, precautions have been taken to insure that the input current requirement is as low as possible.

The architecture and pin naming of the AMUX were modeled after the QADC64 operating in external multiplexer mode. Use of this feature of the QADC64 is described in **5.7 External Multiplexing Operation**, and is illustrated in **Figure 5-13**. When the AMUX is used with the QADC64, it replaces the external multiplexers. The software model is identical and the system performance is improved.

5.13.1 Signal Descriptions

5.13.1.1 External Pins (Connected to Pads)

ANX0–ANX15, analog input pins, extended. These 16 analog input channels are divided into 2 groups of 8 inputs: ANX0–14 even, ANX1–15 odd. Each group is connected to a separate multiplexer within the AMUX. These pads are unique to the AMUX.

V_{DDA}, V_{SSA}, analog power, is shared by the QADC64 and the AMUX.

V_{DD}, V_{SS}, digital power, is shared by the QADC64 and the AMUX.

5.13.1.2 Internal Pins (Connected to QADC64)

See table **Table 5-22** for a summary of this section.

AMMA[2:0]. Multiplexer address inputs. These pins are decoded into 8 select lines that steer the 4 multiplexers within the AMUX in parallel. The final stage selection between the 4 multiplexers is done within the QADC64. On the MC68F375 only 2 analog multiplexers are selected.

Assuming that these lines change simultaneously, break-before-make switching ensures that on each channel change, all multiplexer outputs are opened before the new one is selected. This prevents 2 inputs in the same mux from being connected to the mux output at any one time.

AMCHAN[1:0]. Address lines to select between the 4 multiplexers. This selection appears redundant since the final selection between the 4 multiplexers is done by the QADC64 module. This redundancy is added to reduce the dynamic charging current required from the input pin. These pins perform a disable function more than a select function. Only 1 of the 4 multiplexers can be active at a time. These address lines disable the other 3 multiplexers so that none of their switches are closed. As the selected multiplexer cycles through its various inputs, the other multiplexers draw no dynamic charging current through their inputs. Without this disable function (as in the external multiplexer case), pins on all 4 multiplexers would draw current even though only 1 multiplexer would actually be in use.

Assuming that AMCHAN[1:0] change simultaneously with AMMA[2:0], break before make switching ensures that on each channel change the following occurs:

1. All 4 multiplexers are disabled
2. The desired input ANX0-15 is selected
3. The appropriate multiplexer is enabled

AMPMUX. When low this signal disables all multiplexer inputs. It is used in conjunction with AMCHAN[1:0] to reduce dynamic charging current into the AMUX analog pins. This signal is normally low when the direct inputs to the QADC64 are selected.

AMADB[1:0]. The 2 multiplexer outputs from the AMUX.

Table 5-22 AMUX I/O Functionality

AMPMUX	AMCHAN[1:0]	AMMA[2:0]	AMADB3	AMADB2	AMADB1	AMADB0
0	XX	XXX	Z	Z	Z	Z
1	00	000	Z	Z	Z	ANX0
1	00	001	Z	Z	Z	ANX2
1	00	010	Z	Z	Z	ANX4
1	00	011	Z	Z	Z	ANX6
1	00	100	Z	Z	Z	ANX8
1	00	101	Z	Z	Z	ANX10
1	00	110	Z	Z	Z	ANX12
1	00	111	Z	Z	Z	ANX14
1	01	000	Z	Z	ANX1	Z
1	01	001	Z	Z	ANX3	Z
1	01	010	Z	Z	ANX5	Z
1	01	011	Z	Z	ANX7	Z
1	01	100	Z	Z	ANX9	Z
1	01	101	Z	Z	ANX11	Z
1	01	110	Z	Z	ANX13	Z
1	01	111	Z	Z	ANX15	Z

5.13.2 Mixed AMUX/External Multiplexing

The QADC64/AMUX combination uses a mixed internal/external multiplexing. In mixed multiplexing, the AMUX outputs will be tied to port B pads B[1:0]. Note that the on-chip AMUX is susceptible to noise injection on the user board through the pad input + ESD (electro-static discharge) protection circuitry + the substrate (package) capacitance on its inputs. Care should be taken in the layout of any connections to the AN0/ANW/PQB0 and AN1/ANX/PQB1 pins. If the on-chip AMUX is not disabled, these pins could be used for analog inputs. See **5.13.3 Pin Connection and Performance Considerations** for loading implications.

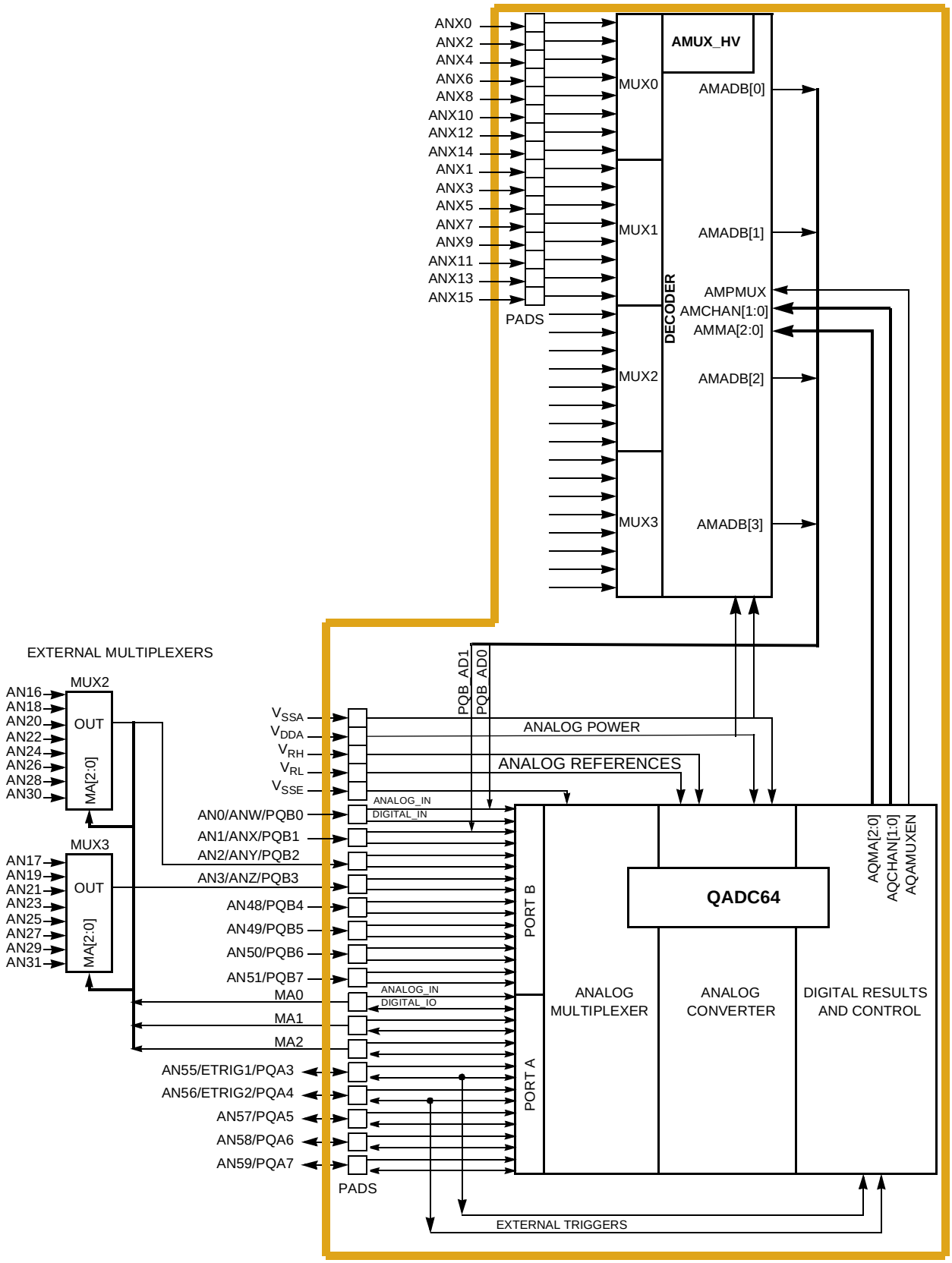


Figure 5-13 AMUX/QADC64 Configured for Mixed Multiplexing

5.13.3 Pin Connection and Performance Considerations

The performance of an ANXx pin is almost the same as that of an ANx input to the QADC64. There are two differences: there is an increased resistance because of the additional multiplexer switch in series, and there is no buffering between channels within the groups of 8. If the increased resistance becomes a problem it can be compensated for by increasing the input sample time (IST[7:6] in the Conversion Command Word Table of the QADC64). The lack of buffering between channels causes additional dynamic charging current to flow into the analog pins during successive conversions of different voltages. A schematic of one group of eight inputs which illustrates the cause of this charging current is shown in figure **Figure 5-14**.

If alternating between 2 AMUX channels within the group of eight, one at V_{rh} and the other at V_{rl}, the equivalent capacitance C_{EQ} must be charged/discharged between V_{rh} and V_{rl}. This generates an average current which flows across the sum of the filter, source, and mux resistances. The error due to the charging current is expressed by the following equation:

$$ERROR = I_{AVG} * R_{EQ} = \Delta V * sample_rate * C_{EQ} * R_{EQ}$$

Where: $C_{EQ} = C_{MUXOUT} + C_P$

$$R_{EQ} = R_{SOURCE} + R_{FILTER} + R_{MUXOUT}$$

$$\Delta V = V_{rh} - V_{rl}$$

For an error equal to 1/2 LSB, the following equation must be met:

$$C_{EQ} * R_{EQ} < 1/(2048*sample_rate)$$

As an example, if C_{EQ} = 2.0 pF and R_{EQ} = 50K, the maximum sample_rate is 4.9 KHz.

This charging current is also present on normal QADC64 channels, but to a lesser extent because C_{MUXOUT} and R_{MUXOUT} are both zero.

For a discussion of other pin connection and performance considerations, please refer to **5.3 QADC64 Pin Functions**.

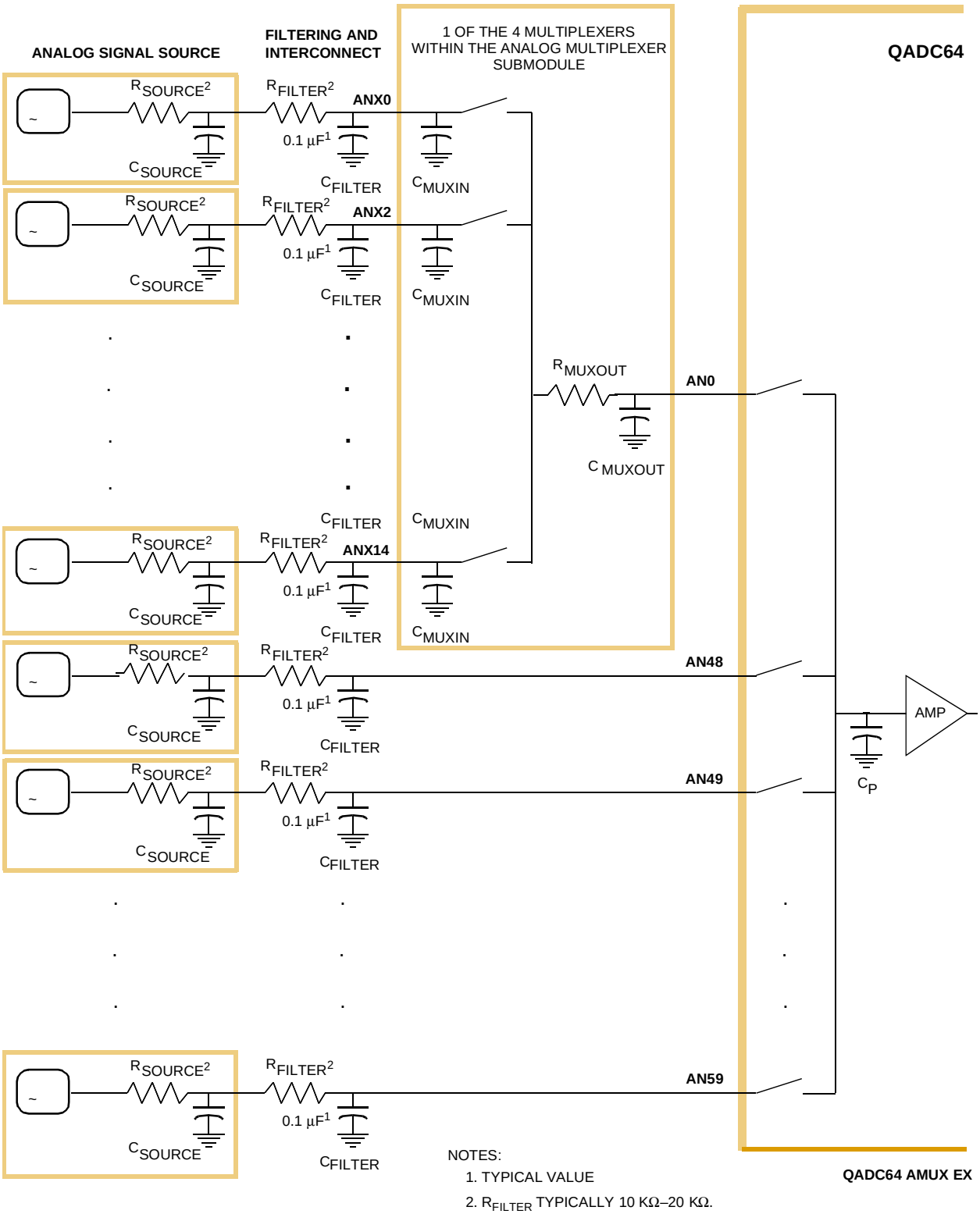


Figure 5-14 Analog Multiplexer Submodule Charging Current Illustration

SECTION 6
QUEUED SERIAL MULTI-CHANNEL MODULE

6.1 Overview

The queued serial multi-channel module (QSMCM) provides three serial communication interfaces: the queued serial peripheral interface (QSPI) and two serial communications interfaces (SCI1 and SCI2). These submodules communicate with the CPU via a common slave bus interface unit (SBIU).

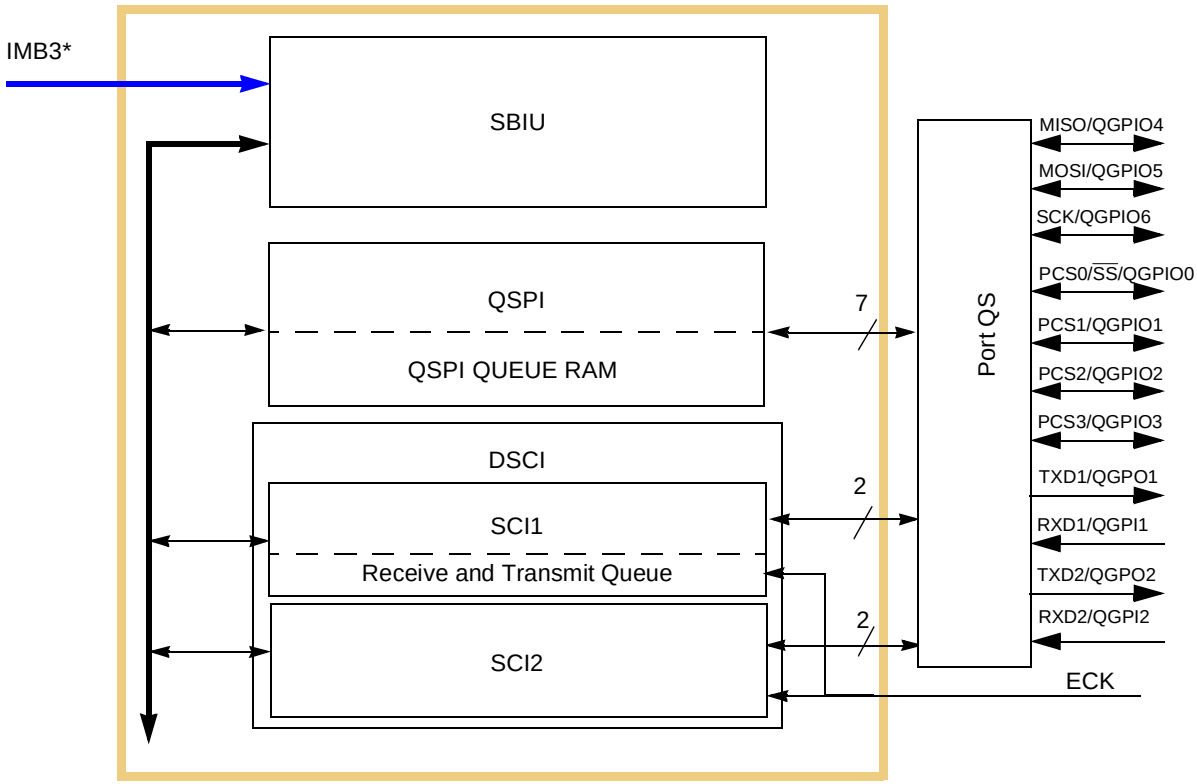
The QSPI is a full-duplex, synchronous serial interface for communicating with peripherals and other MCUs. It is enhanced from the original SPI in the QSMCM (queued serial module) to include a total of 160 bytes of queue RAM to accommodate more receive, transmit, and control information. The QSPI is fully compatible with the SPI systems found on other Motorola devices.

The dual, independent SCIs are used to communicate with external devices and other MCUs via an asynchronous serial bus. Each SCI is a full-duplex universal asynchronous receiver transmitter (UART) serial interface. The original QSMCM SCI is enhanced by the addition of an SCI and a common external baud clock source.

The SCI1 has the ability to use the resultant baud clock from SCI2 as the input clock source for the SCI1 baud rate generator. Also, the SCI1 has an additional mode of operation that allows queuing of transmit and receive data frames. If the queue feature is enabled, a set of 16 entry queues is allocated for the receive and/or transmit operation.

6.2 Block Diagram

Figure 6-1 depicts the major components of the QSMCM.



*Note: SBIU Bus and interface to IMB3 are each 16 bits wide.

Figure 6-1 QSMCM Block Diagram

6.3 Signal Descriptions

The QSMCM has 12 external pins, as shown in **Figure 6-1**. Seven of the pins, if not in use for their submodule function, can be used as general-purpose I/O port pins. The RXDx and TXDx pins can alternately serve as general-purpose input-only and output-only signals, respectively. ECK is a dedicated clock pin that is not usable on the MC68F375.

For detailed descriptions of QSMCM signals, refer to **6.6 QSMCM Pin Control Registers**, **6.7.3 QSPI Pins**, and **6.8.6 SCI Pins**.

6.4 Memory Map

The QSMCM memory map, shown in **Table 6-1**, includes the global registers, the QSPI and dual SCI control and status registers, and the QSPI RAM. The QSMCM memory map can be divided into supervisor-only data space and assignable data space. The address offsets shown are from the base address of the QSMCM module. Refer to **Figure 1-2** for a diagram of the MC68F375 internal memory map.

Table 6-1 QSMCM Register Map

Access[1]	Address	MSB[2] 0	LSB 15
S	0xYF FC00	QSMCM Module Configuration Register (QSMCMMCR) See **Table 6-3** for bit descriptions.	
T	0xYF FC02	QSMCM Test Register (QTEST)	
S/U	0xYF FC04	QSM Interrupt Level Register (QILR) See **Table 6-4** for bit descriptions.	QSM Interrupt Vector Register (QIVR) See **Table 6-5** for bit descriptions.
S	0xYF FC06	Reserved	Queued SPI Interrupt Level (QSPI_IL) See **Table 6-6** for bit descriptions.
S/U	0xYF FC08	SCI1Control Register 0 (SCC1R0) See **Table 6-6** for bit descriptions.	
S/U	0xYF FC0A	SCI1Control Register 1 (SCC1R1) See **Table 6-24** for bit descriptions.	
S/U	0xYF FC0C	SCI1 Status Register (SC1SR) See **Table 6-25** for bit descriptions.	
S/U	0xYF FC0E	SCI1 Data Register (SC1DR) See **Table 6-26** for bit descriptions.	
S/U	0xYF FC10	Reserved	
S/U	0xYF FC12	Reserved	
S/U	0xYF FC14	Reserved	QSMCM Port Q Data Register (PORTQS) See **6.6.1 Port QS Data Register (PORTQS)** for bit descriptions.
S/U	0xYF FC16	QSMCM Pin Assignment Register (PQS-PAR) See **Table 6-10** for bit descriptions.	QSMCM Data Direction Register (DDRQS) See **Table 6-11** for bit descriptions.
S/U	0xYF FC18	QSPI Control Register 0 (SPCR0) See **Table 6-13** for bit descriptions.	
S/U	0xYF FC1A	QSPI Control Register 1 (SPCR1) See **Table 6-15** for bit descriptions.	
S/U	0xYF FC1C	QSPI Control Register 2 (SPCR2) See **Table 6-16** for bit descriptions.	
S/U	0xYF FC1E	QSPI Control Register 3 (SPCR3) See **Table 6-17** for bit descriptions.	QSPI Status Register (SPSR) See **Table 6-18** for bit descriptions.
S/U	0xYF FC20	SCI2 Control Register 0 (SCC2R0)	
S/U	0xYF FC22	SCI2 Control Register 1 (SCC2R1)	
S/U	0xYF FC24	SCI2 Status Register (SC2SR)	
S/U	0xYF FC26	SCI2 Data Register (SC2DR)	
S/U	0xYF FC28	QSCI1 Control Register (QSCI1CR) See **Table 6-31** for bit descriptions.	
S/U	0xYF FC2A	QSCI1 Status Register (QSCI1SR) See **Table 6-32** for bit descriptions.	

Table 6-1 QSMCM Register Map (Continued)

Access[1]	Address	MSB[2] 0	LSB 15
S/U	0xYF FC2C – 0xYF FC4A	Transmit Queue Locations (SCTQ)	
S/U	0xYF FC4C – 0xYF FC6A	Receive Queue Locations (SCRQ)	
S/U	0xYF FC6C – 0xYF FD3F	Reserved	
S/U	0xYF FD40 – 0xYF FD7F	Receive Data RAM (REC.RAM)[3]	
S/U	0xYF FD80 – 0xYF FDBF	Transmit Data RAM (TRAN.RAM)[3]	
S/U	0xYF FDC0 – 0xYF FDDF	Command RAM (COMD.RAM)[3]	

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers, such as SPCR3 and SPSR, are on 8-bit boundaries. 16-bit registers such as SPCR0 are on 16-bit boundaries.
3. Note that QRAM offsets have been changed from the original (modular family) QSMCM.

The supervisor-only data space segment contains the QSMCM global registers. These registers define parameters needed by the QSMCM to integrate with the MCU. Access to these registers is permitted only when the CPU is operating in supervisor mode.

Assignable data space can be either restricted to supervisor-only access or unrestricted to both supervisor and user accesses. The supervisor (SUPV) bit in the QSMCM module configuration register (QSMCMMCR) designates the assignable data space as either supervisor or unrestricted. If SUPV is set, then the space is designated as supervisor-only space. Access is then permitted only when the CPU is operating in supervisor mode. If SUPV is clear, both user and supervisor accesses are permitted. To clear SUPV, the CPU must be in supervisor mode.

The QSMCM assignable data space segment contains the control and status registers for the QSPI and SCI submodules, as well as the QSPI RAM. All registers and RAM can be accessed on byte (8-bits), half-word (16-bits), and word (32-bit) boundaries. Word accesses require two consecutive IMB3 bus cycles.

6.5 QSMCM Global Registers

The QSMCM global registers contain system parameters used by the QSPI and SCI submodules for interfacing to the CPU and the intermodule bus. The global registers are listed in **Table 6-2 QSMCM Global Registers**

Table 6-2 QSMCM Global Registers

Access[1]	Address	MSB[2]	LSB
S	0xYF FC00	QSMCM Module Configuration Register (QSMCMMCR) See **Table 6-3** for bit descriptions.	
T	0xYF FC02	QSMCM Test Register (QTEST)	
S	0xYF FC04	Dual SCI Interrupt Level (QDSCI_IL) See **Table 6-4** for bit descriptions.	QSM Interrupt Vector Register (QIVR) See **Table 6-5** for bit descriptions.
S	0xYF FC06	Reserved	Queued SPI Interrupt Level (QSPI_IL) See **Table 6-6** for bit descriptions.

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers reside on 8-bit boundaries. 16-bit registers reside on 16-bit boundaries.

6.5.1 Low-Power Stop Operation

When the STOP bit in QSMCMMCR is set, the system clock input to the QSMCM is disabled and the module enters a low-power operating state. QSMCMMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable in low-power stop mode. However, writes to RAM or any register are guaranteed valid while STOP is asserted. STOP can be written by the CPU and is cleared by reset.

System software must bring each submodule to an orderly stop before setting STOP to avoid data corruption. The SCI receiver and transmitter should be disabled after transfers in progress are complete. The QSPI can be halted by setting the HALT bit in SPCR3 and then setting STOP after the HALTA flag is set.

6.5.2 Freeze Operation

The FRZ1 bit in QSMCMMCR determines how the QSMCM responds when the IMB3 FREEZE signal is asserted. FREEZE is asserted when the CPU enters background debug mode. Setting FRZ1 causes the QSPI to halt on the first transfer boundary following FREEZE assertion. FREEZE causes the SCI1 transmit queue to halt on the first transfer boundary following FREEZE assertion.

6.5.3 Access Protection

The SUPV bit in the QMCR defines the assignable QSMCM registers as either supervisor-only data space or unrestricted data space.

When the SUPV bit is set, all registers in the QSMCM are placed in supervisor-only space. For any access from within user mode, the IMB3 address acknowledge ($\overline{\text{AACK}}$) signal is asserted and a bus error is generated.

Because the QSMCM contains a mix of supervisor and user registers, $\overline{\text{AACK}}$ is asserted for either supervisor or user mode accesses, and the bus cycle remains internal. If a supervisor-only register is accessed in user mode, the module responds as if an access had been made to an unauthorized register location, and a bus error is generated.

6.5.4 QSMCM Interrupts

Both the QSPI and SCI can generate interrupt requests. Each has a separate interrupt request priority register. A single vector register is used to generate exception vector numbers.

The values of the ILQSPI and ILSCI fields in QILR determine the priority of QSPI and SCI interrupt requests. The values in these fields correspond to internal interrupt request signals IRQ[7:1]. A value of 0b111 causes IRQ7 to be asserted when a QSM interrupt request is made. Lower field values cause correspondingly lower-numbered interrupt request signals to be asserted. Setting the ILQSPI or ILSCI field values to 0b000 disables interrupts for the respective section. If ILQSPI and ILSCI have the same non-zero value, and the QSPI and SCI make simultaneous interrupt requests, the QSPI has priority.

When the CPU32 acknowledges an interrupt request, it places the value in the status register interrupt priority (IP) mask on the address bus. The QSM compares the IP mask value to the priority of the request to determine whether it should contend for arbitration priority. Arbitration priority is determined by the value of the IARB field in QSMCR. Each module that generates interrupts must have a non-zero IARB value. Arbitration is performed by means of serial contention between values stored in individual module IARB fields.

When the QSM wins interrupt arbitration, it responds to the CPU32 interrupt acknowledge cycle by placing an interrupt vector number on the data bus. The vector number is used to calculate displacement into the CPU32 exception vector table. SCI and QSPI vector numbers are generated from the value in the QIVR INTV field. The values of bits INTV[7:1] are the same for QSPI and SCI. The value of INTV0 is supplied by the QSM when an interrupt request is made. INTV0 = 0 for SCI interrupt requests; INTV0 = 1 for QSPI interrupt requests.

At reset, INTV[7:0] is initialized to 0x0F, the uninitialized interrupt vector number. To enable interrupt-driven serial communication, a user-defined vector number must be written to QIVR, and interrupt handler routines must be located at the addresses pointed to by the corresponding vector. Writes to INTV0 have no effect. Reads of INTV0 return a value of one.

Refer to **SECTION 3 CENTRAL PROCESSOR UNIT** and **SECTION 4 SINGLE-CHIP INTEGRATION MODULE 2 (SCIM2E)** for more information about exceptions and interrupts.

6.5.5 QSMCM Configuration Register (QSMCMMCR)

The QSMCMMCR contains parameters for interfacing to the CPU and the intermodule bus. This register can be modified only when the CPU is in supervisor mode.

QSMCMMCR — QSMCM Configuration Register **0xYF FC00**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	FRZ1	RESERVED						SUPV	RESERVED			IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Table 6-3 QSMCMMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Stop enable. Refer to **6.5.1 Low-Power Stop Operation**. 0 = Normal clock operation 1 = Internal clocks stopped
14	FRZ1	Freeze1 bit. Refer to **6.5.2 Freeze Operation**. 0 = Ignore the FREEZE signal 1 = Halt the QSMCM (on transfer boundary)
13:8	—	Reserved
7	SUPV	Supervisor /Unrestricted. Refer to **6.5.3 Access Protection**. 0 = Assigned registers are unrestricted (user access allowed) 1 = Assigned registers are restricted (only supervisor access allowed)
6:4	—	Reserved
3:0	IARB	Interrupt arbitration number. IARB determines QSMCM interrupt arbitration priority. An IARB field can be assigned a value from 0b0001 (lowest priority) to 0b1111 (highest value). Note that the logic associated with the IARB field is implemented for bus masters with interrupt acknowledge cycles (IACK).

6.5.6 QSMCM Test Register (QTEST)

The QTEST register is used for factory testing of the MCU.

6.5.7 QSMCM Interrupt Level Registers (QILR, QIVR, QSPI_IL)

The values of ILQSPI[2:0] and ILSCI[2:0] in QILR and QSPI_IL determine the priority of QSPI and SCI interrupt requests.

QIVR determines the value of the interrupt vector number the QSMCM supplies when it responds to an interrupt acknowledge cycle. At reset, QIVR is initialized to 0x0F, the uninitialized interrupt vector number. To use interrupt-driven serial communication, a user-defined vector number must be written to QIVR.

QILR — QSM Interrupt Levels Register **0xYF FC04**

7	6	5	4	3	2	1	0
RESERVED					ILSCI[2:0]		
RESET:							
0	0	0	0	0	0	0	0

Table 6-4 QILR Bit Settings

Bit(s)	Name	Description
7:3	—	Reserved
2:0	ILSCI[2:0]	Interrupt level of SCI. When an interrupt request is made,the ILSCI value determines which of the interrupt request signals is asserted. When a request is acknowledged, the QSMCM compares this value to a mask value supplied by the CPU32 to determine whether to respond. The field must have a value in the range 0x0 (interrupts disabled) to 0x7 (highest priority). If ILQSPI[2:0] and ILSCI[2:0] have the same non-zero value, and both submodules simultaneously request interrupt service, the QSPI has priority.

QIVR — QSMCM Interrupt Vector Register **0xYF FC05**

7	6	5	4	3	2	1	0
INTV[7:0]							
RESET:							
0	0	0	0	1	1	1	1

Table 6-5 QIVR Bit Settings

Bit(s)	Name	Description
7:0	INTV[7:0]	Interrupt vector number. The values of INTV[7:1] are the same for both QSPI and SCI interrupt requests; the value of INTV0 used during an interrupt acknowledge cycle is supplied by the QSMCM. INTV0 is at logic level zero during an SCI interrupt and at logic level one during a QSPI interrupt. A write to INTV0 has no effect. Reads of INTV0 return a value of one.

QSPI_IL — Queued SPI Interrupt Level Register **0xYF FC07**

7	6	5	4	3	2	1	0
RESERVED					ILQSPI[2:0]		
RESET:							
0	0	0	0	0	0	0	0

Table 6-6 QSPI_IL Bit Settings

Bit(s)	Name	Description
7:3	—	Reserved
2:0	ILQSPI[2:0]	Interrupt level of QSPI. When an interrupt request is made, the ILQSPI value determines which of the interrupt request signals is asserted; when a request is acknowledged, the QSMCM compares this value to a mask value supplied by the CPU32 to determine whether to respond. ILQSPI must have a value in the range 0x0 (interrupts disabled) to 0x7 (highest priority).

6.6 QSMCM Pin Control Registers

Table 6-7 lists the three QSMCM pin control registers.

Table 6-7 QSMCM Pin Control Registers

Address	Register
0x30 5014	QSMCM Port Data Register (PORTQS) See **6.6.1 Port QS Data Register (PORTQS)** for bit descriptions.
0x30 5016	PORTQS Pin Assignment Register (PQSPAR) See **Table 6-11** for bit descriptions.
0x30 5017	PORTQS Data Direction Register (DDRQS) See **Table 6-11** for bit descriptions.

The QSMCM uses 12 pins. Eleven of the pins, when not being used by the serial subsystems, form a parallel port on the MCU. (The ECK pin is a dedicated external clock source.)

The port QS pin assignment register (PQSPAR) governs the usage of QSPI pins. Clearing a bit assigns the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI.

PQSPAR does not affect operation of the SCI. When the SCIx transmitter is disabled, TXDx is a discrete output; when the SCIx receiver is disabled, RXDx is a discrete input. When the SCIx transmitter or receiver is enabled, the associated TXDx or RXDx pin is assigned its SCI function.

The port QS data direction register (DDRQS) determines whether QSPI pins are inputs or outputs. Clearing a bit makes the corresponding pin an input; setting a bit makes the pin an output. DDRQS affects both QSPI function and I/O function. **Table 6-10** summarizes the effect of DDRQS bits on QSPI pin function.

DDRQS does not affect SCI pin function. TXDx pins are always outputs, and RXDx pins are always inputs, regardless of whether they are functioning as SCI pins or as PORTQS pins.

The port QS data register (PORTQS) latches I/O data. PORTQS writes drive pins defined as outputs. PORTQS reads return data present on the pins. To avoid driving undefined data, write the first data to PORTQS before configuring DDRQS.

Table 6-8 Effect of DDRQS on QSPI Pin Function

QSMCM Pin	Mode	DDRQS Bit	Bit State	Pin Function
MISO	Master	DDQS0	0	Serial data input to QSPI
			1	Disables data input
	Slave		0	Disables data output
			1	Serial data output from QSPI
MOSI	Master	DDQS1	0	Disables data output
			1	Serial data output from QSPI
	Slave		0	Serial data input to QSPI
			1	Disables data input
SCK[1]	Master	DDQS2	—	Clock output from QSPI
	Slave		—	Clock input to QSPI
PCS0/$\overline{SS}$	Master	DDQS3	0	Assertion causes mode fault
			1	Chip-select output
	Slave		0	QSPI slave select input
			1	Disables slave select input
PCS[1:3]	Master	DDQS[4:6]	0	Disables chip-select output
			1	Chip-select output
	Slave		0	Inactive
			1	Inactive

NOTES:
1. SCK/QGPIO6 is a digital I/O pin unless the SPI is enabled (SPE set in SPCR1), in which case it becomes the QSPI serial clock SCK.

6.6.1 Port QS Data Register (PORTQS)

PORTQS determines the actual input or output value of a QSMCM port pin if the pin is defined as general-purpose input or output. All QSMCM pins except the ECK pin can be used as general-purpose input and/or output. When the SCIx transmitter is disabled, TXDx is a discrete output; when the SCIx receiver is disabled, RXDx is a discrete input. Writes to this register affect the pins defined as outputs; reads of this register return the actual value of the pins.

PORTQS — Port QS Data Register **0xYF FC14**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED				QDRX D2	QDTX D2	QDRX D1	QDTX D1	0	QDPC S3	QDPC S2	QDPC S1	QDPC S0	QD-SCK	QD-MOSI	QDMI-SO
RESET:															
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

6.6.2 PORTQS Pin Assignment Register (PQSPAR)

PQSPAR determines which of the QSPI pins, with the exception of the SCK pin, are used by the QSPI submodule, and which pins are available for general-purpose I/O.

Pins may be assigned on a pin-by-pin basis. If the QSPI is disabled, the SCK pin is automatically assigned its general-purpose I/O function (QGPIO6).

QSPI pins designated by PQSPAR as general-purpose I/O pins are controlled only by PQSDDR and PQSPDR; the QSPI has no effect on these pins. PQSPAR does not affect the operation of the SCI submodule.

Table 6-9 summarizes the QSMCM pin functions.

Table 6-9 QSMCM Pin Functions

PORTQS Function	QSMCM Function
QGPI2	RXD2
QGPO2	TXD2
QGPI1	RXD1
QGPO1	TXD1
QGPIO6	SCK
QGPIO5	MOSI
QGPIO4	MISO
QGPIO3	PCS3
QGPIO2	PCS2
QGPIO1	PCS1
QGPIO0	PCS0

PQSPAR — PORTQS Pin Assignment Register **0xYF FC16**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	QPAP CS3	QPAP CS2	QPAP CS1	QPAP CS0	0	QPA-MOSI	QPAM ISO	DDRQS*							
RESET:															
0	0	0	0	0	0	0	0								

*See bit descriptions in **Table 6-11**.

Table 6-10 PQSPAR Bit Settings

Bit(s)	Name	Description
0	—	Reserved
1	QPAPCS3	0 = Pin is assigned QGPIO3 1 = Pin is assigned PCS3 function
2	QPAPCS2	0 = Pin is assigned QGPIO2 1 = Pin is assigned PCS2 function
3	QPAPCS1	0 = Pin is assigned QGPIO1 1 = Pin is assigned PCS1 function
4	QPAPCS0	0 = Pin is assigned QGPIO0 1 = Pin is assigned PCS0 function
5	—	Reserved
6	QPAMOSI	0 = Pin is assigned QGPIO5 1 = Pin is assigned MOSI function
7	QPAMISO	0 = Pin is assigned QGPIO4 1 = Pin is assigned MISO function
8:15	DDRQS	PORSTQS data direction register. See **6.6.3 PORTQS Data Direction Register (DDRQS)**.

6.6.3 PORTQS Data Direction Register (DDRQS)

DDRQS assigns QSPI pin as an input or an output regardless of whether the QSPI submodule is enabled or disabled. All QSPI pins are configured during reset as general-purpose inputs.

This register does not affect SCI operation. The TXD1 and TXD2 remain output pins dedicated to the SCI submodules, and the RXD1, RXD2 and ECK pins remain input pins dedicated to the SCI submodules.

DDRQS — PORTQS Data Direction Register **0xYF FC16**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PQSPAR*								0	QDDP CS3	QDDP CS2	QDDP CS1	QDDP CS0	QDDS CK	QD-DMO-SI	QD-DMI-SO
RESET:								0	0	0	0	0	0	0	0

*See bit descriptions in **Table 6-10**.

Table 6-11 DDRQS Bit Settings

Bit(s)	Name	Description
15:8	PQSPAR	PORTSQS pin assignment register. See **6.6.2 PORTQS Pin Assignment Register (PQSPAR)**.
7	—	Reserved
6	QDDPCS3	QSPI pin data direction for the pin PCS3 0 = Pin direction is input 1 = Pin direction is output
5	QDDPCS2	QSPI pin data direction for the pin PCS2 0 = Pin direction is input 1 = Pin direction is output
4	QDDPCS1	QSPI pin data direction for the pin PCS1 0 = Pin direction is input 1 = Pin direction is output
3	QDDPCS0	QSPI pin data direction for the pin PCS0 0 = Pin direction is input 1 = Pin direction is output
2	QDDSCK	QSPI pin data direction for the pin SCK 0 = Pin direction is input 1 = Pin direction is output
1	QPAMOSI	QSPI pin data direction for the pin MOSI 0 = Pin direction is input 1 = Pin direction is output
0	QPAMISO	QSPI pin data direction for the pin MISO 0 = Pin direction is input 1 = Pin direction is output

6.7 Queued Serial Peripheral Interface

The queued serial peripheral interface (QSPI) is used to communicate with external devices through a synchronous serial bus. The QSPI is fully compatible with SPI systems found on other Motorola products, but has enhanced capabilities. The QSPI can perform full duplex three-wire or half duplex two-wire transfers. Several transfer rates, clocking, and interrupt-driven communication options are available. **Figure 6-2** is a block diagram of the QSPI.

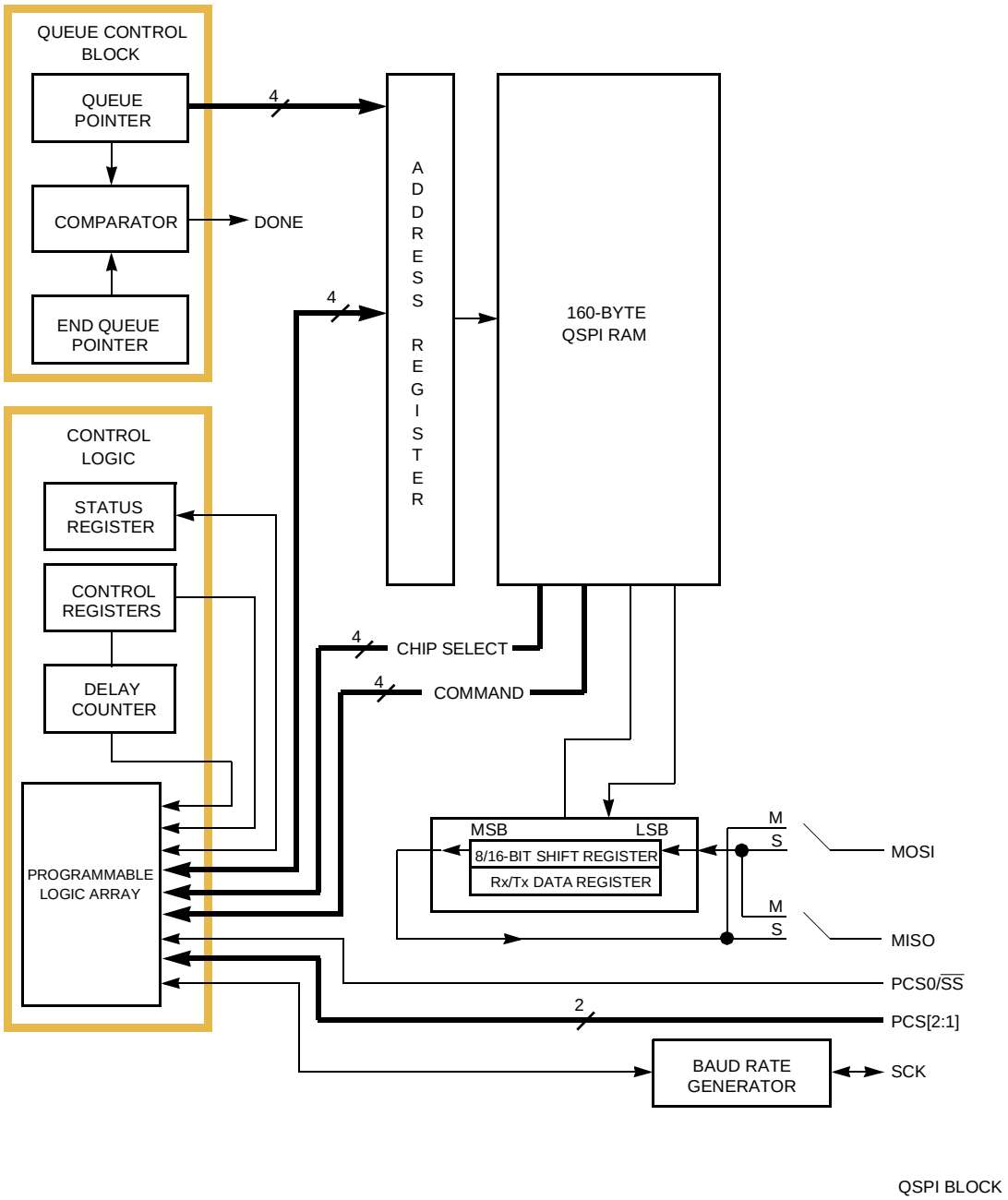


Figure 6-2 QSPI Block Diagram

Serial transfers of eight to 16 bits can be specified. Programmable transfer length simplifies interfacing to devices that require different data lengths.

An inter-transfer delay of approximately 0.8 to 204 µs (using a 40-MHz system clock) can be programmed. The default delay is 17 clocks (0.425 µs at 40 MHz). Programmable delay simplifies the interface to devices that require different delays between transfers.

A dedicated 160-byte RAM is used to store received data, data to be transmitted, and a queue of commands. The CPU can access these locations directly. This allows serial peripherals to be treated like memory-mapped parallel devices.

The command queue allows the QSPI to perform up to 32 serial transfers without CPU intervention. Each queue entry contains all the information needed by the QSPI to independently complete one serial transfer.

A pointer identifies the queue location containing the data and command for the next serial transfer. Normally, the pointer address is incremented after each serial transfer, but the CPU can change the pointer value at any time. Support for multiple-tasks can be provided by segmenting the queue.

The QSPI has four peripheral chip-select pins. The chip-select signals simplify interfacing by reducing CPU intervention. If the chip-select signals are externally decoded, 16 independent select signals can be generated.

Wrap-around mode allows continuous execution of queued commands. In wrap-around mode, newly received data replaces previously received data in the receive RAM. Wrap-around mode can simplify the interface with A/D converters by continuously updating conversion values stored in the RAM.

Continuous transfer mode allows transfer of an uninterrupted bit stream. From 8 to 512 bits can be transferred without CPU intervention. Longer transfers are possible, but minimal intervention is required to prevent loss of data. A standard delay of 17 system clocks (0.8 µs with a 40-MHz system clock) is inserted between the transfer of each queue entry.

6.7.1 QSPI Registers

The QSPI memory map, shown in **Table 6-12**, includes the QSMCM global and pin control registers, four QSPI control registers (SPCR[0:3]), the status register (SPSR), and the QSPI RAM. Registers and RAM can be read and written by the CPU. The memory map can be divided into supervisor-only data space and assignable data space. The address offsets shown are from the base address of the QSMCM module. Refer to **Figure 1-2** for a diagram of the MC68F375 internal memory map.

Table 6-12 QSPI Register Map

Access[1]	Address	MSB[2]	LSB
S/U	0xYF FC18	QSPI Control Register 0 (SPCR0) See **Table 6-13** for bit descriptions.	
S/U	0xYF FC1A	QSPI Control Register 1 (SPCR1) See **Table 6-15** for bit descriptions.	
S/U	0xYF FC1C	QSPI Control Register 2 (SPCR2) See **Table 6-16** for bit descriptions.	
S/U	0xYF FC1E/ 0xYF FC1F	QSPI Control Register 3 (SPCR3) See **Table 6-17** for bit descriptions.	QSPI Status Register (SPSR) See **Table 6-18** for bit descriptions.
S/U	0xYF FD40 – 0xYF FD7F	Receive Data RAM (32 half-words)	
S/U	0xYF FD80 – 0xYF FDBF	Transmit Data RAM (32 half-words)	
S/U	0xYF FDC0 – 0xYF FDDF	Command RAM (32 bytes)	

NOTES:
1. S = Supervisor access only
   S/U = Supervisor access only or unrestricted user access (assignable data space).
2. 8-bit registers, such as SPCR3 and SPSR, are on 8-bit boundaries. 16-bit registers such as SPCR0 are on 16-bit boundaries.

To ensure proper operation, set the QSPI enable bit (SPE) in SPCR1 only after initializing the other control registers. Setting this bit starts the QSPI.

Rewriting the same value to a control register does not affect QSPI operation with the exception of writing NEWQP in SPCR2. Rewriting the same value to these bits causes the RAM queue pointer to restart execution at the designated location.

Before changing control bits, the user should halt the QSPI. Writing a different value into a control register other than SPCR2 while the QSPI is enabled may disrupt operation. SPCR2 is buffered, preventing any disruption of the current serial transfer. After the current serial transfer is completed, the new SPCR2 value becomes effective.

6.7.1.1 QSPI Control Register 0

SPCR0 contains parameters for configuring the QSPI before it is enabled. The CPU has read/write access to SPCR0, but the QSPI has read access only. SPCR0 must be initialized before QSPI operation begins. Writing a new value to SPCR0 while the QSPI is enabled disrupts operation.

SPCR0 — QSPI Control Register 0 **0xYF FC18**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSTR	WOMQ	BITS				CPOL	CPHA	SPBR							
RESET:															
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0

Table 6-13 SPCR0 Bit Settings

Bit(s)	Name	Description
15	MSTR	Master/slave mode select 0 = QSPI is a slave device and only responds to externally generated serial transfers. 1 = QSPI is the system master and can initiate transmission to external SPI devices.
14	WOMQ	Wired-OR mode for QSPI pins. This bit controls the QSPI pins regardless of whether they are used as general-purpose outputs or as QSPI outputs, and regardless of whether the QSPI is enabled or disabled. 0 = Pins designated for output by DDRQS operate in normal mode. 1 = Pins designated for output by DDRQS operate in open drain mode.
13:10	BITS	Bits per transfer. In master mode, when BITSE is set in a command RAM byte, BITS determines the number of data bits transferred. When BITSE is cleared, eight bits are transferred regardless of the value in BITS. In slave mode, the BITS field always determines the number of bits the QSPI will receive during each transfer before storing the received data. Data transfers from 8 to 16 bits are supported. Illegal (reserved) values default to eight bits. **Table 6-14** shows the number of bits per transfer.
9	CPOL	Clock polarity. CPOL is used to determine the inactive state of the serial clock (SCK). It is used with CPHA to produce a desired clock/data relationship between master and slave devices. 0 = The inactive state of SCK is logic zero. 1 = The inactive state of SCK is logic one.
8	CPHA	Clock phase. CPHA determines which edge of SCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices. 0 = Data is captured on the leading edge of SCK and changed on the trailing edge of SCK. 1 = Data is changed on the leading edge of SCK and captured on the trailing edge of SCK
7:0	SPBR	Serial clock baud rate. The QSPI uses a modulus counter to derive the SCK baud rate from the MCU system clock. Baud rate is selected by writing a value from 2 to 255 into SPBR. The following equation determines the SCK baud rate: $\text{SCK Baud Rat} = \frac{f_{SYS}}{2 \times SPBR}$ Refer to **6.7.5.2 Baud Rate Selection** for more information.

Table 6-14 Bits Per Transfer

BITS[3:0]	Bits per Transfer
0000	16
0001 to 0111	Reserved (defaults to 8)
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

6.7.1.2 QSPI Control Register 1

SPCR1 enables the QSPI and specifies transfer delays. The CPU has read/write access to SPCR1, but the QSPI has read access only to all bits except SPE. SPCR1 must be written last during initialization because it contains SPE. The QSPI automatically clears this bit after it completes all serial transfers or when a mode fault occurs. Writing a new value to SPCR1 while the QSPI is enabled disrupts operation.

SPCR1 — QSPI Control Register 1 **0xYF FC1A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
SPE	DSCKL							DTL							
RESET:															
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

Table 6-15 SPCR1 Bit Settings

Bit(s)	Name	Description
15	SPE	QSPI enable. Refer to **6.7.4.1 Enabling, Disabling, and Halting the SPI**. 0 = QSPI is disabled. QSPI pins can be used for general-purpose I/O. 1 = QSPI is enabled. Pins allocated by PQSPAR are controlled by the QSPI.
14:8	DSCKL	Delay before SCK. When the DSCK bit is set in a command RAM byte, this field determines the length of the delay from PCS valid to SCK transition. The following equation determines the actual delay before SCK: $\text{PCS to SCK Delay} = \frac{\text{DSCKL}}{f_{SYS}}$ where DSCKL equals is in the range of 1 to 127. Refer to **6.7.5.3 Delay Before Transfer** for more information.
7:0	DTL	Length of delay after transfer. When the DT bit is set in a command RAM byte, this field determines the length of the delay after a serial transfer. The following equation is used to calculate the delay: $\text{Delay after Transfer} = \frac{32 \times \text{DTL}}{f_{SYS}}$ where DTL is in the range of 1 to 255. A zero value for DTL causes a delay-after-transfer value of $8192 \div f_{SYS}$ (204.8 µµs with a 40-MHz system clock). Refer to **6.7.5.4 Delay After Transfer** for more information.

6.7.1.3 QSPI Control Register 2

SPCR2 contains QSPI queue pointers, wraparound mode control bits, and an interrupt enable bit. The CPU has read/write access to SPCR2, but the QSPI has read access only. Writes to this register are buffered. New SPCR2 values become effective only after completion of the current serial transfer. Rewriting NEWQP in SPCR2 causes execution to restart at the designated location. Reads of SPCR2 return the current value of the register, not the buffer.

SPCR2 — QSPI Control Register 2 **0xYF FC1C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
SPIFIE	WREN	WRTO	ENDQP					RESERVED			NEWQP				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-16 SPCR2 Bit Settings

Bit(s)	Name	Description
15	SPIFIE	SPI finished interrupt enable. Refer to **6.7.4.2 QSPI Interrupts**. 0 = QSPI interrupts disabled 1 = QSPI interrupts enabled
14	WREN	Wrap enable. Refer to **6.7.5.7 Master Wraparound Mode**. 0 = Wraparound mode disabled. 1 = Wraparound mode enabled.
13	WRTO	Wrap to. When wraparound mode is enabled and after the end of queue has been reached, WRTO determines which address the QSPI executes next. The end of queue is determined by an address match with ENDQP. 0 = Wrap to pointer address 0x0 1 = Wrap to address in NEWQP
12:8	ENDQP	Ending queue pointer. This field determines the last absolute address in the queue to be completed by the QSPI. After completing each command, the QSPI compares the queue pointer value of the just-completed command with the value of ENDQP. If the two values match, the QSPI sets SPIF to indicate it has reached the end of the programmed queue. Refer to **6.7.4 QSPI Operation** for more information.
7:5	—	Reserved
4:0	NEWQP	New queue pointer value. This field contains the first QSPI queue address. Refer to **6.7.4 QSPI Operation** for more information.

6.7.1.4 QSPI Control Register 3

SPCR3 contains the loop mode enable bit, halt and mode fault interrupt enable, and the halt control bit. The CPU has read/write access to SPCR3, but the QSPI has read access only. SPCR3 must be initialized before QSPI operation begins. Writing a new value to SPCR3 while the QSPI is enabled disrupts operation.

SPCR3 — QSPI Control Register 3 **0xYF FC1E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED					LOOPQ	HMIE	HALT	SPSR*							
RESET:															
0	0	0	0	0	0	0	0								

*See bit descriptions in **Table 6-18**.

Table 6-17 SPCR3 Bit Settings

Bit(s)	Name	Description
15:11	—	Reserved
10	LOOPQ	QSPI loop mode. LOOPQ controls feedback on the data serializer for testing. 0 = Feedback path disabled. 1 = Feedback path enabled.
9	HMIE	HALTA and MODF interrupt enable. HMIE enables interrupt requests generated by the HALTA status flag or the MODF status flag in SPSR. 0 = HALTA and MODF interrupts disabled. 1 = HALTA and MODF interrupts enabled.
8	HALT	Halt QSPI. When HALT is set, the QSPI stops on a queue boundary. It remains in a defined state from which it can later be restarted. Refer to **6.7.4.1 Enabling, Disabling, and Halting the SPI**. 0 = QSPI operates normally. 1 = QSPI is halted for subsequent restart.
7:0	—	SPSR. See**Table 6-18** for bit descriptions.

6.7.1.5 QSPI Status Register

The SPSR contains information concerning the current serial transmission. Only the QSPI can set bits in this register. To clear status flags, the CPU reads SPSR with the flags set and then writes the SPSR with zeros in the appropriate bits. Writes to CPTQP have no effect.

SPSR — QSPI Status Register **0xYF FC1F**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
SPCR3*								SPIF	MODF	HAL-TA	CPTQP				
								0	0	0	0	0	0	0	0

*See bit descriptions in **Table 6-17**.

Table 6-18 SPSR Bit Settings

Bit(s)	Name	Description
15:8	SPCR3	See bit descriptions in **Table 6-17**.
7	SPIF	QSPI finished flag. SPIF is set after execution of the command at the address in ENDQP in SPCR2. If wraparound mode is enabled (WREN = 1), the SPIF is set, after completion of the command defined by ENDQP, each time the QSPI cycles through the queue. 0 = QSPI is not finished 1 = QSPI is finished
6	MODF	Mode fault flag. The QSPI asserts MODF when the QSPI is in master mode (MSTR = 1) and the $\overline{SS}$ input pin is negated by an external driver. Refer to **6.7.8 Mode Fault** for more information. 0 = Normal operation 1 = Another SPI node requested to become the network SPI master while the QSPI was enabled in master mode ($\overline{SS}$ input taken low).
5	HALTA	Halt acknowledge flag. HALTA is set when the QSPI halts in response to setting the HALT bit in SPCR3. HALTA is also set when the IMB3 FREEZE signal is asserted, provided the FRZ1 bit in the QSMCMMCR is set. To prevent undefined operation, the user must not modify any QSPI control registers or RAM while the QSPI is halted. If HMIE in SPCR3 is set the QSPI sends interrupt requests to the CPU when HALTA is asserted. 0 = QSPI is not halted. 1 = QSPI is halted
4:0	CPTQP	Completed queue pointer. CPTQP points to the last command executed. It is updated when the current command is complete. When the first command in a queue is executing, CPTQP contains either the reset value 0x0 or a pointer to the last command completed in the previous queue. If the QSPI is halted, CPTQP may be used to determine which commands have not been executed. The CPTQP may also be used to determine which locations in the receive data segment of the QSPI RAM contain valid received data.

6.7.2 QSPI RAM

The QSPI contains a 160-byte block of dual-ported static RAM that can be accessed by both the QSPI and the CPU. Because of this dual access capability, up to two wait states may be inserted into CPU access time if the QSPI is in operation.

The size and type of access of the QSPI RAM by the CPU affects the QSPI access time. The QSPI allows byte, half-word, and word accesses. Only word accesses of the RAM by the CPU are coherent because these accesses are an indivisible operation. If the CPU makes a coherent access of the QSPI RAM, the QSPI cannot access the QSPI RAM until the CPU is finished. However, a word or misaligned word access is not coherent because the CPU must break its access of the QSPI RAM into two parts, which allows the QSPI to access the QSPI RAM between the two accesses by the CPU.

The RAM is divided into three segments: receive data RAM, transmit data RAM, and command data RAM. Receive data is information received from a serial device external to the MCU. Transmit data is information stored for transmission to an external device. Command data defines transfer parameters. **Figure 6-3** shows RAM organization.

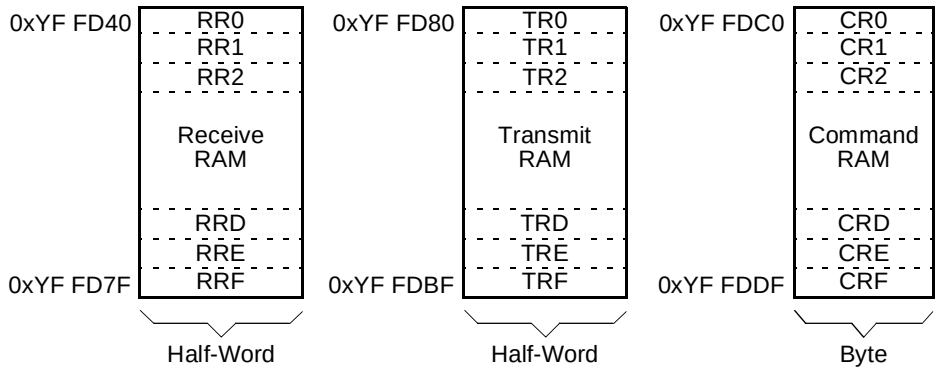


Figure 6-3 QSPI RAM

6.7.2.1 Receive RAM

Data received by the QSPI is stored in this segment, to be read by the CPU. Data stored in the receive RAM is right-justified, i.e., the least significant bit is always in the right-most bit position within the word regardless of the serial transfer length. Unused bits in a receive queue entry are set to zero by the QSPI upon completion of the individual queue entry. The CPU can access the data using byte, half-word, or word addressing.

The CPTQP value in SPSR shows which queue entries have been executed. The CPU uses this information to determine which locations in receive RAM contain valid data before reading them.

6.7.2.2 Transmit RAM

Data that is to be transmitted by the QSPI is stored in this segment. The CPU normally writes one word of data into this segment for each queue command to be executed. If the corresponding peripheral, such as a serial input port, is used solely to input data, then this segment does not need to be initialized.

Data must be written to transmit RAM in a right-justified format. The QSPI cannot modify information in the transmit RAM. The QSPI copies the information to its data serializer for transmission. Information remains in transmit RAM until overwritten.

6.7.2.3 Command RAM

Command RAM is used by the QSPI in master mode. The CPU writes one byte of control information to this segment for each QSPI command to be executed. The QSPI cannot modify information in command RAM.

Command RAM consists of 32 bytes. Each byte is divided into two fields. The peripheral chip-select field, enables peripherals for transfer. The command control field provides transfer options.

A maximum of 32 commands can be in the queue. These bytes are assigned an address from 0x00 to 0x1F. Queue execution by the QSPI proceeds from the address in NEWQP through the address in ENDQP. (Both of these fields are in SPCR2.)

CR[0:F] — Command RAM **0xYF FDC0 – 0xYF FDDF**

7	6	5	4	3	2	1	0
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0[1]
—	—	—	—	—	—	—	—
CONT	BITSE	DT	DSCK	PCS3	PCS2	PCS1	PCS0[1]
Command Control				Peripheral Chip Select			

NOTES:
1. The PCS0 bit represents the dual-function PCS0/$\overline{SS}$.

Table 6-19 Command RAM Bit Settings

Bit(s)	Name	Description
0	CONT	Continue 0 = Control of chip selects returned to PORTQS after transfer is complete. 1 = Peripheral chip selects remain asserted after transfer is complete.
1	BITSE	Bits per transfer enable 0 = Eight bits 1 = Number of bits set in BITS field of SPCR0.
2	DT	Delay after transfer 0 = Delay after transfer is $17 \div f_{SYS}$. 1 = SPCR1 DTL[7:0] specifies delay after transfer PCS valid to SCK.
3	DSCK	PCS to SCK Delay 0 = PCS valid to SCK delay is one-half SCK. 1 = SPCR1 DSCKL[6:0] specifies delay from PCS valid to SCK.
4:7	PCS[3:0]	Peripheral chip selects. Use peripheral chip-select bits to select an external device for serial data transfer. More than one peripheral chip select may be activated at a time, and more than one peripheral chip can be connected to each PCS pin, provided proper fanout is observed. PCS0 shares a pin with the slave select ($\overline{SS}$) signal, which initiates slave mode serial transfer. If $\overline{SS}$ is taken low when the QSPI is in master mode, a mode fault occurs.

Refer to **6.7.5 Master Mode Operation** for more information on the command RAM.

6.7.3 QSPI Pins

Seven pins are associated with the QSPI. When not needed by the QSPI, they can be configured for general-purpose I/O. **Table 6-20** identifies the QSPI pins and their functions. Register DDRQS determines whether the pins are designated as input or output. The user must initialize DDRQS for the QSPI to function correctly.

Table 6-20 QSPI Pin Functions

Pin Names	Mnemonic	Mode	Function
Master in slave out	MISO	Master Slave	Serial data input to QSPI Serial data output from QSPI
Master out slave in	MOSI	Master Slave	Serial data output from QSPI Serial data input to QSPI
Serial clock	SCK[1]	Master Slave	Clock output from QSPI clock Input to QSPI
Peripheral chip selects	PCS[1:3]	Master	Outputs select peripheral(s)
Peripheral chip select[2] Slave select[3]	PCS0/ $\overline{SS}$	Master Slave	Output selects peripheral(s) Input selects the QSPI
Slave select[4]	$\overline{SS}$	Master	May cause mode fault

NOTES:

1. All QSPI pins (except SCK) can be used as general-purpose I/O if they are not used by the QSPI while the QSPI is operating. SCK can only be used for general-purpose I/O if the QSPI is disabled.
2. An output (PCS0) when the QSPI is in master mode.
3. An input ($\overline{SS}$) when the QSPI is in slave mode.
4. An input ($\overline{SS}$) when the QSPI is in master mode; useful in multimaster systems.

6.7.4 QSPI Operation

The QSPI uses a dedicated 160-byte block of static RAM accessible by both the QSPI and the CPU to perform queued operations. The RAM is divided into three segments: 32 command control bytes, 64 transmit data bytes, and 64 receive data bytes.

Once the CPU has set up a queue of QSPI commands, written the transmit data segment with information to be sent, and enabled the QSPI, the QSPI operates independently of the CPU. The QSPI executes all of the commands in its queue, sets a flag indicating completion, and then either interrupts the CPU or waits for CPU intervention.

QSPI RAM is organized so that one byte of command data, one word of transmit data, and one word of receive data correspond to each queue entry, 0x0 to 0x2F.

The CPU initiates QSPI operation by setting up a queue of QSPI commands in command RAM, writing transmit data into transmit RAM, then enabling the QSPI. The QSPI executes the queued commands, sets a completion flag (SPIF), and then either interrupts the CPU or waits for intervention.

There are four queue pointers. The CPU can access three of them through fields in QSPI registers. The new queue pointer (NEWQP), contained in SPCR2, points to the first command in the queue. An internal queue pointer points to the command currently being executed. The completed queue pointer (CPTQP), contained in SPSR, points to the last command executed. The end queue pointer (ENDQP), contained in SPCR2, points to the final command in the queue.

The internal pointer is initialized to the same value as NEWQP. During normal operation, the command pointed to by the internal pointer is executed, the value in the internal pointer is copied into CPTQP, the internal pointer is incremented, and then the

sequence repeats. Execution continues at the internal pointer address unless the NEWQP value is changed. After each command is executed, ENDQP and CPTQP are compared. When a match occurs, the SPIF flag is set and the QSPI stops and clears SPE, unless wraparound mode is enabled.

At reset, NEWQP is initialized to 0x0. When the QSPI is enabled, execution begins at queue address 0x0 unless another value has been written into NEWQP. ENDQP is initialized to 0x0 at reset but should be changed to the last queue entry before the QSPI is enabled. NEWQP and ENDQP can be written at any time. When NEWQP changes, the internal pointer value also changes. However, if NEWQP is written while a transfer is in progress, the transfer is completed normally. Leaving NEWQP and ENDQP set to 0x0 transfers only the data in transmit RAM location 0x0.

6.7.4.1 Enabling, Disabling, and Halting the SPI

The SPE bit in the SPCR1 enables or disables the QSPI submodule. Setting SPE causes the QSPI to begin operation. If the QSPI is a master, setting SPE causes the QSPI to begin initiating serial transfers. If the QSPI is a slave, the QSPI begins monitoring the PCS0/$\overline{SS}$ pin to respond to the external initialization of a serial transfer.

When the QSPI is disabled, the CPU may use the QSPI RAM. When the QSPI is enabled, both the QSPI and the CPU have access to the QSPI RAM. The CPU has both read and write access to all 160 bytes of the QSPI RAM. The QSPI can read-only the transmit data segment and the command control segment and can write-only the receive data segment of the QSPI RAM.

The QSPI turns itself off automatically when it is finished by clearing SPE. An error condition called mode fault (MODF) also clears SPE. This error occurs when PCS0/$\overline{SS}$ is configured for input, the QSPI is a system master (MSTR = 1), and PCS0/$\overline{SS}$ is driven low externally.

Setting the HALT bit in SPCR3 stops the QSPI on a queue boundary. The QSPI halts in a known state from which it can later be restarted. When HALT is set, the QSPI finishes executing the current serial transfer (up to 16 bits) and then halts. While halted, if the command control bit (CONT of the QSPI RAM) for the last command was asserted, the QSPI continues driving the peripheral chip select pins with the value designated by the last command before the halt. If CONT was cleared, the QSPI drives the peripheral chip-select pins to the value in register PORTQS.

If HALT is set during the last command in the queue, the QSPI completes the last command, sets both HALTA and SPIF, and clears SPE. If the last queue command has not been executed, asserting HALT does not set SPIF or clear SPE. QSPI execution continues when the CPU clears HALT.

To stop the QSPI, assert the HALT bit in SPCR3, then wait until the HALTA bit in SPSR is set. SPE can then be safely cleared, providing an orderly method of shutting down the QSPI quickly after the current serial transfer is completed. The CPU can disable the QSPI immediately by clearing SPE. However, loss of data from a current serial transfer may result and confuse an external SPI device.

6.7.4.2 QSPI Interrupts

The QSPI has three possible interrupt sources but only one interrupt vector. These sources are SPIF, MODF, and HALTA. When the CPU responds to a QSPI interrupt, the user must ascertain the interrupt cause by reading the SPSR. Any interrupt that was set may then be cleared by writing to SPSR with a zero in the bit position corresponding to the interrupt source.

The SPIFIE bit in SPCR2 enables the QSPI to generate an interrupt request upon assertion of the SPIF status flag. Because it is buffered, the value written to SPIFIE applies only upon completion of the queue (the transfer of the entry indicated by ENDPQ). Thus, if a single sequence of queue entries is to be transferred (i.e., no WRAP), then SPIFIE should be set to the desired state before the first transfer.

If a sub-queue is to be used, the same CPU write that causes a branch to the sub-queue may enable or disable the SPIF interrupt for the sub-queue. The primary queue retains its own selected interrupt mode, either enabled or disabled.

The SPIF interrupt must be cleared by clearing SPIF. Subsequent interrupts may then be prevented by clearing SPIFIE. Clearing SPIFIE does not immediately clear an interrupt already caused by SPIF.

6.7.4.3 QSPI Flow

The QSPI operates in either master or slave mode. Master mode is used when the MCU initiates data transfers. Slave mode is used when an external device initiates transfers. Switching between these modes is controlled by MSTR in SPCR0. Before entering either mode, appropriate QSMCM and QSPI registers must be initialized properly.

In master mode, the QSPI executes a queue of commands defined by control bits in each command RAM queue entry. Chip-select pins are activated, data is transmitted from the transmit RAM and received by the receive RAM.

In slave mode, operation proceeds in response to SS pin assertion by an external SPI bus master. Operation is similar to master mode, but no peripheral chip selects are generated, and the number of bits transferred is controlled in a different manner. When the QSPI is selected, it automatically executes the next queue transfer to exchange data with the external device correctly.

Although the QSPI inherently supports multi-master operation, no special arbitration mechanism is provided. A mode fault flag (MODF) indicates a request for SPI master arbitration. System software must provide arbitration. Note that unlike previous SPI systems, MSTR is not cleared by a mode fault being set nor are the QSPI pin output drivers disabled. The QSPI and associated output drivers must be disabled by clearing SPE in SPCR1.

Figure 6-4 shows QSPI initialization. **Figure 6-5** through **Figure 6-9** show QSPI master and slave operation. The CPU must initialize the QSMCM global and pin registers and the QSPI control registers before enabling the QSPI for either mode of operation. The command queue must be written before the QSPI is enabled for master mode

operation. Any data to be transmitted should be written into transmit RAM before the QSPI is enabled. During wraparound operation, data for subsequent transmissions can be written at any time.

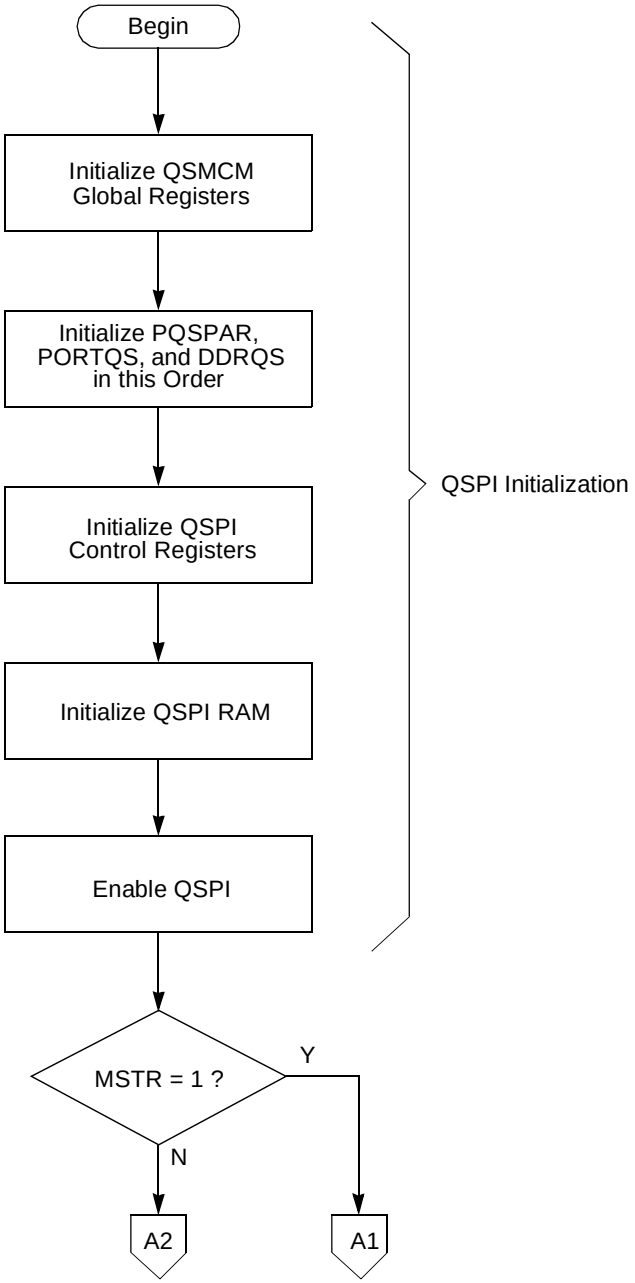


Figure 6-4 Flowchart of QSPI Initialization Operation

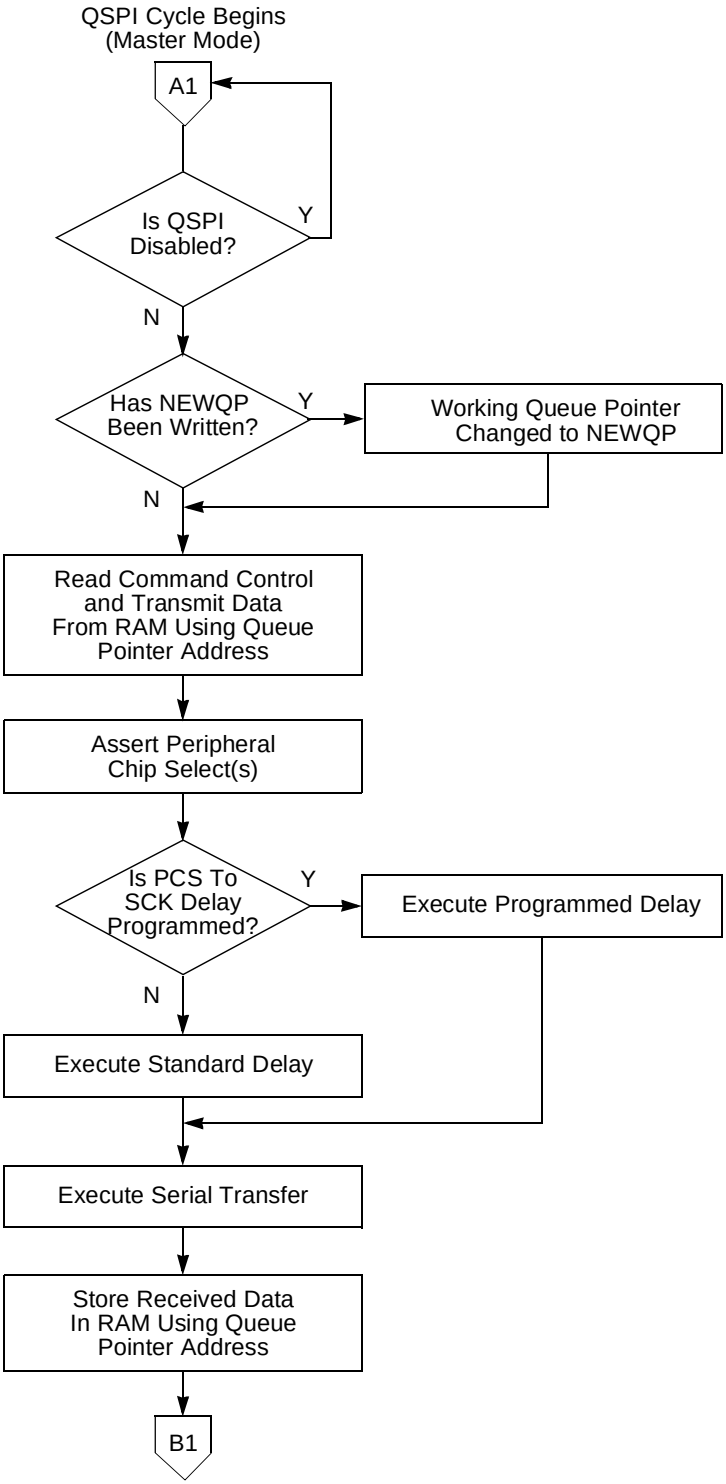


Figure 6-5 Flowchart of QSPI Master Operation (Part 1)

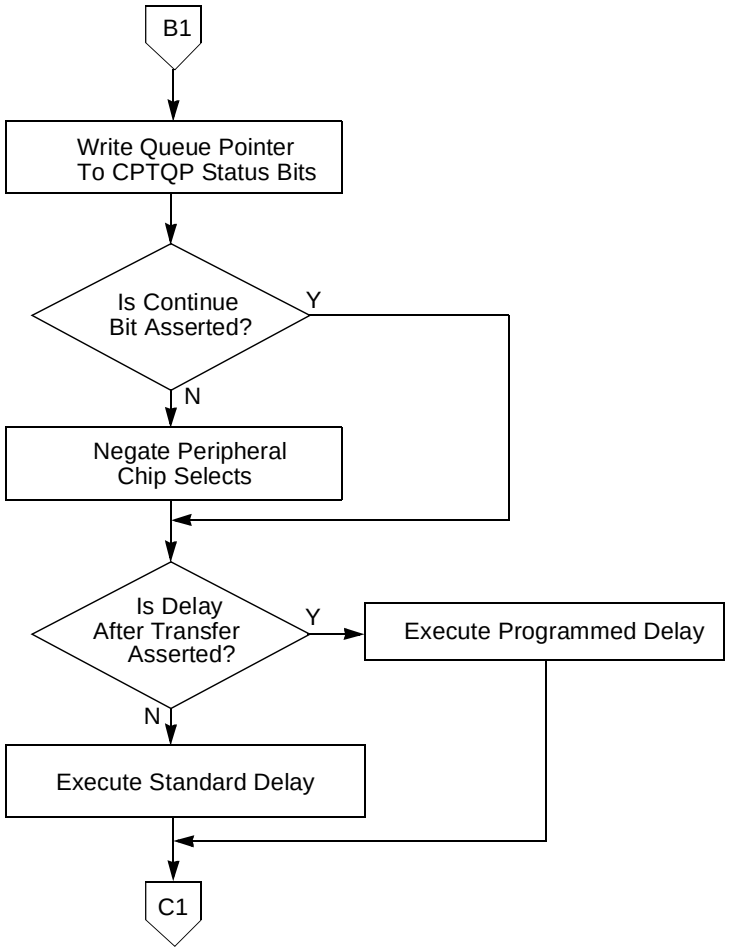


Figure 6-6 Flowchart of QSPI Master Operation (Part 2)

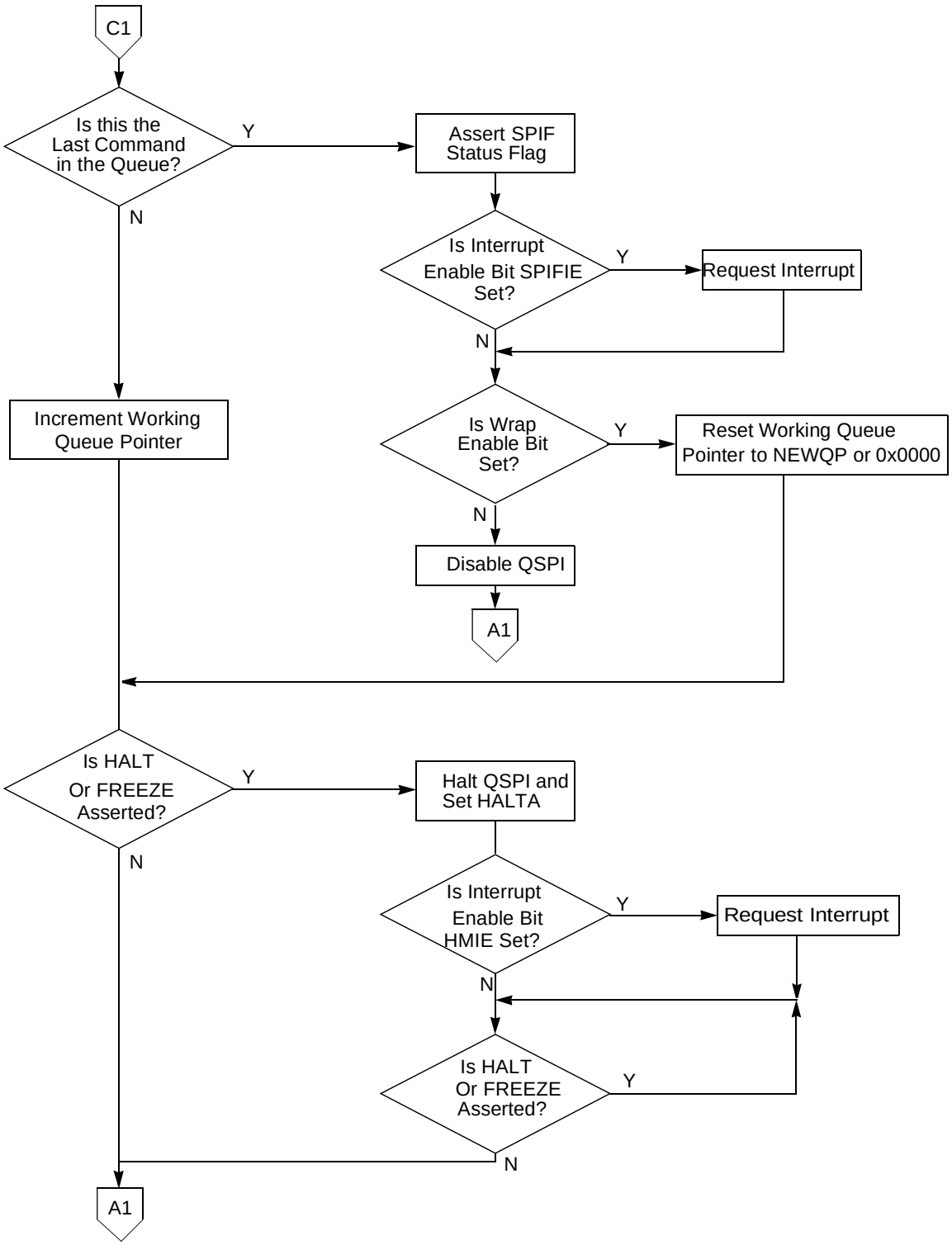


Figure 6-7 Flowchart of QSPI Master Operation (Part 3)

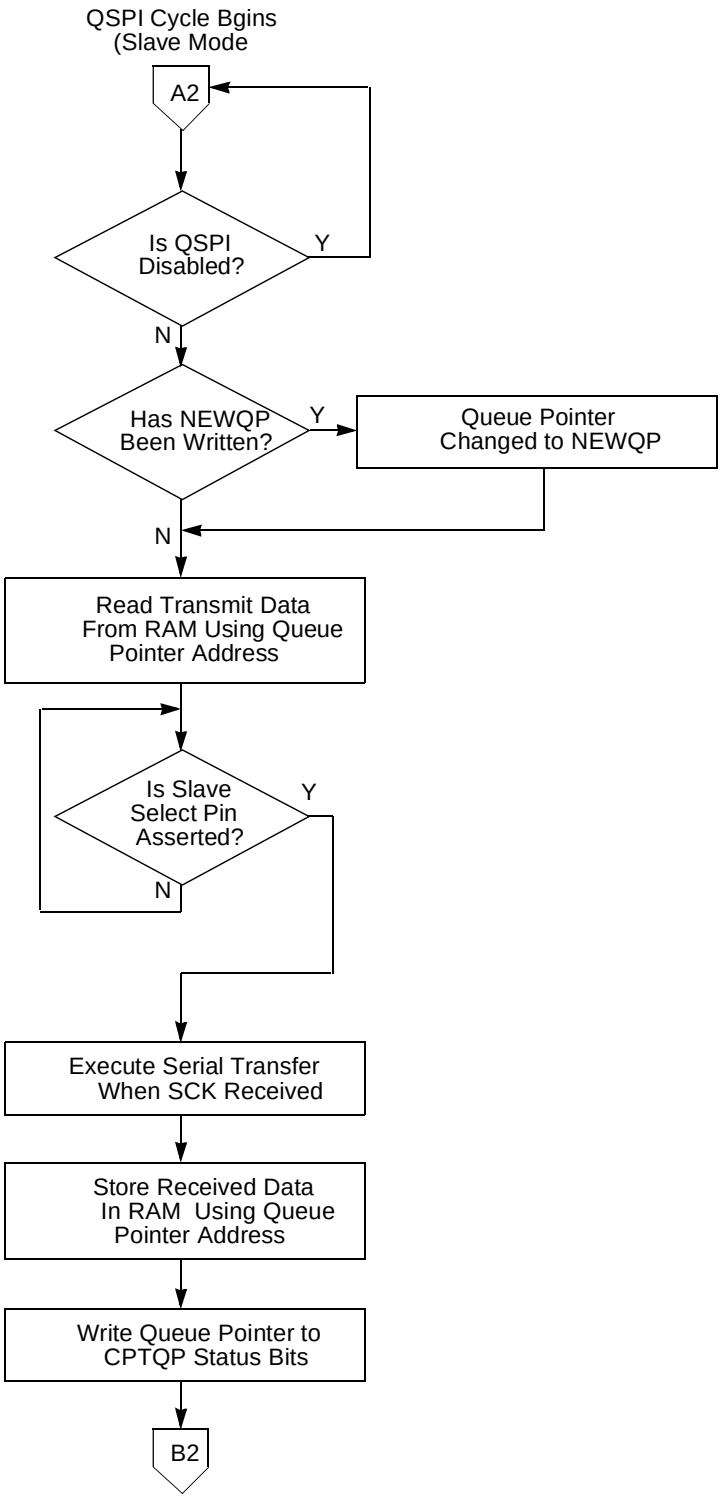


Figure 6-8 Flowchart of QSPI Slave Operation (Part 1)

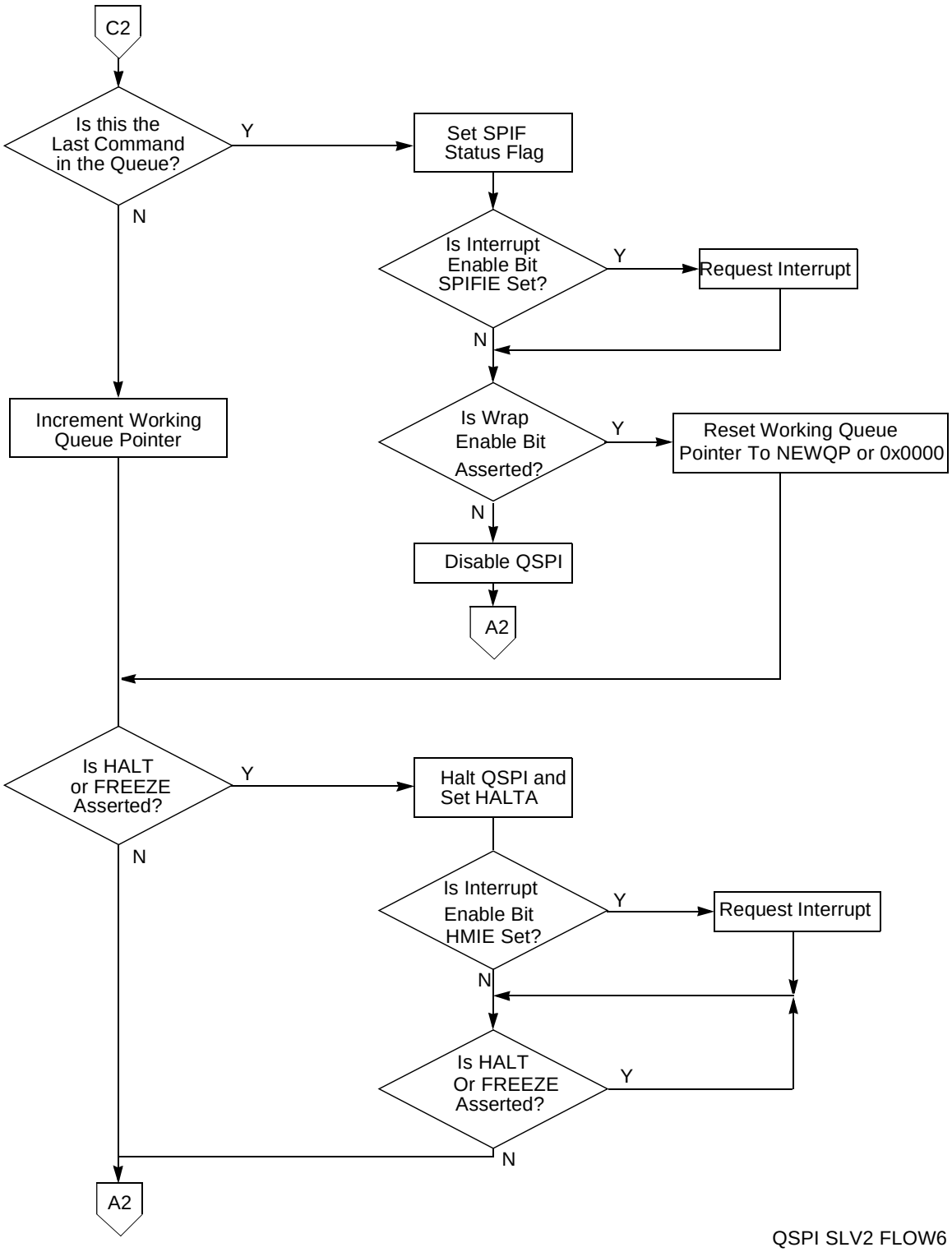


Figure 6-9 Flowchart of QSPI Slave Operation (Part 2)

Normally, the SPI bus performs synchronous bi-directional transfers. The serial clock on the SPI bus master supplies the clock signal SCK to time the transfer of data. Four

possible combinations of clock phase and polarity can be specified by the CPHA and CPOL bits in SPCR0.

Data is transferred with the most significant bit first. The number of bits transferred per command defaults to eight, but can be set to any value from eight to sixteen bits by writing a value into the BITS field in SPCR0 and setting BITSE in command RAM.

Typically, SPI bus outputs are not open drain unless multiple SPI masters are in the system. If needed, the WOMQ bit in SPCR0 can be set to provide wired-OR, open drain outputs. An external pull-up resistor should be used on each output line. WOMQ affects all QSPI pins regardless of whether they are assigned to the QSPI or used as general-purpose I/O.

6.7.5 Master Mode Operation

Setting the MSTR bit in SPCR0 selects master mode operation. In master mode, the QSPI can initiate serial transfers, but cannot respond to externally initiated transfers. When the slave select input of a device configured for master mode is asserted, a mode fault occurs.

Before QSPI operation begins, PQSPAR must be written to assign the necessary pins to the QSPI. The pins necessary for master mode operation are MISO, MOSI, SCK, and one or more of the chip-select pins. MISO is used for serial data input in master mode, and MOSI is used for serial data output. Either or both may be necessary, depending on the particular application. SCK is the serial clock output in master mode and must be assigned to the QSPI for proper operation.

The PORTQS data register must next be written with values that make the QGPIO6/SCK (bit 13 QDSCK of PORTQS) and QGPIO[3:0]/PCS[3:0] (bits 12:9 QDPCS[3:0] of PORTQS) outputs inactive when the QSPI completes a series of transfers. Pins allocated to the QSPI by PQSPAR are controlled by PORTQS when the QSPI is inactive. PORTQS I/O pins driven to states opposite those of the inactive QSPI signals can generate glitches that momentarily enable or partially clock a slave device.

For example, if a slave device operates with an inactive SCK state of logic one (CPOL = 1) and uses active low peripheral chip-select PCS0, the QDSCK and QDPCS0 bits in PORTQS must be set to 0b11. If QDSCK and QDPCS0 = 0b00, falling edges will appear on QGPIO6/SCK and GPIO0/PCS0 as the QSPI relinquishes control of these pins and PORTQS drives them to logic zero from the inactive SCK and PCS0 states of logic one.

Before master mode operation is initiated, QSMCM register DDRQS is written last to direct the data flow on the QSPI pins used. Configure the SCK, MOSI and appropriate chip-select pins PCS[3:0] as outputs. The MISO pin must be configured as an input.

After pins are assigned and configured, write appropriate data to the command queue. If data is to be transmitted, write the data to transmit RAM. Initialize the queue pointers as appropriate.

QSPI operation is initiated by setting the SPE bit in SPCR1. Shortly after SPE is set, the QSPI executes the command at the command RAM address pointed to by NEWQP. Data at the pointer address in transmit RAM is loaded into the data serializer and transmitted. Data that is simultaneously received is stored at the pointer address in receive RAM.

When the proper number of bits have been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads the next data for transfer from transmit RAM. The command pointed to by the incremented working queue pointer is executed next, unless a new value has been written to NEWQP. If a new queue pointer value is written while a transfer is in progress, that transfer is completed normally.

When the CONT bit in a command RAM byte is set, PCS pins are continuously driven to specified states during and between transfers. If the chip-select pattern changes during or between transfers, the original pattern is driven until execution of the following transfer begins. When CONT is cleared, the data in register PORTQS is driven between transfers. The data in PORTQS must match the inactive states of SCK and any peripheral chip-selects used.

When the QSPI reaches the end of the queue, it sets the SPIF flag. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled.

6.7.5.1 Clock Phase and Polarity

In master mode, data transfer is synchronized with the internally-generated serial clock SCK. Control bits, CPHA and CPOL, in SPCR0, control clock phase and polarity. Combinations of CPHA and CPOL determine upon which SCK edge to drive outgoing data from the MOSI pin and to latch incoming data from the MISO pin.

6.7.5.2 Baud Rate Selection

Baud rate is selected by writing a value from two to 255 into the SPBR field in SPCR0. The QSPI uses a modulus counter to derive the SCK baud rate from the MCU system clock.

The following expressions apply to the SCK baud rate:

$$\text{SCK Baud Rate} = \frac{f_{SYS}}{2 \times \text{SPBR}}$$

or

$$\text{SPBR} = \frac{f_{SYS}}{2 \times \text{SCK Baud Rate Desired}}$$

Giving SPBR a value of zero or one disables the baud rate generator. SCK is disabled and assumes its inactive state. At reset, the SCK baud rate is initialized to one eighth of the system clock frequency.

Table 6-21 provides some example SCK baud rates with a 40-MHz system clock.

Table 6-21 Example SCK Frequencies with a 40-MHz System Clock

Division Ratio	SPBR Value	SCK Frequency
4	2	10.00 MHz
6	3	6.67 MHz
8	4	5.00 MHz
14	7	2.86 MHz
28	14	1.43 MHz
58	29	689 kHz
280	140	143 kHz
510	255	78.43 kHz

6.7.5.3 Delay Before Transfer

The DSCK bit in each command RAM byte inserts either a standard (DSCK = 0) or user-specified (DSCK = 1) delay from chip-select assertion until the leading edge of the serial clock. The DSCKL field in SPCR1 determines the length of the user-defined delay before the assertion of SCK. The following expression determines the actual delay before SCK:

$$\text{PCS to SCK Delay} = \frac{\text{DSCKL}}{f_{SYS}}$$

where DSCKL is in the range from 1 to 127.

NOTE

A zero value for DSCKL causes a delay of 128 system clocks, which equals 3.2 µs for a 40-MHz system clock. Because of design limits, a DSCKL value of one defaults to the same timing as a value of two.

When DSCK equals zero, DSCKL is not used. Instead, the PCS valid-to-SCK transition is one-half the SCK period.

6.7.5.4 Delay After Transfer

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion. Writing a value to the DTL field in SPCR1 specifies a delay period. The DT bit in each command RAM byte determines whether the standard delay period

(DT = 0) or the specified delay period (DT = 1) is used. The following expression is used to calculate the delay:

$$\text{Delay after Transfer} = \frac{32 \times \text{DTL}}{f_{SYS}}$$

where DTL is in the range from one to 255.

A zero value for DTL causes a delay-after-transfer value of 8192 ÷ system clock frequency (204.8 µs with a 40-MHz system clock).

If DT is zero in a command RAM byte, a standard delay is inserted.

$$\text{Standard Delay after Transfer} = \frac{17}{f_{SYS}}$$

Delay after transfer can be used to provide a peripheral deselect interval. A delay can also be inserted between consecutive transfers to allow serial A/D converters to complete conversion.

Adequate delay between transfers must be specified for long data streams because the QSPI requires time to load a transmit RAM entry for transfer. Receiving devices need at least the standard delay between successive transfers. If the system clock is operating at a slower rate, the delay between transfers must be increased proportionately.

6.7.5.5 Transfer Length

There are two transfer length options. The user can choose a default value of eight bits, or a programmed value from eight (0b1000) to 16 (0b0000) bits, inclusive. Reserved values (from 0b0001 to 0b0111) default to eight bits. The programmed value must be written into the BITS field in SPCR0. The BITSE bit in each command RAM byte determines whether the default value (BITSE = 0) or the BITS value (BITSE = 1) is used.

6.7.5.6 Peripheral Chip Selects

Peripheral chip-select signals are used to select an external device for serial data transfer. Chip-select signals are asserted when a command in the queue is executed. Signals are asserted at a logic level corresponding to the value of the PCS[3:0] bits in each command byte. More than one chip-select signal can be asserted at a time, and more than one external device can be connected to each PCS pin, provided proper fanout is observed. PCS0 shares a pin with the slave select SS signal, which initiates slave mode serial transfer. If SS is taken low when the QSPI is in master mode, a mode fault occurs.

To configure a peripheral chip select, set the appropriate bit in PQSPAR, then configure the chip-select pin as an output by setting the appropriate bit in DDRQS. The value

of the bit in PORTQS that corresponds to the chip-select pin determines the base state of the chip-select signal. If the base state is zero, chip-select assertion must be active high (PCS bit in command RAM must be set); if base state is one, assertion must be active low (PCS bit in command RAM must be cleared). PORTQS bits are cleared during reset. If no new data is written to PORTQS before pin assignment and configuration as an output, the base state of chip-select signals is zero and chip-select pins are configured for active-high operation.

6.7.5.7 Master Wraparound Mode

Wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address 0x0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2.

In wraparound mode, the QSPI cycles through the queue continuously, even while the QSPI is requesting interrupt service. SPE is not cleared when the last command in the queue is executed. New receive data overwrites previously received data in receive RAM. Each time the end of the queue is reached, the SPIF flag is set. SPIF is not automatically reset. If interrupt-driven QSPI service is used, the service routine must clear the SPIF bit to end the current interrupt request. Additional interrupt requests during servicing can be prevented by clearing SPIFIE, but SPIFIE is buffered. Clearing it does not end the current request.

Wraparound mode is exited by clearing the WREN bit or by setting the HALT bit in SPCR3. Exiting wraparound mode by clearing SPE is not recommended, as clearing SPE may abort a serial transfer in progress. The QSPI sets SPIF, clears SPE, and stops the first time it reaches the end of the queue after WREN is cleared. After HALT is set, the QSPI finishes the current transfer, then stops executing commands. After the QSPI stops, SPE can be cleared.

6.7.6 Slave Mode

Clearing the MSTR bit in SPCR0 selects slave mode operation. In slave mode, the QSPI is unable to initiate serial transfers. Transfers are initiated by an external SPI bus master. Slave mode is typically used on a multi-master SPI bus. Only one device can be bus master (operate in master mode) at any given time.

Before QSPI operation is initiated, QSMCM register PQSPAR must be written to assign necessary pins to the QSPI. The pins necessary for slave mode operation are MISO, MOSI, SCK, and PCS0/SS. MISO is used for serial data output in slave mode, and MOSI is used for serial data input. Either or both may be necessary, depending on the particular application. SCK is the serial clock input in slave mode and must be assigned to the QSPI for proper operation. Assertion of the active-low slave select signal SS initiates slave mode operation.

Before slave mode operation is initiated, DDRQS must be written to direct data flow on the QSPI pins used. Configure the MOSI, SCK and PCS0/SS pins as inputs. The MISO pin must be configured as an output.

After pins are assigned and configured, write data to be transmitted into transmit RAM. Command RAM is not used in slave mode, and does not need to be initialized. Set the queue pointers, as appropriate.

When SPE is set and MSTR is clear, a low state on the slave select PCS0/SS pin begins slave mode operation at the address indicated by NEWQP. Data that is received is stored at the pointer address in receive RAM. Data is simultaneously loaded into the data serializer from the pointer address in transmit RAM and transmitted. Transfer is synchronized with the externally generated SCK. The CPHA and CPOL bits determine upon which SCK edge to latch incoming data from the MISO pin and to drive outgoing data from the MOSI pin.

Because the command RAM is not used in slave mode, the CONT, BITSE, DT, DSCK, and peripheral chip-select bits have no effect. The PCS0/SS pin is used only as an input.

The SPBR, DT and DSCKL fields in SPCR0 and SPCR1 bits are not used in slave mode. The QSPI drives neither the clock nor the chip-select pins and thus cannot control clock rate or transfer delay.

Because the BITSE option is not available in slave mode, the BITS field in SPCR0 specifies the number of bits to be transferred for all transfers in the queue. When the number of bits designated by BITS[3:0] has been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads new transmit data from transmit RAM into the data serializer. The working queue pointer address is used the next time PCS0/SS is asserted, unless the CPU32 writes to NEWQP first.

The QSPI shifts one bit for each pulse of SCK until the slave select input goes high. If SS goes high before the number of bits specified by the BITS field is transferred, the QSPI resumes operation at the same pointer address the next time SS is asserted. The maximum value that the BITS field can have is 16. If more than 16 bits are transmitted before SS is negated, pointers are incremented and operation continues.

The QSPI transmits as many bits as it receives at each queue address, until the BITS value is reached or SS is negated. SS does not need to go high between transfers as the QSPI transfers data until reaching the end of the queue, whether SS remains low or is toggled between transfers.

When the QSPI reaches the end of the queue, it sets the SPIF flag. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled.

Slave wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address 0x0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2. Slave wraparound operation is identical to master wraparound operation.

6.7.6.1 Description of Slave Operation

After reset, the QSMCM registers and the QSPI control registers must be initialized as described above. Although the command control segment is not used, the transmit and receive data segments may, depending upon the application, need to be initialized. If meaningful data is to be sent out from the QSPI, the user should write the data to the transmit data segment before enabling the QSPI.

If SPE is set and MSTR is not set, a low state on the slave select (PCS0/$\overline{SS}$) pin commences slave mode operation at the address indicated by NEWQP. The QSPI transmits the data found in the transmit data segment at the address indicated by NEWQP, and the QSPI stores received data in the receive data segment at the address indicated by NEWQP. Data is transferred in response to an external slave clock input at the SCK pin.

Because the command control segment is not used, the command control bits and peripheral chip-select codes have no effect in slave mode operation. The QSPI does not drive any of the four peripheral chip-selects as outputs. PCS0/$\overline{SS}$ is used as an input.

Although CONT cannot be used in slave mode, a provision is made to enable receipt of more than 16 data bits. While keeping the QSPI selected (PCS0/$\overline{SS}$ is held low), the QSPI stores the number of bits, designated by BITS, in the current receive data segment address, increments NEWQP, and continues storing the remaining bits (up to the BITS value) in the next receive data segment address.

As long as PCS0/$\overline{SS}$ remains low, the QSPI continues to store the incoming bit stream in sequential receive data segment addresses, until either the value in BITS is reached or the end-of-queue address is used with wraparound mode disabled.

When the end of the queue is reached, the SPIF flag is asserted, optionally causing an interrupt. If wraparound mode is disabled, any additional incoming bits are ignored.

If wraparound mode is enabled, storing continues at either address 0x0 or the address of NEWQP, depending on the WRTO value. When using this capability to receive a long incoming data stream, the proper delay between transfers must be used. The QSPI requires time, approximately 0.425 µs with a 40-MHz system clock, to prefetch the next transmit RAM entry for the next transfer. Therefore, the user may select a baud rate that provides at least a 0.6-µs delay between successive transfers to ensure no loss of incoming data. If the system clock is operating at a slower rate, the delay between transfers must be increased proportionately.

Because the BITSE option in the command control segment is no longer available, BITS sets the number of bits to be transferred for all transfers in the queue until the CPU changes the BITS value. As mentioned above, until PCS0/$\overline{SS}$ is negated (brought high), the QSPI continues to shift one bit for each pulse of SCK. If PCS0/$\overline{SS}$ is negated before the proper number of bits (according to BITS) is received, the next time the QSPI is selected it resumes storing bits in the same receive-data segment address where it left off. If more than 16 bits are transferred before negating the PCS0/$\overline{SS}$, the QSPI stores the number of bits indicated by BITS in the current receive data

segment address, then increments the address and continues storing as described above. Note that PCS0/$\overline{SS}$ does not necessarily have to be negated between transfers.

Once the proper number of bits (designated by BITS) are transferred, the QSPI stores the received data in the receive data segment, stores the internal working queue pointer value in CPTQP, increments the internal working queue pointer, and loads the new transmit data from the transmit data segment into the data serializer. The internal working queue pointer address is used the next time PCS0/$\overline{SS}$ is asserted, unless the CPU writes to the NEWQP first.

The DT and DSCK command control bits are not used in slave mode. As a slave, the QSPI does not drive the clock line nor the chip-select lines and, therefore, does not generate a delay.

In slave mode, the QSPI shifts out the data in the transmit data segment. The transmit data is loaded into the data serializer (refer to **Figure 6-1**) for transmission. When the PCS0/$\overline{SS}$ pin is pulled low the MISO pin becomes active and the serializer then shifts the 16 bits of data out in sequence, most significant bit first, as clocked by the incoming SCK signal. The QSPI uses CPHA and CPOL to determine which incoming SCK edge the MOSI pin uses to latch incoming data, and which edge the MISO pin uses to drive the data out.

The QSPI transmits and receives data until reaching the end of the queue (defined as a match with the address in ENDQP), regardless of whether PCS0/SS remains selected or is toggled between serial transfers. Receiving the proper number of bits causes the received data to be stored. The QSPI always transmits as many bits as it receives at each queue address, until the BITS value is reached or PCS0/SS is negated.

6.7.7 Slave Wraparound Mode

When the QSPI reaches the end of the queue, it always sets the SPIF flag, whether wraparound mode is enabled or disabled. An optional interrupt to the CPU is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wraparound mode is enabled. A description of SPIFIE bit can be found in 4.3.3 QSPI Control Register 2 (SPCR2).

In wraparound mode, the QSPI cycles through the queue continuously. Each time the end of the queue is reached, the SPIF flag is set. If the CPU fails to clear SPIF, it remains set, and the QSPI continues to send interrupt requests to the CPU (assuming SPIFIE is set). The user may avoid causing CPU interrupts by clearing SPIFIE.

As SPIFIE is buffered, clearing it after the SPIF flag is asserted does not immediately stop the CPU interrupts, but only prevents future interrupts from this source. To clear the current interrupt, the CPU must read QSPI register SPSR with SPIF asserted, followed by a write to SPSR with zero in SPIF (clear SPIF). Execution continues in wraparound mode even while the QSPI is requesting interrupt service from the CPU. The internal working queue pointer is incremented to the next address and the com-

mands are executed again. SPE is not cleared by the QSPI. New receive data overwrites previously received data located in the receive data segment.

Wraparound mode is properly exited in two ways: a) The CPU may disable wraparound mode by clearing WREN. The next time end of the queue is reached, the QSPI sets SPIF, clears SPE, and stops; and, b) The CPU sets HALT. This second method halts the QSPI after the current transfer is completed, allowing the CPU to negate SPE. The CPU can immediately stop the QSPI by clearing SPE; however, this method is not recommended, as it causes the QSPI to abort a serial transfer in process.

6.7.8 Mode Fault

MODF is asserted by the QSPI when the QSPI is the serial master (MSTR = 1) and the slave select (PCS0/$\overline{SS}$) input pin is pulled low by an external driver. This is possible only if the PCS0/$\overline{SS}$ pin is configured as input by QDDR. This low input to $\overline{SS}$ is not a normal operating condition. It indicates that a multimaster system conflict may exist, that another MCU is requesting to become the SPI network master, or simply that the hardware is incorrectly affecting PCS0/$\overline{SS}$. SPE in SPCR1 is cleared, disabling the QSPI. The QSPI pins revert to control by QPDR. If MODF is set and HMIE in SPCR3 is asserted, the QSPI generates an interrupt to the CPU.

The CPU may clear MODF by reading SPSR with MODF asserted, followed by writing SPSR with a zero in MODF. After correcting the mode fault problem, the QSPI can be re-enabled by asserting SPE.

The PCS0/$\overline{SS}$ pin may be configured as a general-purpose output instead of input to the QSPI. This inhibits the mode fault checking function. In this case, MODF is not used by the QSPI.

6.8 Serial Communication Interface

The dual, independent, serial communication interface (DSCI) communicates with external devices through an asynchronous serial bus. The two SCI modules are functionally equivalent, except that the SCI1 also provides 16-deep queue capabilities for the transmit and receive operations. The SCIs are fully compatible with other Motorola SCI systems. The DSCI has all of the capabilities of previous SCI systems as well as several significant new features.

Figure 6-10 is a block diagram of the SCI transmitter. **Figure 6-11** is a block diagram of the SCI receiver.

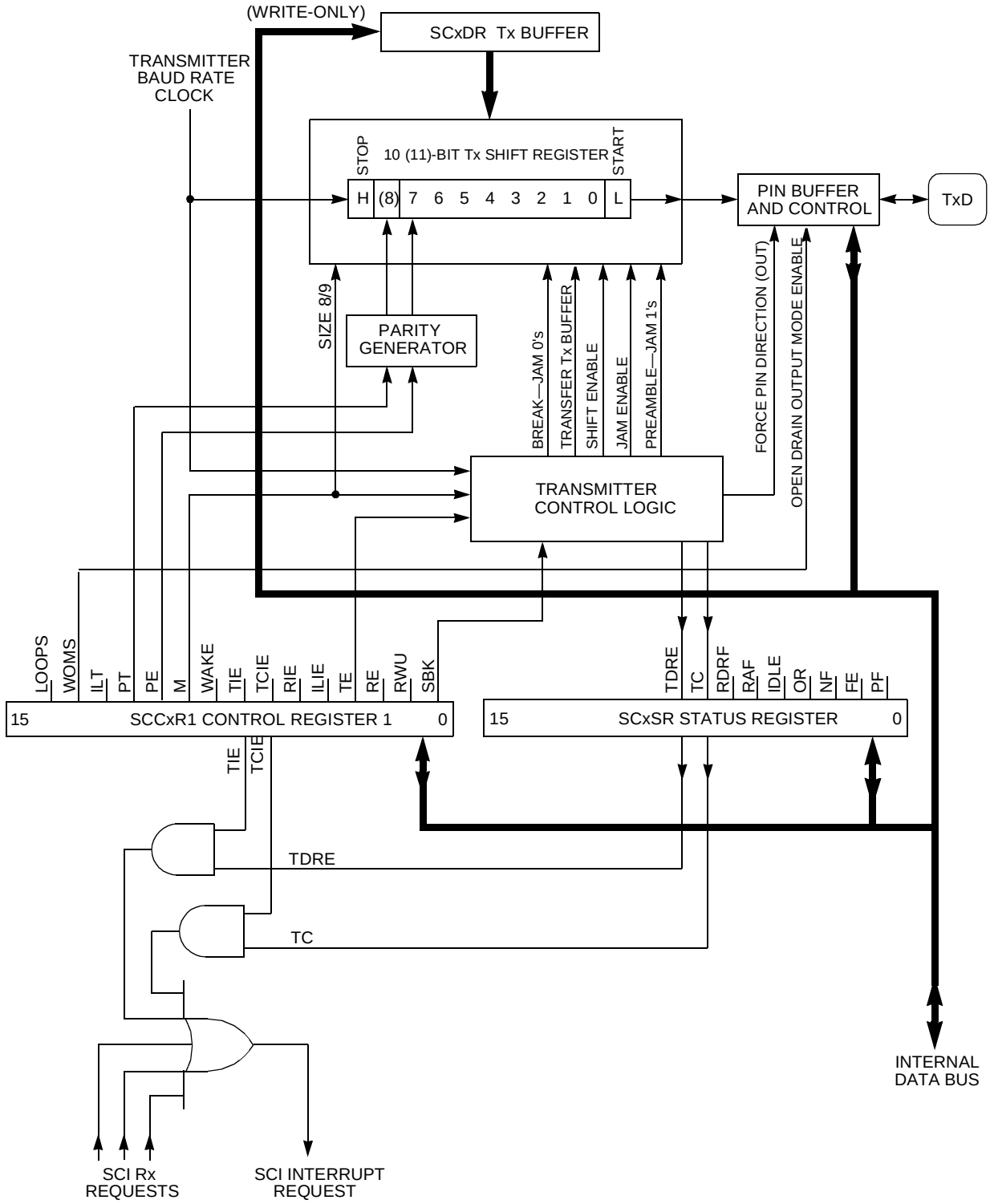


Figure 6-10 SCI Transmitter Block Diagram

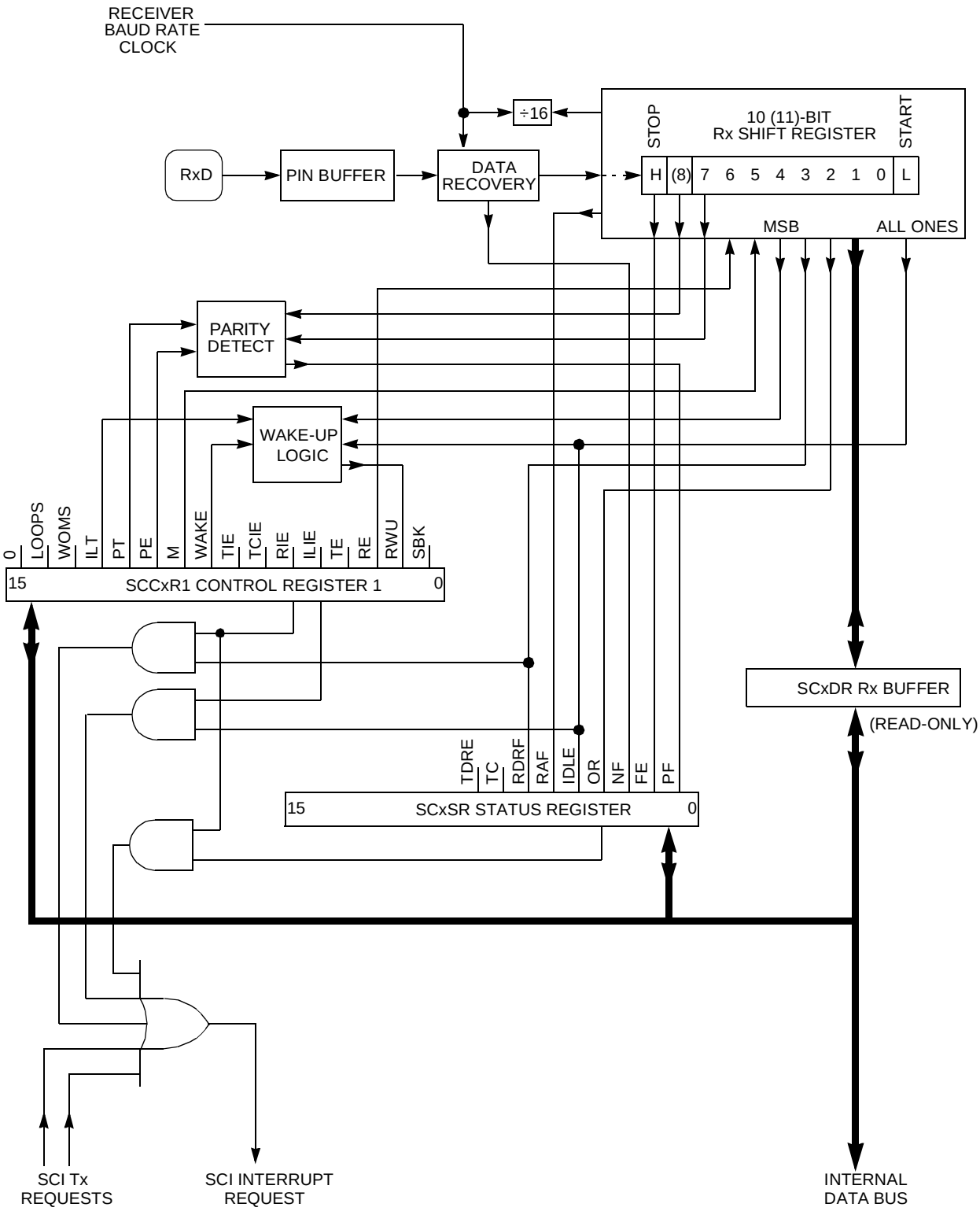


Figure 6-11 SCI Receiver Block Diagram

6.8.1 SCI Registers

The SCI programming model includes the QSMCM global and pin control registers and the DSCI registers.

The DSCI registers, listed in **Table 6-22**, consist of five control registers, three status registers, and 34 data registers. All registers may be read or written at any time by the CPU. Rewriting the same value to any DSCI register does not disrupt operation; however, writing a different value into a DSCI register when the DSCI is running may disrupt operation. To change register values, the receiver and transmitter should be disabled with the transmitter allowed to finish first. The status flags in register SCxSR can be cleared at any time.

Table 6-22 SCI Registers

Address	Name	Usage
0xYF FC08	SCC1R0	SCI1 Control Register 0 See **Table 6-23** for bit descriptions.
0xYF FC0A	SCC1R1	SCI1 Control Register 1 See **Table 6-24** for bit descriptions.
0xYF FC0C	SC1SR	SCI1 Status Register See **Table 6-25** for bit descriptions.
00xYF FC0E (non-queue mode only)	SC1DR	SCI1 Data Register Transmit Data Register (TDR1)* Receive Data Register (RDR1)* See **Table 6-26** for bit descriptions.
0xYF FC20	SCC2R0	SCI2 Control Register 0
0xYF FC22	SCC2R1	SCI2 Control Register 1
0xYF FC24	SC2SR	SCI2 Status Register
0xYF FC26	SC2DR	SCI2 Data Register Transmit Data Register (TDR2)* Receive Data Register (RDR2)*
0xYF FC28	QSCI1CR	QSCI1 Control Register Interrupts, wrap, queue size and enables for receive and transmit, QTPNT. See **Table 6-31** for bit descriptions.
0xYF FC2A	QSCI1SR	QSCI1 Status Register OverRun error flag, queue status flags, QRPNT, and QPEND. See **Table 6-32** for bit descriptions.
0xYF FC2C — 0xYF FC4A	QSCI1 Transmit Queue Memory Area	QSCI1 Transmit Queue Data locations (on half-word boundary)
0xYF FC4C — 0xYF FC6A	QSCI1 Receive Queue Memory Area	QSCI1 Receive Queue Data locations (on half-word boundary)

*Reads access the RDRx; writes access the TDRx.

During SCIx initialization, two bits in the SCCxR1 should be written last: the transmitter enable (TE) and receiver enable (RE) bits, which enable SCIx. Registers SCCxR0 and SCCxR1 should both be initialized at the same time or before TE and RE are asserted.

A single half-word write to SCCxR1 can be used to initialize SCIx and enable the transmitter and receiver.

6.8.2 SCI Control Register 0

SCCxR0 contains the SCIx baud rate selection field and two bits controlling the clock source. The baud rate must be set before the SCI is enabled. The CPU can read and write SCCxR0 at any time.

Changing the value of SCCxR0 bits during a transfer operation can disrupt the transfer. Before changing register values, allow the SCI to complete the current transfer, then disable the receiver and transmitter.

SCCxR0 — SCI Control Register 0 **0xYF FC08, 0xYF FC22**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
OTHR	LNK-BD	0	SCxBR												
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Table 6-23 SCCxR0 Bit Settings

Bit(s)	Name	Description
15	OTHR	This bit is reserved and should always be programmed to 0 at all times.
14	LNKBD	This bit is reserved and should always be programmed to 0 at all times.
13	—	Reserved
12:0	SCxBR	SCI baud rate. The SCI baud rate is programmed by writing a 13-bit value to this field. Writing a value of zero to SCxBR disables the baud rate generator. Baud clock rate is calculated as follows: $\text{SCI Baud Rate} = \frac{f_{SYS}}{32 \times \text{SCxBR}}$ where SCxBR is in the range of 1 to 8191. Refer to **6.8.7.3 Baud Clock** for more information.

6.8.3 SCI Control Register 1

SCCxR1 contains SCIx configuration parameters, including transmitter and receiver enable bits, interrupt enable bits, and operating mode enable bits. The CPU can read or write this register at any time. The SCI can modify the RWU bit under certain circumstances.

Changing the value of SCCxR1 bits during a transfer operation can disrupt the transfer. Before changing register values, allow the SCI to complete the current transfer, then disable the receiver and transmitter.

SCCxR1 — SCI Control Register 1 **0xYF FC0A, 0xYF FC22**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	LOOPS	WOMS	ILT	PT	PE	M	WAKE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-24 SCCxR1 Bit Settings

Bit(s)	Name	Description
15	—	Reserved
14	LOOPS	Loop mode 0 = Normal SCI operation, no looping, feedback path disabled. 1 = SCI test operation, looping, feedback path enabled.
13	WOMS	Wired-OR mode for SCI Pins 0 = If configured as an output, TXD is a normal CMOS output. 1 = If configured as an output, TXD is an open drain output.
12	ILT	Idle-line detect type. Refer to **6.8.7.7 Idle-Line Detection**. 0 = Short idle-line detect (start count on first one). 1 = Long idle-line detect (start count on first one after stop bit(s)).
11	PT	Parity type. Refer to **6.8.7.4 Parity Checking**. 0 = Even parity. 1 = Odd parity.
10	PE	Parity enable. Refer to **6.8.7.4 Parity Checking**. 0 = SCI parity disabled. 1 = SCI parity enabled.
9	M	Mode select. Refer to **6.8.7.2 Serial Formats**. 0 = 10-bit SCI frame. 1 = 11-bit SCI frame.
8	WAKE	Wakeup by address mark. Refer to **6.8.7.8 Receiver Wake-Up**. 0 = SCI receiver awakened by idle-line detection. 1 = SCI receiver awakened by address mark (last bit set).
7	TIE	Transmit interrupt enable 0 = SCI TDRE interrupts disabled. 1 = SCI TDRE interrupts enabled.
6	TCIE	Transmit complete interrupt enable 0 = SCI TC interrupts disabled. 1 = SCI TC interrupts enabled.
5	RIE	Receiver interrupt enable 0 = SCI RDRF and OR interrupts disabled. 1 = SCI RDRF and OR interrupts enabled.
4	ILIE	Idle-line interrupt enable 0 = SCI IDLE interrupts disabled. 1 = SCI IDLE interrupts enabled.
3	TE	Transmitter enable 0 = SCI transmitter disabled (TXD pin can be used as general-purpose output) 1 = SCI transmitter enabled (TXD pin dedicated to SCI transmitter).
2	RE	Receiver Enable 0 = SCI receiver disabled (RXD pin can be used as general-purpose input). 1 = SCI receiver enabled (RXD pin is dedicated to SCI receiver).

Table 6-24 SCCxR1 Bit Settings (Continued)

Bit(s)	Name	Description
1	RWU	Receiver wakeup. Refer to **6.8.7.8 Receiver Wake-Up**. 0 = Normal receiver operation (received data recognized). 1 = Wakeup mode enabled (received data ignored until receiver is awakened).
0	SBK	Send break 0 = Normal operation. 1 = Break frame(s) transmitted after completion of current frame.

6.8.4 SCI Status Register (SCxSR)

SCxSR contains flags that show SCI operating conditions. These flags are cleared either by SCIx hardware or by a read/write sequence. The sequence consists of reading the SCxSR (either the upper byte, lower byte, or the entire half-word) with a flag bit set, then reading (or writing, in the case of flags TDRE and TC) the SCxDR (either the lower byte or the half-word).

The contents of the two 16-bit registers SCxSR and SCxDR appear as upper and lower half-words, respectively, when the SCxSR is read into a 32-bit register. An upper byte access of SCxSR is meaningful only for reads. Note that a word read can simultaneously access both registers SCxSR and SCxDR. This action clears the receive status flag bits that were set at the time of the read, but does not clear the TDRE or TC flags. To clear TC, the SCxSR read must be followed by a write to register SCxDR (either the lower byte or the half-word). The TDRE flag in the status register is read-only.

If an internal SCI signal for setting a status bit comes after the CPU has read the asserted status bits but before the CPU has read or written the SCxDR, the newly set status bit is not cleared. Instead, SCxSR must be read again with the bit set and SCxDR must be read or written before the status bit is cleared.

NOTE

None of the status bits are cleared by reading a status bit while it is set and then writing zero to that same bit. Instead, the procedure outlined above must be followed. Note further that reading either byte of SCxSR causes all 16 bits to be accessed, and any status bits already set in either byte are armed to clear on a subsequent read or write of SCxDR.

SCxSR — SCIx Status Register **0xYF FC0C, 0xYF FC24**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED							TDRE	TC	RDRF	RAF	IDLE	OR	NF	FE	PF
RESET:															
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0

Table 6-25 SCxSR Bit Settings

Bit(s)	Name	Description
15:9	—	Reserved
8	TDRE	Transmit data register empty. TDRE is set when the byte in TDRx is transferred to the transmit serial shifter. If this bit is zero, the transfer is yet to occur and a write to TDRx will overwrite the previous value. New data is not transmitted if TDRx is written without first clearing TDRE. 0 = Transmit data register still contains data to be sent to the transmit serial shifter. 1 = A new character can now be written to the transmit data register. For transmit queue operation, this bit should be ignored by software.
7	TC	Transmit complete. TC is set when the transmitter finishes shifting out all data, queued preambles (mark/idle-line), or queued breaks (logic zero). 0 = SCI transmitter is busy. 1 = SCI transmitter is idle. For transmit queue operation, TC is cleared when SCxSR is read with TC set, followed by a write to SCTQ[0:15].
6	RDRF	Receive data register full. RDRF is set when the contents of the receive serial shifter are transferred to register RDRx. If one or more errors are detected in the received word, the appropriate flag(s) (NF, FE, or PF) are set within the same clock cycle. 0 = Receive data register is empty or contains previously read data. 1 = Receive data register contains new data. For receiver queue operation, this bit should be ignored by software.
5	RAF	Receiver active flag. RAF indicates whether the receiver is busy. This flag is set when the receiver detects a possible start bit and is cleared when the chosen type of idle line is detected. RAF can be used to reduce collisions in systems with multiple masters. 0 = SCI receiver is idle. 1 = SCI receiver is busy.
4	IDLE	Idle line detected. IDLE is set when the receiver detects an idle-line condition (reception of a minimum of 10 or 11 consecutive ones as specified by ILT in SCCxR1). This bit is not set by the idle-line condition when RWU in SCCxR1 is set. Once cleared, IDLE is not set again until after RDRF is set (after the line is active and becomes idle again). If a break is received, RDRF is set, allowing a subsequent idle line to be detected again. Under certain conditions, the IDLE flag may be set immediately following the negation of RE in SCCxR1. System designs should ensure this causes no detrimental effects. 0 = SCI receiver did not detect an idle-line condition. 1 = SCI receiver detected an idle-line condition. For receiver queue operation, IDLE is cleared when SCxSR is read with IDLE set, followed by a read of SCRQ[0:15].
3	OR	Overrun error. OR is set when a new byte is ready to be transferred from the receive serial shifter to register RDRx, and RDRx is already full (RDRF is still set). Data transfer is inhibited until OR is cleared. Previous data in RDRx remains valid, but additional data received during an overrun condition (including the byte that set OR) is lost. Note that whereas the other receiver status flags (NF, FE, and PF) reflect the status of data already transferred to RDRx, the OR flag reflects an operational condition that resulted in a loss of data to RDRx. 0 = RDRF is cleared before new data arrives. 1 = RDRF is not cleared before new data arrives.

Table 6-25 SCxSR Bit Settings (Continued)

Bit(s)	Name	Description
2	NF	Noise error flag. NF is set when the receiver detects noise on a valid start bit, on any of the data bits, or on the stop bit(s). It is not set by noise on the idle line or on invalid start bits. Each bit is sampled three times for noise. If the three samples are not at the same logic level, the majority value is used for the received data value, and NF is set. NF is not set until the entire frame is received and RDRF is set. Although no interrupt is explicitly associated with NF, an interrupt can be generated with RDRF, and the interrupt handler can check NF. 0 = No noise detected in the received data. 1 = Noise detected in the received data. For receiver queue operation NF is cleared when SCxSR is read with NF set, followed by a read of SCRQ[0:15].
1	FE	Framing error. FE is set when the receiver detects a zero where a stop bit (one) was expected. A framing error results when the frame boundaries in the received bit stream are not synchronized with the receiver bit counter. FE is not set until the entire frame is received and RDRF is set. Although no interrupt is explicitly associated with FE, an interrupt can be generated with RDRF, and the interrupt handler can check FE. 0 = No framing error detected in the received data. 1 = Framing error or break detected in the received data.
0	PF	Parity error. PF is set when the receiver detects a parity error. PF is not set until the entire frame is received and RDRF is set. Although no interrupt is explicitly associated with PF, an interrupt can be generated with RDRF, and the interrupt handler can check PF. 0 = No parity error detected in the received data. 1 = Parity error detected in the received data.

6.8.5 SCI Data Register (SCxDR)

The SCxDR consists of two data registers located at the same address. The receive data register (RDRx) is a read-only register that contains data received by the SCI serial interface. Data is shifted into the receive serial shifter and is transferred to RDRx. The transmit data register (TDRx) is a write-only register that contains data to be transmitted. Data is first written to TDRx, then transferred to the transmit serial shifter, where additional format bits are added before transmission.

SCxDR — SCI Data Register **0xYF FC0E, 0xYF FC26**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED							R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:															
0	0	0	0	0	0	0	U	U	U	U	U	U	U	U	U

Table 6-26 SCxSR Bit Settings

Bit(s)	Name	Description
15:9	—	Reserved
8:0	R[8:0]/ T[8:0]	R[7:0]/T[7:0] contain either the eight data bits received when SCxDR is read, or the eight data bits to be transmitted when SCxDR is written. R8/T8 are used when the SCI is configured for nine-bit operation (M = 1). When the SCI is configured for 8-bit operation, R8/T8 have no meaning or effect. Accesses to the lower byte of SCxDR triggers the mechanism for clearing the status bits or for initiating transmissions whether byte, half-word, or word accesses are used.

6.8.6 SCI Pins

The RXD1 and RXD2 pins are the receive data pins for the SCI1 and SCI2, respectively. TXD1 and TXD2 are the transmit data pins for the two SCI modules. An external clock pin, ECK, is common to both SCIs. The pins and their functions are listed in **Table 6-27**.

Table 6-27 SCI Pin Functions

Pin Names	Mnemonic	Mode	Function
Receive Data	RXD1, RXD2	Receiver disabled Receiver enabled	General purpose input Serial data input to SCI
Transmit Data	TXD1, TXD2	Transmitter disabled Transmitter enabled	General purpose output Serial data output from SCI
External Clock	ECK	Receiver disabled Receiver enabled Transmitter disabled Transmitter enabled	Not used Alternate input source to baud Not used Alternate input source to baud

6.8.7 SCI Operation

The SCI can operate in polled or interrupt-driven mode. Status flags in SCxSR reflect SCI conditions regardless of the operating mode chosen. The TIE, TCIE, RIE, and ILIE bits in SCCxR1 enable interrupts for the conditions indicated by the TDRE, TC, RDRF, and IDLE bits in SCxSR, respectively.

6.8.7.1 Definition of Terms

- Bit-time — The time required to transmit or receive one bit of data, which is equal to one cycle of the baud frequency.
- Start bit — One bit-time of logic zero that indicates the beginning of a data frame. A start bit must begin with a one-to-zero transition and be preceded by at least three receive time samples of logic one.
- Stop bit— One bit-time of logic one that indicates the end of a data frame.
- Frame — A complete unit of serial information. The SCI can use 10-bit or 11-bit frames.
- Data frame — A start bit, a specified number of data or information bits, and at least one stop bit.
- Idle frame — A frame that consists of consecutive ones. An idle frame has no start

bit.

- Break frame — A frame that consists of consecutive zeros. A break frame has no stop bits.

6.8.7.2 Serial Formats

All data frames must have a start bit and at least one stop bit. Receiving and transmitting devices must use the same data frame format. The SCI provides hardware support for both 10-bit and 11-bit frames. The M bit in SCCxR1 specifies the number of bits per frame.

The most common data frame format for NRZ (non-return to zero) serial interfaces is one start bit, eight data bits (LSB first), and one stop bit (ten bits total). The most common 11-bit data frame contains one start bit, eight data bits, a parity or control bit, and one stop bit. Ten-bit and 11-bit frames are shown in **Table 6-28**.

Table 6-28 Serial Frame Formats

10-bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	—	2
1	7	1	1
1	8	—	1
11-Bit Frames			
Start	**Data**	**Parity/Control**	**Stop**
1	7	1	2
1	8	1	1

6.8.7.3 Baud Clock

The SCI baud rate is programmed by writing a 13-bit value to the SCxBR field in SCI control register zero (SCCxR0). The baud rate is derived from the MCU system clock by a modulus counter. Writing a value of zero to SCxBR[12:0] disables the baud rate generator. The baud rate is calculated as follows:

$$\text{SCI Baud Rate} = \frac{f_{SYS}}{32 \times \text{SCxBR}}$$

or

$$\text{SCxBR} = \frac{f_{SYS}}{32 \times \text{SCI Baud Rate Desired}}$$

where SCxBR is in the range {1, 2, 3, ..., 8191}.

The SCI receiver operates asynchronously. An internal clock is necessary to synchronize with an incoming data stream. The SCI baud rate generator produces a receive time sampling clock with a frequency 16 times that of the SCI baud rate. The SCI deter-

mines the position of bit boundaries from transitions within the received waveform, and adjusts sampling points to the proper positions within the bit period.

Table 6-29 shows theoretical baud rates for a 40-MHz system clock (the MC68F375 has a maximum system clock rate of 33 MHz).

Table 6-29 Examples of SCIx Baud Rates[1]

Nominal Baud Rate	Actual Baud Rate	Percent Error	Value of SCxBR
1,250,000.00	1,250,000.00	0.00	1
57,600.00	56,818.18	-1.36	22
38,400.00	37,878.79	-1.36	33
32,768.00	32,894.74	0.39	38
28,800.00	29,069.77	0.94	43
19,200.00	19,230.77	0.16	65
14,400.00	14,367.81	-0.22	87
9,600.00	9,615.38	0.16	130
4,800.00	4,807.69	0.16	260
2,400.00	2,399.23	-0.03	521
1,200.00	1,199.62	-0.03	1042
600.00	600.09	0.02	2083
300.00	299.98	-0.01	4167

NOTES:
1. These rates are based on a 40-MHz system clock.

6.8.7.4 Parity Checking

The PT bit in SCCxR1 selects either even (PT = 0) or odd (PT = 1) parity. PT affects received and transmitted data. The PE bit in SCCxR1 determines whether parity checking is enabled (PE = 1) or disabled (PE = 0). When PE is set, the MSB of data in a frame (i.e., the bit preceding the stop bit) is used for the parity function. For transmitted data, a parity bit is generated. For received data, the parity bit is checked. When parity checking is enabled, the PF bit in the SCI status register (SCxSR) is set if a parity error is detected.

Enabling parity affects the number of data bits in a frame, which can in turn affect frame size. **Table 6-30** shows possible data and parity formats.

Table 6-30 Effect of Parity Checking on Data Size

M	PE	Result
0	0	8 data bits
0	1	7 data bits, 1 parity bit
1	0	9 data bits
1	1	8 data bits, 1 parity bit

6.8.7.5 Transmitter Operation

The transmitter consists of a serial shifter and a parallel data register (TDRx) located in the SCI data register (SCxDR). The serial shifter cannot be directly accessed by the

CPU. The transmitter is double-buffered, which means that data can be loaded into the TDRx while other data is shifted out. The TE bit in SCCxR1 enables (TE = 1) and disables (TE = 0) the transmitter.

The shifter output is connected to the TXD pin while the transmitter is operating (TE = 1, or TE = 0 and transmission in progress). Wired-OR operation should be specified when more than one transmitter is used on the same SCI bus. The WOMS bit in SCCxR1 determines whether TXD is an open drain (wired-OR) output or a normal CMOS output. An external pull-up resistor on TXD is necessary for wired-OR operation. WOMS controls TXD function, regardless of whether the pin is used by the SCI or as a general-purpose output pin.

Data to be transmitted is written to SCxDR, then transferred to the serial shifter. Before writing to TDRx, the user should check the transmit data register empty (TDRE) flag in SCxSR. When TDRE = 0, the TDRx contains data that has not been transferred to the shifter. Writing to SCxDR again overwrites the data. If TDRE = 1, then TDRx is empty, and new data may be written to TDRx, clearing TDRE.

As soon as the data in the transmit serial shifter has shifted out and if a new data frame is in TDRx (TDRE = 0), then the new data is transferred from TDRx to the transmit serial shifter and TDRE is set automatically. An interrupt may optionally be generated at this point.

The transmission complete (TC) flag in SCxSR shows transmitter shifter state. When TC = 0, the shifter is busy. TC is set when all shifting operations are completed. TC is not automatically cleared. The processor must clear it by first reading SCxSR while TC is set, then writing new data to SCxDR, or writing to SCTQ[0:15] for transmit queue operation.

The state of the serial shifter is checked when the TE bit is set. If TC = 1, an idle frame is transmitted as a preamble to the following data frame. If TC = 0, the current operation continues until the final bit in the frame is sent, then the preamble is transmitted. The TC bit is set at the end of preamble transmission.

The SBK bit in SCCxR1 is used to insert break frames in a transmission. A non-zero integer number of break frames are transmitted while SBK is set. Break transmission begins when SBK is set, and ends with the transmission in progress at the time either SBK or TE is cleared. If SBK is set while a transmission is in progress, that transmission finishes normally before the break begins. To ensure the minimum break time, toggle SBK quickly to one and back to zero. The TC bit is set at the end of break transmission. After break transmission, at least one bit-time of logic level one (mark idle) is transmitted to ensure that a subsequent start bit can be detected.

If TE remains set, after all pending idle, data and break frames are shifted out, TDRE and TC are set and TXD is held at logic level one (mark).

When TE is cleared, the transmitter is disabled after all pending idle, data, and break frames are transmitted. The TC flag is set, and control of the TXD pin reverts to PQSPAR and DDRQS. Buffered data is not transmitted after TE is cleared. To avoid losing data in the buffer, do not clear TE until TDRE is set.

Some serial communication systems require a mark on the TXD pin even when the transmitter is disabled. Configure the TXD pin as an output, then write a one to either QDTX1 or QDTX2 of the PORTQS register. See **6.6.1**. When the transmitter releases control of the TXD pin, it reverts to driving a logic one output.

To insert a delimiter between two messages, to place non-listening receivers in wake-up mode between transmissions, or to signal a re-transmission by forcing an idle-line, clear and then set TE before data in the serial shifter has shifted out. The transmitter finishes the transmission, then sends a preamble. After the preamble is transmitted, if TDRE is set, the transmitter marks idle. Otherwise, normal transmission of the next sequence begins.

Both TDRE and TC have associated interrupts. The interrupts are enabled by the transmit interrupt enable (TIE) and transmission complete interrupt enable (TCIE) bits in SCCxR1. Service routines can load the last data frame in a sequence into SCxDR, then terminate the transmission when a TDRE interrupt occurs.

Two SCI messages can be separated with minimum idle time by using a preamble of 10 bit-times (11 if a 9-bit data format is specified) of marks (logic ones). Follow these steps:

1. Write the last data frame of the first message to the TDRx
2. Wait for TDRE to go high, indicating that the last data frame is transferred to the transmit serial shifter
3. Clear TE and then set TE back to one. This queues the preamble to follow the stop bit of the current transmission immediately.
4. Write the first data frame of the second message to register TDRx

In this sequence, if the first data frame of the second message is not transferred to TDRx prior to the finish of the preamble transmission, then the transmit data line (TXDx pin) marks idle (logic one) until TDRx is written. In addition, if the last data frame of the first message finishes shifting out (including the stop bit) and TE is clear, TC goes high and transmission is considered complete. The TXDx pin reverts to being a general-purpose output pin.

6.8.7.6 Receiver Operation

The RE bit in SCCxR1 enables (RE = 1) and disables (RE = 0) the receiver. The receiver contains a receive serial shifter and a parallel receive data register (RDRx) located in the SCI data register (SCxDR). The serial shifter cannot be directly accessed by the CPU. The receiver is double-buffered, allowing data to be held in the RDRx while other data is shifted in.

Receiver bit processor logic drives a state machine that determines the logic level for each bit-time. This state machine controls when the bit processor logic is to sample the RXD pin and also controls when data is to be passed to the receive serial shifter. A receive time clock is used to control sampling and synchronization. Data is shifted into the receive serial shifter according to the most recent synchronization of the receive time clock with the incoming data stream. From this point on, data movement

is synchronized with the MCU system clock. Operation of the receiver state machine is detailed in the *Queued Serial Module Reference Manual* (QSMRM/AD).

The number of bits shifted in by the receiver depends on the serial format. However, all frames must end with at least one stop bit. When the stop bit is received, the frame is considered to be complete, and the received data in the serial shifter is transferred to the RDRx. The receiver data register flag (RDRF) is set when the data is transferred.

The stop bit is always a logic one. If a logic zero is sensed during this bit-time, the FE flag in SCxSR is set. A framing error is usually caused by mismatched baud rates between the receiver and transmitter or by a significant burst of noise. Note that a framing error is not always detected; the data in the expected stop bit-time may happen to be a logic one.

Noise errors, parity errors, and framing errors can be detected while a data stream is being received. Although error conditions are detected as bits are received, the noise flag (NF), the parity flag (PF), and the framing error (FE) flag in SCxSR are not set until data is transferred from the serial shifter to the RDRx.

RDRF must be cleared before the next transfer from the shifter can take place. If RDRF is set when the shifter is full, transfers are inhibited and the overrun error (OR) flag in SCxSR is set. OR indicates that the RDRx needs to be serviced faster. When OR is set, the data in the RDRx is preserved, but the data in the serial shifter is lost.

When a completed frame is received into the RDRx, either the RDRF or OR flag is always set. If RIE in SCCxR1 is set, an interrupt results whenever RDRF is set. The receiver status flags NF, FE, and PF are set simultaneously with RDRF, as appropriate. These receiver flags are never set with OR because the flags apply only to the data in the receive serial shifter. The receiver status flags do not have separate interrupt enables, since they are set simultaneously with RDRF and must be read by the user at the same time as RDRF.

When the CPU reads SCxSR and SCxDR in sequence, it acquires status and data, and also clears the status flags. Reading SCxSR acquires status and arms the clearing mechanism. Reading SCxDR acquires data and clears SCxSR.

6.8.7.7 Idle-Line Detection

During a typical serial transmission, frames are transmitted isochronically and no idle time occurs between frames. Even when all the data bits in a frame are logic ones, the start bit provides one logic zero bit-time during the frame. An idle line is a sequence of contiguous ones equal to the current frame size. Frame size is determined by the state of the M bit in SCCxR1.

The SCI receiver has both short and long idle-line detection capability. Idle-line detection is always enabled. The idle-line type (ILT) bit in SCCxR1 determines which type of detection is used. When an idle-line condition is detected, the IDLE flag in SCxSR is set.

For short idle-line detection, the receiver bit processor counts contiguous logic one bit-times whenever they occur. Short detection provides the earliest possible recognition of an idle-line condition, because the stop bit and contiguous logic ones before and after it are counted. For long idle-line detection, the receiver counts logic ones after the stop bit is received. Only a complete idle frame causes the IDLE flag to be set.

In some applications, software overhead can cause a bit-time of logic level one to occur between frames. This bit-time does not affect content, but if it occurs after a frame of ones when short detection is enabled, the receiver flags an idle line.

When the ILIE bit in SCCxR1 is set, an interrupt request is generated when the IDLE flag is set. The flag is cleared by reading SCxSR and SCxDR in sequence. For receiver queue operation, IDLE is cleared when SCxSR is read with IDLE set, followed by a read of SCRQ[0:15]. IDLE is not set again until after at least one frame has been received (RDRF = 1). This prevents an extended idle interval from causing more than one interrupt.

6.8.7.8 Receiver Wake-Up

The receiver wake-up function allows a transmitting device to direct a transmission to a single receiver or to a group of receivers by sending an address frame at the start of a message. Hardware activates each receiver in a system under certain conditions. Resident software must process address information and enable or disable receiver operation.

A receiver is placed in wake-up mode by setting the RWU bit in SCCxR1. While RWU is set, receiver status flags and interrupts are disabled. Although the software can clear RWU, it is normally cleared by hardware during wake-up.

The WAKE bit in SCCxR1 determines which type of wake-up is used. When WAKE = 0, idle-line wake-up is selected. When WAKE = 1, address-mark wake-up is selected. Both types require a software-based device addressing and recognition scheme.

Idle-line wake-up allows a receiver to sleep until an idle line is detected. When an idle line is detected, the receiver clears RWU and wakes up. The receiver waits for the first frame of the next transmission. The data frame is received normally, transferred to the RDRx, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. For idle-line wake-up to work, there must be a minimum of one frame of idle line between transmissions. There must be no idle time between frames within a transmission.

Address mark wake-up uses a special frame format to wake up the receiver. When the MSB of an address-mark frame is set, that frame contains address information. The first frame of each transmission must be an address frame. When the MSB of a frame is set, the receiver clears RWU and wakes up. The data frame is received normally, transferred to the RDRx, and the RDRF flag is set. If software does not recognize the address, it can set RWU and put the receiver back to sleep. Address mark wake-up allows idle time between frames and eliminates idle time between transmissions. However, there is a loss of efficiency because of an additional bit-time per frame.

6.8.7.9 Internal Loop Mode

The LOOPS bit in SCCxR1 controls a feedback path in the data serial shifter. When LOOPS is set, the SCI transmitter output is fed back into the receive serial shifter. TXD is asserted (idle line). Both transmitter and receiver must be enabled before entering loop mode.

6.9 SCI Queue Operation

6.9.1 Queue Operation of SCI1 for Transmit and Receive

The SCI1 serial module allows for queueing on transmit and receive data frames. In the standard mode, in which the queue is disabled, the SCI1 operates as previously defined (i.e. transmit and receive operations done via SC1DR). However, if the SCI1 queue feature is enabled (by setting the QTE and/or QRE bits within QSCI1CR) a set of 16 entry queues is allocated for the receive and/or transmit operation. Through software control the queue is capable of continuous receive and transfer operations within the SCI1 serial unit.

6.9.2 Queued SCI1 Status and Control Registers

The SCI1 queue uses the following registers:

- QSCI1 control register (QSCI1CR, address offset 0x28)
- QSCI1 status register (QSCI1SR, address offset 0x2A)

6.9.2.1 QSCI1 Control Register

QSCI1CR — QSCI1 Control Register **0xYF FC28**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
QTPNT				QTH-FI	QBH-FI	QTHEI	QB-HEI	0	QTE	QRE	QTWE	QTSZ			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 6-31 QSCI1CR Bit Settings

Bit(s)	Name	Description
15:12	QTPNT	Queue transmit pointer. QTPNT is a 4-bit counter used to indicate the next data frame within the transmit queue to be loaded into the SC1DR. This feature allows for ease of testability. This field is writable in test mode only; otherwise it is read-only.
11	QTHFI	Receiver queue top-half full interrupt. When set, QTHFI enables an SCI1 interrupt whenever the QTHF flag in QSCI1SR is set. The interrupt is blocked by negating QTHFI. This bit refers to the queue locations SCRQ[0:7]. 0 = QTHF interrupt inhibited 1 = Queue top-half full (QTHF) interrupt enabled
10	QBHFI	Receiver queue bottom-half full interrupt. When set, QBHFI enables an SCI1 interrupt whenever the QBHF flag in QSCI1SR is set. The interrupt is blocked by negating QBHFI. This bit refers to the queue locations SCRQ[8:15]. 0 = QBHF interrupt inhibited 1 = Queue bottom-half full (QBHF) interrupt enabled

Table 6-31 QSCI1CR Bit Settings (Continued)

Bit(s)	Name	Description
9	QTHEI	Transmitter queue top-half empty interrupt. When set, QTHEI enables an SCI1 interrupt whenever the QTHE flag in QSCI1SR is set. The interrupt is blocked by negating QTHEI. This bit refers to the queue locations SCTQ[0:7]. 0 = QTHE interrupt inhibited 1 = Queue top-half empty (QTHE) interrupt enabled
8	QBHEI	Transmitter queue bottom-half empty interrupt. When set, QBHEI enables an SCI1 interrupt whenever the QBHE flag in QSCI1SR is set. The interrupt is blocked by negating QBHEI. This bit refers to the queue locations SCTQ[8:15]. 0 = QBHE interrupt inhibited 1 = Queue bottom-half empty (QBHE) interrupt enabled
7	—	Reserved
6	QTE	Queue transmit enable. When set, the transmit queue is enabled and the TDRE bit should be ignored by software. The TC bit is redefined to indicate when the entire queue is finished transmitting. When clear, the SCI1 functions as described in the previous sections and the bits related to the queue (Section 5.5 and its subsections) should be ignored by software with the exception of QTE. 0 = Transmit queue is disabled 1 = Transmit queue is enabled
5	QRE	Queue receive enable. When set, the receive queue is enabled and the RDRF bit should be ignored by software. When clear, the SCI1 functions as described in the previous sections and the bits related to the queue (Section 5.5 and its subsections) should be ignored by software with the exception of QRE. 0 = Receive queue is disabled 1 = Receive queue is enabled
4	QTWE	Queue transmit wrap enable. When set, the transmit queue is allowed to restart transmitting from the top of the queue after reaching the bottom of the queue. After each wrap of the queue, QTWE is cleared by hardware. 0 = Transmit queue wrap feature is disabled 1 = Transmit queue wrap feature is enabled
3:0	QTSZ	Queue transfer size. The QTSZ bits allow programming the number of data frames to be transmitted. From 1 (QTSZ = 0b0000) to 16 (QTSZ = 0b1111) data frames can be specified. QTSZ is loaded into QPEND initially or when a wrap occurs.

6.9.2.2 QSCI1 Status Register

QSCI1SR — QSCI1 Status Register **0xYF FC2A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED			QOR	QTHF	QBHF	QTHE	QBHE	QRPNT				QPEND			
RESET:															
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

Table 6-32 QSCI1SR Bit Settings

Bit(s)	Name	Description
15:13	—	Reserved
12	QOR	Receiver queue overrun error. The QOR is set when a new data frame is ready to be transferred from the SC1DR to the queue and the queue is already full (QTHF or QBHF are still set). Data transfer is inhibited until QOR is cleared. Previous data transferred to the queue remains valid. Additional data received during a queue overrun condition is not lost provided the receive queue is re-enabled before OR (SC1SR) is set. The OR flag is set when a new data frame is received in the shifter but the data register (SC1DR) is still full. The data in the shifter that generated the OR assertion is overwritten by the next received data frame, but the data in the SC1DR is not lost. 0 = The queue is empty before valid data is in the SC1DR 1 = The queue is not empty when valid data is in the SC1DR
11	QTHF	Receiver queue top-half full. QTHF is set when the receive queue locations SCRQ[0:7] are completely filled with new data received via the serial shifter. QTHF is cleared when register QSCI1SR is read with QTHF set, followed by a write of QTHF to zero. 0 = The queue locations SCRQ[0:7] are partially filled with newly received data or is empty 1 = The queue locations SCRQ[0:7] are completely full of newly received data
10	QBHF	Receiver queue bottom-half full. QBHF is set when the receive queue locations SCRQ[8:15] are completely filled with new data received via the serial shifter. QBHF is cleared when register QSCI1SR is read with QBHF set, followed by a write of QBHF to zero. 0 = The queue locations SCRQ[8:15] are partially filled with newly received data or is empty 1 = The queue locations SCRQ[8:15] are completely full of newly received data
9	QTHE	Transmitter queue top-half empty. QTHE is set when all the data frames in the transmit queue locations SCTQ[0:7] have been transferred to the transmit serial shifter. QTHE is cleared when register QSCI1SR is read with QTHE set, followed by a write of QTHE to zero. 0 = The queue locations SCTQ[0:7] still contain data to be sent to the transmit serial shifter 1 = New data may now be written to the queue locations SCTQ[0:7]
8	QBHE	Transmitter queue bottom-half empty. QBHE is set when all the data frames in the transmit queue locations SCTQ[8:15] has been transferred to the transmit serial shifter. QBHE is cleared when register QSCI1SR is read with QBHE set, followed by a write of QBHE to zero. 0 = The queue locations SCTQ[8:15] still contain data to be sent to the transmit serial shifter 1 = New data may now be written to the queue locations SCTQ[8:15]
7:4	QRPNT	Queue receive pointer. QRPNT is a 4-bit counter used to indicate the position where the next valid data frame will be stored within the receive queue. This field is writable in test mode only; otherwise it is read-only.
3:0	QPEND	Queue pending. QPEND is a 4-bit decrementer used to indicate the number of data frames in the queue that are awaiting transfer to the SC1DR. This field is writable in test mode only; otherwise it is read-only. From 1 (QPEND = 0b0000) to 16 (or done, QPEND = 1111) data frames can be specified.

6.9.3 QSCI1 Transmitter Block Diagram

The block diagram of the enhancements to the SCI transmitter is shown in **Figure 6-12**.

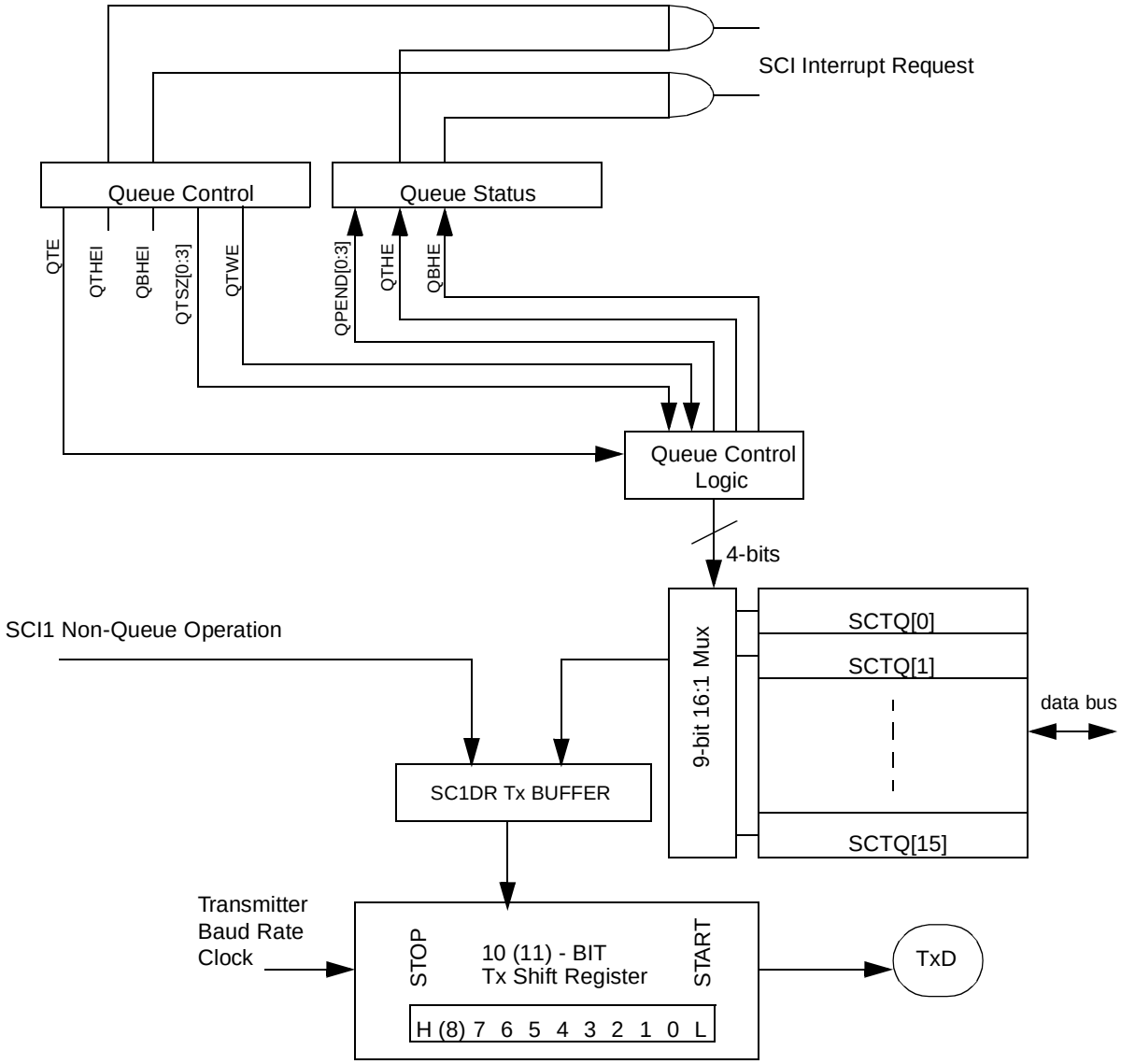


Figure 6-12 Queue Transmitter Block Enhancements

6.9.4 QSCI1 Additional Transmit Operation Features

- Available on a single SCI channel (SCI1) implemented by the queue transmit enable (QTE) bit set by software. When enabled, (QTE = 1) the TDRE bit should be ignored by software and the TC bit is redefined (as described later).
- When the queue is disabled (QTE = 0), the SCI functions in single buffer transfer mode where the queue size is set to one (QTSZ = 0000), and TDRE and TC function as previously defined. Locations SCTQ[0:15] can be used as general purpose 9-bit registers. All other bits pertaining to the queue should be ignored by software.
- Programmable queue up to 16 transmits (SCTQ[0:15]) which may allow for infinite and continuous transmits.
- Available transmit wrap function to prevent message breaks for transmits greater

than 16. This is achieved by the transmit wrap enable (QTWE) bit. When QTWE is set, the hardware is allowed to restart transmitting from the top of the queue (SCTQ[0]). After each wrap, QTWE is cleared by hardware.

— Transmissions of more than 16 data frames must be performed in multiples of 16 (QTSZ = 0b1111) except for the last set of transmissions. For any single non-continuous transmissions of 16 or less or the last transmit set composed of 16 or fewer data frames, the user is allowed to program QTSZ to the corresponding value of 16 or less where QTWE = 0.

- Interrupt generation when the top half (SCTQ[0:7]) of the queue has been emptied (QTHE) and the bottom half (SCTQ[8:15]) of the queue has been emptied (QBHE). This may allow for uninterrupted and continuous transmits by indicating to the CPU that it can begin refilling the queue portion that is now emptied.
  — The QTHE bit is set by hardware when the top half is empty or the transmission has completed. The QTHE bit is cleared when the QSCI1SR is read with QTHE set, followed by a write of QTHE to zero.
  — The QBHE bit is set by hardware when the bottom half is empty or the transmission has completed. The QBHE bit is cleared when the QSCI1SR is read with QBHE set, followed by a write of QBHE to zero.
  — In order to implement the transmit queue, QTE must be set (QSCI1CR), TE must be set (SCC1R1), QTHE must be cleared (QSCI1SR), and TDRE must be set (SC1SR).
- Enable and disable options for the interrupts QTHE and QBHE as controlled by QTHEI and QBHEI respectfully.
- Programmable 4-bit register queue transmit size (QTSZ) for configuring the queue to any size up to 16 transfers at a time. This value may be rewritten after transmission has started to allow for the wrap feature.
- 4-bit status register to indicate the number of data transfers pending (QPEND). This register counts down to all 0's where the next count rolls over to all 1's. This counter is writable in test mode; otherwise it is read-only.
- 4-bit counter (QTPNT) is used as a pointer to indicate the next data frame within the transmit queue to be loaded into the SC1DR. This counter is writable in test mode; otherwise it is read-only.
- A transmit complete (TC) bit re-defined when the queue is enabled (QTE = 1) to indicate when the entire queue (including when wrapped) is finished transmitting. This is indicated when QPEND = 1111 and the shifter has completed shifting data out. TC is cleared when the SCxSR is read with TC = 1 followed by a write to SCTQ[0:15]. If the queue is disabled (QTE = 0), the TC bit operates as originally designed.
- When the transmit queue is enabled (QTE = 1), writes to the transmit data register (SC1DR) have no effect.

6.9.5 QSCI1 Transmit Flow Chart Implementing the Queue

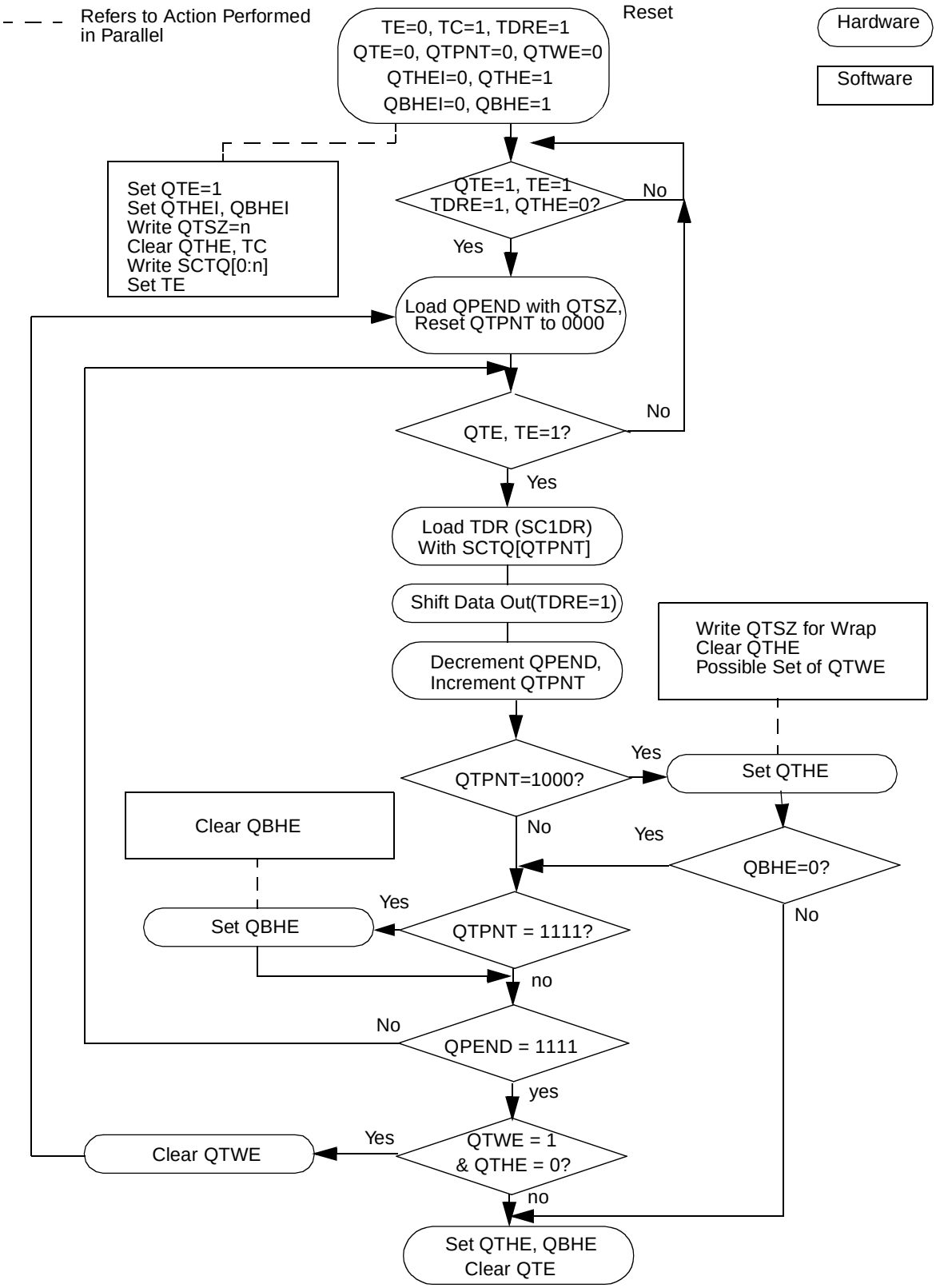


Figure 6-13 Queue Transmit Flow

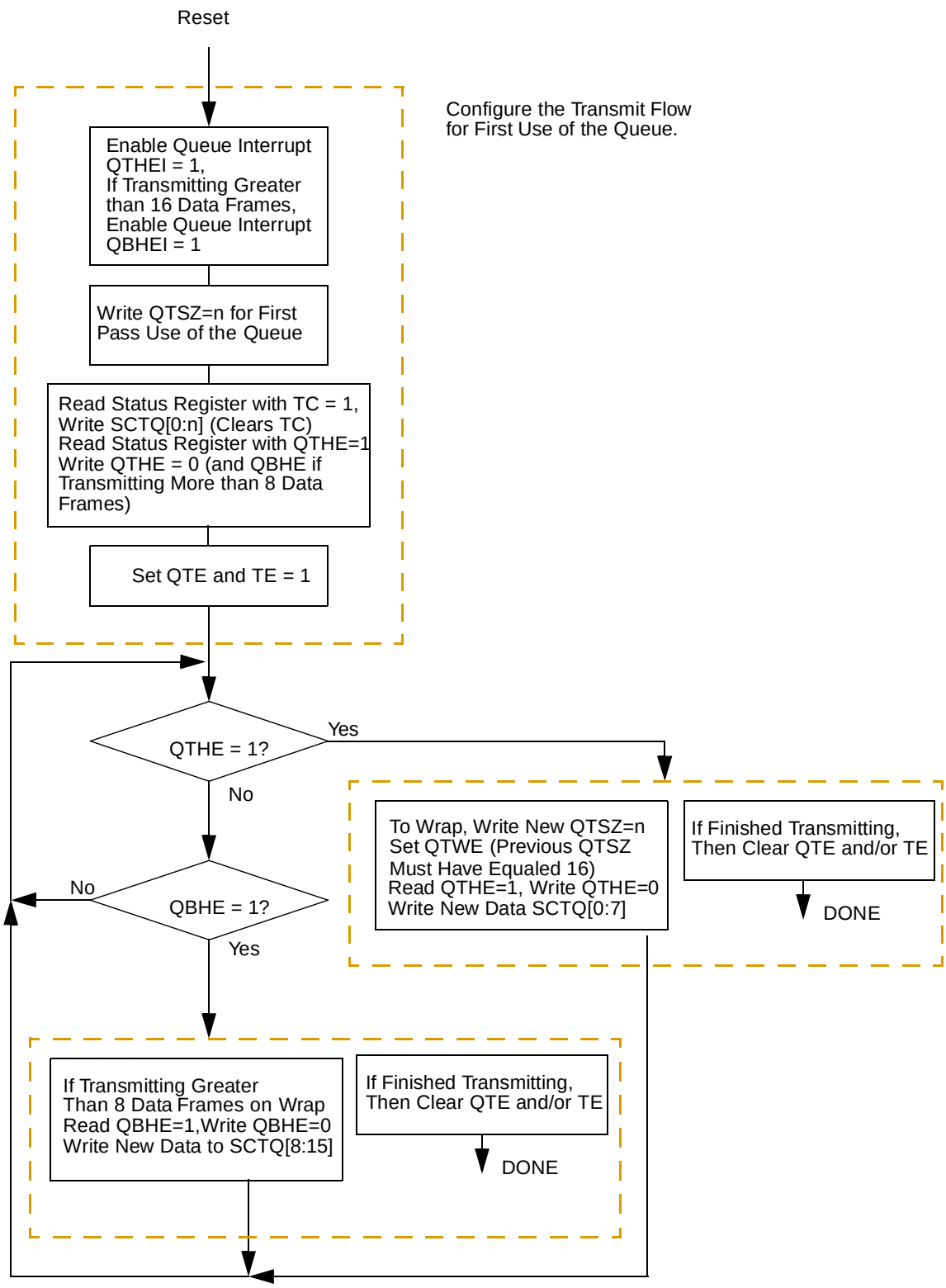


Figure 6-14 Queue Transmit Software Flow

6.9.6 Example QSCI1 Transmit for 17 Data Bytes

Figure 6-15 below shows a transmission of 17 data frames. The bold type indicates the current value for QTPNT and QPEND. The italic type indicates the action just performed by hardware. Regular type indicates the actions that should be performed by software before the next event.

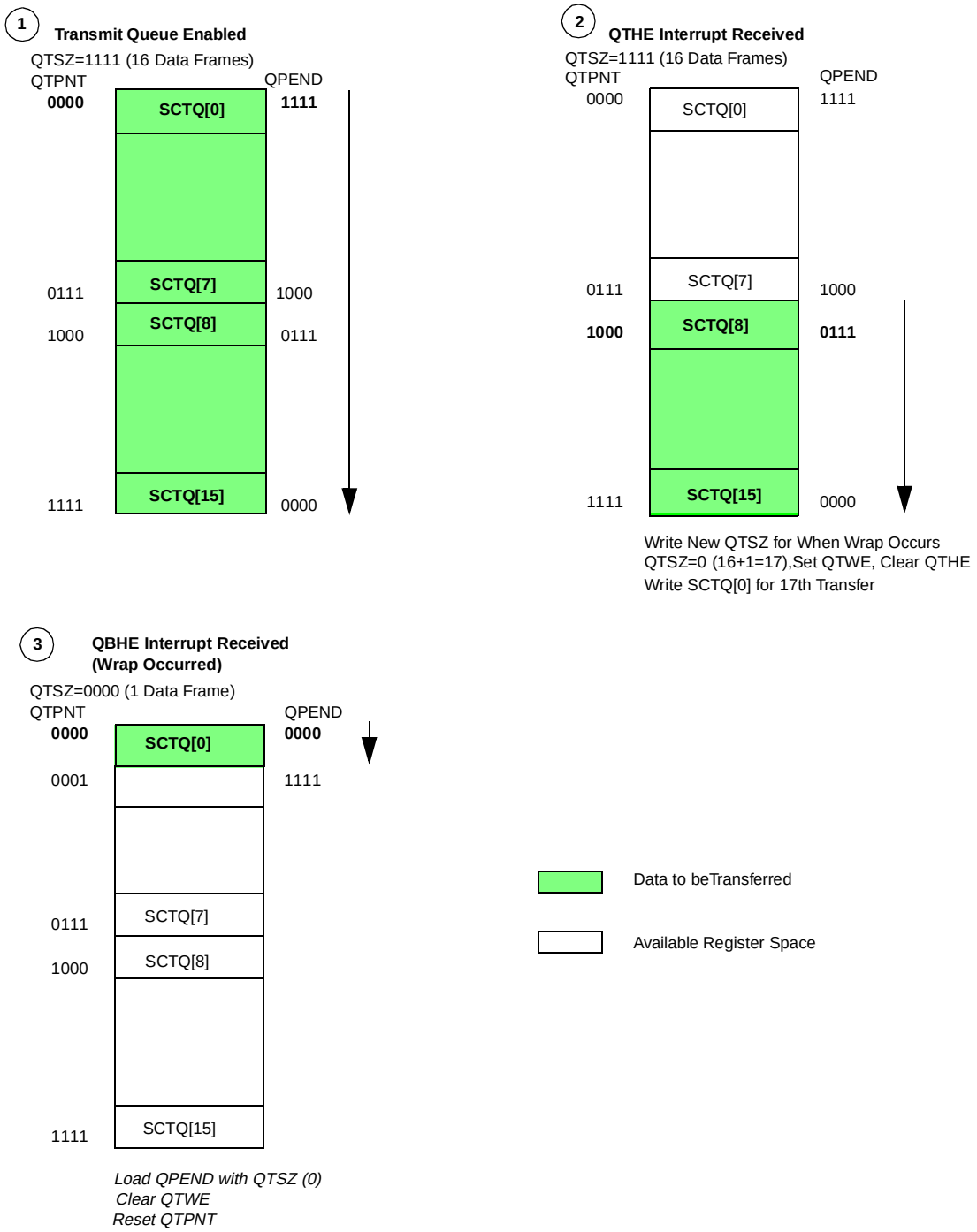


Figure 6-15 Queue Transmit Example for 17 Data Bytes

6.9.7 Example SCI Transmit for 25 Data Bytes

Figure 6-16 below is an example of a transmission of 25 data frames.

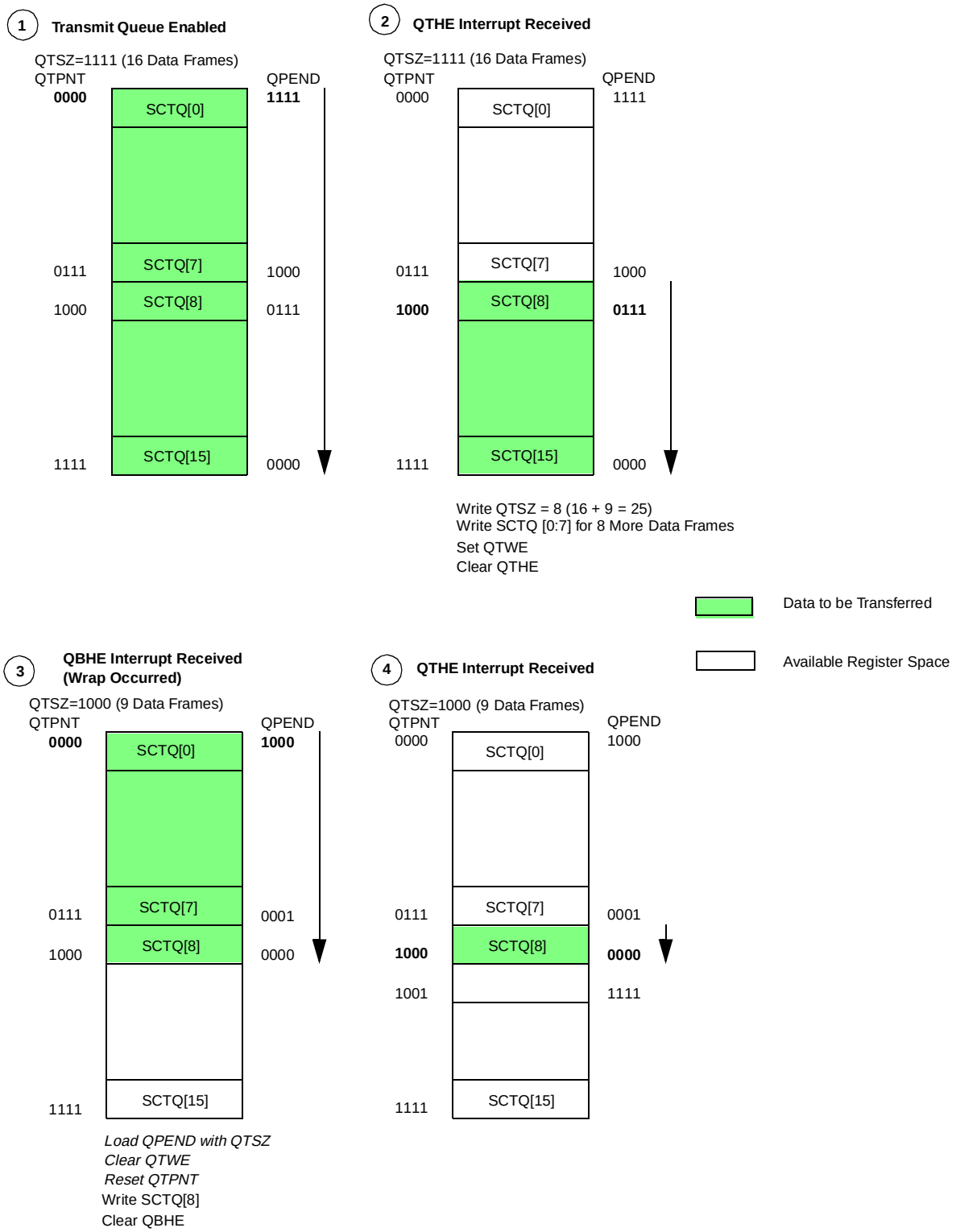


Figure 6-16 Queue Transmit Example for 25 Data Frames

6.9.8 QSCI1 Receiver Block Diagram

The block diagram of the enhancements to the SCI receiver is shown below in **Figure 6-17**.

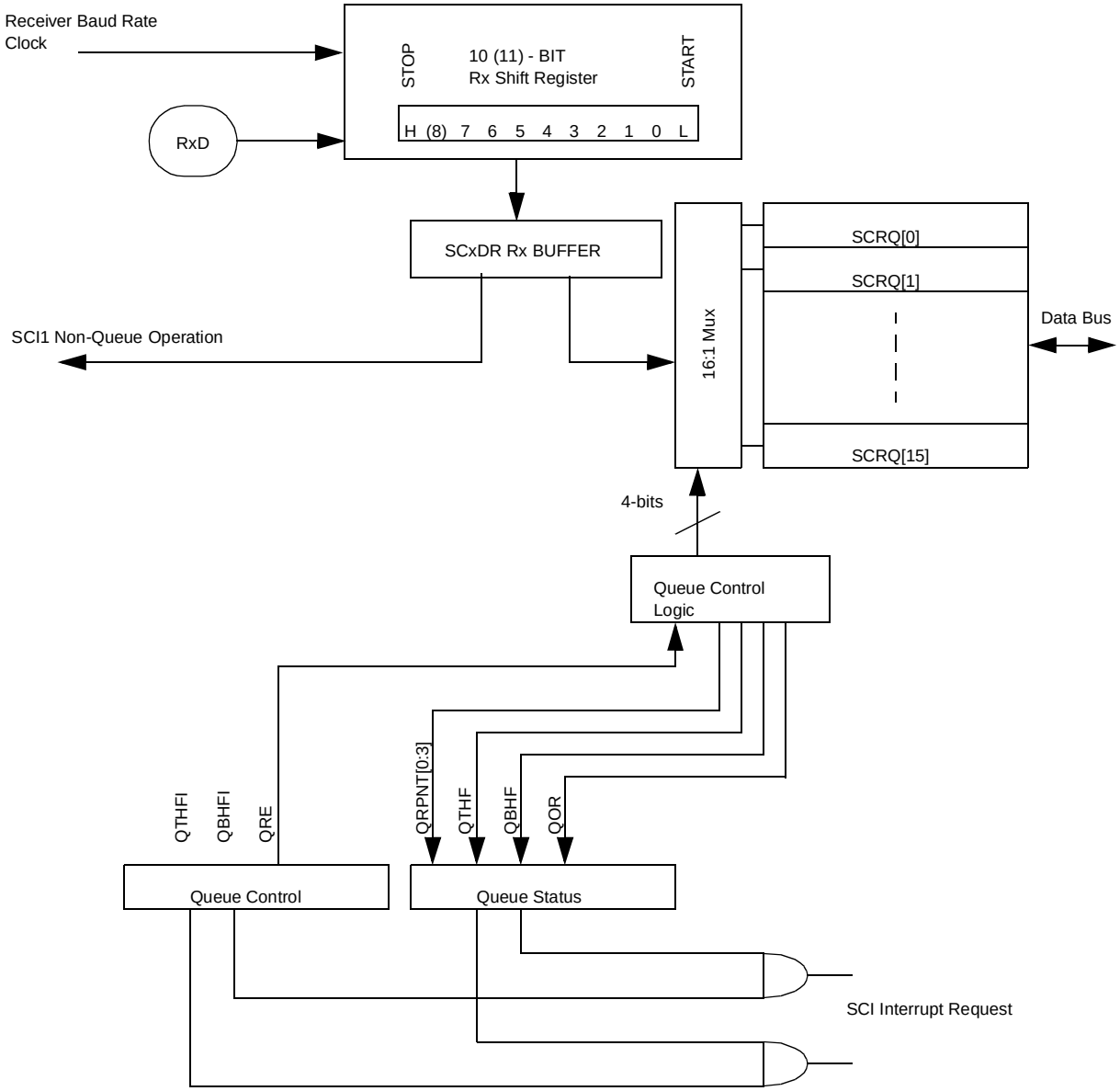


Figure 6-17 Queue Receiver Block Enhancements

6.9.9 QSCI1 Additional Receive Operation Features

- Available on a single SCI channel (SCI1) implemented by the queue receiver enable (QRE) bit set by software. When the queue is enabled, software should ignore the RDRF bit.
- When the queue is disabled (QRE = 0), the SCI functions in single buffer receive mode (as originally designed) and RDRF and OR function as previously defined.

Locations SCRQ[0:15] can be used as general purpose 9-bit registers. Software should ignore all other bits pertaining to the queue.

- Only data that has no errors (FE and PF both false) is allowed into the queue. The status flags FE and PF, if set, reflect the status of data not allowed into the queue. The receive queue is disabled until the error flags are cleared via the original SCI mechanism and the queue is re-initialized. The pointer QRPNT indicates the queue location where the data frame would have been stored.
- Queue size capable to receive up to 16 data frames (SCRQ[0:15]) which may allow for infinite and continuous receives.
- Interrupt generation can occur when the top half (SCRQ[0:7]) of the queue has been filled (QTHF) and the bottom half (SCRQ[8:15]) of the queue has been filled (QBHF). This may allow for uninterrupted and continuous receives by indicating to the CPU to start reading the queue portion that is now full.
  - The QTHF bit is set by hardware when the top half is full or the receive has completed. The QTHF bit is cleared when the SCxSR is read with QTHF set, followed by a write of QTHF to zero.
  - The QBHF bit is set by hardware when the bottom half is full or the receive has completed. The QBHF bit is cleared when the SCxSR is read with QBHF set, followed by a write of QBHF to zero.
- In order to implement the receive queue, the following conditions must be met: QRE must be set (QSCI1CR); RE must be set (SCC1R1); QOR and QTHF must be cleared (QSCI1SR); and OR, PF, and FE must be cleared (SC1SR).
- Enable and disable options for the interrupts QTHF and QBHF as controlled by the QTHFI and QBHFI, respectfully.
- 4-bit counter (QRPNT) is used as a pointer to indicate where the next valid data frame will be stored.
- A queue overrun error flag (QOR) to indicate when the queue is already full when another data frame is ready to be stored into the queue (similar to the OR bit in single buffer mode). The QOR bit can be set for QTHF = 1 or QBHF = 1, depending on where the store is being attempted.
- The queue can be exited when an idle line is used to indicate when a group of serial transmissions is finished. This can be achieved by using the ILIE bit to enable the interrupt when the IDLE flag is set. The CPU can then clear QRE and/or RE allowing the receiver queue to be exited.
- For receiver queue operation, IDLE is cleared when SC1SR is read with IDLE set, followed by a read of SCRQ[0:15].
- For receiver queue operation, NF is cleared when the SC1SR is read with NF set, followed by a read of SCRQ[0:15]. When noise occurs, the data is loaded into the receive queue, and operation continues unaffected. However, it may not be possible to determine which data frame in the receive queue caused the noise flag to be asserted.
- The queue is successfully filled (16 data frames) if error flags (FE and PF) are clear, QTHF and QBHF are set, and QRPNT is reset to all zeroes.
- QOR indicates that a new data frame has been received in the data register (SC1DR), but it cannot be placed into the receive queue due to either the QTHF or QBHF flag being set (QSCI1SR). Under this condition, the receive queue is disabled (QRE = 0). Software may service the receive queue and clear the appropri-

ate flag (QTHF, QBHF). Data is not lost provided that the receive queue is re-enabled before OR (SC1SR) is set, which occurs when a new data frame is received in the shifter but the data register (SC1DR) is still full. The data in the shifter that generated the OR assertion is overwritten by the next received data frame, but the data in the SC1DR is not lost.

6.9.10 QSCI1 Receive Flow Chart Implementing The Queue

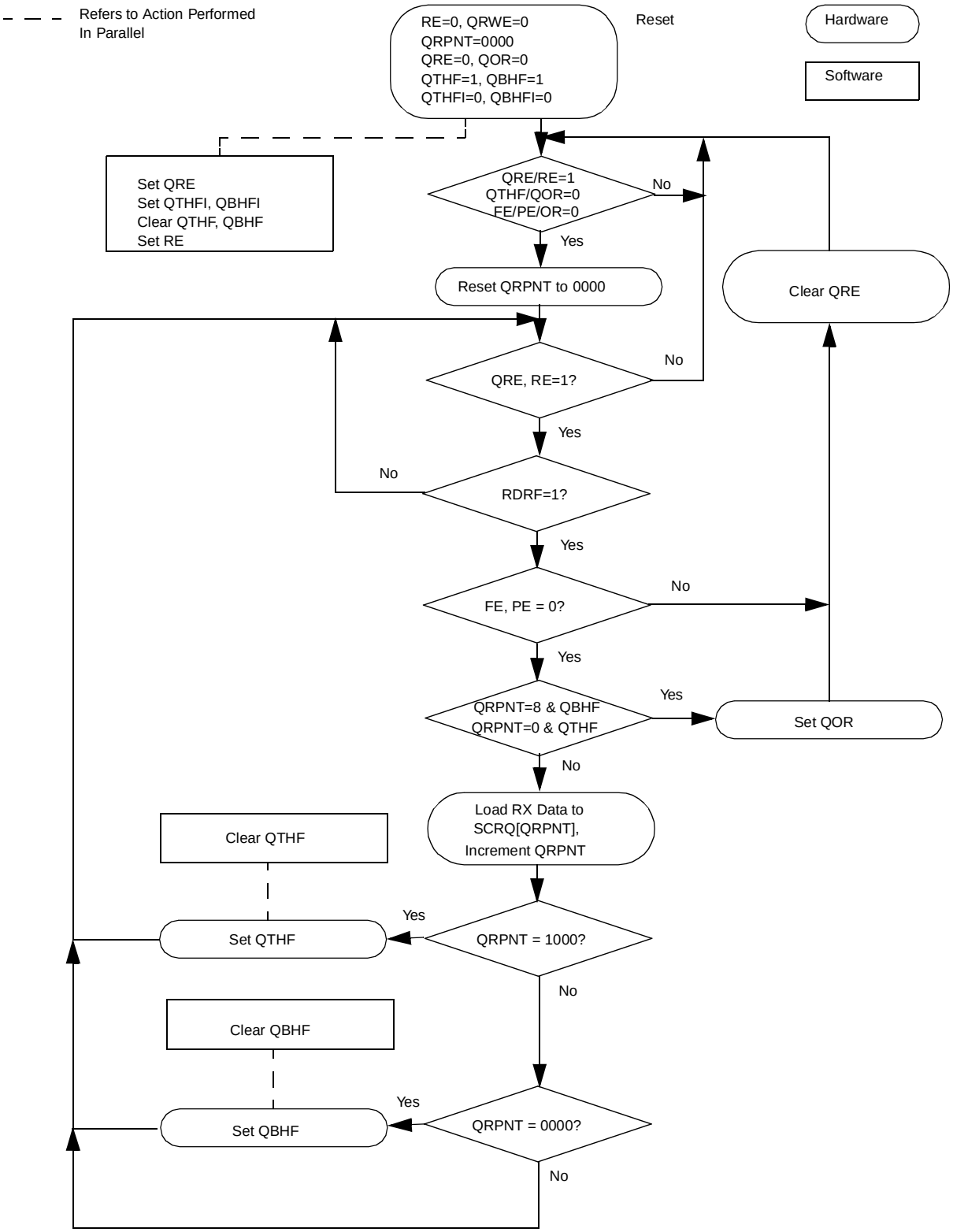


Figure 6-18 Queue Receive Flow

6.9.11 QSCI1 Receive Queue Software Flow Chart

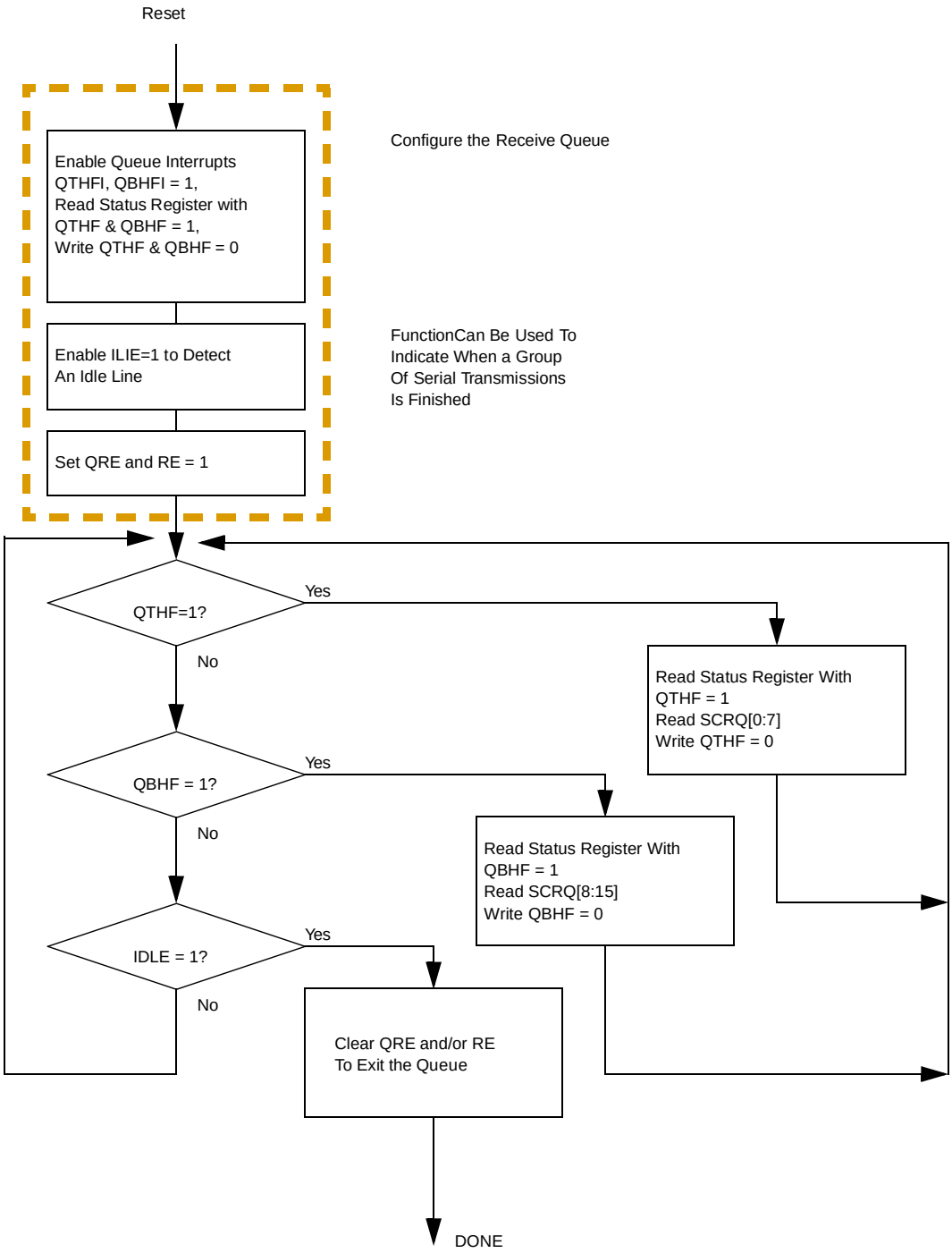


Figure 6-19 Queue Receive Software Flow

6.9.12 Example QSCI1 Receive Operation of 17 Data Frames

Figure 6-20 shows an example receive operation of 17 data frames. The bold type indicates the current value for the QRPNT. Action of the queue may be followed by starting at the top of the figure and going left to right and then down the page.

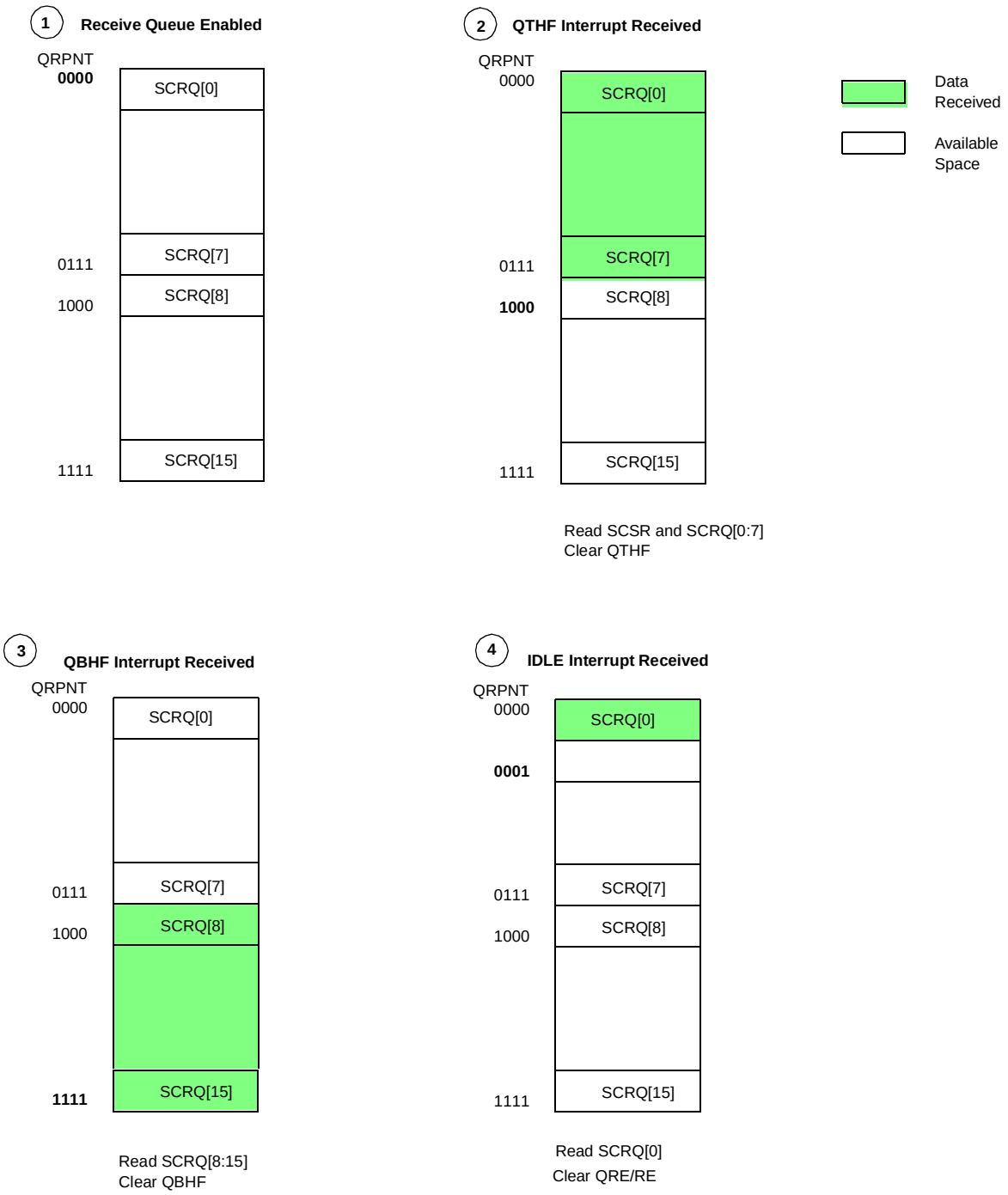


Figure 6-20 Queue Receive Example for 17 Data Bytes

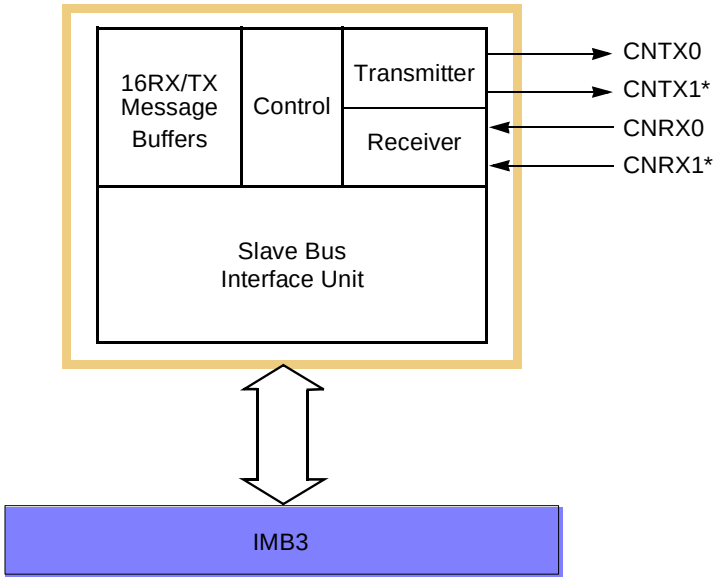
SECTION 7
CAN 2.0B CONTROLLER MODULE

7.1 Overview

The MC68F375 contains two CAN 2.0B controller modules (TouCAN). Each TouCAN is a communication controller that implements the controller area network (CAN) protocol, an asynchronous communications protocol used in automotive and industrial control systems. It is a high speed (1 Mbit/sec), short distance, priority based protocol that can run over a variety of mediums (for example, fiber optic cable or an unshielded twisted pair of wires). The TouCAN supports both the standard and extended identifier (ID) message formats specified in the CAN protocol specification, revision 2.0, part B.

Each TouCAN module contains 16 message buffers, which are used for transmit and receive functions. It also contains message filters, which are used to qualify the received message IDs when comparing them to the receive buffer identifiers.

Figure 7-1 shows a block diagram of a TouCAN module.



*Note: CNTX1 and CNRX1 are not available on MC68F375.

Figure 7-1 TouCAN Block Diagram

7.2 Features

Each TouCAN module provides these features:

- Full implementation of CAN protocol specification, version 2.0 A/B
  - — Standard data and remote frames (up to 109 bits long)
  - — Extended data and remote frames (up to 127 bits long)
  - — 0 to 8 bytes data length
  - — Programmable bit rate up to 1 Mbit/sec
- 16 RX/TX message buffers of 0-8 bytes data length
- Content-related addressing
- No read/write semaphores required
- Three programmable mask registers: global (for message buffers 0 through 13), special for message buffer 14, and special for message buffer 15
- Programmable transmit-first scheme: lowest ID or lowest buffer number
- “Time stamp”, based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Programmable I/O modes
- Maskable interrupts
- Independent of the transmission medium (external transceiver is assumed)
- Open network architecture
- Multimaster concept
- High immunity to EMI
- Short latency time for high-priority messages
- Low power sleep mode with programmable wakeup on bus activity
- Outputs have open drain drivers
- Can be used to implement recommended practices SAE J1939 and SAE J2284
- Can also be used to implement popular industrial automation standards such as DeviceNet™ and Smart Distributed System
- Motorola IMB-family modular architecture

7.3 External Pins

The TouCAN module interface to the CAN bus consists of four pins: CANTX0 and CANTX1, which transmit serial data, and CANRX0 and CANRX1, which receive serial data. **Figure 7-2** shows a typical CAN system.

NOTE

CNTX1 and CNRX1 are not available on the MC68F375.

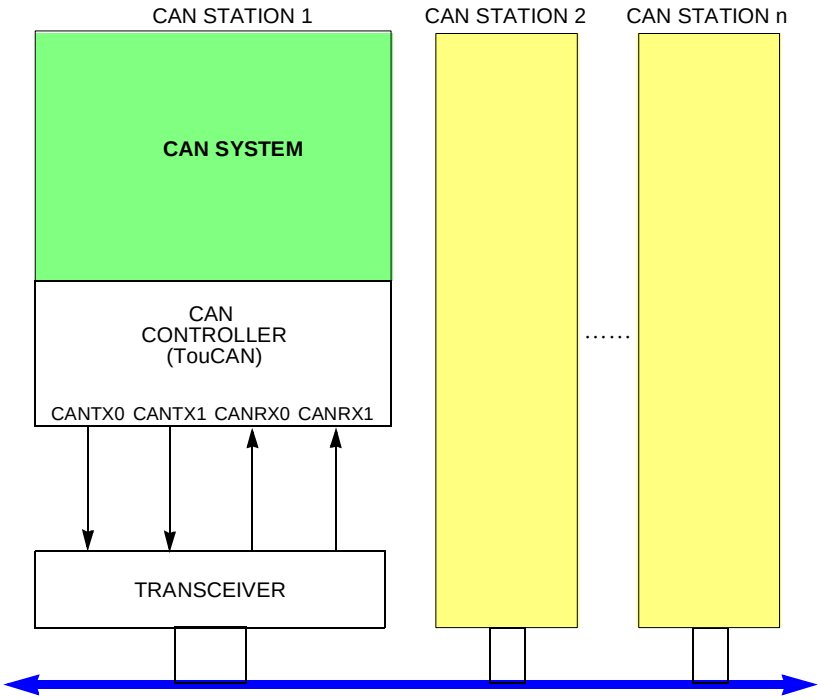


Figure 7-2 Typical CAN Network

Each CAN station is connected physically to the CAN bus through a transceiver. The transceiver provides the transmit drive, waveshaping, and receive/compare functions required for communicating on the CAN bus. It can also provide protection against damage to the TouCAN caused by a defective CAN bus or a defective CAN station.

7.4 TouCAN Architecture

The TouCAN module uses a flexible design that allows each of its 16 message buffers to be designated either a transmit (TX) buffer or a receive (RX) buffer. In addition, to reduce the CPU overhead required for message handling, each message buffer is assigned an interrupt flag bit to indicate that the transmission or reception completed successfully.

7.4.1 TX/RX Message Buffer Structure

Figure 7-3 displays the extended (29 bit) ID message buffer structure. **Figure 7-4** displays the standard (11 bit) ID message buffer structure.

	15 … 8	7 … 4	3 … 0					
$0	TIME STAMP	CODE	LENGTH					CONTROL/STATUS
$2	ID[28-18]		SRR	IDE	ID[17-15]			ID_HIGH
$4	ID[14-0]						RTR	ID_LOW
$6	DATA BYTE 0		DATA BYTE 1					
$8	DATA BYTE 2		DATA BYTE 3					
$A	DATA BYTE 4		DATA BYTE 5					
$C	DATA BYTE 6		DATA BYTE 7					
$E	RESERVED							

Figure 7-3 Extended ID Message Buffer Structure

	15 … 8	7 … 4	3 … 0					
$0	TIME STAMP	CODE	LENGTH					CONTROL/STATUS
$2	ID[28:18]		RTR	0	0	0	0	ID_HIGH
$4	16-BIT TIME STAMP							ID_LOW
$6	DATA BYTE 0		DATA BYTE 1					
$8	DATA BYTE 2		DATA BYTE 3					
$A	DATA BYTE 4		DATA BYTE 5					
$C	DATA BYTE 6		DATA BYTE 7					
$E	RESERVED							

Figure 7-4 Standard ID Message Buffer Structure

7.4.1.1 Common Fields for Extended and Standard Format Frames

Table 7-1 describes the message buffer fields that are common to both extended and standard identifier format frames.

Table 7-1 Common Extended/Standard Format Frames

Field	Description
Time Stamp	Contains a copy of the high byte of the free running timer, which is captured at the beginning of the identifier field of the frame on the CAN bus.
Code	Refer to **Table 7-2** and **Table 7-3**.
RX Length	Length (in bytes) of the RX data stored in offset 0x6 through 0xD of the buffer.This field is written by the TouCAN module, copied from the DLC (data length code) field of the received frame.
TX Length	Length (in bytes) of the data to be transmitted, located in offset $6 through 0xD of the buffer. This field is written by the CPU and is used as the DLC field value. If RTR (remote transmission request) = 1, the frame is a remote frame and will be transmitted without data field, regardless of the value in TX length.
Data	This field can store up to eight data bytes for a frame. For RX frames, the data is stored as it is received from the bus. For TX frames, the CPU provides the data to be transmitted within the frame.
Reserved	The CPU controls access to this word entry field (16 bits).

Table 7-2 Message Buffer Codes for Receive Buffers

RX Code Before RX New Frame	Description	RX Code After RX New Frame	Comment
0000	NOT ACTIVE — message buffer is not active.	—	—
0100	EMPTY — message buffer is active and empty.	0010	—
0010	FULL — message buffer is full.	0110	If a CPU read occurs before the new frame, new receive code is 0010.
0110	OVERRUN — second frame was received into a full buffer before the CPU read the first one.		
0XY1[1]	BUSY — message buffer is now being filled with a new receive frame. This condition will be cleared within 20 cycles.	0010	An empty buffer was filled (XY was 10).
		0110	A full/overrun buffer was filled (Y was 1).

NOTES:
1. For TX message buffers, upon read, the BUSY bit should be ignored.

Table 7-3 Message Buffer Codes for Transmit Buffers

RTR	Initial TX Code	Description	Code After Successful Transmission
X	1000	Message buffer not ready for transmit.	—
0	1100	Data frame to be transmitted once, unconditionally.	1000
1	1100	Remote frame to be transmitted once, and message buffer becomes an RX message buffer for data frames.	0100
0	1010[1]	Data frame to be transmitted only as a response to a remote frame, always.	1010
0	1110	Data frame to be transmitted only once, unconditionally, and then only as a response to remote frame, always.	1010

NOTES:
1. When a matching remote request frame is detected, the code for such a message buffer is changed to be 1110.

7.4.1.2 Fields for Extended Format Frames

Table 7-4 describes the message buffer fields used only for extended identifier format frames.

Table 7-4 Extended Format Frames

Field	Description
ID[28:18]/[17:15]	Contains the 14 most significant bits of the extended identifier, located in the ID HIGH word of the message buffer.
Substitute Remote Request (SRR)	Contains a fixed recessive bit, used only in extended format. Should be set to one by the user for TX buffers. It will be stored as received on the CAN bus for RX buffers.
ID Extended (IDE)	If extended format frame is used, this field should be set to one. If zero, standard format frame should be used.
ID[14:0]	Bits [14:0] of the extended identifier, located in the ID LOW word of the message buffer.
Remote Transmission Request (RTR)	This bit is located in the least significant bit of the ID LOW word of the message buffer; 0 = Data Frame, 1 = Remote Frame.

7.4.1.3 Fields for Standard Format Frames

Table 7-5 describes the message buffer fields used only for standard identifier format frames.

Table 7-5 Standard Format Frames

Field	Description
16-Bit Time Stamp	The ID LOW word, which is not needed for standard format, is used in a standard format buffer to store the 16-bit value of the free-running timer which is captured at the beginning of the identifier field of the frame on the CAN bus.
ID[28:18]	Contains bits [28:18] of the identifier, located in the ID HIGH word of the message buffer. The four least significant bits in this register (corresponding to the IDE bit and ID[17:15] for an extended identifier message) must all be written as logic zeros to ensure proper operation of the TouCAN.
RTR	This bit is located in the ID HIGH word of the message buffer; 0 = data frame, 1 = remote frame.
RTR/SRR Bit Treatment	If the TouCAN transmits this bit as a one and receives it as a zero, an “arbitration loss” is indicated. If the TouCAN transmits this bit as a zero and is receives it as a one, a bit error is indicated. If the TouCAN transmits a value and receives a matching response, a successful bit transmission is indicated.

7.4.1.4 Serial Message Buffers

To allow double buffering of messages, the TouCAN has two shadow buffers called serial message buffers. The TouCAN uses these two buffers for buffering both received messages and messages to be transmitted. Only one serial message buffer is active at a time, and its function depends upon the operation of the TouCAN at that time. At no time does the user have access to or visibility of these two buffers.

7.4.1.5 Message Buffer Activation/Deactivation Mechanism

Each message buffer must be activated once the user configures it for the desired operation. A buffer is activated by writing the appropriate code to the control/status word for that buffer. Once the buffer is activated, it will begin participating in the normal transmit and receive processes.

A buffer is deactivated by writing the appropriate deactivation code to the control/status word for that buffer. A buffer is typically deactivated when the user desires to reconfigure the buffer (for example to change the buffer's function from RX to TX or TX to RX). The buffer should also be deactivated before changing a receive buffer's message identifier or before loading a new message to be transmitted into a transmit buffer.

For more details on activation and deactivation of message buffers and the effects on message buffer operation, refer to **7.5 TouCAN Operation**.

7.4.1.6 Message Buffer Lock/Release/Busy Mechanism

In addition to the activation/deactivation mechanism, the TouCAN also uses a lock/release/busy mechanism to ensure data coherency during the receive process. The mechanism includes a lock status for each message buffer and uses the two serial message buffers to facilitate frame transfers within the TouCAN.

Reading the control/status word of a receive message buffer triggers the lock for that buffer. While locked, a received message cannot be transferred into that buffer from one of the serial message buffers.

If a message transfer between the message buffer and a serial message buffer is in progress when the control/status word is read, the BUSY status is indicated in the code field, and the lock is not activated.

The user can release the lock on a message buffer in one of two ways. Reading the control/status word of another message buffer locks that buffer, releasing the previously locked buffer. A global release can also be performed on any locked message buffer by reading the free-running timer.

Once a lock is released, any message transfers between a serial message buffer and a message buffer that were delayed due to that buffer being locked will take place. For more details on the message buffer locking mechanism, and the effects on message buffer operation, refer to **7.5 TouCAN Operation**.

7.4.2 Receive Mask Registers

The receive mask registers are used as acceptance masks for received frame IDs. The following masks are defined:

- A global mask, used for receive buffers 0-13
- Two separate masks for buffers 14 and 15

The value of the mask registers should not be changed during normal operation. If the mask register data is changed after the masked identifier of a received message is

matched to a locked message buffer, that message will be transferred into that message buffer once it is unlocked, regardless of whether that message's masked identifier still matches the receive buffer identifier. **Table 7-6** shows mask bit values.

Table 7-6 Receive Mask Register Bit Values

Mask Bit	Values
0	The corresponding incoming ID bit is "don't care".
1	The corresponding ID bit is checked against the incoming ID bit to see if a match exists.

Table 7-7 shows mask examples for normal and extended messages. Refer to **7.8 Programmer's Model** for more information on RX mask registers.

Table 7-7 Mask Examples for Normal/Extended Messages

Message Buffer (MB) /Mask	Base ID ID[28:18]	IDE	Extended ID ID[17:0]	Match
MB2	1 1 1 1 1 1 1 1 0 0 0	0	—	—
MB3	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
MB4	0 0 0 0 0 0 1 1 1 1 1	0	—	—
MB5	0 0 0 0 0 0 1 1 1 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
MB14	1 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—
RX Global Mask	1 1 1 1 1 1 1 1 1 1 0		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1	—
RX Message In	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	3[1]
	1 1 1 1 1 1 1 1 0 0 1	0	—	2[2]
	1 1 1 1 1 1 1 1 0 0 1	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0	—[3]
	0 1 1 1 1 1 1 1 0 0 0	0	—	—[4]
	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—[5]
RX 14 Mask	0 1 1 1 1 1 1 1 1 1 1		1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0	—
RX Message In	1 0 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	—[6]
	0 1 1 1 1 1 1 1 0 0 0	1	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	14[7]

NOTES:
1. Match for extended format (MB3).
2. Match for standard format (MB2).
3. No match for MB3 because of ID0.
4. No match for MB2 because of ID28.
5. No match for MB3 because of ID28, match for MB14.
6. No match for MB14 because of ID27.
7. Match for MB14.

7.4.3 Bit Timing

The TouCAN module uses three 8-bit registers to set up the bit timing parameters required by the CAN protocol. Control registers 1 and 2 (CANCTRL1, CANCTRL2) contain the PROPSEG, PSEG1, PSEG2, and the RJW fields which allow the user to

configure the bit timing parameters. The prescaler divide register (PRESDIV) allows the user to select the ratio used to derive the S-clock from the system clock. The time quanta clock operates at the S-clock frequency. **Table 7-8** provides examples of system clock, CAN bit rate, and S-clock bit timing parameters. Refer to **7.8 Programmer's Model** for more information on the bit timing registers.

Table 7-8 Example System Clock, CAN Bit Rate and S-Clock Frequencies

System Clock Frequency (MHz)	CAN Bit-Rate (MHz)	Possible S-Clock Frequency (MHz)	Possible Number of Time Quanta/Bit	PRESDIV Value + 1
25	1	25	25	1
20	1	10, 20	10, 20	2, 1
16	1	8, 16	8, 16	2, 1
25	0.125	1, 1.25, 2.5	8,10, 20	25, 20,10
20	0.125	1, 2, 2.5	8, 16, 20	20, 10, 8
16	0.125	1, 2	8,16	16, 8

7.4.3.1 Configuring the TouCAN Bit Timing

The following considerations must be observed when programming bit timing functions.

- If the programmed PRESDIV value results in a single system clock per one time quantum, then the PSEG2 field in CANCTRL2 register must not be programmed to zero.
- If the programmed PRESDIV value results in a single system clock per one time quantum, then the information processing time (IPT) equals three time quanta; otherwise it equals two time quanta. If PSEG2 equals two, then the TouCAN transmits one time quantum late relative to the scheduled sync segment.
- If the prescaler and bit timing control fields are programmed to values that result in fewer than 10 system clock periods per CAN bit time and the CAN bus loading is 100%, then anytime the rising edge of a start-of-frame (SOF) symbol transmitted by another node occurs during the third bit of the intermission between messages, the TouCAN may not be able to prepare a message buffer for transmission in time to begin its own transmission and arbitrate against the message which transmitted the early SOF.
- The TouCAN bit time must be programmed to be greater than or equal to nine system clocks, or correct operation is not guaranteed.

7.4.4 Error Counters

The TouCAN has two error counters, the transmit (TX) error counter and the receive (RX) error counter. Refer to **7.8 Programmer's Model** for more information on error counters.The rules for increasing and decreasing these counters are described in the CAN protocol, and are fully implemented in the TouCAN. Each counter has the following features:

- 8-bit up/down counter

- Increment by 8 (RX error counter also increments by one)
- Decrement by one
- Avoid decrement when equal to zero
- RX error counter reset to a value between 119 and 127 inclusive, when the TouCAN transitions from error passive to error active
- Following reset, both counters reset to zero
- Detect values for error passive, bus off and error active transitions
- Cascade usage of TX error counter with an additional internal counter to detect the 128 occurrences of 11 consecutive recessive bits necessary to transition from bus off into error active.

Both counters are read only (except in test/freeze/halt modes).

The TouCAN responds to any bus state as described in the CAN protocol, transmitting an error active or error passive flag, delaying its transmission start time (error passive) and avoiding any influence on the bus when in the bus off state. The following are the basic rules for TouCAN bus state transitions:

- If the value of the TX error counter or RX error counter increments to a value greater than or equal to 128, the fault confinement state (FCS[1:0]) field in the error status register is updated to reflect an error passive state.
- If the TouCAN is in an error passive state, and either the TX error counter or RX error counter decrements to a value less than or equal to 127 while the other error counter already satisfies this condition, the FCS[1:0] field in the error status register is updated to reflect an error active state.
- If the value of the TX error counter increases to a value greater than 255, the FCS[1:0] field in the error status register is updated to reflect a bus off state, and an interrupt may be issued. The value of the TX error counter is reset to zero.
- If the TouCAN is in the bus off state, the TX error counter and an additional internal counter are cascaded to count 128 occurrences of 11 consecutive recessive bits on the bus. To do this, the TX error counter is first reset to zero, and then the internal counter begins counting consecutive recessive bits. Each time the internal counter counts 11 consecutive recessive bits, the TX error counter is incremented by one and the internal counter is reset to zero. When the TX error counter reaches the value of 128, the FCS[1:0] field in the error status register is updated to be error active, and both error counters are reset to zero. Any time a dominant bit is detected following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero but does not affect the TX error counter value.
- If only one node is operating in a system, the TX error counter is incremented with each message it attempts to transmit, due to the resulting acknowledgment errors. However, acknowledgment errors never cause the TouCAN to change from the error passive state to the bus off state.
- If the RX error counter increments to a value greater than 127, it stops incrementing, even if more errors are detected while being a receiver. After the next successful message reception, the counter is reset to a value between 119 and 127, to enable a return to the error active state.

7.4.5 Time Stamp

The value of the free-running 16-bit timer is sampled at the beginning of the identifier field on the CAN bus. For a message being received, the time stamp is stored in the time stamp entry of the receive message buffer at the time the message is written into that buffer. For a message being transmitted, the time stamp entry is written into the transmit message buffer once the transmission has completed successfully.

The free-running timer can optionally be reset upon the reception of a frame into message buffer 0. This feature allows network time synchronization to be performed.

7.5 TouCAN Operation

The basic operation of the TouCAN can be divided into three areas:

- Reset and initialization of the module
- Transmit message handling
- Receive message handling

Example sequences for performing each of these processes is given in the following paragraphs.

7.5.1 TouCAN Reset

The TouCAN can be reset in two ways:

- Hard reset, using one of the IMB3 reset lines.
- Soft reset, using the SOFTRST bit in the module configuration register.

Following the negation of reset, the TouCAN is not synchronized with the CAN bus, and the HALT, FRZ, and FRZACK bits in the module configuration register are set. In this state, the TouCAN does not initiate frame transmissions or receive any frames from the CAN bus. The contents of the message buffers are not changed following reset.

Any configuration change or initialization requires that the TouCAN be frozen by either the assertion of the HALT bit in the module configuration register or by reset.

7.5.2 TouCAN Initialization

Initialization of the TouCAN includes the initial configuration of the message buffers and configuration of the CAN communication parameters following a reset, as well as any reconfiguration which may be required during operation. The following is a general initialization sequence for the TouCAN:

1. Initialize all operation modes
   a. Initialize the transmit and receive pin modes in control register 0 (CANCTRL0).
   b. Initialize the bit timing parameters PROPSEG, PSEGS1, PSEG2, and RJW in control registers 1 and 2 (CANCTRL[1:2]).
   c. Select the S-clock rate by programming the PRESDIV register.
   d. Select the internal arbitration mode (LBUF bit in CANCTRL1).

2. Initialize message buffers
   a. The control/status word of all message buffers must be written either as an active or inactive message buffer.
   b. All other entries in each message buffer should be initialized as required.
3. Initialize mask registers for acceptance mask as needed
4. Initialize TouCAN interrupt handler
   a. Initialize the interrupt configuration register (CANICR) with a specific request level
   b. Set the required mask bits in the IMASK register (for all message buffer interrupts), in CANCTRL0 (for bus off and error interrupts), and in CANMCR for the WAKE interrupt.
5. Negate the HALT bit in the module configuration register
   a. At this point, the TouCAN attempts to synchronize with the CAN bus.

NOTE

In both the transmit and receive processes, the first action in preparing a message buffer must be to deactivate the buffer by setting its code field to the proper value. This step is mandatory to ensure data coherency.

7.5.3 Transmit Process

The transmit process includes preparation of a message buffer for transmission, as well as the internal steps performed by the TouCAN to decide which message to transmit. For the user, this involves loading the message and ID to be transmitted into a message buffer and then activating that buffer as an active transmit buffer. Once this is done, the TouCAN performs all additional steps necessary to transmit the message onto the CAN bus.

The user should prepare or change a message buffer for transmission by executing the following steps.

1. Write the control/status word to hold the transmit buffer inactive (code = 0b1000)
2. Write the ID_HIGH and ID_LOW words
3. Write the data bytes
4. Write the control/status word (active TX code, TX length)

NOTE

Steps 1 and 4 are mandatory to ensure data coherency.

Once an active transmit code is written to a transmit message buffer, that buffer begins participating in an internal arbitration process as soon as the receiver senses that the CAN bus is free, or at the inter-frame space. If there are multiple messages awaiting

transmission, this internal arbitration process selects the message buffer from which the next frame is transmitted.

When this process is over and a message buffer is selected for transmission, the frame from that message buffer is transferred to the serial message buffer for transmission.

The TouCAN transmit s no more than eight data bytes, even if the transmit length contains a value greater than eight.

At the end of a successful transmission, the value of the free-running timer (which was captured at the beginning of the identifier field on the CAN bus), is written into the time stamp field in the message buffer. The code field in the control/status word of the message buffer is updated and a status flag is set in the IFLAG register.

7.5.3.1 Transmit Message Buffer Deactivation

Any write access to the control/status word of a transmit message buffer during the process of selecting a message buffer for transmission immediately deactivates that message buffer, removing it from the transmission process.

If the user deactivates a transmit message buffer while a message is being transferred from it to a serial message buffer, the message is not transmitted.

If the user deactivates the transmit message buffer after the message is transferred to the serial message buffer, the message is transmitted, but no interrupt is requested, and the transmit code is not updated.

If a message buffer containing the lowest ID is deactivated while that message is undergoing the internal arbitration process to determine which message should be sent, then that message may not be transmitted.

7.5.3.2 Reception of Transmitted Frames

The TouCAN receives a frame it has transmitted if an empty message buffer with a matching identifier exists.

7.5.4 Receive Process

During the receive process, the following events occur:

- The user configures the message buffers for reception.
- The TouCAN transfers received messages from the serial message buffers to the receive message buffers with matching IDs.
- The user retrieves these messages.

The user should prepare or change a message buffer for frame reception by executing the following steps.

1. Write the control/status word to hold the receive buffer inactive (code = 0b0000).
2. Write the ID_HIGH and ID_LOW words.

3. Write the control/status word to mark the receive message buffer as active and empty.

NOTE

Steps 1 and 3 are mandatory for data coherency.

Once these steps are performed, the message buffer functions as an active receive buffer and participates in the internal matching process, which takes place every time the TouCAN receives an error-free frame. In this process, all active receive buffers compare their ID value to the newly received one. If a match is detected, the following actions occur:

1. The frame is transferred to the first (lowest entry) matching receive message buffer.
2. The value of the free-running timer (captured at the beginning of the identifier field on the CAN bus) is written into the time stamp field in the message buffer.
3. The ID field, data field, and RX length field are stored.
4. The code field is updated.
5. The status flag is set in the IFLAG register.

The user should read a received frame from its message buffer in the following order:

1. Control/status word (mandatory, as it activates the internal lock for this buffer)
2. ID (optional, since it is needed only if a mask was used)
3. Data field word(s)
4. Free-running timer (optional, as it releases the internal lock)

If the free running timer is not read, that message buffer remains locked until the read process starts for another message buffer. Only a single message buffer is locked at a time. When a received message is read, the only mandatory read operation is that of the control/status word. This ensures data coherency.

If the BUSY bit is set in the message buffer code, the CPU should defer accessing that buffer until this bit is negated. Refer to **Table 7-2**.

NOTE

The user should check the status of a message buffer by reading the status flag in the IFLAG register and not by reading the control/status word code field for that message buffer. This prevents the buffer from being locked inadvertently.

Because the received identifier field is always stored in the matching receive message buffer, the contents of the identifier field in a receive message buffer may change if one or more of the ID bits are masked.

7.5.4.1 Receive Message Buffer Deactivation

Any write access to the control/status word of a receive message buffer during the process of selecting a message buffer for reception immediately deactivates that message buffer, removing it from the reception process.

If a receive message buffer is deactivated while a message is being transferred into it, the transfer is halted and no interrupt is requested. If this occurs, that receive message buffer may contain mixed data from two different frames.

Data must never be written into a receive message buffer. If this occurs while a message is being transferred from a serial message buffer, the control/status word will reflect a full or overrun condition, but no interrupt is requested.

7.5.4.2 Locking and Releasing Message Buffers

The lock/release/busy mechanism is designed to guarantee data coherency during the receive process. The following examples demonstrate how the the lock/release/busy mechanism affects TouCAN operation.

1. Reading a control/status word of a message buffer triggers a lock for that message buffer. A new received message frame which matches the message buffer cannot be written into this message buffer while it is locked.
2. To release a locked message buffer, the CPU either locks another message buffer by reading its control/status word or globally releases any locked message buffer by reading the free-running timer.
3. If a receive frame with a matching ID is received during the time the message buffer is locked, the receive frame is not immediately transferred into that message buffer, but remains in the serial message buffer. There is no indication when this occurs.
4. When a locked message buffer is released, if a frame with a matching identifier exists within the serial message buffer, then this frame is transferred to the matching message buffer.
5. If two or more receive frames with matching IDs are received while a message buffer with a matching ID is locked, the last received frame with that ID is kept within the serial message buffer, while all preceding ones are lost. There is no indication when this occurs.
6. If the user reads the control/status word of a receive message buffer while a frame is being transferred from a serial message buffer, the BUSY code is indicated. The user should wait until this code is cleared before continuing to read from the message buffer to ensure data coherency. In this situation, the read of the control/status word does not lock the message buffer.

Polling the control/status word of a receive message buffer can lock it, preventing a message from being transferred into that buffer. If the control/status word of a receive message buffer is read, it should be followed by a read of the control/status word of another buffer, or by a read of the free-running timer, to ensure that the locked buffer is unlocked.

7.5.5 Remote Frames

The remote frame is a message frame that is transmitted to request a data frame. The TouCAN can be configured to transmit a data frame automatically in response to a remote frame, or to transmit a remote frame and then wait for the responding data frame to be received.

To transmit a remote frame, the user initializes a message buffer as a transmit message buffer with the RTR bit set to one. Once this remote frame is transmitted successfully, the transmit message buffer automatically becomes a receive message buffer, with the same ID as the remote frame that was transmitted.

When the TouCAN receives a remote frame, it compares the remote frame ID to the IDs of all transmit message buffers programmed with a code of 1010. If there is an exact matching ID, the data frame in that message buffer is transmitted. If the RTR bit in the matching transmit message buffer is set, the TouCAN transmits a remote frame as a response.

A received remote frame is not stored in a receive message buffer. It is only used to trigger the automatic transmission of a frame in response. The mask registers are not used in remote frame ID matching. All ID bits (except RTR) of the incoming received frame must match for the remote frame to trigger a response transmission.

7.5.6 Overload Frames

The TouCAN does not initiate overload frame transmissions unless it detects the following conditions on the CAN bus:

- A dominant bit in the first or second bit of intermission.
- A dominant bit in the seventh (last) bit of the end-of-frame (EOF) field in receive frames
- A dominant bit in the eighth (last) bit of the error frame delimiter or overload frame delimiter

7.6 Special Operating Modes

The TouCAN module has three special operating modes:

- Debug mode
- Low-power stop mode
- Auto-power save mode

7.6.1 Debug Mode

Debug mode is entered when the FRZ1 bit in CANMCR is set and one of the following events occurs:

- The HALT bit in the CANMCR is set; or
- The IMB3 FREEZE line is asserted

Once entry into debug mode is requested, the TouCAN waits until an intermission or idle condition exists on the CAN bus, or until the TouCAN enters the error passive or

bus off state. Once one of these conditions exists, the TouCAN waits for the completion of all internal activity. Once this happens, the following events occur:

- The TouCAN stops transmitting or receiving frames.
- The prescaler is disabled, thus halting all CAN bus communication.
- The TouCAN ignores its RX pins and drives its TX pins as recessive. The TouCAN loses synchronization with the CAN bus and the NOTRDY and FRZACK bits in CANMCR are set.
- The CPU is allowed to read and write the error counter registers.

After engaging one of the mechanisms to place the TouCAN in debug mode, the user must wait for the FRZACK bit to be set before accessing any other registers in the TouCAN; otherwise unpredictable operation may occur.

To exit debug mode, the IMB FREEZE line must be negated or the HALT bit in CANMCR must be cleared.

Once debug mode is exited, the TouCAN resynchronizes with the CAN bus by waiting for 11 consecutive recessive bits before beginning to participate in CAN bus communication.

7.6.2 Low-Power Stop Mode

Before entering low-power stop mode, the TouCAN waits for the CAN bus to be in an idle state, or for the third bit of intermission to be recessive. The TouCAN then waits for the completion of all internal activity (except in the CAN bus interface) to be complete. Then the following events occur:

- The TouCAN shuts down its clocks, stopping most internal circuits, thus achieving maximum power savings.
- The bus interface unit continues to operate, allowing the CPU to access the module configuration register.
- The TouCAN ignores its RX pins and drives its TX pins as recessive.
- The TouCAN loses synchronization with the CAN bus, and the STOPACK and NOTRDY bits in the module configuration register are set.

To exit low-power stop mode:

- Reset the TouCAN either by asserting one of the IMB3 reset lines or by asserting the SOFTRST bit CANMCR.
- Clear the STOP bit in CANMCR.
- The TouCAN module can optionally exit low-power stop mode via the self-wake mechanism. If the SELFWAKE bit in CANMCR was set at the time the TouCAN entered stop mode, then upon detection of a recessive to dominant transition on the CAN bus, the TouCAN clears the STOP bit in CANMCR and its clocks begin running.

When the TouCAN is in low-power stop mode, a recessive to dominant transition on the CAN bus causes the WAKEINT bit in the error and status register (ESTAT) to be set. This event generates an interrupt if the WAKEMSK bit in CANMCR is set.

Consider the following notes regarding low-power stop mode:

- When the self-wake mechanism is activated, the TouCAN tries to receive the frame that woke it up. (It assumes that the dominant bit detected is a start-of-frame bit.) It will not arbitrate for the CAN bus at this time.
- If the STOP bit is set while the TouCAN is in the bus-off state, then the TouCAN enters low-power stop mode and stops counting recessive bit times. The count continues when STOP is cleared.
- To place the TouCAN in low-power stop mode with the self-wake mechanism engaged, write to CANMCR with both STOP and SELFWAKE set, and then wait for the TouCAN to set the STOPACK bit.
- To take the TouCAN out of low-power stop mode when the self-wake mechanism is enabled, write to CANMCR with both STOP and SELFWAKE clear, and then wait for the TouCAN to clear the STOPACK bit.
- The SELFWAKE bit should not be set after the TouCAN has already entered low-power stop mode.
- If both STOP and SELFWAKE are set and a recessive to dominant edge immediately occurs on the CAN bus, the TouCAN may never set the STOPACK bit, and the STOP bit will be cleared.
- To prevent old frames from being sent when the TouCAN awakes from low-power stop mode via the self-wake mechanism, disable all transmit sources, including transmit buffers configured for remote request responses, before placing the TouCAN in low-power stop mode.
- If the TouCAN is in debug mode when the STOP bit is set, the TouCAN assumes that debug mode should be exited. As a result, it tries to synchronize with the CAN bus, and only then does it await the conditions required for entry into low-power stop mode.
- Unlike other modules, the TouCAN does not come out of reset in low-power stop mode. The basic TouCAN initialization procedure should be executed before placing the module in low-power stop mode. (Refer to **7.5.2 TouCAN Initialization**.)
- If the TouCAN is in low-power stop mode with the self-wake mechanism engaged and is operating with a single system clock per time quantum, there can be extreme cases in which TouCAN wake-up on recessive to dominant edge may not conform to the CAN protocol. TouCAN synchronization is shifted one time quantum from the wake-up event. This shift lasts until the next recessive to dominant edge, which resynchronizes the TouCAN to be in conformance with the CAN protocol. The same holds true when the TouCAN is in auto-power save mode and awakens on a recessive to dominant edge.

7.6.3 Auto-Power Save Mode

Auto power save mode enables normal operation with optimized power savings. Once the auto power save (APS) bit in CANMCR is set, the TouCAN looks for a set of conditions in which there is no need for the clocks to be running. If these conditions are met, the TouCAN stops its clocks, thus saving power. The following conditions activate auto-power save mode.

- No RX/TX frame in progress.
- No transfer of RX/TX frames to and from a serial message buffer, and no TX

frame awaiting transmission in any message buffer.
- No CPU access to the TouCAN module.
- The TouCAN is not in debug mode, low-power stop mode, or the bus off state.

While its clocks are stopped, if the TouCAN senses that any one of the aforementioned conditions is no longer true, it restarts its clocks. The TouCAN then continues to monitor these conditions and stops or restarts its clocks accordingly.

7.7 Interrupts

The TouCAN is capable of generating one interrupt level on the IMB3. This level is programmed into the priority level bits in the interrupt configuration register (CANICR). This value determines which interrupt signal is driven onto the bus when an interrupt is requested.

When an interrupt is requested, the CPU32 initiates an IACK cycle. The TouCAN decodes the IACK cycle and compares the CPU32 recognized level to the level that it is currently requesting. If a match occurs, then arbitration begins. If the TouCAN wins arbitration, it generates a uniquely encoded interrupt vector that indicates which event is requesting service. This encoding scheme is as follows:

- The higher-order bits of the interrupt vector come from the IVBA[2:0] field in CANICR.
- The low-order five bits are an encoded value that indicate which of the 19 TouCAN interrupt sources is requesting service.

Figure 7-5 shows a block diagram of the interrupt hardware.

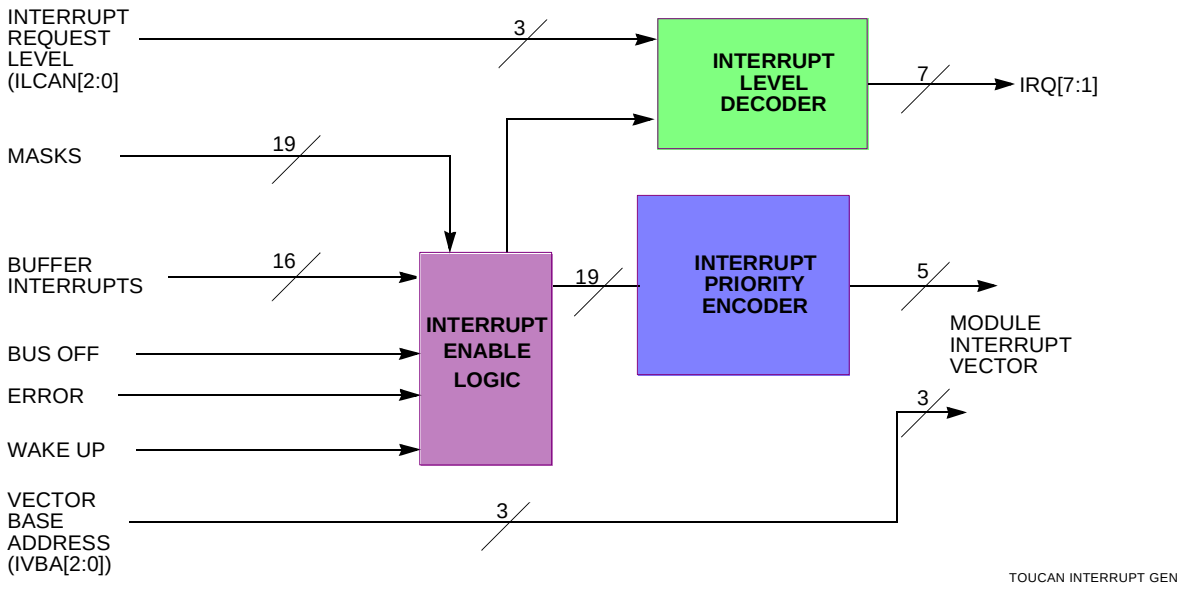


Figure 7-5 TouCAN Interrupt Vector Generation

Each one of the 16 message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between transmit and receive interrupts for a particular buffer. Each of the buffers is assigned a bit in the IFLAG register. An IFLAG bit is set when the corresponding buffer completes a successful transmission/reception. An IFLAG bit is cleared when the CPU32 reads IFLAG while the associated bit is set, and then writes it back as zero (and no new event of the same type occurs between the read and the write actions).

The other three interrupt sources (bus off, error and wake up) act in the same way, and have flag bits located in the error and status register (ESTAT). The bus off and error interrupt mask bits (BOFFMSK and ERRMSK) are located in CANCTRL0, and the wake up interrupt mask bit (WAKEMSK) is located in the module configuration register. **Table 7-9** shows TouCAN interrupt priorities and their corresponding vector addresses.

Table 7-9 Interrupt Sources and Vector Addresses

Interrupt Source	Vector Number
Buffer 0	0bXXX00000 (Highest priority)
Buffer 1	0bXXX00001
Buffer 2	0bXXX00010
Buffer 3	0bXXX00011
Buffer 4	0bXXX00100
Buffer 5	0bXXX00101
Buffer 6	0bXXX00110
Buffer 7	0bXXX00111
Buffer 8	0bXXX01000
Buffer 9	0bXXX01001
Buffer 10	0bXXX01010
Buffer 11	0bXXX01011
Buffer 12	0bXXX01100
Buffer 13	0bXXX01101
Buffer 14	0bXXX01110
Buffer 15	0bXXX01111
Bus off	0bXXX10000
Error	0bXXX10001
Wake-up	0bXXX10010 (Lowest priority)

7.8 Programmer's Model

Table 7-10 shows the TouCAN address map. Refer to **Figure 1-2** to locate the TouCAN modules in the MC68F375 address map.

The column labeled "Access" indicates the privilege level at which the CPU must be operating to access the register. A designation of "S" indicates that supervisor mode is required. A designation of "S/U" indicates that the register can be programmed for either supervisor mode access or unrestricted access.

The address space for each TouCAN module is split, with 128 bytes starting at the base address, and an extra 256 bytes starting at the base address +128. The upper 256 are fully used for the message buffer structures. Of the lower 128 bytes, some are not used. Registers with bits marked as “reserved” should always be written as logic 0.

Typically, the TouCAN control registers are programmed during system initialization, before the TouCAN becomes synchronized with the CAN bus. The configuration registers can be changed after synchronization by halting the TouCAN module. This is done by setting the HALT bit in the TouCAN module configuration register (CANMCR). The TouCAN responds by asserting the CANMCR NOTRDY bit. Additionally, the control registers can be modified while the MCU is in background debug mode.

NOTE

The TouCAN has no hard-wired protection against invalid bit/field programming within its registers. Specifically, no protection is provided if the programming does not meet CAN protocol requirements.

Table 7-10 TouCAN Register Ma p

Access	Address[1]	15 8	7 0
S	0xYFF080	TouCAN Module Configuration Register (TCNMCR) See **Table 7-11** for bit descriptions.	
S	0xYFF082	TouCAN Test Register (TTR)	
S	0xYFF084	TouCAN Interrupt Configuration Register (CANICR)	
S/U	0xYFF086	Control Register 0 (CANCTRL0) See **Table 7-13** and **Table 7-16** for bit descriptions.	Control Register 1 (CANCTRL1)
S/U	0xYFF088	Control and Prescaler Divider Register (PRESDIV) See **Table 7-17** and **Table 7-18** for bit descriptions.	Control Register 2 (CTRL2)
S/U	0xYFF08A	Free-Running Timer Register (TIMER) See **Table 7-19** for bit descriptions.	
—	—	Reserved	
S/U	0xYFF090	Receive Global Mask – High (RXGMASKHI) See **Table 7-20** for bit descriptions.	
S/U	0xYFF092	Receive Global Mask – Low (RXGMASKLO) See **Table 7-20** for bit descriptions.	
S/U	0xYFF094	Receive Buffer 14 Mask – High (RX14MASKHI) See **7.8.9 Receive Buffer 14 Mask Registers** for bit descriptions.	
S/U	0xYFF096	Receive Buffer 14 Mask – Low (RX14MASKLO) See **7.8.9 Receive Buffer 14 Mask Registers** for bit descriptions.	
S/U	0xYFF098	Receive Buffer 15 FMask – High (RX15MASKHI) See **7.8.10 Receive Buffer 15 Mask Registers** for bit descriptions.	
S/U	0xYFF09A	Receive Buffer 15 Mask – Low (RX15MASKLO) See **7.8.10 Receive Buffer 15 Mask Registers** for bit descriptions.	
—	—	Reserved	
S/U	0xYFF0A0	Error and Status Register (ESTAT) See **Table 7-21** for bit descriptions.	
S/U	0xYFF0A2	Interrupt Masks (IMASK) See **Table 7-24** for bit descriptions.	
S/U	0xYFF0A4	Interrupt Flags (IFLAG) See **Table 7-25** for bit descriptions.	
S/U	0xYFF0A6	Receive Error Counter (RXECTR) See **Table 7-26** for bit descriptions.	Transmit Error Counter (TXECTR)

NOTES:

1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIM2E configuration register (SCIMMCR).

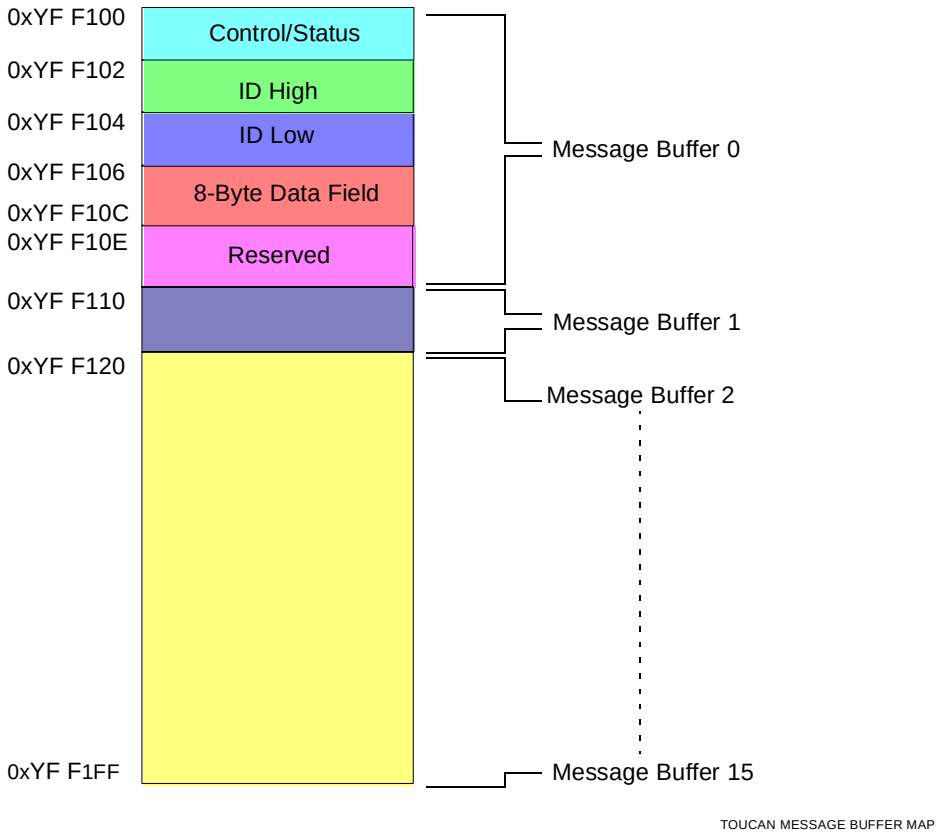


Figure 7-6 TouCAN Message Buffer Memory Map

7.8.1 TouCAN Module Configuration Register

TCNMCR — TouCAN Module Configuration Register **0xYF F080**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	FRZ	NOT USED	HALT	NOT RDY	WAKE MSK	SOFT RST	FRZ ACK	SUPV	SELF WAKE	APS	STOP ACK	IARB[3:0]			
RESET:															
0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0

Table 7-11 TCNMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Low-power stop mode enable. The STOP bit may only be set by the CPU. It may be cleared either by the CPU or by the TouCAN, if the SELFWAKE bit is set. 0 = Enable TouCAN clocks 1 = Disable TouCAN clocks
14	FRZ	FREEZE assertion response. When FRZ = 1, the TouCAN can enter debug mode when the IMB3 FREEZE line is asserted or the HALT bit is set. Clearing this bit field causes the TouCAN to exit debug mode. Refer to **7.6.1 Debug Mode** for more information. 0 = TouCAN ignores the IMB3 FREEZE signal and the HALT bit in the module configuration register. 1 = TouCAN module enabled to enter debug mode.
13	—	Reserved
12	HALT	Halt TouCAN S-Clock. Setting the HALT bit has the same effect as assertion of the IMB3 FREEZE signal on the TouCAN without requiring that FREEZE be asserted. This bit is set to one after reset. It should be cleared after initializing the message buffers and control registers. TouCAN message buffer receive and transmit functions are inactive until this bit is cleared. When HALT is set, write access to certain registers and bits that are normally read-only is allowed. 0 = The TouCAN operates normally 1 = TouCAN enters debug mode if FRZ = 1
11	NOTRDY	TouCAN not ready. This bit indicates that the TouCAN is either in low-power stop mode or debug mode. This bit is read-only and is set only when the TouCAN enters low-power stop mode or debug mode. It is cleared once the TouCAN exits either mode, either by synchronization to the CAN bus or by the self-wake mechanism. 0 = TouCAN has exited low-power stop mode or debug mode. 1 = TouCAN is in low-power stop mode or debug mode.
10	WAKEMSK	Wakeup interrupt mask. The WAKEMSK bit enables wake-up interrupt requests. 0 = Wake up interrupt is disabled. 1 = Wake up interrupt is enabled.
9	SOFTRST	Soft reset. When this bit is asserted, the TouCAN resets its internal state machines (sequencer, error counters, error flags, and timer) and the host interface registers (CANMCR, CANICR, CANTCR, IMASK, and IFLAG). The configuration registers that control the interface with the CAN bus are not changed (CANCTRL[0:2] and PRESDIV). Message buffers and receive message masks are also not changed. This allows SOFTRST to be used as a debug feature while the system is running. Setting SOFTRST also clears the STOP bit in CANMCR. After setting SOFTRST, allow one complete bus cycle to elapse for the internal TouCAN circuitry to completely reset before executing another access to CANMCR. The TouCAN clears this bit once the internal reset cycle is completed. 0 = Soft reset cycle completed 1 = Soft reset cycle initiated
8	FRZACK	TouCAN disable. When the TouCAN enters debug mode, it sets the FRZACK bit. This bit should be polled to determine if the TouCAN has entered debug mode. When debug mode is exited, this bit is negated once the TouCAN prescaler is enabled. This is a read-only bit. 0 = The TouCAN has exited debug mode and the prescaler is enabled. 1 = The TouCAN has entered debug mode, and the prescaler is disabled.
7	SUPV	Supervisor/user data space. The SUPV bit places the TouCAN registers in either supervisor or user data space. 0 = Registers with access controlled by the SUPV bit are accessible in either user or supervisor privilege mode. 1 = Registers with access controlled by the SUPV bit are restricted to supervisor mode.

Table 7-11 TCNMCR Bit Settings (Continued)

Bit(s)	Name	Description
6	SELFWAKE	Self wake enable. This bit allows the TouCAN to wake up when bus activity is detected after the STOP bit is set. If this bit is set when the TouCAN enters low-power stop mode, the TouCAN will monitor the bus for a recessive to dominant transition. If a recessive to dominant transition is detected, the TouCAN immediately clears the STOP bit and restarts its clocks. If a write to CANMCR with SELFWAKE set occurs at the same time a recessive-to-dominant edge appears on the CAN bus, the bit will not be set, and the module clocks will not stop. The user should verify that this bit has been set by reading CANMCR. Refer to **7.6.2 Low-Power Stop Mode** for more information on entry into and exit from low-power stop mode. 0 = Self wake disabled. 1 = Self wake enabled.
5	APS	Auto power save. The APS bit allows the TouCAN to automatically shut off its clocks to save power when it has no process to execute, and to automatically restart these clocks when it has a task to execute without any CPU intervention. 0 = Auto power save mode disabled; clocks run normally. 1 = Auto power save mode enabled; clocks stop and restart as needed.
4	STOPACK	Stop acknowledge. When the TouCAN is placed in low-power stop mode and shuts down its clocks, it sets the STOPACK bit. This bit should be polled to determine if the TouCAN has entered low-power stop mode. When the TouCAN exits low-power stop mode, the STOPACK bit is cleared once the TouCAN's clocks are running. 0 = The TouCAN is not in low-power stop mode and its clocks are running. 1 = The TouCAN has entered low-power stop mode and its clocks are stopped
3:0	IARB[3:0]	Interrupt Arbitration ID. The IARB field is used to arbitrate between simultaneous interrupt requests of the same priority. Each module that can generate interrupt requests must be assigned a unique, non-zero IARB field value.

7.8.2 TouCAN Interrupt Configuration Register

CANICR — TouCAN Interrupt Configuration Register **0xYF F084**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED					ILCAN[2:0]			IVBA[2:0]			RESERVED				
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

NOTE

If the TouCAN issues an interrupt request after reset and before IVBA[2:0] is initialized, it will drive 0x0F as the "uninitialized" interrupt vector in response to a CPU32 interrupt acknowledge cycle, regardless of the specific event.

Table 7-12 CANICR Bit Settings

Bit(s)	Name	Description
15:11	—	Reserved
10:8	ILCAN[2:0]	When the TouCAN generates an interrupt request, ILCAN[2:0] determines which of the interrupt request signals is asserted. When a request is acknowledged, the TouCAN compares ILCAN[2:0] to a mask value supplied by the CPU32 to determine whether to respond. ILCAN[2:0] must have a value in the range of 0x0 (interrupts disabled) to 0x7 (highest priority).
7:5	IVBA[2:0]	The interrupt vector base address specifies the high-order three bits of all the vector numbers generated by the different TouCAN interrupt sources.
4:0	—	Reserved

7.8.3 Control Register 0

CANCTRL0 — Control Register 0 **0xYF F086**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
BOFF MSK	ERR MSK	RESERVED		RXMOD		TXMODE		CANCTRL1							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 7-13 CANCTRL0 Bit Settings

Bit(s)	Name	Description
15	BOFFMSK	Bus off interrupt mask. The BOFF MASK bit provides a mask for the bus off interrupt. 0 = Bus off interrupt disabled. 1 = Bus off interrupt enabled.
14	ERRMSK	Error interrupt mask. The ERRMSK bit provides a mask for the error interrupt. 0 = Error interrupt disabled. 1 = Error interrupt enabled.
13:12	—	Reserved
11:10	RXMODE	Receive pin configuration control. These bits control the configuration of the CANRX0 and CANRX1 pins. Refer to the **Table 7-14**.
9:8	TXMODE	Transmit pin configuration control. This bit field controls the configuration of the CANTX0 and CANTX1 pins. Refer to **Table 7-15**.
7:0	CANCTRL1	See **Table 7-16**.

Table 7-14 RX MODE[1:0] Configuration

Pin	RX1	RX0	Receive Pin Configuration
CANRX1	0	X	A logic 0 on the CANRX1 pin is interpreted as a dominant bit; a logic 1 on the CANRX1 pin is interpreted as a recessive bit
	1	X	A logic 1 on the CANRX1 pin is interpreted as a dominant bit; a logic 0 on the CANRX1 pin is interpreted as a recessive bit
CANRX0	X	0	A logic 0 on the CANRX0 pin is interpreted as a dominant bit; a logic 1 on the CANRX0 pin is interpreted as a recessive bit
	X	1	A logic 1 on the CANRX0 pin is interpreted as a dominant bit; a logic 0 on the CANRX0 pin is interpreted as a recessive bit

Table 7-15 Transmit Pin Configuration

TXMODE[1:0]	Transmit Pin Configuration
00	Full CMOS[1]; positive polarity (CANTX0 = 0, CANTX1 = 1 is a dominant level)
01	Full CMOS; negative polarity (CANTX0 = 1, CANTX1 = 0 is a dominant level)
1X	Open drain[2]; positive polarity

NOTES:

1. Full CMOS drive indicates that both dominant and recessive levels are driven by the chip.
2. Open drain drive indicates that only a dominant level is driven by the chip. During a recessive level, the CANTX0 and CANTX1 pins are disabled (three stated), and the electrical level is achieved by external pull-up/pull-down devices. The assertion of both TX mode bits causes the polarity inversion to be cancelled (open drain mode forces the polarity to be positive).

7.8.4 Control Register 1

CANCTRL1 — Control Register 1 **0xYF F086**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CANCTRL0								SAMP	Re-served	TSYNC	LBUF	0D	PROPSE		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-16 CANCTRL1 Bit Settings

Bit(s)	Name	Description
15:8	CANCTRL0	See **Table 7-13**
7	SAMP	Sampling mode. The SAMP bit determines whether the TouCAN module will sample each received bit one time or three times to determine its value. 0 = One sample, taken at the end of phase buffer segment 1, is used to determine the value of the received bit. 1 = Three samples are used to determine the value of the received bit. The samples are taken at the normal sample point and at the two preceding periods of the S-clock.
6	—	Reserved
5	TSYNC	Timer synchronize mode. The TSYNC bit enables the mechanism that resets the free-running timer each time a message is received in message buffer 0. This feature provides the means to synchronize multiple TouCAN stations with a special “SYNC” message (global network time). 0 = Timer synchronization disabled. 1 = Timer synchronization enabled. Note: there can be a bit clock skew of four to five counts between different TouCAN modules that are using this feature on the same network.
4	LBUF	Lowest buffer transmitted first. The LBUF bit defines the transmit-first scheme. 0 = Message buffer with lowest ID is transmitted first. 1 = Lowest numbered buffer is transmitted first.
3	—	Reserved
2:0	PROPSEG	Propagation segment time. PROPSEG defines the length of the propagation segment in the bit time. The valid programmed values are 0 to 7. The propagation segment time is calculated as follows: Propagation Segment Time = (PROPSEG + 1) Time Quanta where 1 Time Quantum = 1 Serial Clock (S-Clock) Period

7.8.5 Prescaler Divide Register

PRESDIV — Prescaler Divide Register **0xYF F088**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PRESDIV								CANCTRL2							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 7-17 PRESDIV Bit Settings

Bit(s)	Name	Description
15:8	PRESDIV	Prescaler divide factor. PRESDIV determines the ratio between the system clock frequency and the serial clock (S-clock). The S-clock is determined by the following calculation: $$\text{S-clock} = \frac{f_{sys}}{\text{PRESDIV} + 1}$$ The reset value of PRESDIV is 0x00, which forces the S-clock to default to the same frequency as the system clock. The valid programmed values are 0 through 255.
7:0	CANCTRL2	See **Table 7-18**.

7.8.6 Control Register 2

CANCTRL2 — Control Register 2 **0xYF F088**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PRESDIV								RJW		PSEG			PSEG2		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 7-18 CANCTRL2 Bit Settings

Bit(s)	Name	Description
15:8	PRESDIV	See **Table 7-17**.
7:6	RJW	Resynchronization jump width. The RJW field defines the maximum number of time quanta a bit time may be changed during resynchronization. The valid programmed values are 0 through 3. The resynchronization jump width is calculated as follows: Resynchronizaton Jump Width = (RJW + 1) Time Quanta
5:3	PSEG1	PSEG1[2:0] — Phase buffer segment 1. The PSEG1 field defines the length of phase buffer segment 1 in the bit time. The valid programmed values are 0 through 7. The length of phase buffer segment 1 is calculated as follows: Phase Buffer Segment 1 = (PSEG1 + 1) Time Quanta
2:0	PSEG2	PSEG2 — Phase Buffer Segment 2. The PSEG2 field defines the length of phase buffer segment 2 in the bit time. The valid programmed values are 0 through 7. The length of phase buffer segment 2 is calculated as follows: Phase Buffer Segment 2 = (PSEG2 + 1) Time Quanta

7.8.7 Free Running Timer

TIMER — Free Running Timer Register **0xYF F08A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
TIMER															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Table 7-19 TIMER Bit Settings

Bit(s)	Name	Description
15:0	TIMER	The free running timer counter can be read and written by the CPU. The timer starts from zero after reset, counts linearly to 0xFFFF, and wraps around. The timer is clocked by the TouCAN bit-clock. During a message, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it increments at the nominal bit rate. The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. The captured value is written into the “time stamp” entry in a message buffer after a successful reception or transmission of a message.

7.8.8 Receive Global Mask Registers

RXGMSKHI — Receive Global Mask Register High **0xYF F090**
RXGMSKLO — Receive Global Mask Register Low **0xYF F092**

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	LSB 16
MID28	MID27	MID26	MID25	MID24	MID23	MID22	MID21	MID20	MID19	MID18	0	1	MID17	MID16	MID15
RESET:															
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0	0
RESET:															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Table 7-20 RXGMSKHI, RXGMSKLO Bit Settings

Bit(s)	Name	Description
31:0	MIDx	The receive global mask registers use four bytes. The mask bits are applied to all receive-identifiers, excluding receive-buffers 14 and 15, which have their own specific mask registers. Base ID mask bits MID[28:18] are used to mask standard or extended format frames. Extended ID bits MID[17:0] are used to mask only extended format frames. The RTR/SRR bit of a received frame is never compared to the corresponding bit in the message buffer ID field. However, remote request frames (RTR = 1) once received, are never stored into the message buffers. RTR mask bit locations in the mask registers (bits 20 and 0) are always zero, regardless of any write to these bits. The IDE bit of a received frame is always compared to determine if the message contains a standard or extended identifier. Its location in the mask registers (bit 19) is always one, regardless of any write to this bit.

7.8.9 Receive Buffer 14 Mask Registers

RX14MSKHI — Receive Buffer 14 Mask Register High **0xYF F094**
RX14MSKLO — Receive Buffer 14 Mask Register Low **0xYF F096**

The receive buffer 14 mask registers have the same structure as the receive global mask registers and are used to mask buffer 14.

7.8.10 Receive Buffer 15 Mask Registers

RX15MSKHI — Receive Buffer 15 Mask Register High **0xYF F098**
RX15MSKLO — Receive Buffer 15 Mask Register Low **0xYF F09A**

The receive buffer 15 mask registers have the same structure as the receive global mask registers and are used to mask buffer 15.

7.8.11 Error and Status Register

ESTAT — Error and Status Register **0xYF F0A0**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
BITER		ACK ERR	CRC ERR	FORM ERR	STUF F ERR	TX WARN	RX WARN	IDLE	TX/ RX	FCS		0	BOFF INT	ERR INT	WAKE INT
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

This register reflects various error conditions, general status, and has the enable bits for three of the TouCAN interrupt sources. The reported error conditions are those which have occurred since the last time the register was read. A read clears these bits to zero.

Table 7-21 ESTAT Bit Settings

Bit(s)	Name	Description
15:14	BITERR	Transmit bit error. The BITERR[1:0] field is used to indicate when a transmit bit error occurs. Refer to **Table 7-22**. **NOTE**: The transmit bit error field is not modified during the arbitration field or the ACK slot bit time of a message, or by a transmitter that detects dominant bits while sending a passive error frame.
13	ACKERR	Acknowledge error. The ACKERR bit indicates whether an acknowledgment has been correctly received for a transmitted message. 0 = No ACK error was detected since the last read of this register. 1 = An ACK error was detected since the last read of this register.
12	CRCERR	Cyclic redundancy check error. The CRCERR bit indicates whether or not the CRC of the last transmitted or received message was valid. 0 = No CRC error was detected since the last read of this register. 1 = A CRC error was detected since the last read of this register.
11	FORMERR	Message format error. The FORMERR bit indicates whether or not the message format of the last transmitted or received message was correct. 0 = No format error was detected since the last read of this register. 1 = A format error was detected since the last read of this register.
10	STUFERR	Bit stuff error. The STUFFERR bit indicates whether or not the bit stuffing that occurred in the last transmitted or received message was correct. 0 = No bit stuffing error was detected since the last read of this register. 1 = A bit stuffing error was detected since the last read of this register.
9	TXWARN	Transmit error status flag. The TXWARN status flag reflects the status of the TouCAN transmit error counter. 0 = Transmit error counter < 96. 1 = Transmit error counter ≥ 96.

Table 7-21 ESTAT Bit Settings (Continued)

Bit(s)	Name	Description
8	RXWARN	Receiver error status flag. The RXWARN status flag reflects the status of the TouCAN receive error counter. 0 = Receive error counter < 96. 1 = Receive error counter ≥ 96.
7	IDLE	Idle status. The IDLE bit indicates when there is activity on the CAN bus. 0 = The CAN bus is not idle. 1 = The CAN bus is idle.
6	TX/$\overline{\text{RX}}$	Transmit/receive status. The TX/RX bit indicates when the TouCAN module is transmitting or receiving a message. TX/RX has no meaning when IDLE = 1. 0 = The TouCAN is receiving a message if IDLE = 0. 1 = The TouCAN is transmitting a message if IDLE = 0.
5:4	FCS	Fault confinement state. The FCS[1:0] field describes the state of the TouCAN. Refer to **Table 7-23**. If the SOFTRST bit in CANMCR is asserted while the TouCAN is in the bus off state, the error and status register is reset, including FCS[1:0]. However, as soon as the TouCAN exits reset, FCS[1:0] bits will again reflect the bus off state. Refer to**7.4.4 Error Counters** for more information on entry into and exit from the various fault confinement states.
3	—	Reserved
2	BOFFINT	Bus off interrupt. The BOFFINT bit is used to request an interrupt when the TouCAN enters the bus off state. 0 = No bus off interrupt requested. 1 = When the TouCAN state changes to bus off, this bit is set, and if the BOFFMSK bit in CANCTRL0 is set, an interrupt request is generated. This interrupt is not requested after reset.
1	ERRINT	Error Interrupt. The ERRINT bit is used to request an interrupt when the TouCAN detects a transmit or receive error. 0 = No error interrupt request. 1 = If an event which causes one of the error bits in the error and status register to be set occurs, the error interrupt bit is set. If the ERRMSK bit in CANCTRL0 is set, an interrupt request is generated. To clear this bit, first read it as a one, then write as a zero. Writing a one has no effect.
0	WAKEINT	Wake interrupt. The WAKEINT bit indicates that bus activity has been detected while the TouCAN module is in low-power stop mode. 0 = No wake interrupt requested. 1 = When the TouCAN is in low-power stop mode and a recessive to dominant transition is detected on the CAN bus, this bit is set. If the WAKEMSK bit is set in CANMCR, an interrupt request is generated.

Table 7-22 Transmit Bit Error Status

BITERR[1:0]	Bit Error Status
00	No transmit bit error
01	At least one bit sent as dominant was received as recessive
10	At least one bit sent as recessive was received as dominant
11	Not used

Table 7-23 Fault Confinement State Encoding

FCS[1:0]	Bus State
00	Error active
01	Error passive
1X	Bus off

7.8.12 Interrupt Mask Register

IMASK — Interrupt Mask Register **0xYF F0A2**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
IMASKH								IMASKL							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-24 IMASK Bit Settings

Bit(s)	Name	Description
15:8, 7:0	IMASKH, IMASKL	IMASK contains two 8-bit fields, IMASKH and IMASKL. IMASK can be accessed with a 16-bit read or write, and IMASKH and IMASKL can be accessed with byte reads or writes. IMASK contains one interrupt mask bit per buffer. It allows the CPU to designate which buffers will generate interrupts after successful transmission/reception. Setting a bit in IMASK enables interrupt requests for the corresponding message buffer.

7.8.13 Interrupt Flag Register

IFLAG — Interrupt Flag Register **0xYF F0A4**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
IFLAGH								IFLAGL							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-25 IFLAG Bit Settings

Bit(s)	Name	Description
15:8, 7:0	IFLAGH, IFLAGL	IFLAG contains two 8-bit fields, IFLAGH and IFLAGL. IFLAG can be accessed with a 16-bit read or write, and IFLAGH and IFLAGL can be accessed with byte reads or writes. IFLAG contains one interrupt flag bit per buffer. Each successful transmission/reception sets the corresponding IFLAG bit and, if the corresponding IMASK bit is set, an interrupt request will be generated. To clear an interrupt flag, first read the flag as a one, and then write it as a zero. Should a new flag setting event occur between the time that the CPU reads the flag as a one and writes the flag as a zero, the flag is not cleared. This register can be written to zeros only.

7.8.14 Error Counters

RXECTR — Receive Error Counter **0xYF F0A6**
TXECTR — Transmit Error Counter **0xYF F0A6**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RXECTR								TXECTR							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7-26 RXECTR, TXECTR Bit Settings

Bit(s)	Name	Description
15:8, 7:0	RXECTR, TXECTR	Both counters are read only, except when the TouCAN is in test or debug mode.

SECTION 8
TIME PROCESSOR UNIT 3

The time processor unit 3 (TPU3), an enhanced version of the original TPU, is an intelligent, semi-autonomous microcontroller designed for timing control. The TPU3 is fully compatible to the TPU2. Operating simultaneously with the CPU, the TPU3 module process micro-instructions, schedules and processes real-time hardware events, performs input and output, and accesses shared data without CPU intervention. Consequently, for each timer event, the CPU setup and service times are minimized or eliminated. **Figure 8-1** is a simplified block diagram of the TPU3.

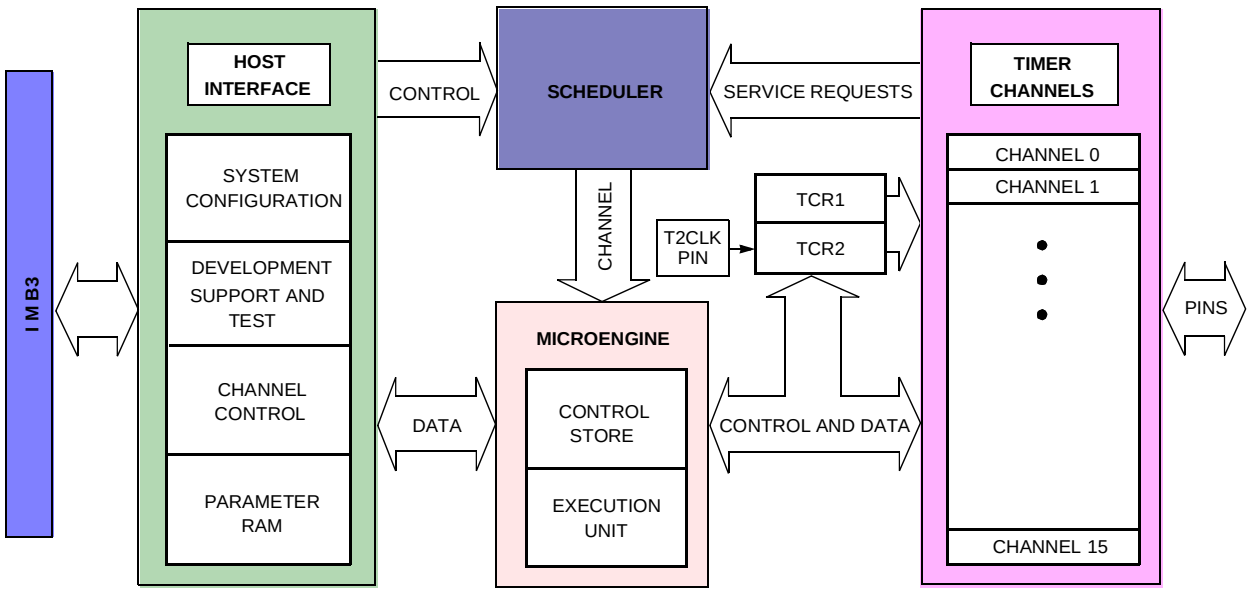


Figure 8-1 TPU3 Block Diagram

8.1 Overview

The TPU3 can be viewed as a special-purpose microcomputer that performs a programmable series of two operations, match and capture. Each occurrence of either operation is called an event. A programmed series of events is called a function. TPU functions replace software functions that would require CPU interrupt service.

The microcode ROM TPU3 functions that are available in the MC68F375 are described in **APPENDIX D TPU ROM FUNCTIONS**.

8.2 TPU3 Components

The TPU3 consists of two 16-bit time bases, 16 independent timer channels, a task scheduler, a microengine, and a host interface. In addition, a dual-ported parameter RAM is used to pass parameters between the module and the CPU.

8.2.1 Time Bases

Two 16-bit counters provide reference time bases for all output compare and input capture events. Prescalers for both time bases are controlled by the CPU via bit fields in the TPU3 module configuration register (TPUMCR) and TPU module configuration register two (TPUMCR2). Timer count registers TCR1 and TCR2 provide access to the current counter values. TCR1 and TCR2 can be read by TPU microcode but are not directly available to the CPU. The TCR1 clock is always derived from the system clock. The TCR2 clock can be derived from the system clock or from an external input via theT2CLK clock pin. The duration between active edges on the T2CLK clock pin must be at least nine system clocks.

8.2.2 Timer Channels

The TPU3 has 16 independent channels, each connected to an MCU pin. The channels have identical hardware and are functionally equivalent in operation. Each channel consists of an event register and pin control logic. The event register contains a 16-bit capture register, a 16-bit compare/match register, and a 16-bit greater-than-or-equal-to comparator. The direction of each pin, either output or input, is determined by the TPU microengine. Each channel can either use the same time base for match and capture, or can use one time base for match and the other for capture.

8.2.3 Scheduler

When a service request is received, the scheduler determines which TPU3 channel is serviced by the microengine. A channel can request service for one of four reasons: for host service, for a link to another channel, for a match event, or for a capture event. The host system assigns each active channel one of three priorities: high, middle, or low. When multiple service requests are received simultaneously, a priority-scheduling mechanism grants service based on channel number and assigned priority.

8.2.4 Microengine

The microengine is composed of a control store and an execution unit. Control-store ROM holds the microcode for each factory-masked time function. When assigned to a channel by the scheduler, the execution unit executes microcode for a function assigned to that channel by the CPU. Microcode can also be executed from the dual-port RAM (DPTRAM) module instead of the control store. The DPTRAM allows emulation and development of custom TPU microcode without the generation of a microcode ROM mask. Refer to **8.3.6 Emulation Support** for more information.

8.2.5 Host Interface

The host interface registers allow communication between the CPU and the TPU3, both before and during execution of a time function. The registers are accessible from the IMB through the TPU3 bus interface unit. Refer to **8.4 Programming Model** for register bit/field definitions and address mapping.

8.2.6 Parameter RAM

Parameter RAM occupies 256 bytes at the top of the system address map. Channel parameters are organized as 128 16-bit words. Channels zero through 15 each have eight parameters. The parameter RAM address map in **8.4.18 TPU3 Parameter RAM** shows how parameter words are organized in memory.

The CPU specifies function parameters by writing to the appropriate RAM address. The TPU3 reads the RAM to determine channel operation. The TPU3 can also store information to be read by the CPU in the parameter RAM. Detailed descriptions of the parameters required by each time function are beyond the scope of this manual. Refer to the *TPU Reference Manual* (TPURM/AD) and the Motorola TPU Literature Package (TPULITPAK/D) for more information.

8.3 TPU Operation

All TPU3 functions are related to one of the two 16-bit time bases. Functions are synthesized by combining sequences of match events and capture events. Because the primitives are implemented in hardware, the TPU3 can determine precisely when a match or capture event occurs, and respond rapidly. An event register for each channel provides for simultaneous match/capture event occurrences on all channels.

When a match or input capture event requiring service occurs, the affected channel generates a service request to the scheduler. The scheduler determines the priority of the request and assigns the channel to the microengine at the first available time. The microengine performs the function defined by the content of the control store or emulation RAM, using parameters from the parameter RAM.

8.3.1 Event Timing

Match and capture events are handled by independent channel hardware. This provides an event accuracy of one time-base clock period, regardless of the number of channels that are active. An event normally causes a channel to request service. The time needed to respond to and service an event is determined by which channels and the number of channels requesting service, the relative priorities of the channels requesting service, and the microcode execution time of the active functions. Worst-case event service time (latency) determines TPU3 performance in a given application. Latency can be closely estimated. For more information, refer to the *TPU Reference Manual* (TPURM/AD).

8.3.2 Channel Orthogonality

Most timer systems are limited by the fixed number of functions assigned to each pin. All TPU3 channels contain identical hardware and are functionally equivalent in oper-

ation, so that any channel can be configured to perform any time function. Any function can operate on the calling channel, and, under program control, on another channel determined by the program or by a parameter. The user controls the combination of time functions.

8.3.3 Interchannel Communication

The autonomy of the TPU3 is enhanced by the ability of a channel to affect the operation of one or more other channels without CPU intervention. Interchannel communication can be accomplished by issuing a link service request to another channel, by controlling another channel directly, or by accessing the parameter RAM of another channel.

8.3.4 Programmable Channel Service Priority

The TPU3 provides a programmable service priority level to each channel. Three priority levels are available. When more than one channel of a given priority requests service at the same time, arbitration is accomplished according to channel number. To prevent a single high-priority channel from permanently blocking other functions, other service requests of the same priority are performed in channel order after the lowest-numbered, highest-priority channel is serviced.

8.3.5 Coherency

For data to be coherent, all available portions of the data must be identical in age, or must be logically related. As an example, consider a 32-bit counter value that is read and written as two 16-bit words. The 32-bit value is read-coherent only if both 16-bit portions are updated at the same time, and write-coherent only if both portions take effect at the same time. Parameter RAM hardware supports coherent access of two adjacent 16-bit parameters. The host CPU must use a long-word operation to guarantee coherency.

8.3.6 Emulation Support

Although factory-programmed time functions can perform a wide variety of control tasks, they may not be ideal for all applications. The TPU3 provides emulation capability that allows the user to develop new time functions. Emulation mode is entered by setting the EMU bit in TPUMCR. In emulation mode, an auxiliary bus connection is made between the DPTRAM and the TPU3, and access to DPTRAM via the intermodule bus is disabled. A 9-bit address bus, a 32-bit data bus, and control lines transfer information between the modules. To ensure exact emulation, DPTFLASH module access timing remains consistent with access timing of the TPU microcode ROM control store.

To support changing TPU application requirements, Motorola has established a TPU function library. The function library is a collection of TPU functions written for easy assembly in combination with each other or with custom functions. Refer to Motorola Programming Note TPUPN00/D, ***Using the TPU Function Library and TPU Emulation Mode*** for information about developing custom functions and accessing the TPU

function library. Refer to the ***Motorola TPU Literature Package*** (TPULITPAK/D) for more information about specific functions.

8.3.7 TPU3 Interrupts

Each of the TPU channels can generate an interrupt service request. Interrupts for each channel must be enabled by writing to the appropriate control bit in the channel interrupt enable register (CIER). The channel interrupt status register (CISR) contains one interrupt status flag per channel. Time functions set the flags. Setting a flag bit causes the TPU to make an interrupt service request if the corresponding channel interrupt enable bit is set and the interrupt request level is non-zero.

The value of the channel interrupt request level (CIRL) field in the TPU interrupt configuration register (TICR) determines the priority of all TPU interrupt service requests. CIRL values correspond to MCU interrupt request signals IRQ[7:1]. IRQ7 is the highest-priority request signal; IRQ1 has the lowest priority. Assigning a value of 0b111 to CIRL causes IRQ7 to be asserted when a TPU interrupt request is made; lower field values cause corresponding lower-priority interrupt request signals to be asserted. Assigning CIRL a value of 0b000 disables all interrupts.

The CPU32 recognizes only interrupt requests of a priority greater than the value contained in the interrupt priority (IP) mask in the status register. When the CPU32 acknowledges an interrupt request, the priority of the acknowledged interrupt is written to the IP mask and is driven out onto the IMB address lines.

When the IP mask value driven out on the address lines is the same as the CIRL value, the TPU contends for arbitration priority. The IARB field in TPUMCR contains the TPU arbitration number. Each module that can make an interrupt service request must be assigned a unique non-zero IARB value in order to implement an arbitration scheme. Arbitration is performed by means of serial assertion of IARB field bit values. The IARB of TPUMCR is initialized to 0x0 during reset.

When the TPU wins arbitration, it must respond to the CPU32 interrupt acknowledge cycle by placing an interrupt vector number on the data bus. The vector number is used to calculate displacement into the exception vector table. Vectors are formed by concatenating the 4-bit value of the CIBV field in TICR with the 4-bit number of the channel requesting interrupt service. Since the CIBV field has a reset value of 0x0, it must be assigned a value corresponding to the upper nibble of a block of 16 user-defined vector numbers before TPU interrupts are enabled. Otherwise, a TPU interrupt service request could cause the CPU32 to take one of the reserved vectors in the exception vector table.

For more information about the exception vector table, refer to **3.9.1 Exception Vectors**.

8.3.8 Prescaler Control for TCR1

Timer count register 1 (TCR1) is clocked from the output of a prescaler. The following fields control TCR1:

- The PSCK and TCR1P fields in TPUMCR
- The DIV2 field in TPUMCR2
- The EPSCKE and EPSCK fields in TPUMCR3.

The rate at which TCR1 is incremented is determined as follows:

- The user selects either the standard prescaler (by clearing the enhanced prescaler enable bit, EPSCKE, in TPUMCR3) or the enhanced prescaler (by setting EPSCKE).
  - — If the standard prescaler is selected (EPSCKE = 0), the the PSCK bit determines whether the standard prescaler divides the system clock input by 32 (PSCK = 0) or four (PSCK = 1)
  - — If the enhanced prescaler is selected (EPSCKE = 1), the EPSCK bits select a value by which the system clock is divided. The lowest frequency for TCR1 clock is system clock divided by 64x8. The highest frequency for TCR1 clock is system clock divided by two (2x1). See **Table 8-1**.

Table 8-1 Enhanced TCR1 Prescaler Divide Values

EPSCK Value	Divide System Clock By
0x00	2
0x01	4
0x02	6
0x03	8
0x04, 0x05,...0x1d	10,12,...60
0x1e	62
0x1f	64

  - — The output of either the standard prescaler or the enhanced prescaler is then divided by 1, 2, 4, or 8, depending on the value of the TCR1P field in the TPUMCR.

Table 8-2 TCR1 Prescaler Values

TCR1P Value	Divide by
0b00	1
0b01	2
0b10	4
0b11	8

  - — If the DIV2 bit is one, the TCR1 counter increments at a rate of the internal clock divided by two. If DIV2 is zero, the TCR1 increment rate is defined by the output of the TCR1 prescaler (which, in turn, takes as input the output of either the standard or enhanced prescaler).

Figure 8-2 shows a diagram of the TCR1 prescaler control block.

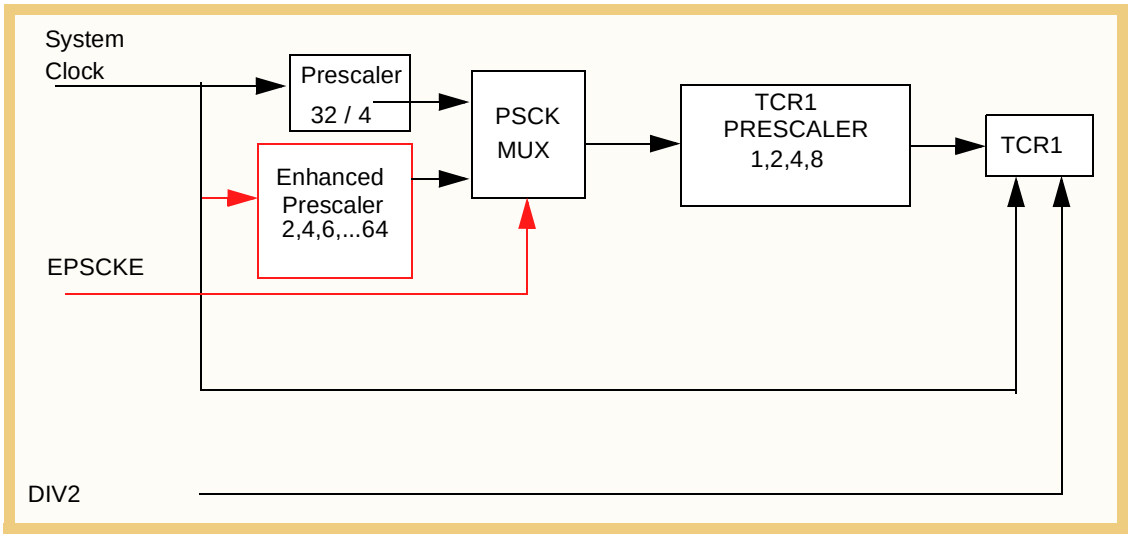


Figure 8-2 TCR1 Prescaler Control

8.3.9 Prescaler Control for TCR2

Timer count register 2 (TCR2), like TCR1, is clocked from the output of a prescaler. The T2CG (TCR2 clock/gate control) bit and the T2CSL (TCR2 counter clock edge) bit in TPUMCR determine T2CR2 pin functions. Refer to **Table 8-3**.

Table 8-3 TCR2 Counter Clock Source

T2CSL	T2CG	TCR2 Clock
0	0	Rise transition T2CLK
0	1	Gated system clock
1	0	Fall transition T2CLK
1	1	Rise & fall transition T2CLK

The function of the T2CG bit is shown in **Figure 8-3**.

When T2CG is set, the external T2CLK pin functions as a gate of the DIV8 clock (the TPU3 system clock divided by eight). In this case, when the external TCR2 pin is low, the DIV8 clock is blocked, preventing it from incrementing TCR2. When the external TCR2 pin is high, TCR2 is incremented at the frequency of the DIV8 clock. When T2CG is cleared, an external clock from the TCR2 pin, which has been synchronized and fed through a digital filter, increments TCR2. The duration between active edges on the T2CLK clock pin must be at least nine system clocks.

The TCR2PSCK2 bit in TPUMCR3 determines whether the clock source is divided by two before it is fed into the TCR2 prescaler. The TCR2 field in TPUMCR specifies the value of the prescaler: 1, 2, 4, or 8. Channels using TCR2 have the capability to

resolve down to the TPU3 system clock divided by eight. **Figure 8-3** illustrates the TCR2 pre-divider and pre-scaler control.

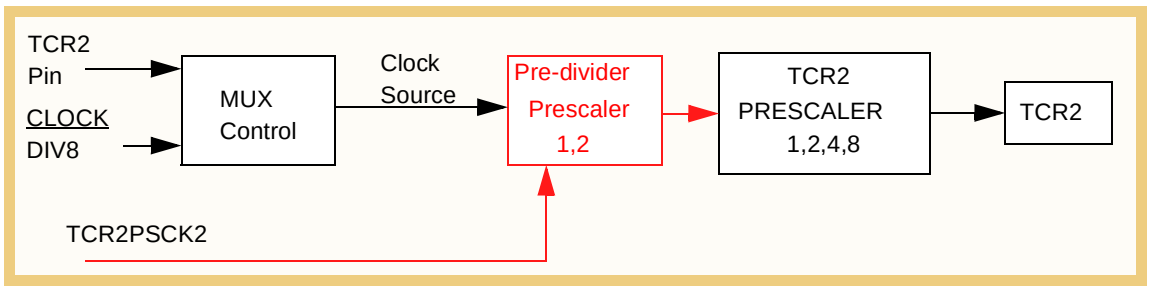


Figure 8-3 TCR2 Prescaler Control

Table 8-4 is a summary of prescaler output (assuming a divide-by-one value for the pre-divider prescaler.

Table 8-4 TCR2 Prescaler Control

TCR2 Prescaler	Divide By	Internal Clock Divided By	External Clock Divided By
00	1	8	1
01	2	16	2
10	4	32	4
11	8	64	8

8.4 Programming Model

The TPU3 memory map contains three groups of registers:

- System configuration registers
- Channel control and status registers
- Development support and test verification registers

All registers except the channel interrupt status register (CISR) must be read or written by means of half-word (16-bit) or word (32-bit) accesses. The address space of the TPU3 memory map occupies 512 bytes. Unused registers within the 512-byte address space return zeros when read.

Table 8-5 shows the TPU3 address map.

Table 8-5 TPU3 Register Map

Address	MSB 15 — Register — LSB 0
0xYF FE00	TPU3 Module Configuration Register (TPUMCR) See **Table 8-6** for bit descriptions.
0xYF FE02	TPU3 Test Configuration Register (TCR)
0xYF FE04	Development Support Control Register (DSCR) See **Table 8-7** for bit descriptions.
0xYF FE06	Development Support Status Register (DSSR) See **Table 8-8** for bit descriptions.
0xYF FE08	TPU3 Interrupt Configuration Register (TICR) See **Table 8-9** for bit descriptions.
0xYF FE0A	Channel Interrupt Enable Register (CIER) See **Table 8-10** for bit descriptions.
0xYF FE0C	Channel Function Selection Register 0 (CFSR0) See **Table 8-11** for bit descriptions.
0xYF FE0E	Channel Function Selection Register 1 (CFSR1) See **Table 8-11** for bit descriptions.
0xYF FE10	Channel Function Selection Register 2 (CFSR2) See **Table 8-11** for bit descriptions.
0xYF FE12	Channel Function Selection Register 3 (CFSR3) See **Table 8-11** for bit descriptions.
0xYF FE14	Host Sequence Register 0 (HSQR0) See **Table 8-12** for bit descriptions.
0xYF FE16	Host Sequence Register 1 (HSQR1) See **Table 8-12** for bit descriptions.
0xYF FE18	Host Service Request Register 0 (HSRR0) See **Table 8-13** for bit descriptions.
0xYF FE1A	Host Service Request Register 1 (HSRR1) See **Table 8-13** for bit descriptions.
0xYF FE1C	Channel Priority Register 0 (CPR0) See **Table 8-14** for bit descriptions.
0xYF FE1E	Channel Priority Register 1 (CPR1) See **Table 8-14** for bit descriptions.
0xYF FE20	Channel Interrupt Status Register (CISR) See **Table 8-16** for bit descriptions.
0xYF FE22	Link Register (LR)
0xYF FE24	Service Grant Latch Register (SGLR)
0xYF FE26	Decoded Channel Number Register (DCNR)
0xYF FE28	TPU Module Configuration Register 2 (TPUMCR2) See **Table 8-17** for bit descriptions.
0xYF FE2A	TPU Module Configuration 3 (TPUMCR3) See **Table 8-20** for bit descriptions.
0xYF FE2C	Internal Scan Data Register (ISDR)
0xYF FE2E	Internal Scan Control Register (ISCR)
0xYF FF00 – 0xYF FF0F	Channel 0 Parameter Registers
0xYF FF10 – 0xYF FF1F	Channel 1 Parameter Registers

Table 8-5 TPU3 Register Map (Continued)

Address	MSB 15 — LSB 0 Register
0xYF FF20 – 0xYF FF2F	Channel 2 Parameter Registers
0xYF FF30 – 0xYF FF3F	Channel 3 Parameter Registers
0xYF FF40 – 0xYF FF4F	Channel 4 Parameter Registers
0xYF FF50 – 0xYF FF5F	Channel 5 Parameter Registers
0xYF FF60 – 0xYF FF6F	Channel 6 Parameter Registers
0xYF FF70 – 0xYF FF7F	Channel 7 Parameter Registers
0xYF FF80 – 0xYF FF8F	Channel 8 Parameter Registers
0xYF FF90 – 0xYF FF9F	Channel 9 Parameter Registers
0xYF FFA0 – 0xYF FFAF	Channel 10 Parameter Registers
0xYF FFB0 – 0xYF FFBF	Channel 11 Parameter Registers
0xYF FFC0 – 0xYF FFCF	Channel 12 Parameter Registers
0xYF FFD0 – 0xYF FFDF	Channel 13 Parameter Registers
0xYF FFE0 – 0xYF FFEF	Channel 14 Parameter Registers
0xYF FFF0 – 0xYF FFFF	Channel 15 Parameter Registers

8.4.1 TPU Module Configuration Register

TPUMCR — TPU Module Configuration Register **0xYF FE00**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	TCR1P		TCR2P		EMU	T2CG	STF	SUPV	PSCK	TPU3	T2CSL	IARB[3:0]			
RESET:															
0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Table 8-6 TPUMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Low-power stop mode enable. If the STOP bit in TPUMCR is set, the TPU3 shuts down its internal clocks, shutting down the internal microengine. TCR1 and TCR2 cease to increment and retain the last value before the stop condition was entered. The TPU3 asserts the stop flag (STF) in TPUMCR to indicate that it has stopped. 0 = Enable TPU3 clocks 1 = Disable TPU3 clocks
14:13	TCR1P	Timer count register 1 prescaler control. TCR1 is clocked from the output of a prescaler. The prescaler divides its input by 1, 2, 4, or 8. This is a write-once field unless the PWOD bit in TPUMCR3 is set. 00 = Divide by 1 01 = Divide by 2 10 = Divide by 4 11 = Divide by 8 Refer to **8.3.8 Prescaler Control for TCR1** for more information.
12:11	TCR2P	Timer count register 2 prescaler control. TCR2 is clocked from the output of a prescaler. The prescaler divides this input by 1, 2, 4, or 8. This is a write-once field unless the PWOD bit in TPUMCR3 is set. 00 = Divide by 1 01 = Divide by 2 10 = Divide by 4 11 = Divide by 8 Refer to **8.3.9 Prescaler Control for TCR2** for more information.
10	EMU	Emulation control. In emulation mode, the TPU3 executes microinstructions from DPTRAM exclusively. Access to the DPTRAM via the IMB3 is blocked, and the DPTRAM is dedicated for use by the TPU3. After reset, this bit can be written only once. 0 = TPU3 and DPTRAM operate normally 1 = TPU3 and DPTRAM operate in emulation mode
9	T2CG	TCR2 clock/gate control 0 = TCR2 pin used as clock source for TCR2 1 = TCR2 pin used as gate of DIV8 clock for TCR2 Refer to **8.3.9 Prescaler Control for TCR2** for more information.
8	STF	Stop flag. 0 = TPU3 is operating normally 1 = TPU3 is stopped (STOP bit has been set)
7	SUPV	Supervisor data space 0 = Assignable registers are accessible from user or supervisor privilege level 1 = Assignable registers are accessible from supervisor privilege level only
6	PSCK	Standard prescaler clock. Note that this bit has no effect if the extended prescaler is selected (EPSCKE = 1). 0 = $f_{SYS} \div 32$ is input to TCR1 prescaler, if standard prescaler is selected 1 = $f_{SYS} \div 4$ is input to TCR1 prescaler, if standard prescaler is selected
5	TPU3	TPU3 enable. The TPU3 enable bit provides compatibility with the TPU. If running TPU code on the TPU3, the microcode size should not be greater than 2 Kbytes and the TPU3 enable bit should be cleared to zero. The TPU3 enable bit is write-once after reset. The reset value is one, meaning that the TPU3 will operate in TPU3 mode. 0 = TPU mode; zero is the TPU reset value 1 = TPU3 mode; one is the TPU3 reset value **NOTE**: The programmer should not change this value unless necessary when developing custom TPU microcode.

Table 8-6 TPUMCR Bit Settings (Continued)

Bit(s)	Name	Description
4	T2CSL	TCR2 counter clock edge. This bit and the T2CG control bit determine the clock source for TCR2. Refer to **8.3.9 Prescaler Control for TCR2** for details.
3:0	IARB[3:0]	Interrupt Arbitration ID. The IARB field is used to arbitrate between simultaneous interrupt requests of the same priority. Each module that can generate interrupt requests must be assigned a unique, non-zero IARB field value.

8.4.2 TPU3 Test Configuration Register

TCR — TPU3 Test Configuration Register **0x30 4002, 0x30 4402**

Used for factory test only.

8.4.3 Development Support Control Register

DSCR — Development Support Control Register **0xYF FE04**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
HOT4	RESERVED				BLC	CLKS	FRZ		CCL	BP	BC	BH	BL	BM	BT
RESET:															
0					0	0	0	0	0	0	0	0	0	0	0

Table 8-7 DSCR Bit Settings

Bit(s)	Name	Description
15	HOT4	Hang on T4 0 = Exit wait on T4 state caused by assertion of HOT4 1 = Enter wait on T4 state
14:11	—	Reserved
10	BLC	Branch latch control 0 = Latch conditions into branch condition register before exiting halted state 1 = Do not latch conditions into branch condition register before exiting the halted state or during the time-slot transition period
9	CLKS	Stop clocks (to TCRs) 0 = Do not stop TCRs 1 = Stop TCRs during the halted state
8:7	FRZ	FREEZE assertion response. The FRZ bits specify the TPU microengine response to the IMB3 FREEZE signal 00 = Ignore freeze 01 = Reserved 10 = Freeze at end of current microcycle 11 = Freeze at next time-slot boundary
6	CCL	Channel conditions latch. CCL controls the latching of channel conditions (MRL and TDL) when the CHAN register is written. 0 = Only the pin state condition of the new channel is latched as a result of the write CHAN register microinstruction 1 = Pin state, MRL, and TDL conditions of the new channel are latched as a result of a write CHAN register microinstruction
5	BP	µPC breakpoint enable 0 = Breakpoint not enabled 1 = Break if µPC equals µPC breakpoint register

Table 8-7 DSCR Bit Settings (Continued)

Bit(s)	Name	Description
4	BC	Channel breakpoint enable 0 = Breakpoint not enabled 1 = Break if CHAN register equals channel breakpoint register at beginning of state or when CHAN is changed through microcode
3	BH	Host service breakpoint enable 0 = Breakpoint not enabled 1 = Break if host service latch is asserted at beginning of state
2	BL	Link service breakpoint enable 0 = Breakpoint not enabled 1 = Break if link service latch is asserted at beginning of state
1	BM	MRL breakpoint enable 0 = Breakpoint not enabled 1 = Break if MRL is asserted at beginning of state
0	BT	TDL breakpoint enable 0 = Breakpoint not enabled 1 = Break if TDL is asserted at beginning of state

8.4.4 Development Support Status Register

DSSR — Development Support Status Register **0xYF FE06**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED								BKPT	PCBK	CHBK	SRBK	TPUF	RESERVED		
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-8 DSSR Bit Settings

Bit(s)	Name	Description
15:8	—	Reserved
7	BKPT	Breakpoint asserted flag. If an internal breakpoint caused the TPU3 to enter the halted state, the TPU3 asserts the BKPT signal on the IMB and sets the BKPT flag. BKPT remains set until the TPU3 recognizes a breakpoint acknowledge cycle, or until the IMB FREEZE signal is asserted.
6	PCBK	µPC breakpoint flag. PCBK is asserted if a breakpoint occurs because of a µPC (microprogram counter) register match with the µPC breakpoint register. PCBK is negated when the BKPT flag is cleared.
5	CHBK	Channel register breakpoint flag. CHBK is asserted if a breakpoint occurs because of a CHAN register match with the CHAN register breakpoint register. CHBK is negated when the BKPT flag is cleared.
4	SRBK	Service request breakpoint flag. SRBK is asserted if a breakpoint occurs because of any of the service request latches being asserted along with their corresponding enable flag in the development support control register. SRBK is negated when the BKPT flag is cleared.
3	TPUF	TPU3 FREEZE flag. TPUF is set whenever the TPU3 is in a halted state as a result of FREEZE being asserted. This flag is automatically negated when the TPU3 exits the halted state because of FREEZE being negated.
2:0	—	Reserved

8.4.5 TPU3 Interrupt Configuration Register

TICR — TPU3 Interrupt Configuration Register **0xYF FE08**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED					CIRL			CIBV				RESERVED			
RESET:															
					0	0	0	0	0	0	0				

Table 8-9 TICR Bit Settings

Bit(s)	Name	Description
15:11	—	Reserved
10:8	CIRL	Channel interrupt request level. This three-bit field specifies the interrupt request level for all channels. T field is used in conjunction with the ILBS field to determine the request level of TPU3 interrupts.
7:4	CIBV	Channel interrupt base vector. The TPU is assigned 16 unique interrupt vector numbers, one vector number for each channel. The CIBV field specifies the most significant nibble of all 16 TPU channel interrupt vector numbers. The lower nibble of the TPU interrupt vector number is determined by the channel number on which the interrupt occurs.
3:0	—	Reserved.

8.4.6 Channel Interrupt Enable Register

The channel interrupt enable register (CIER) allows the CPU to enable or disable the ability of individual TPU3 channels to request interrupt service. Setting the appropriate bit in the register enables a channel to make an interrupt service request; clearing a bit disables the interrupt.

CIER — Channel Interrupt Enable Register **0xYF FE0A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-10 CIER Bit Settings

Bit(s)	Name	Description
15:0	CH[15:0]	Channel interrupt enable/disable 0 = Channel interrupts disabled 1 = Channel interrupts enabled **Note:** The MSB (bit 0 in big-endian mode) represents CH15, and the LSB (bit 15 in big-endian mode) represents CH0.

8.4.7 Channel Function Select Registers

Encoded 4-bit fields within the channel function select registers specify one of 16 time functions to be executed on the corresponding channel. Encodings for predefined functions will be provided in a subsequent draft of this document.

CFSR0 — Channel Function Select Register 0 **0xYF FE0C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15				CH 14				CH 13				CH 12			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR1 — Channel Function Select Register 1 **0xYF FE0E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 11				CH 10				CH 9				CH 8			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR2 — Channel Function Select Register 2 **0xYF FE10**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 7				CH 6				CH 5				CH 4			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CFSR3 — Channel Function Select Register 3 **0xYF FE12**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 3				CH 2				CH 1				CH 0			
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-11 CFSRx Bit Settings

Name	Description
CH[15:0]	Encoded time function for each channel. Encoded four-bit fields in the channel function select registers specify one of 16 time functions to be executed on the corresponding channel.

8.4.8 Host Sequence Registers

The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified. Meanings of host sequence bits and host service request bits for predefined time functions will be provided in a subsequent draft of this document.

HSQR0 — Host Sequence Register 0 **0xYF FE14**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HSQR1 — Host Sequence Register 1 **0xYF FE16**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-12 HSQRx Bit Settings

Name	Description
CH[15:0]	Encoded host sequence. The host sequence field selects the mode of operation for the time function selected on a given channel. The meaning of the host sequence bits depends on the time function specified.

8.4.9 Host Service Request Registers

The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits is determined by time function microcode. Refer to the ***TPU Reference Manual*** (TPURM/AD) and the ***Motorola TPU Literature Package*** (TPULITPAK/D) for more information.

HSRR0 — Host Service Request Register 0 **0xYF FE18**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HSRR1 — Host Service Request Register 1 **0xYF FE1A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-13 HSSRx Bit Settings

Name	Description
CH[15:0]	Encoded type of host service. The host service request field selects the type of host service request for the time function selected on a given channel. The meaning of the host service request bits depends on the time function specified. A host service request field cleared to 0b00 signals the host that service is completed by the microengine on that channel. The host can request service on a channel by writing the corresponding host service request field to one of three non-zero states. The CPU must monitor the host service request register until the TPU3 clears the service request to 0b00 before any parameters are changed or a new service request is issued to the channel.

8.4.10 Channel Priority Registers

The channel priority registers (CPR1, CPR2) assign one of three priority levels to a channel or disable the channel.

CPR0 — Channel Priority Register 0 **0xYF FE1C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15		CH 14		CH 13		CH 12		CH 11		CH 10		CH 9		CH 8	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPR1 — Channel Priority Register 1 **0xYF FE1E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 7		CH 6		CH 5		CH 4		CH 3		CH 2		CH 1		CH 0	
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-14 CPRx Bit Settings

Name	Description
CH[15:0]	Encoded channel priority levels. **Table 8-15** indicates the number of time slots guaranteed for each channel priority encoding.

Table 8-15 Channel Priorities

CHx[1:0]	Service	Guaranteed Time Slots
00	Disabled	—
01	Low	1 out of 7
10	Middle	2 out of 7
11	High	4 out of 7

8.4.11 Channel Interrupt Status Register

The channel interrupt status register (CISR) contains one interrupt status flag per channel. Time functions specify via microcode when an interrupt flag is set. Setting a flag causes the TPU3 to make an interrupt service request if the corresponding CIER bit is set. To clear a status flag, read CISR, then write a zero to the appropriate bit. CISR is the only TPU3 register that can be accessed on a byte basis.

CISR — Channel Interrupt Status Register **0xYF FE20**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CH 15	CH 14	CH 13	CH 12	CH 11	CH 10	CH 9	CH 8	CH 7	CH 6	CH 5	CH 4	CH 3	CH 2	CH 1	CH 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-16 CISR Bit Settings

Bit(s)	Name	Description
15:0	CH[15:0]	Channel interrupt status 0 = Channel interrupt not asserted 1 = Channel interrupt asserted

8.4.12 Link Register

LR — Link Register **0xYF FE22**

Used for factory test only.

8.4.13 Service Grant Latch Register

SGLR — Service Grant Latch Register **0xYF FE24**

Used for factory test only.

8.4.14 Decoded Channel Number Register

DCNR — Decoded Channel Number Register **0xYF FE26**

Used for factory test only.

8.4.15 TPU3 Module Configuration Register 2

TPUMCR2 — TPU Module Configuration Register 2 **0xYF FE28**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED							DIV2	SOFT RST	ETBANK		FPSCK			T2CF	DTPU
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 8-17 TPUMCR2 Bit Settings

Bit(s)	Name	Description
15:9	—	Reserved
8	DIV2	Divide by 2 control. When asserted, the DIV2 bit, along with the TCR1P bit and the PSCK bit in the TPUMCR, determines the rate of the TCR1 counter in the TPU3. If set, the TCR1 counter increments at a rate of two system clocks. If negated, TCR1 increments at the rate determined by control bits in the TCR1P and PSCK fields. 0 = TCR1 increments at rate determined by control bits in the TCR1P and PSCK fields of the TPUMCR register 1 = Causes TCR1 counter to increment at a rate of the system clock divided by two
7	SOFT RST	Soft reset. The TPU3 performs an internal reset when both the SOFT RST bit in the TPUMCR2 and the STOP bit in TPUMCR are set. The CPU must write zero to the SOFT RST bit to bring the TPU3 out of reset. The SOFT RST bit must be asserted for at least nine clocks. 0 = Normal operation 1 = Puts TPU3 in reset until bit is cleared **NOTE**: Do not attempt to access any other TPU3 registers when this bit is asserted. When this bit is asserted, it is the only accessible bit in the register.
6:5	ETBANK	Entry table bank select. This field determines the bank where the microcoded entry table is situated. After reset, this field is 0b00. This control bit field is write once after reset. ETBANK is used when the microcode contains entry tables not located in the default bank 0. To execute the ROM functions on this MCU, ETBANK[1:0] must be 00. Refer to **Table 8-18**. **NOTE**: This field should not be modified by the programmer unless necessary because of custom microcode.
4:2	FPSCK	Filter prescaler clock. The filter prescaler clock control bit field determines the ratio between system clock frequency and minimum detectable pulses. The reset value of these bits is zero, defining the filter clock as four system clocks. Refer to **Table 8-19**.
1	T2CF	T2CLK pin filter control. When asserted, the T2CLK input pin is filtered with the same filter clock that is supplied to the channels. This control bit is write once after reset. 0 = Uses fixed four-clock filter 1 = T2CLK input pin filtered with same filter clock that is supplied to the channels
0	DTPU	Disable TPU3 pins. When the disable TPU3 control pin is asserted, pin TP15 is configured as an input disable pin. When the TP15 pin value is zero, all TPU3 output pins are three-stated, regardless of the pins function. The input is not synchronized. This control bit is write once after reset. 0 = TP15 functions as normal TPU3 channel 1 = TP15 pin configured as output disable pin. When TP15 pin is low, all TPU3 output pins are in a high-impedance state, regardless of the pin function.

Table 8-18 Entry Table Bank Location

ETBANK	Bank
00	0
01	1
10	2
11	3

Table 8-19 System Clock Frequency/Minimum Guaranteed Detected Pulse

Filter Control	Divide By	20 MHz	33 MHz
000	4	200 ns	121 ns
001	8	400 ns	242 ns
010	16	800 ns	485 ns
011	32	1.6 µs	970 ns
100	64	3.2 µs	1.94 µs
101	128	6.4 µs	3.88 µs
110	256	12.8 µs	7.76 µs
111	512	25.6 µs	15.51 µs

8.4.16 TPU Module Configuration Register 3

TPUMCR3 — TPU Module Configuration Register 3 **0xYF FE2A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED							PWOD	TCR2 PCK2	EP-SCKE	Re-served	EPSCK				
RESET:					0	0	0	0	0	0	0				

Table 8-20 TPUMCR3 Bit Settings

Bit(s)	Name	Description
15:9	—	Reserved
8	PWOD	Prescaler write-once disable bit. The PWOD bit does not lock the EPSCK field and the EPSCKE bit. 0 = Prescaler fields in MCR are write-once 1 = Prescaler fields in MCR can be written anytime
7	TCR2PSCK2	TCR2 prescaler 2 0 = Prescaler clock source is divided by one 1 = Prescaler clock source is divided by two
6	EPSCKE	Enhanced pre-scaler enable 0 = Disable enhanced prescaler (use standard prescaler) 1 = Enable enhanced prescaler. System clock will be divided by the value in EPSCK field.
5	—	Reserved
4:0	EPSCK	Enhanced prescaler value that will be loaded into the enhanced prescaler counter. Prescaler value = (EPSCK + 1) x 2. Refer to **8.3.8 Prescaler Control for TCR1** for details.

8.4.17 TPU3 Test Registers

The following TPU3 registers are used for factory test only:

- Internal Scan Data Register (ISDR, address 0xYF FE2C)
- Internal Scan Control Register (ISCR, address offset 0xYF FE2E)

8.4.18 TPU3 Parameter RAM

The channel parameter registers are organized as one hundred 16-bit words of RAM. Channels 0 to 15 have eight parameters. The parameter registers constitute a shared work space for communication between the CPU and the TPU3. The TPU3 can only access data in the parameter RAM, refer to **Table 8-21**.

Table 8-21 Parameter RAM Address Map[1]

Channel	Parameter							
Number	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	00	02	04	06	08	0A	0C	0E
1	10	12	14	16	18	1A	1C	1E
2	20	22	24	26	28	2A	2C	2E
3	30	32	34	36	38	3A	3C	3E
4	40	42	44	46	48	4A	4C	4E
5	50	52	54	56	58	5A	5C	5E
6	60	62	64	66	68	6A	6C	6E
7	70	72	74	76	78	7A	7C	7E
8	80	82	84	86	88	8A	8C	8E
9	90	92	94	96	98	9A	9C	9E
10	A0	A2	A4	A6	A8	AA	AC	AE
11	B0	B2	B4	B6	B8	BA	BC	BE
12	C0	C2	C4	C6	C8	CA	CC	CE
13	D0	D2	D4	D6	D8	DA	DC	DE
14	E0	E2	E4	E6	E8	EA	EC	EE
15	F0	F2	F4	F6	F8	FA	FC	FE

NOTES:

1. The base address of the parameter RAM is 0xYF FF00. The parameter RAM addresses should be added to the base address.

8.5 Time Functions

Descriptions of the MC68F375 pre-programmed time functions are shown in **APPENDIX D TPU ROM FUNCTIONS**.

SECTION 9
DUAL-PORT TPU RAM (DPTRAM)

9.1 Introduction

The dual-port RAM module with TPU microcode storage support (DPTRAM) consists of a control register block and a 6-Kbyte array of static RAM, which can be used either as a microcode storage for TPU or as a general-purpose memory.

The DPTRAM module acts as a common memory on the IMB3 and allows the transfer of data to the two TPU3 modules. Therefore, the DPTRAM interface includes an IMB3 bus interface and two TPU3 interfaces. When the RAM is being used in microcode mode, the array is only accessible to the TPU3 via a separate local bus, and not via the IMB3.

The dual-port TPU3 RAM (DPTRAM) is intended to serve as fast, two-clock access, general-purpose RAM memory for the MCU. When used as general-purpose RAM, this module is accessed via the MCU's internal bus.

The DPTRAM module is powered by V_{DDL} in normal operation.

NOTE

The VDDDPTRAM pin does not have circuitry to allow for standby operation and should be connected to V_{DDL}.

The DPTRAM may also be used as the microcode control store for up to two TPU3 modules when placed in a special emulation mode. In this mode the DPTRAM array may only be accessed by either or both of the TPU3 units simultaneously via separate emulation buses, and not via the IMB3. The MC68F375 has only one TPU3.

The DPTRAM contains a multiple input signature calculator (MISC) in order to provide RAM data corruption checking. The MISC reads each RAM address and generates a 32-bit data-dependent signature. This signature can then be checked by the host.

The DPTRAM supports soft defects detection (SDD).

9.2 Features

- Six Kbytes of static RAM
- Only accessible by the CPU if neither TPU3 is in emulation mode
- Low-power stop operation
  - — Entered by setting the STOP bit in the DPTMCR
  - — Applies only to IMB3 accesses and not to accesses from either TPU3 interface
- TPU microcode mode
  - — The DPTRAM array acts as a microcode storage for the TPU module. This provides a means executing TPU code out of DPTRAM instead of program-

ming it in the TPU ROM.

- Includes built in check logic which scans the array contents and calculates the RAM signature
- IMB3 bus interface
- Two TPU3 interface units, only one is used on the MC68F375
- Bytes, half-word or word accessible

9.3 DPTRAM Configuration and Block Diagram

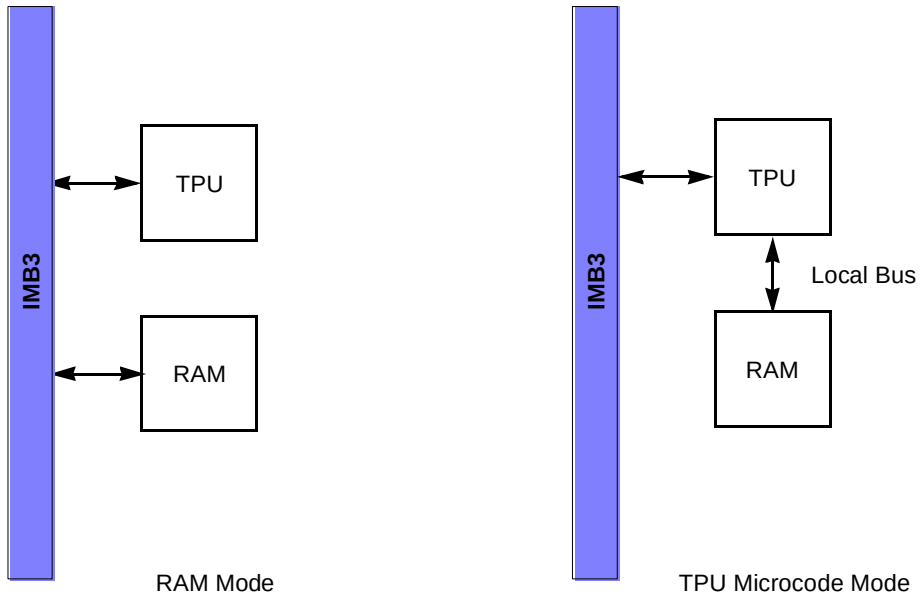


Figure 9-1 DPTRAM Configuration

9.4 Programming Model

The DPTRAM module consists of two separately addressable sections. The first is a set of memory-mapped control and status registers used for configuration (DPTMCR, RAMBAR, MISRH, MISRL, MISCNT) and testing (DPTTCR) of the DPTRAM array. The second section is the array itself.

All DPTRAM module control and status registers are located in supervisor data space. User reads or writes of these will result in a bus error.

When the TPU3 is using the RAM array for microcode control store, none of these control registers have any effect on the operation of the RAM array.

All addresses within the 64-byte control block will respond when accessed properly. Unimplemented addresses will return zeros for read accesses. Likewise, unimplemented bits within registers will return zero when read and will not be affected by write operations.

Table 9-1 shows the DPTRAM control and status registers. The addresses shown are offsets from the base address for the module. Refer to **Figure 1-2** to locate the DPTRAM control block in the MC68F375 address map. The DPTRAM array occupies a 6-Kbyte block.

Table 9-1 DPTRAM Register Map

R/W Access	Address	Register	Reset Value
Supv R/W	0xYF F880	DPT RAM nodule configuration register (DPTMCR) See **Table 9-2** for bit descriptions.	0x0100
Test	0xYF F882	Test configuration register (DPTTCR)	0x0000
Supv R/W	0xYF F884	RAM base address register (DPTBAR) See **Table 9-3** for bit descriptions.	0x0001
Supv Read Only	0xYF F886	Multiple Input Signature Register High (MISRH) See **9.4.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	0x0000
Supv Read Only	0xYF F888	Multiple Input Signature Register Low (MISRL) See **9.4.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	0x0000
Supv Read Only	0xYF F88A	Multiple Input Signature Counter (MISCNT) See **9.4.5 MISC Counter (MISCNT)** for bit descriptions.	Last memory address

9.4.1 DPTRAM Module Configuration Register (DPTMCR)

This register defines the basic configuration of the DPTRAM module. The DPTMCR contains bits to configure the DPT RAM module for stop operation and for proper access rights to the array. The register also contains the MISC control bits.

DPTMCR — DPT Module Configuration Register **0xYF F880**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	Reserved				MISF	MIS-EN	RASP	Reserved							
RESET:															
0					0	0	1	0	0	0	0	0	0	0	0

Table 9-2 DPTMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Low power stop (sleep) mode 0 = DPTRAM clocks running 1 = DPTRAM clocks shut down Only the STOP bit in the DPTMCR may be accessed while the STOP bit is asserted. Accesses to other DPTRAM registers may result in unpredictable behavior. Note also that the STOP bit should be set and cleared independently of the other control bits in this register to guarantee proper operation. Changing the state of other bits while changing the state of the STOP bit may result in unpredictable behavior. Refer to **9.5.4 Stop Operation** for more information.
14:11	—	Reserved
10	MISF	Multiple input signature flag. MISF is readable at any time. This flag bit should be polled by the host to determine if the MISC has completed reading the RAM. If MISF is set, the host should read the MISRH and MISRL registers to obtain the RAM signature. 0 = First signature not ready 1 = MISC has read entire RAM. Signature is latched in MISRH and MISRL and is ready to be read.
9	MISEN	Multiple input signature enable. MISEN is readable and writable at any time. The MISC will only operate when this bit is set and the MC68F375 is in TPU3 emulation mode. When enabled, the MISC will continuously cycle through the RAM addresses, reading each and adding the contents to the MISR. In order to save power, the MISC can be disabled by clearing the MISEN bit. 0 = MISC disabled 1 = MISC enabled
8	RASP	Ram area supervisor/user program/data. The RAM array may be placed in supervisor or unrestricted Space. When placed in supervisor space, (RASP = 1), only a supervisor may access the array. If a supervisor program is accessing the array, normal read/write operation will occur. If a user program is attempting to access the array, the access will be ignored and the address may be decoded externally. 0 = Both supervisor and user access to RAM allowed 1 = Supervisor access only to RAM allowed
7:0	—	Reserved

9.4.2 DPTRAM Test Register

DPTTCR — Test Register **0x30 0002**

DPTTCR is used only during factory testing of the MCU.

9.4.3 Ram Base Address Register (DPTBAR)

The DPTBAR register is used to specify the 16 MSBs of the starting DPT RAM array location in the memory map.

This register can be written only once after a reset and must be written after the DPRTAM is enabled (DPTMCR STOP = 0b0). This prevents runaway software from inadvertently re-mapping the array. Since the locking mechanism is triggered by the first write after reset, the base address of the array should be written in a single operation. Writing only one half of the register will prevent the other half from being written.

DPTBAR — RAM Array Base Address Register **0xYF F884**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	Reserved				RAMDS
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 9-3 DPTBAR Bit Settings

Bit(s)	Name	Description
15:5	A[8:18]	RAM array base address. These bits specify the 11 high-order bits (address lines ADDR[8:18] in little-endian notation) of the 24-bit base address of the RAM array. This allows the array to be placed on a 8-Kbyte boundary anywhere in the memory map. It is the users responsibility not to overlap the RAM array memory map with other modules on the chip.
4:1	—	Reserved. (Bits 11:12 represent A[12:11] in DPTRAM implementation that require them.)
0	RAMDS	RAM disabled. RAMDS is a read-only status bit. The RAM array is disabled after a master reset since the RAMBAR register may be incorrect. When the array is disabled, it will not respond to any addresses on the IMB3. Access to the RAM control register block is not affected when the array is disabled. RAMDS is cleared by the DPTRAM module when a base address is written to the array address field of RAMBAR. RAMDS = 0: RAM enabled RAMDS = 1: RAM disabled

9.4.4 MISR High (MISRH) and MISR Low (MISRL)

The MISRH and MISRL together contain the 32-bit RAM signature calculated by the MISC. These registers are read-only and should be read by the host when the MISF bit in the MCR is set. Note that the naming of the D[31:0] bits represents little-endian bit encoding.

Exiting TPU3 emulation mode results in the reset of both MISRH and MISRL

MISRH — Multiple Input Signature Register High **0xYF F886**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MISRL — Multiple Input Signature Register Low **0xYF F888**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9.4.5 MISC Counter (MISCNT)

The MISCNT contains the address of the current MISC memory access. This registers is read-only. Note that the naming of the A[31:0] bits represents little-endian bit encoding.

Exiting TPU3 emulation mode or clearing the MISEN bit in the DPTMCR results in the reset of this register.

MISCNT — MISC Counter **0xYF F88A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	RESERVED		

RESET:

Last Memory Address

9.5 Operation

The DPTRAM module has several modes of operation. The following sections describe DPTRAM operation in each of these modes.

9.5.1 Normal Operation

In normal operation, the DPTRAM is powered by V_{DDL} and may be accessed via the IMB3 by a bus master.

Read or write accesses of 8, 16, or 32 bits are supported. In normal operation, neither TPU3 accesses the array, nor do they have any effect on the operation of the DPTRAM module.

9.5.2 Standby Operation

The DPTRAM on the MC68F375 does not support standby operation. The VDDDPTRAM should always be powered up and down with VDDL.

9.5.3 Reset Operation

When a synchronous reset occurs, a bus master is allowed to complete the current access. Thus a write bus cycle (byte or half word) that is in progress when a synchronous reset occurs will be completed without error. Once a write already in progress has been completed, further writes to the RAM array are inhibited.

NOTE

A word (32-bit) write will be completed coherently only if the reset occurs during the second (16-bit) write bus cycle. If reset occurs during the first write bus cycle, only the first half word will be written to the RAM array and the second write will not be allowed to occur. In this case, the word data contained in the DPTRAM will not be coherent. The first half word will contain the most significant half of the new word information and the second half word will contain the least significant half of the old word information.

If a reset is generated by an asynchronous reset such as the loss of clocks or software watchdog time-out, the contents of the RAM array are not guaranteed. (Refer to **4.7 Reset** for a description of the MC68F375 reset sources, operation, control, and status.)

Reset will also reconfigure some of the fields and bits in the DPTRAM control registers to their default reset state. See the description of the control registers to determine the effect of reset on these registers.

9.5.4 Stop Operation

Setting the STOP control bit in the DPTMCR causes the module to enter its lowest power-consuming state. The DPTMCR can still be written to allow the STOP control bit to be cleared.

In stop mode, the DPTRAM array cannot be read or written. All data in the array is retained. The BIU continues to operate to allow the CPU to access the STOP bit in the DPTMCR. The system clock remains stopped until the STOP bit is cleared or the DPTRAM module is reset.

The STOP bit is initialized to logical zero during reset. Only the STOP bit in the DPTMCR can be accessed while the STOP bit is asserted. Accesses to other DPTRAM registers may result in unpredictable behavior. Note also that the STOP bit should be set and cleared independently of the other control bits in this register to guarantee proper operation. Changing the state of other bits while changing the state of the STOP bit may result in unpredictable behavior.

9.5.5 Freeze Operation

The FREEZE line on the IMB3 has no effect on the DPTRAM module. When the freeze line is set, the DPTRAM module will operate in its current mode of operation. If the DPTRAM module is not disabled, (RAMDS = 0), it may be accessed via the IMB3. If the DPTRAM array is being used by the TPU in emulation mode, the DPTRAM will still be able to be accessed by the TPU microengine.

9.5.6 TPU3 Emulation Mode Operation

To emulate TPU3 time functions, the user stores the microinstructions required for all time functions to be used, in the RAM array. This must be done with the DPTRAM in its normal operating mode and accessible from the IMB3. After the time functions are

stored in the array, the user places one or both of the TPU3 units in emulation mode. The RAM array is then controlled by the TPU3 units and disconnected from the IMB3.

To use the DPTRAM for microcode accesses, set the EMU bit in the corresponding TPU3 module configuration register. Through the auxiliary buses, the TPU3 units can access word instructions simultaneously at a rate of up to 40 MHz.

When the RAM array is being used by either or both of the TPU3 units, all accesses via the IMB3 are disabled. The control registers have no effect on the RAM array. Accesses to the array are ignored, allowing an external RAM to replace the function of the general-purpose RAM array.

The contents of the RAM are validated using a multiple input signature calculator (MISC). MISC reads of the RAM are performed only when the MC68F375 is in emulation mode and the MISC is enabled (MISEN = 1 in the DPTMCR).

Refer to **8.3.6 Emulation Support** for more information in TPU3 and DPTRAM operation in emulation mode.

9.6 Multiple Input Signature Calculator (MISC)

The integrity of the RAM data is ensured through the use of a MISC. The RAM data is read in reverse address order and a unique 32-bit signature is generated based on the output of these reads. MISC reads are performed when one of the TPU3 modules does not request back-to-back accesses to the RAM provided that the MISEN bit in the MC68F375 MCR is set.

The MISC generates the DPTRAM signature based on the following polynomial:

$$G(x) = 1 + x + x^2 + x^{22} + x^{31}$$

After the entire RAM has been read and a signature has been calculated, the MISC sets the MISF bit in the MC68F375 MCR. The host should poll this bit and enter a handling routine when the bit is found to be set.

The signature should be then read from the MISRH and MISRL registers and the host determines if it matches the predetermined signature.

The MISRH and MISRL registers are updated each time the MISC completes reading the entire RAM regardless of whether or not the previous signature has been read or not. This ensures that the host reads the most recently generated signature.

The MISC can be disabled by clearing the MISEN bit in the MC68F375 MCR. Note that the reset state of the MC68F375 MISEN is disabled.

SECTION 10
CDR MoneT FLASH FOR THE IMB3 (CMFI)

10.1 Overview

The CDR MoneT FLASH for the IMB3 (CMFI) is designed to be used with the Inter-module Bus 3 (IMB3) and consequently any bus master capable of operating the IMB3.

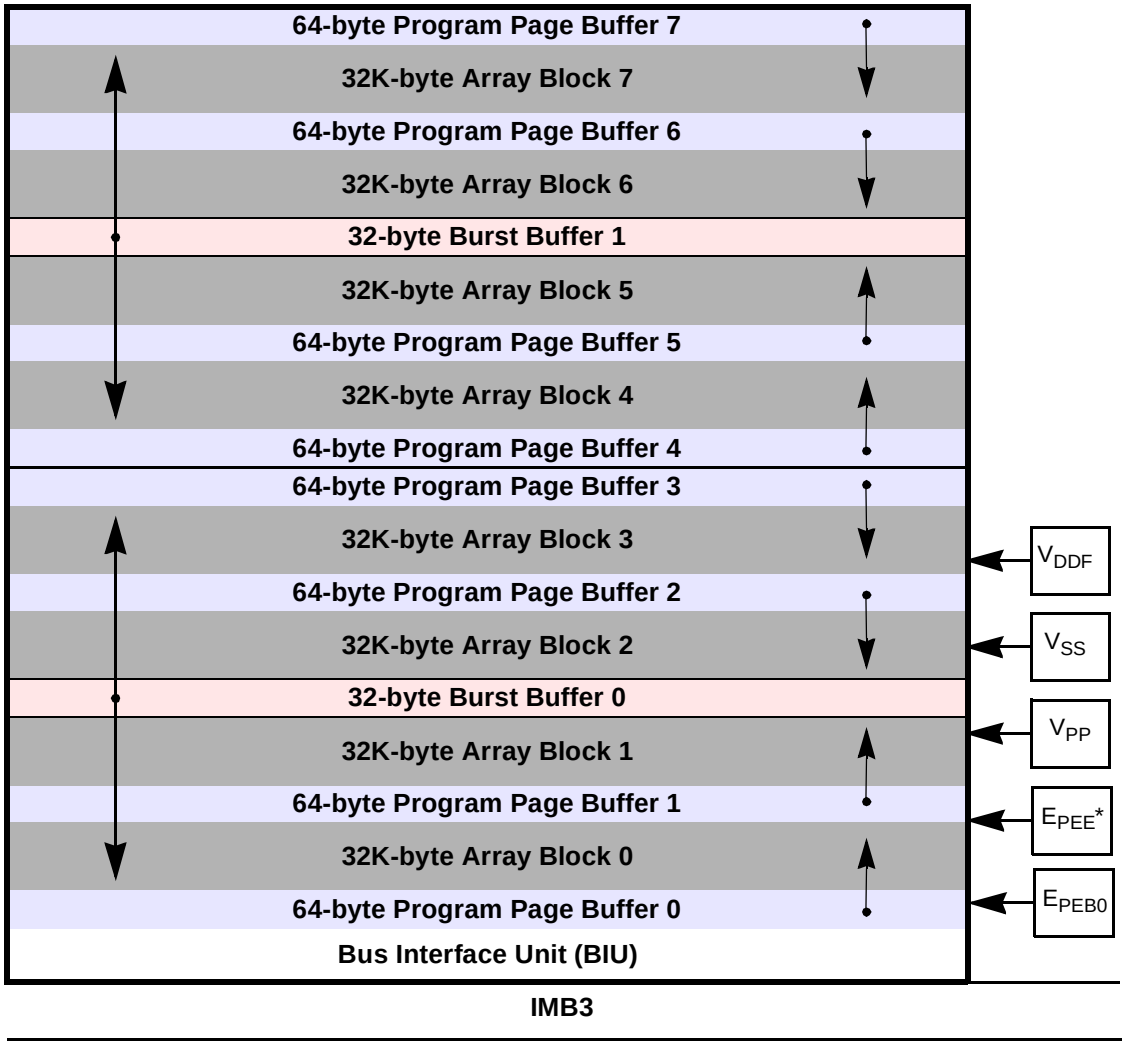
The CMFI array uses the MOTOROLA 1 transistor (MoneT) bit cell configured as 256 Kbytes (262,144 bytes) of non-volatile memory (NVM). The CMFI array is divided into eight 32-Kbyte (32,768-bytes) array blocks.

10.1.1 Overview Description

The primary function of the CMFI EEPROM module is to serve as electrically programmable and erasable NVM to store program instructions and/or data. It is a class of non-volatile solid state silicon memory devices consisting of an array of isolated elements, a means for selectively adding and removing electrical charge to the elements and a means of selectively sensing the stored charge in the elements. When power is removed from the device, the stored charge of the isolated elements will be retained.

The CMFI EEPROM module is arranged into two major sections as shown in **Figure 10-1**. The first section is the MoneT array used to store system programs and data. The second section is the bus interface unit (BIU) that controls access and operation of the CMFI array through a standard IMB3 interface and external signals for:

- E_{PEB0} – Block 0 protect signal
- E_{PEE} - Program enable (note: The EPEE pin is not available on the MC68F375 and is always enabled.)
- V_{PP} – Supplying program and erase power.
- V_{DDF} - Flash operating voltage



* The E_{PEE} pin does not exist on the MC68F375 and is always enabled.

Figure 10-1 Block Diagram for a CMFI EEPROM in the 256-Kbyte Configuration.

The CMFI EEPROM module array is divided into array blocks to allow for independent block erase and multiple block programming. The size of an array block in the CMFI module is a fixed 32 Kbytes. The total CMFI EEPROM array is distributed into 8 blocks. Information is transferred to the CMFI EEPROM through the IMB3 by a long-word (32 bits).

To improve system performance, the BIU accesses information in the array at 32 bytes per access. These 32 bytes are copied into a burst buffer aligned to the low order addresses, IADDR[4:0]. The CMFI contains two non-overlapping burst buffers. The first burst buffer is associated to the lower array blocks. The second burst buffer is associated with the higher array blocks. Read access time of the data in the current burst buffers is 1 system clock, while the time to copy new data into a burst buffer and

access the required information is 2 system clocks. Reads will always begin with a 2 clock access. If the IMB3 indicates a burst access the following access(es) will be 1 clock until the CMFI reaches the end of the burst buffer or the IMB3 terminates the burst access. During the burst reads, the CMFI increments the address by one word each access. The end of the burst buffer is indicated by the highest location within the burst buffer being read, ADDR[4:0] = 0x1F. All burst accesses are aligned to the IMB3 data bus, ignoring the byte address(es). To prevent the BIU from unnecessarily accessing the array, the CMFI EEPROM shall monitor the IMB3 address to determine if the required information is in one of the two current burst buffers and the access is valid for the module. This process is designed to reduce power consumption by the CMFI.

In normal operation write accesses to the CMFI array are not recognized.

The CMFI EEPROM module requires an external program or erase voltage, V_{PP}, to program or erase the array or any of its control register shadow bits. Special control logic is included to require a specific series of read and write accesses before program or erase operation is allowed.

To improve program performance, the CMFI programs up to eight unique 64-byte pages simultaneously in eight separate array blocks. These 64 bytes are aligned to the low order addresses, IADDR[5:0], to form a program page buffer. Each of the pages being programmed simultaneously are located at the same block offset address, IADDR[23:15|14]. Erase is performed on one or more of the selected array blocks simultaneously.

10.1.2 Features of the CMFI

- MOTOROLA's 1 transistor, MoneT, FLASH bit cell.
- -40 to 125° C operating temperature range.
- V_{DD} 3. 0 V to 3. 6 V operating range.
  - — Operational at 2. 7 V**.**
  - — Up to 40 MHz operation at V_{DD} = 3. 0 V, 150° C = T_J.
- Shadow and bootstrap information stored in special FLASH NVM locations.
- 256-Kbyte array size.
- Array distributed in 8 blocks.
  - — Erase by array block(s).
  - — Common array block size of 32 Kbytes.
  - — Array lock protection for program and erase operations.
    - Built-in margin reads for both program and erase verify reads.
    - Array access disabled while programming or erasing.
  - — Array address attributes restriction control.
    - Select between supervisor and supervisor/user spaces.
    - Select between data and instruction/data spaces.
- Program up to 512 bytes at a time.
  - — Program up to eight 64-byte pages simultaneously.
  - — Pages located at the same offset address.
- Self-timed program and erase pulses.
  - — Internal pulse width timing control using system clock frequencies from 8.0

MHz to 40.0 MHz.
  — Program pulses from 4.0 µs to 2.73 ms.
  — Erase pulses from 4.096 ms to 2.796 s.
- External 4.75 to 5.25 V V_{PP} program and erase power supply.
- Array block 0 enable is selected from one of two sources:
  — A pin external to the device (EPEB0)
  — The inverted state of the CMFI PROTECT bit.
- Data word length of 16 bits.
- Supports IMB3 burst read accesses.
  — Contains two separate non-sequential burst buffers.
  — Burst buffer size of 32 bytes.
  — Burst terminated at end of buffer or by IMB3.
- Software mapping to establish array base address.
- Emulation support
  — Support for integration modules with external emulation through a special chip select.
  — Supports internal emulation through memory overlay option.
- Low power disable via the integration module.
- Wait states for integration from slower external memory.

10.1.3 Glossary of terms used in the CMFI EEPROM Specification

Array block — CMFI array subdivision: a 32-Kbyte contiguous block of information. Each array block may be erased independently.

BIU — Bus interface unit controls access and operation of the CMFI array through a standard IMB3 interface.

Burst read — Array read operation that requires 2 clocks for the first data access and 1 clock for the following data accesses.

CMFI — The CDR MoneT FLASH EEPROM for the IMB3.

Erase interlock write — A write to any CMFI array address after initializing the erase sequence.

Erase margin read — Special burst buffer updates of the CMFI array where the CMFI EEPROM hardware adjusts the reference of the sense amplifier to check for correct erase operation. All CMFI array burst buffer updates between the erase interlock write and clearing the SES bit are erase margin reads.

IM — Integration module.

IMB3 — A motherboard-on-a-chip for embedded controller designs.

Notable features of the bus architecture include: burst data transfers, multiple bus masters, exception processing support, address space partitioning, multiple interrupt levels, vectored interrupts, and extendable bus cycles via wait state insertion.

The IMB3 provides a flexible, high performance bus capable of supporting a family of parts.

Initialize program/erase sequence — The write to the high voltage control register that changes the SES bit from a 0 to a 1.

Master reset — The hardware reset that resets the entire CMFI.

MoneT — The CMFI EEPROM's FLASH bit cell.

Over programmed — By exceeding the specified programming time and/or voltage, a CMFI bit may be over programmed. This bit causes erased bits in the same column on the same array block to read as programmed.

Programming write — A word write to a CMFI array address to transfer information into a program page buffer. The CMFI EEPROM accepts programming writes after initializing the program sequence until the EHV bit is changed from a 0 to a 1.

Program margin read — Special burst buffer updates of the CMFI array where the CMFI EEPROM hardware adjusts the reference of the sense amplifier to check for correct program operation. All CMFI burst buffer updates between the first programming write and clearing the SES bit are program margin reads.

Program page buffer — 64 bytes of information used to program the CMFI array. This information is aligned to a 64-byte boundary within the CMFI array. Each CMFI module has 1 program page buffer per block.

Read burst buffer — 32-byte block of information that is read from the CMFI array. This information is aligned to a 32-byte boundary within the CMFI array. Each CMFI module has two non-sequential burst buffers.

Reserved registers — A location within the control register block which may have one or more bits that are reserved for use by Motorola. These bits are not available for normal use.

Shadow information — An extra row (256 bytes) of the CMFI array used to provide reset configuration information. This row may be accessed by setting the SIE bit in the module configuration register and accessing the CMFI array, see **10.4.3 CMFI EEPROM Configuration Register (CMFIMCR)**. The shadow information is always in the lowest array block of the CMFI array.

System reset — A reset generated under software control that clears the high voltage enable (EHV) bit of the CMFICTL register and forces the BIU into a state ready to receive a new IMB3 access.

10.2 CMFI EEPROM Interface

The CMFI module contains a slave BIU to the IMB3. The BIU controls access and operation of the array through standard IMB3 reads and writes of the array and register blocks in the CMFI module. Additionally, the CMFI uses external signals to provide control and power to the module. These other external signals include an optional sig-

nal to externally control program or erase operations to array block 0 (E_{PEB0}). Three other pins: V_{SSF} V_{DD3F} and V_{PP} provide power to the module.

10.2.1 External Interface

The CMFI EEPROM module uses external signals to provide some external control of operations and provide power. These signals are listed in **Table 10-1**.

Table 10-1 CMFI EEPROM Module External Signals

Description	MNEMONIC	Comments
Internal memory patch signals	INTPATCHB	These signals signify that the current bus cycle will be provided by the internal patch memory instead of the CMFI. The CMFI EEPROM BIU will force an aborted access to the CMFI if any of the four patch signals = “1” to remain synchronized with the IMB3. Thus the CMFI will not assert the data, data transfer acknowledge or burst transfer acknowledge if any of the internal memory patch signals = “1”.
Master program and erase enable	E_{PEE}	This optional signal externally controls program or erase operation. To enable these operations the signal should be at the logic "1” level, while a logic "0” level disables these operations in the CMFI module. The E_{PEE} signal includes a pull down device to keep a logic "0” unless the pin is driven to a logic "1” and a digital filter to protect from external noise. On the MC68F375, this signal is connected to V_{DD} to allow program and erase operations at all times.
Block 0 program and erase enable	E_{PEB0}	This optional signal will externally control program or erase operations to array block 0. To enable these operations the signal should be at the logic “1” level, while a logic “0” level disables these operations in the CMFI Module. This signal may be generated by any desired method and must remain valid throughout the program or erase software starting their respective operation. The E_{PEB0} pin will include a pull down device to keep a logic “0” unless the pin is driven to a logic "1” and a digital filter to protect from external noise. When not connected to E_{PEB0} this signal will be connected to V_{DD} to allow program and erase operations.
CMFI ground	V_{SSF}	To reduce noise in the read path no other circuits should be connected to the CMFI V_{SSF} supply. A maximum of 2 CMFI Modules may be connected to a V_{SSF} supply pin. This V_{SSF} pin must be isolated from all other V_{SS} pins inside the device.
CMFI power supply	V_{DDF}	To reduce noise in the read path no other circuits should be connected to the CMFI V_{DDF} supply pin. A maximum of 2 CMFI Modules may be connected to a V_{DDF} supply pin. This V_{DDF} pin must be isolated from all other V_{DD} pins inside the device. The specified voltage range during operation is 3. 0 V to 3. 6 V.
Program and erase high voltage supply	V_{PP}	V_{PP} provides the high voltage (4. 75 V to 5. 25 V) used during program and erase operations of the CMFI Module. A maximum of 2 CMFI Modules may be connected to a V_{PP} supply pin. The suggested voltage at the V_{PP} pin should be equal to the V_{DD} voltage during all operations except program and erase.

10.3 Programmer's Model

The CMFI EEPROM module consists of two addressed sections. The first is the 32-byte control registers section used to configure, program, erase and test the CMFI EEPROM array, while the second is the array. **Table 10-2** represents how the IMB3 addresses correspond to the CMFI EEPROM control register and array mapping.

Table 10-2 CMFI EEPROM Memory Map with 32-Bit Word

<table>
<tr><th>Control Register</th><th colspan="3">32-Kbyte Blocks</th><th>IMB3 Address</th></tr>
<tr><td rowspan="19">Control Register Hardware Mapping Addresses</td><td colspan="3" rowspan="5">Array Mapping Addresses</td><td>23</td></tr>
<tr><td>22</td></tr>
<tr><td>21</td></tr>
<tr><td>20</td></tr>
<tr><td>19</td></tr>
<tr><td colspan="2"></td><td>18</td></tr>
<tr><td colspan="2"></td><td>Array Hardware Mapping or Block Address[1]</td><td>17</td></tr>
<tr><td colspan="2">Array Hardware Mapping or Block Address[1]</td><td rowspan="2">Block Addresses</td><td>16</td></tr>
<tr><td colspan="2" rowspan="2">Block Addresses</td><td>15</td></tr>
<tr><td rowspan="7">Row Addresses</td><td>14</td></tr>
<tr><td colspan="2" rowspan="6">Row Addresses</td><td>13</td></tr>
<tr><td>12</td></tr>
<tr><td>11</td></tr>
<tr><td>10</td></tr>
<tr><td>9</td></tr>
<tr><td>8</td></tr>
<tr><td colspan="3" rowspan="2">Column Addresses</td><td>7</td></tr>
<tr><td>6</td></tr>
<tr><td>32-Byte Read Page Select</td><td rowspan="4">Program Page Word Addresses</td><td>32-Byte Read Page Select</td><td>5</td></tr>
<tr><td rowspan="3">Control Register Select Addresses</td><td rowspan="3">Read Burst Buffer Word Addresses</td><td rowspan="3">Read Burst Buffer Word Addresses</td><td>4</td></tr>
<tr><td>3</td></tr>
<tr><td>2</td></tr>
<tr><td colspan="4" rowspan="2">Byte Addresses</td><td>1</td></tr>
<tr><td>0</td></tr>
</table>

NOTES:
1. The high order address of the block addresses selects the read burst buffer.

10.4 CMFI EEPROM Control Block

The 32-byte control block contains registers which are used to control CMFI EEPROM module operation. Configuration information is specified and programmed independently from the contents of the CMFI EEPROM array. There are three registers provided for configuration and control of the CMFI EEPROM module:

- The module configuration register (CMFIMCR)
- Array base address register (CMFIBAR)
- CMFI EEPROM high voltage control register (CMFICTLx).

Control bits in these registers are provided to control array operation, programming and erasing.

Some of the control registers have shadow information words which physically exist in a spare CMFI EEPROM row. On master reset, some of the registers and fields within certain registers are loaded with default reset information from the shadow information words. Writing to a register does not alter the contents of the corresponding shadow information word. Using the address of the corresponding control register, the shadow information word is programmed in the same manner as a location in the CMFI EEPROM array. When data is latched into the programming latches while programming a shadow information word, it will not be written to the register itself. Data which is programmed into the CMFIMCR, CMFIBAR or CMFICTL register's shadow information word will not be copied into the register until the next master reset.

The last write to a programming buffer prior to setting EHV determines the value to be programmed. The registers that are loaded from shadow information words during master reset are identified in the individual register field and control bit descriptions. The shadow information words are erased whenever the low block (block 0) of the array is erased.

10.4.1 CMFI EEPROM Module Control Block Addressing

The module control block is addressed by comparing the module control mapping (IMODMAP) to IADDR[23] while IADDR[22:5] are decoded by the CMFI EEPROM module. If the CMFI EEPROM control block address is decoded, the CMFI EEPROM module will assert the address acknowledge (IAACKB) signal. The value of IADDR[22:5] is defined for each device that has a CMFI EEPROM module. These bits are fixed for a particular device (MCU or peripheral), and are specified by Motorola. The value of the module control mapping (IMODMAP) is specified by a control bit in the module configuration register, see **10.4.3 CMFI EEPROM Configuration Register (CMFIMCR)**. The exact register addressing within the control block is determined by IADDR[4:0]. The control block is restricted to supervisor data space (IFC[2:0] = 0b101). Any other address space read or write of the registers will not assert address acknowledge. See **Table 10-3** for control register offset addresses.

Table 10-3 CMFI Control Register Addressing

Address	Control Register	
0xYF F800[1]	Module Configuration (CMFIMCR)	
0xYF F802	RESERVED	
0xYF F804	CMFITST	
0xYF F806	RESERVED	
0xYF F808	CMFIBAR	Base Address High (CMFIBAH)
0xYF F80A		Base Address Low (CMFIBAL)
0xYF F80C	CMFICTL	High Voltage Control 1 (CMFICTL1)
0xYF F80E		High Voltage Control 2 (CMFICTL2)
0xYF F810 to 0xYF F816	CMFIBS[3:0]	
0xYF F818 to 0xYF F81E	RESERVED	

Registers with shadow information word (erased bit state: 0b1).

NOTES:

1. The "Y" in the address is determined by the state of the MM bit in the SCIMMCR.

10.4.2 Reserved Register Accesses

The IMB3 does not support accesses to reserved registers. An internal BERR protocol will terminate with a bus error for all accesses to a reserved register. Also, the data returned by the BIU is undefined and a write access will have no effect for a reserved register.

10.4.3 CMFI EEPROM Configuration Register (CMFIMCR)

The CMFI EEPROM module configuration register is used to control the operation of the CMFI EEPROM array and the BIU.

CMFIMCR — CMFI EEPROM Configuration Register **0xYF F800**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	PROTECT	SIE	$\overline{\text{BOOT}}$	$\overline{\text{LOCK}}$	EMUL	ASPC		WAIT		0	0	0	0	0	0
RESET:[1]															
U[2]	1	0	U[3]	1	U[4]	U[5]		U[6]		—	—	—	—	—	—

NOTES:

1. The default values of some bits in the CMFIMCR are read from the location 0 of the shadow row.
2. Reset state is defined by a shadow bit **or** the state of D15 during reset mode configuration.
3. Reset state is defined by a shadow bit.
4. Reset state is defined by D[10] or the state of D[13] during reset mode configuration.
5. Reset state is defined by a shadow bit, bit is write protected by $\overline{\text{LOCK}}$ and STOP.
6. Reset state is defined by a shadow bit, bit is write protected by $\overline{\text{LOCK}}$.

Table 10-4 CMFIMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Stop control. When the STOP control bit is a 1, the CMFI EEPROM array is disabled. It will not respond to the base address stored in CMFIBAR. STOP will prevent read accesses to the array and to the shadow information words, but has no effect on accesses to the control registers. Attempts to read any shadow information word while STOP = 1 will produce indeterminate results. With STOP = 1 the CMFI may enter the lower power clock stop operation, see **4.4.9 Low Power Stop Mode**. If STOP is set during programming or erasing, the program and erase voltage will automatically be turned off by clearing the EHV bit. The state of this bit after master reset is the logical OR of the inverted state of D[15] and the STOP shadow bit, STOP = D[15] or STOP shadow bit. If STOP is set to a 1 by D[15] or the STOP shadow bit during master reset, the array may be re-enabled by clearing STOP after master reset. This bit is read/write always. 0 = The CMFI EEPROM module is in normal mode of operation. 1 = Causes the CMFI EEPROM module to enter low power STOP operation.
14	PROTECT	Prevent array program/erase. The CMFI EEPROM array and shadow information are protected from program and erase operation by setting PROTECT = 1. The CMFI BIU will perform all programming and erase interlocks except the program and erase voltages will not be applied to locations within the array if PROTECT = 1 Read always, Write when $\overline{\text{LOCK}}$ = 1 and SES = 0. 0 = All NVM bits are unprotected. 1 = All NVM bits are protected.

Table 10-4 CMFIMCR Bit Settings (Continued)

Bit(s)	Name	Description
13	SIE	Shadow information enable. The SIE bit is write protected by the start end sequence (SES) bit for programming operation. Writes will have no effect if SES = 1 and PE = 0. The SIE bit can be read whenever the registers are enabled. When an array location is read in this mode, the shadow information will be read from a location determined by the column, 32-byte read page select, and word addresses (IADDR[7:0]) of the access. Accessing the CMFI control block registers will access the registers and not the shadow information. The default reset state of SIE is normal array access (SIE = 0). The address range of the shadow information is the entire address range of the CMFI EEPROM array but the high order array addresses, IADDR[17|16|15:7], are not used to encode the location. The first 32 bytes (IADDR[7:0] = 0x00 to 0x1F) of the 256 bytes of shadow locations are withheld by Motorola for the register shadow information words. The remaining 224 bytes are available for general use. This is shown in **Figure 10-3**. The upper address bits (IADDR[7:6]) are forced to 0 during reset. When SIE = 1, only the program page buffer associated with the lowest block can be programmed. The other program page buffers cannot be accessed and will not apply any programming voltages to their CMFI array blocks while programming the shadow information. The shadow information is typically in block 0 except for when the 192K-byte and 96K-byte arrays are mapped high, then the shadow information is in block 2. 0 = Normal array access. 1 = Disables normal array access and selects the shadow information.
12	$\overline{\text{BOOT}}$	Boot control. After reset, the $\overline{\text{BOOT}}$ bit may be cleared or set via a write to CMFIMCR; however, it will not affect bootstrap operation. If the STOP bit is set (STOP = 1) then bootstrap operation will be terminated. While the CMFI EEPROM is configured to provide the bootstrap information and read access to the control block, the CMFI array will not provide correct data. Control block writes will not be affected by bootstrap operation. 0 = The CMFI will respond to bootstrap address after reset. 1 = The CMFI will not respond to bootstrap address after reset.
11	$\overline{\text{LOCK}}$	Lock control. In normal operation once the $\overline{\text{LOCK}}$ bit is asserted ($\overline{\text{LOCK}}$ = 0) the write-lock can only be disabled again by a master reset. The $\overline{\text{LOCK}}$ bit is writable if the device is in background debug mode (IFREEZEB = 0). When the $\overline{\text{LOCK}}$ control bit in the CMFIMCR register is asserted ($\overline{\text{LOCK}}$ = 0) the write-lock register bits ASPC, WAIT, PROTECT, EMUL and CMFIBAH are locked. Writes to these bits will have no effect. Read always, clear once unless in background debug mode. 0 = Write-locked registers are protected. 1 = Write-lock is disabled.
10	EMUL	Emulation operation. When the EMUL control bit in the CMFIMCR register is a 1, the CMFI EEPROM is placed in emulation operation. Emulation operation may be entered by writing EMUL to a 1 on devices that support emulation operation; otherwise, writes have no operational effect. The state of this bit after master reset is the logical NOR of EMULIN and EMULEN, EMUL = $\overline{\text{EMULIN}}$ and $\overline{\text{EMULEN}}$. Emulation operation allows the array to be emulated externally, with access controlled by the CMFI EEPROM.
9:8	ASPC	Array space. The array may be specified to exist in supervisor or unrestricted space. The default reset state of ASPC is programmed in a CMFI EEPROM shadow bit by the user. ASPC may be written by the bus master any time STOP = 1 and $\overline{\text{LOCK}}$ = 1. The ASPC bits govern accesses to the array, but have no effect on how control registers and shadow registers are accessed. 00 = Unrestricted data space (IFC = x01), unrestricted program space (IFC = x10) 01 = Unrestricted program space (IFC = x10) 10 = Supervisor data space (IFC = 101), supervisor program space (IFC = 110) 11 = Supervisor program space (IFC = 110)

Table 10-4 CMFIMCR Bit Settings (Continued)

Bit(s)	Name	Description
7:6	WAIT	Wait states. The WAIT field is used to specify the number of wait states inserted by the BIU during accesses. These wait states are added to the bus cycle between the IMB3 asserting data strobe (IDSB) and the CMFI EEPROM writing or reading data. For burst accesses, the wait states are inserted for the first data access only. A wait state has a duration of one system clock cycle. This feature allows the migration of storage space from a slower emulation or development system memory to the MC68F375 without the need for re-timing the system. The program and erase margin reads will extend the bus cycle to their respective timings regardless of the value of WAIT. Read always, writable if $\overline{\text{LOCK}}$ = 1. 00 = Minimum bus cycles = 3 clocks, 1 inserted wait states 01 = Minimum bus cycles = 4 clocks, 2 inserted wait states 10 = Minimum bus cycles = 5 clocks, 3 inserted wait states 11 = Minimum bus cycles = 2 clocks, 0 inserted wait states
5:0	—	Reserved

Figure 10-2

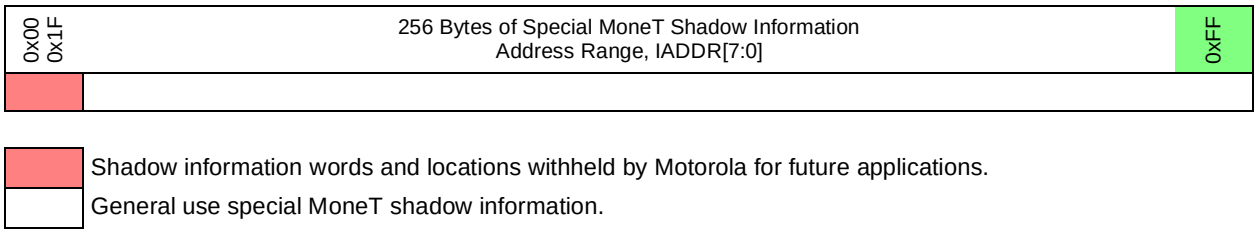


Figure 10-3 Shadow Information

WARNING

If a CMFI EEPROM enables the lock protection mechanism ($\overline{\text{LOCK}}$ = 0) before PROTECT is cleared the device must use background debug mode (IFREEZEB = 0) to program or erase the CMFI EEPROM.

10.4.4 CMFI EEPROM Test Register (CMFITST)

The CMFI EEPROM test register is used to control the test operation of the CMFI EEPROM array and BIU. Only 6 bits are read/writeable in the CMFITST register (in supervisor mode only).

CMFITST — CMFI EEPROM Test Register **0xYF F804**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED				NVR[1]	PAWS[2]			RESERVED	STE[1,3]	GDB[1]	RESERVED				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NOTES:
1. The NVR, STE, and GDB bits are not accessible in all revisions of the MC68F375 (prior to the J61X mask set).
2. The PAWS bits are not accessible in all revisions of the MC68F375.
3. The STE bit should always be programmed as a 0.

Table 10-5 CMFITST Bit Settings

Bit(s)	Name	Description
15:12	—	Reserved
11	NVR	Negative voltage range. This bit switches between the low and high voltage range of the negative charge pump in programming and erasing the CMF Flash module when GDB = 0b1. This bit is writeable when HVS = 0b0. 0 = High Range (more negative) 1 = Low Range
10:8	PAWS[0:2]	Program amplitude/width modulation select. The PAWS bits can be used to select the programming voltage applied to the drain of the EEPROM bitcell. These bits should be left-set to 000. For information about PAWS programming modes, see **Table 10-6**.
7	—	Reserved
6	STE	This bit is reserved for Motorola factory testing and should always be programmed to 0b0. 0 = Normal Operation 1 = Factory test mode use only. This setting could disturb contents of flash
5	GDB	Gate/drain bias select. This bit works in conjunction with the PAWS bits to select between positive and negative ramped voltages for programing and erasing. This bit is writeable when SES = 0b0. 0 = Positive voltage ramp selected on the bitcell drain 1 = Negative voltage ramp selected on the bitcell gate
4:0	—	Reserved

Table 10-6 CMF Programming Algorithm (v6 and Later)

No. of Pulses	Pulse Width	NVR	PAWs	GDB	PAWs Mode	Description
4	256 µs	1	100	1	Mode 4NL	Negative gate ramp (low range)
4	256 µs	1	101	1	Mode 5NL	
4	256 µs	1	110	1	Mode 6NL	
4	256 µs	1	111	1	Mode 7NL	
20	50 µs	0	100	1	Mode 4NL	Negative gate ramp (high range)
20	50 µs	0	101	1	Mode 5NL	
20	50 µs	0	110	1	Mode 6NL	
max. 10,000	50 µs	0	111	1	Mode 7NL	

Table 10-7 CMF Erase Algorithm (v6)

No. of Pulses	Pulse Width	NVR	PAWs	GDB	PAWs Mode	Description
1	100 ms[1]	1	100	1	Mode 4NL	Negative gate ramp (low range)
1	100 ms[1]	1	101	1	Mode 5NL	
1	100 ms[1]	1	110	1	Mode 6NL	
1	100 ms[1]	1	111	1	Mode 7NL	
1	100 ms[1]	0	100	1	Mode 4NL	Negative gate ramp (high range)
1	100 ms[1]	0	101	1	Mode 5NL	
1	100 ms[1]	0	110	1	Mode 6NL	
20	100 ms[2]	0	111	1	Mode 7NL	

NOTES:
1. No margin read after pulse.
2. Do margin read after each pulse.

10.4.5 CMFI Base Address Registers (CMFIBAR)

The CMFI base address register is used to set the base address of the CMFI flash module. It consists of two 16-bit registers, CMFIBAH and CMFIBAL. On a 256-Kbyte CMFI module, the base address of the module should be on a 256-Kbyte boundary, therefore CMFIBAH[7:16] and CMFIBAL[15:12] should be set to zero.

CMFIBAH — CMFI Base Address High Register **0xYF F808**

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	LSB 16
RESERVED								**CMFIBAH[1]**							
RESET:															
0	0	0	0	0	0	0	0	U	U	U	U	U	U	U	U

NOTES:

1. Indicates bits protected by LOCK and STOP. The default state of these bits is read from address 0x000008 of the shadow row on reset.

CMFIBAL — CMFI Base Address Low Register **0xYF F80A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
CMFIBAL[1]				RESERVED											
RESET:															
U	U	U	U	0	0	0	0	0	0	0	0	0	0	0	0

NOTES:

1. Indicates bits protected by $\overline{\text{LOCK}}$ and STOP. The default state of these bits is read from address 0x000008 of the shadow row on reset.

Table 10-8 CMFIBAR (CMFIBAH, CMFIBAL) Bit Settings

Bit(s)	Name	Description
31:24	—	Reserved
23:16	CMFIBAH	The 32-bit base address of the CMFI array memory address block is contained in the CMFIBAH and CMFIBAL array base address registers. The base address register (CMFIBAR) is formed by concatenating the contents of CMFIBAH and CMFIBAL. CMFIBAH contains the high order 16 bits of the address (A[31:16]) and CMFIBAL contains the next lower order bits. The value of CMFIBAH and CMFIBAL are forced to the user defined value programmed in CMFI shadow registers on master RESET. CMFIBAH and CMFIBAL can be written to relocate the CMFI array to an alternate block of memory only when the $\overline{\text{LOCK}}$ bit is 1 and the CMFI module is in STOP mode. NOTE: For a 256K module, CMFIBAH[17:16] and CMFIBAL[15:12] should be programmed to 0.
15:12	CMFIBAL	
11:0	—	Reserved

10.4.6 High Voltage Control Register

The high voltage control register is used to control the program and erase operations of the CMFI array.

CMFICTL1 — CMFI High Voltage Control Register 1 **0xYF F80C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
HVS	0	SCLKR[1]			0	CLKPE[1]		0	CLKPM[1]						
RESET:															
0	0	U[2]			—	U[2]		—	U[2]						

NOTES:
1. These fields are NOT locked by SES if the value of the PAWS bits (in CMFITST) is not 0b000.
2. The default state of these bits will be read from the shadow row location 0xC on reset.

Table 10-9 CMFICTL1 Bit Settings

Bit(s)	Name	Description
15	HVS	High voltage status. The HVS bit is for status only and writes will have no effect. During a program or erase pulse this bit will be a 1 while the pulse is active or during recovery. The BIU will not acknowledge (IAACKB not asserted) an access to an array location if HVS = 1. While HVS = 1 SES cannot be changed and the CMFI cannot enter low power clock stop operation. The program or erase pulse becomes active by setting the EHV bit and is terminated by clearing EHV or by the pulse width timing control. The recovery time is the time that the CMFI EEPROM requires to remove the program or erase voltage from the array or shadow information before switching to another mode of operation. The recovery time is determined by the system clock range (SCLKR[0:2]) and the PE bit. The recovery time is 48 of the scaled clock periods unless SCLKR = 0 then the recovery time is 128 clocks. Once master reset is completed HVS shall indicate no program or erase pulse (HVS = 0). 0 = Program or erase pulse is not applied to the CMFI. 1 = Program or erase pulse is applied to the CMFI.
14	—	Reserved
13:11	SCLKR	System clock range. The SCLKR bits are write protected by the SES bit. Writes to CMFICTL will not change SCLKR if SES = 1. The first term of the timing control is the clock scaling, R. The value of R is determined by the system clock range (SCLKR). SCLKR defines the pulse timer's base clock using the system clock. The following table should be used to set SCLKR based upon the system clock frequency. The system clock period is multiplied by the clock scaling value to generate a 83. 3 ns to 125 ns scaled clock. This scaled clock is used to run the charge pump submodule and the next functional block of the timing control. See **Table 10-11** for SCLKR settings.
10	—	Reserved
9:8	CLKPE	Clock period exponent. The CLKPE[1:0] bits are write protected by the SES bit. Writes to CMFICTL will not change CLKPE[1:0] if SES = 1. The second term of the timing control is the exponential clock multiplier, N. The program pulse number (pulse), clock period exponent (CLKPE[1:0]) CSC, and PE define the exponent in the 2^N multiply of the clock period. The exponent, N, is defined by the equation: $N = 5 + CLKPE[1:0] + (PE \bullet 10)$ See **Table 10-12** for the range of exponents.

Table 10-9 CMFICTL1 Bit Settings (Continued)

Bit(s)	Name	Description
7	—	Reserved
6:0	CLKPM	Clock period multiple select. The CLKPM[6:0] bits are write protected by the SES bit. Writes to CMFICTL will not change CLKPM[6:0] if SES = 1. The third term of the timing control is the linear clock multiplier, M. The clock period multiplier, CLKPM[6:0], defines a linear multiplier for the program or erase pulse. The multiplier, M, is defined by the equation: M = 1 + CLKPM[6:0] This allows for the program/erase pulse to be from 1 to 128 times the pulse set by the system clock period, SCLKR[2:0] and CLKPE[1:0].

CMFICTL2 — CMFI High Voltage Control Register 2 **0xYF F80E**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
BLOCK								0	0	PEEM	B0EM	0	PE	SES	EHV
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10-10 CMFICTL2 Bit Settings

Bit(s)	Name	Description
15:8	BLOCK	Block program and erase select. The BLOCK[7:0] bits are write protected by the SES bit. Writes to CMFICTL will not change BLOCK[7:0] if SES = 1. BLOCK[7:0] selects the CMFI EEPROM array blocks for program and erase operation. Up to eight blocks may be selected for program or erase operation at once. The CMFI EEPROM configuration along with BLOCK[7:0] determine the blocks that will be programmed simultaneously. The CMFI EEPROM array blocks that are selected to be programmed by the program operation are the blocks where BLOCK[M] = 1. The CMFI EEPROM configuration along with BLOCK[7:0] determine the blocks that will be erased simultaneously. The CMFI EEPROM array blocks that are selected to be erased by the erase operation are the blocks where BLOCK[M] = 1. **WARNING** The block bit must be set only for the blocks currently being programed. If the block bits are set for blocks that are not being programmed, the contents of the other blocks could be disturbed. 0 = Array block M is not selected for program or erase. 1 = Array block M is selected for program or erase.
7:6	—	Reserved
5	PEEM	Program erase enable monitor. The CMFI will sample the E_{PEE} signal (always enabled on the MC68F375) when EHV is asserted and hold the E_{PEE} state until EHV is negated. The E_{PEE} signal has a digital filter that requires two consecutive samples to be equal before the output of the filter will change. 0 = High voltage operations are not possible. 1 = High voltage operations are possible.
4	B0EM	Block zero enable monitor. The CMFI will sample B0EM when EHV is asserted and hold the B0EM state until EHV is negated. The optional E_{PEB0} pin has a digital filter similar to the E_{PEE} signal. If B0EM = 1 when EHV is asserted, high voltage operations to CMFI array block 0 such as program or erase are enabled. While, if B0EM = 0 when EHV is asserted high voltage operations to CMFI array block 0 are disabled. 0 = High voltage operations in Array Block 0 are not possible. 1 = High voltage operations in Array Block 0 are possible.
3	—	Reserved

Table 10-10 CMFICTL2 Bit Settings (Continued)

Bit(s)	Name	Description
2	PE	Program or erase select. The PE bit is write protected by the SES bit. Writes to CMFICTL will not change PE if SES = 1. PE configures the CMFI EEPROM for programming or erasing. When PE = 0, the array is configured for programming and if SES = 1 the SIE bit will be write locked. When PE = 1, the array is configured for erasing and SES will not write lock the SIE bit. 0 = Configure for program operation. 1 = Configure for erase operation.
1	SES	Start-end program or erase sequence. The SES bit is write protected by the HVS and EHV bits. Writes to CMFICTL will not change SES if HVS = 1 or EHV = 1. The SES bit is used to signal the start and end of a program or erase sequence. At the start of a program or erase sequence SES is set (written to a 1). At this point the CMFI EEPROM is ready to receive either the programming writes or the erase interlock write. The following bits shall be write locked: PROTECT, BLOCK[7:0], CSC, PE. SES also write locks SCLKR[2:0], CLKPE[1:0] and CLKPM[6:0]. If PE = 0 and SES = 1, SIE will be write locked. The erase interlock write is a write to any CMFI EEPROM array location after SES is set and PE = 1. If the PE bit is a 0 the CMFI BIU will accept programming writes to the CMFI array address for programming. The first programming write shall select the program page offset address (IADDR[14|13:6]) to be programmed along with the data for the programming buffers at the location written. All programming writes after the first shall update the program buffers using the lower address (IADDR[5:2]) and the block address (IADDR[17|16:15|14]) to select the program page buffers to receive the data. For further information see **10.5.2 Program Page Buffers**. After the data has been written to the program buffers the EHV bit is set (written to a 1) to start the programming pulse and lock out further programming writes. If the PE bit is a 1 the CMFI BIU will accept writes to the CMFI array addresses for an erase interlock. An erase interlock write is required before the EHV bit can be set. At the end of the program or erase operation the SES bit must be cleared (written to a 0) to return to normal operation and release the program buffers, PROTECT, SCLKR[2:0], CLKPE[1:0], CLKPM[6:0], BLOCK[7:0], CSC and the PE bit. 0 = Not configured for program or erase operation. 1 = Configure for program or erase operation.
0	EHV	Enable high voltage. EHV can be asserted only after the SES bit has been asserted and a valid programming write(s) or erase hardware interlock write has occurred. If an attempt is made to assert EHV when SES is negated, or if a valid programming write(s) or erase hardware interlock write has not occurred since SES was asserted, EHV will remain negated. The program or erase enable monitor (PEEM) and EHV are used to control the application of the program or erase voltage to the CMFI EEPROM module. High voltage operations to the CMFI EEPROM array, special MoneT shadow locations or FLASH NVM registers can occur only if EHV = 1 and PEEM = 1. Only after the correct hardware and software interlocks have been applied to the CMFI EEPROM can EHV be set. Once EHV is set SES cannot be changed and attempts to read the array will not be acknowledged. The default reset state of EHV disables program or erase pulses (EHV = 0). A master reset while EHV = 1 will terminate the high voltage operation (reset CMFICTL). A system reset or setting the STOP bit to 1 will clear EHV to a 0 terminating the high voltage pulse. The CMFI shall generate the required sequence to disable the high voltage without damage to the high voltage circuits. 0 = Program or erase pulse disabled. 1 = Program or erase pulse enabled.

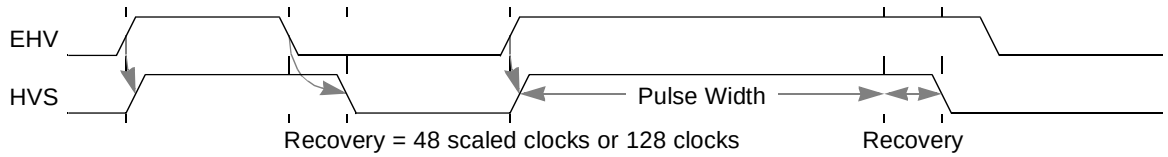


Figure 10-4 Pulse Status Timing

10.4.7 CMFIBS CMFI Bootstrap Words [3:0]

The bootstrap information for the CPU32 processor can be stored in the CMFI shadow row in the four words CMFIBS0–CMFIBS3. CMFIBS0 responds to address 0x000000, CMFIBS1 responds to 0x000002, CMFIBS2 to 0x000004 and CMFIBS3 to 0x000006 on the IMB3. See **10.6.6.3 Programming Shadow Information**.

CMFIBS0 — CMFI Bootstrap Word 0 **0xYF F810**

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	LSB 16
SP[31:16]															
RESET:															
U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]

NOTES:
1. The default state of these bits is read from the shadow row on reset.

CMFIBS1 — CMFI Bootstrap Word 1 **0xYF F812**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
SP[15:0]															
RESET:															
U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]

NOTES:
1. The default state of these bits is read from the shadow row on reset.

CMFIBS2 — CMFI Bootstrap Word 2 **0xYF F814**

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	LSB 16
PC[31:16]															
RESET:															
U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]

NOTES:
1. The default state of these bits is read from the shadow row on reset.

CMFIBS3 — CMFI Bootstrap Word 3 **0xYF F816**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PC[15:0]															
RESET:															
U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]

NOTES:
1. The default state of these bits is read from the shadow row on reset.

10.4.8 Pulse Width Timing Control

To control the pulse widths for program and erase operations the CMFI EEPROM uses the system clock and the timing control in CMFICTL. The control of the program/erase pulse timing is divided into three functions. The total pulse time is defined by the following pulse width equation:

Pulse Width = System Clock Period • R • 2^N • M
Where: R = Clock Scaling, **Table 10-11**,
N = 5 + CLKPE[1:0] + (PE • 10)

Table 10-11 System Clock Range

SCLKR[2:0]	System Clock Frequency (MHz)		Clock Scaling (R)
	Minimum	Maximum	
000	**NOTE: NOT FOR CUSTOMER USE** Program and erase timing control not specified and pulse will not be terminated by the timer control. Recovery time will be specified to be 128 clocks.		1
001	8	12	1
010	12	18	3/2
011	18	24	2
100	24	33	3
101, 110 and 111	Reserved by Motorola for future use		

NOTE

The minimum specified system clock frequency for performing program and erase operations is 8.0 MHz. The CMFI EEPROM does not have any means to monitor the system clock frequency and will not prevent program or erase operation at frequencies below 8.0 MHz. Attempting to program or erase the CMFI EEPROM at system clock frequencies lower than 8.0 MHz will not damage the device if the maximum pulse times and total times are not exceeded. While some bits in the CMFI EEPROM array may change state if programmed or erased at system clock frequencies below 8.0 MHz, the full program or erase transition is not assured.

WARNING

Never stop or alter the IMB3 clock frequency during program or erase operation. Changing the clock frequency during program or erase will result in inaccurate pulse widths and variations in the charge transferred to the array bits.

The value of SCLKR is forced to the user defined value in the corresponding shadow information words on master reset. When SCLKR[2:0] = 000, the pulse width timer control will not terminate the program or erase pulse therefore the user must clear EHV via a software write.

All of the exponents are shown in **Table 10-12**.

Table 10-12 Clock Period Exponent and Pulse Width Range

PE	CLKPE[0:1]	Exponent (N)	Pulse Width Range for all System Clock Frequencies from 8.0 MHz to 33.0 MHz.					
			Minimum Pulse Width			Maximum Pulse Width		
			8 MHz[1] 2^N•1.25E-7	10 MHz[1] 2^N•1E-7	12 MHz[1] 2^N•0.833E-7	8 MHz[1] 2^N•1.25E-07	10 MHz[1] 2^N•1E-7	12 MHz[1] 2^N•0.833E-7
0	00	5	4 µs	3.2 µs	2.7 µs	512 µs	409.6 µs	341.3 µs
	01	6	8 µs	6.4 µs	5.3 µs	1.024 ms	819.2 µs	682.7 µs
	10	7	16 µs	12.8 µs	10.7 µs	2.048 ms	1.6384 ms	1.365 ms
	11	8	32 µs	25.6 µs	21.3 µs	4.096 ms	3.2768 ms	2.731 ms
1	00	15	4.096 ms	3.28 ms	2.73 ms	524.29 ms	419.43 ms	349.5 ms
	01	16	8.192 ms	6.55 ms	5.46 ms	1.05 s	838.86 ms	699.1 ms
	10	17	16.384 ms	13.11 ms	10.92 ms	2.10 s	1.68 s	1.398 s
	11	18	32.768 ms	26.21 ms	21.85 ms	4.19 s	3.35 s	2.796 s

NOTES:

1. CMF clock frequency after SCKLR scaling. Example: A 32 MHz system clock scaled by 3 (SCLKR[0:2] = 0b100) results in an equivalent CMF clock of 10 MHz.

The value of CLKPE[1:0] is forced to the user defined value in the corresponding shadow information words on master reset, as is the value of CLKPM[6:0].

10.4.9 A Technique to Determine SCLKR, CLKPE, and CLKPM

The following example determines the values of the SCLKR, CLKPE, and CLKPM fields for a 51.4 µS pulse program pulse, PE=0, in a system with a 33.0 MHz system clock.

In this example system clock frequency = 33.0 MHz; the system clock period is therefore 30.3 nS.

1. Determine SCLKR:
   From **Table 10-11** a 33.0 MHz system clock uses SCLKR[0:2]=100, R=3.
2. Determine CLKPE:
   From **Table 10-12** a 51.4 µS program pulse, PE=0, can be generated by expo-

nents in the range of N=6. While any of these values can be selected CLKPE[0:1]=00, N=6, will be used for the example.

3. Determine CLKPM:
   Using the selected values of N and R in the pulse width equation and solving for M yields M=8.8. Rounding M to 9 then CLKPM[0:6]=0x8 (0b0001000).

4. Check the results:
   Pulse Width=System Clock Period • R • 2^N • M
   using SCLKR[0:2]=100, CLKPE[0:1]=00, CLKPM[0:6]=0001000 and PE=0 at 33.0 MHz system clock. Pulse Width=30.3 µS • 3 • 2^6 • 9=52.4 µS program pulse.

10.5 CMFI EEPROM Array Addressing

The base address of the memory block in which the CMFI EEPROM array resides is specified in the array base address register (CMFIBAR). The default reset base address is specified in the shadow registers by the user. The only restrictions on the base address are that it must be on a 2^N byte boundary (N = 16, 17 or 18 depending on array size). If the base address is set such that the CMFI EEPROM array overlaps the control register block, accesses to the 32-bytes of the array that overlap the control registers will be ignored, allowing the control block to remain accessible. This is only true with respect to the same CMFI EEPROM module. If the control register blocks of other modules are overlapped by the CMFI EEPROM array, accesses to the overlapped addresses will be indeterminate.

The CMFI EEPROM array is divided into a shadow row and eight 32-Kbyte blocks as shown in **10.6.6.3 Programming Shadow Information**. The shadow register information is stored in shadow block. The array supports multiple-page programming.

Information in the array is accessed in 32-byte burst buffers. Two read burst buffers are aligned to the low order addresses (IADDR[4:0]). The first burst buffer is associated with the lower array blocks. The second burst buffer is associated with the higher array blocks. Read access time from the array will take 2, 3, 4 or 5 clocks as defined by WAIT[1:0] for the first data access. If a burst access the following access are 1 clock accesses from within the 32-byte burst buffer. To prevent the BIU from unnecessarily refreshing the burst buffer from the array, the CMFI EEPROM shall monitor the IMB3 address to determine if the required information is within one of the two read burst buffers and the access is valid for the module.

Write accesses to the CMFI array will not assert the address acknowledge unless they are programming or erase interlock writes.

10.5.1 Read Burst Buffers

The two 32-byte read burst buffers are fully independent and are located in two separate read sections of the array as indicated in **Figure 10-1**. Each burst buffer status and address are monitored in the BIU. The status of the read burst buffers are usually valid, but are made invalid by the following operations:

- Setting/Clearing VT

- Setting/Clearing PBR
- Reset,
- Programming write
- Erase interlock write
- Setting EHV
- Clearing SES
- Setting/Clearing SIE

Each access to the CMFI EEPROM array shall determine if the requested location is within the current burst buffers. If the requested location is not within the read burst buffers then the correct read burst buffer shall be made invalid and a new 32 byte block of information will be fetched from the array. The burst buffer address is updated and status is made valid. If the requested location is within one of the current burst buffers or has been fetched from the array the selected bytes are transferred to the IMB3 completing the access. While bursting data from the read burst buffer the address is incremented by width of the internal data bus. Upon reaching the end of the read burst buffer the CMFI EEPROM shall terminate the burst access.

10.5.2 Program Page Buffers

The CMFI EEPROM can program up to eight 64-byte pages at one time. Each program page buffer is associated with one array block as indicated in the diagrams in **Figure 10-1**. All program page buffers share the same block offset address, IADDR[14|13:6], stored in the BIU. The block offset address is extracted from the address of the first programming write. To select the CMFI EEPROM array block that will be programmed, the program page buffers use the CMFI EEPROM array configuration and BLOCK[7:0]. The data programmed in each array block is determined by the programming writes to the program buffer for each block. All program buffer data is unique whereas the program page offset address is shared by all blocks.

The array block that will be programmed is selected by the BLOCK bit that is a 1. If BLOCK[M] = 1 then program buffer[M] is active and array block[M] will program. If BLOCK[M] = 0 then program buffer[M] is inactive and array block[M] will not program.

Bits in the program page buffers shall select the non-program state if SES = 0. During a program margin read, the program buffers will update bits to the non-program state for bits that correspond to array bits that the program margin read has determined are programmed.

10.6 Operation

The following sections describe the functioning of the CMFI EEPROM during various operational modes. The primary function of the CMFI EEPROM Module is to serve as electrically erasable and programmable non-volatile memory accessed by any bus master capable of using the IMB3.

10.6.1 Power On Reset

The device signals a power on reset (IPORB = 0) to the CMFI EEPROM when a full reset is required. A power on reset is the priority operation for the CMFI EEPROM and will terminate all other operations, including resetting FRI = 0.

10.6.2 Master Reset

The device signals a master reset (IMSTRSTB = 0) to the CMFI EEPROM when a full reset is required. A master reset is the 3rd highest priority operation for the CMFI EEPROM and will terminate all other operations, unless FRI = 1. If FRI = 1 master reset is blocked to the CMFI EEPROM.

The CMFI EEPROM module uses master reset to initialize all register bits to their default reset value. If the CMFI EEPROM is in program or erase operation (EHV = 1) and a master reset is generated, the module will perform the needed interlocks to disable the high voltage without damage to the high voltage circuits. Master reset will terminate any other mode of operation and force the CMFI EEPROM BIU to a state ready to receive IMB3 accesses within 4 clocks of the end of master reset.

During master reset the CMFI EEPROM must perform several tasks to assure correct and reliable operation. In order of execution these tasks are:

1. Recover from high voltage operation.
2. Copy the shadow information into the registers.
3. Reset the BIU state machine to receive the first cycle start within 4 clocks of the final reset clock.

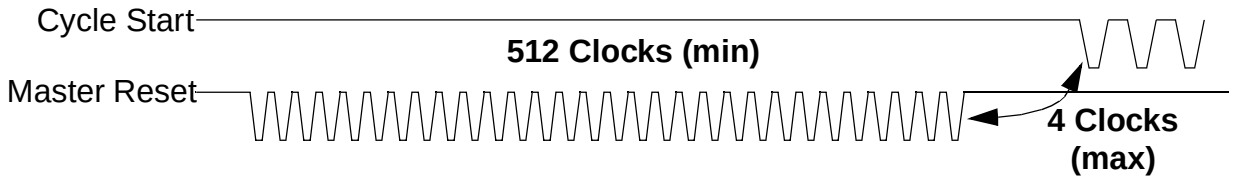


Figure 10-5 Master Reset Configuration Timing

10.6.3 System Reset

The device signals a system reset (ISYSRSTB = 0) to the CMFI EEPROM when required. A system reset is the 4th highest priority operation and forces the BIU into a state ready to receive IMB3 accesses and clears the EHV bit, unless FRI = 1. All other bits shall remain unaltered by a system reset.

10.6.4 Register Read and Write Operation

The CMFI EEPROM control registers are accessible for read or write operation at all times while the device is powered up except during master reset, or system reset.

The access time of a CMFI register is at least 2 clocks depending on the state of WAIT[1:0] for both read and write accesses. Read accesses to reserved registers shall

cause the BIU to generate either a bus error or return all zeros. Write accesses to reserved registers shall have no effect. See section **10.4.1 CMFI EEPROM Module Control Block Addressing** for more information on register accesses.

10.6.5 Array Read Operation

The CMFI EEPROM array is available for read operation under most conditions while the device is powered up. Reads of the array are not allowed during master reset, system reset, and ACCESS = 0 while high voltage is applied to the array or while the CMFI EEPROM is stopped, see section **10.6.8 Stop Operation** for more information on stopping the CMFI EEPROM. At certain points, as defined in the program or erase sequence, reading the array shall result in a margin read. These margin reads return the status of the program or erase operation and not the data in the array.

After reset, programming writes, erase interlock write, setting EHV, clearing SES or setting/clearing SIE (this affects the lower array blocks burst buffer only) the burst buffers will not contain valid information and the correct information will be fetched from the array.

Section **10.5.1 Read Burst Buffers** defines how the two burst buffers in the CMFI EEPROM are associated to array blocks. During burst read operation, the initial read will require at least 2 clocks depending on the state of WAIT[1:0]. Subsequent burst reads will take 1 clock until the end of the burst buffer is reached or the burst is terminated by the IMB3.

10.6.6 Programming

To modify the charge stored in the isolated element of the CMFI bit from a logic 1 state to a logic 0 state, a programming operation is required. This programming operation shall apply the required voltages to change the charge state of the selected bits without changing the logic state of any other bits in the CMFI array. The program operation cannot change the logic 0 state to a logic 1 state; this transition must be done by the erase operation. Programming uses a set of up to 8 program buffers of 64 bytes each to store the required data, an address offset buffer to store the starting address of the block(s) to be programmed and a block select buffer that stores information on which block(s) are to be programmed. Any number of the array blocks may be programmed at one time.

Do not program any page more than once after a successful erase operation. While this will not physically damage the array it shall cause an increased partial disturb time for the unselected bits on the row and columns that are not programmed.

A full erase of all blocks being programmed must be done before the CMFI EEPROM can be used reliably if over programming occurs.

WARNING

If the PROTECT bit is set then the CMFI EEPROM array will not be programmed. Also, if PEEM = 0, no programming voltages will be applied to the array and if B0EM = 0, no programming voltages will be applied to block 0.

10.6.6.1 Program Sequence

The CMFI EEPROM module requires a sequence of writes to the high voltage control registers (CMFICTL1 and CMFICTL2) and to the program page buffer(s) in order to enable the high voltage to the array or shadow information for program operation. The required program sequence follows.

1. Write PROTECT = 0 to disable protection on the CMFI EEPROM array.
2. Write PAWS to 0b100, write NVR = 1, write GDB = 1.
3. Set the initial pulse width bit settings per **Table 10-6**. Using section **10.4.9 A Technique to Determine SCLKR, CLKPE, and CLKPM**, write the pulse width timing control fields for a program pulse to the CMFICTL1 register. Write BLOCK[7:0] to select the array blocks to be programmed and PE = 0 in the CMFICTL2 register.

   WARNING

   Do not select the block bits of blocks not currently being programmed.

4. Write SES = 1 in the CMFICTL2 register. This step can be done with the same write in step 2 but is split out as a separate step in the sequence for looping.
5. Programming writes. Write to the 64-byte array locations to be programmed. This shall update the programming page buffer(s) with the information to be programmed. Only the last write to each word within the program page buffer shall be saved for programming. All accesses of the array after the first write shall be to the same block offset address (IADDR[14|13:6]) regardless of the address provided. Thus the locations accessed after the first programming write are limited to the page locations to be programmed. Off page read accesses of the CMFI array after the first programming write are program margin reads see section **10.6.6.2 Program Margin Reads**.

   All program page buffers share the same block offset address (IADDR[14|13:6]) stored in the BIU. The block offset address is extracted from the address of the first programming write. To select the CMFI EEPROM array block(s) that will be programmed, the program page buffers use the CMFI EEPROM array configuration and BLOCK[7:0]. Subsequent writes fill in the program page buffers using the block address to select the program page buffer and the page word address (IADDR[5:2]) to select the word in the page buffer. The array configuration and BLOCK[7:0] determine which blocks are programmed simultaneously.
6. Write EHV = 1 in the CMFICTL2 register. If a program buffer has not received

a programming write no programming voltages will be applied to the corresponding word in the array. Also, at this point writes to the program page buffers are disabled until SES has been cleared and set.

7. Read the CMFICTL1 register until HVS = 0.
8. Write EHV = 0 in CMFICTL2.
9. Program Verify. Read the words of the pages that are being programmed. These are program margin reads, see section **10.6.6.2 Program Margin Reads**. If any bit is a 1 after reading all of the locations that are being programmed go to step 5. If all the locations verify as programmed go to step 9.

WARNING

After a program pulse, read at least one location with IADDR[5] = 0 and one location with IADDR[5] = 1 on each programmed page. Failure to do so may result in the loss of information in the CMFI EEPROM array. While this will not physically damage the array it will require that a full erase of all blocks being programmed be done before the CMFI EEPROM can be used reliably.

To reduce the time for verification, read only two locations in each array block that is being programmed after reading a non-programmed bit. The first location must be a location with IADDR[5] = 0; while, the second must use IADDR[5] = 1. Also, after a location has been fully verified (all bits are programmed) it is not necessary to verify the location as no further programming voltages will be applied to the drain of the corresponding bits.

10. If the margin read is successful, then write SES = 0 in the CMFCTL register, otherwise do the following:

    a. Write new pulse width parameters (if required per **Table 10-6**) - SCLKR, CLKPE, CLKPM.

    b. Write new values for PAWS, NVR, and GDB (if required per **Table 10-6**).

    c. Go back to step 6 to apply additional programming pulses.

11. If more information needs to be programmed, go back to step 2.

CAUTION

Failure to read each page that is being programmed after each program pulse may result in the loss of information in the CMF EEPROM array. While this will not physically damage the array a full erase of all blocks being programmed must be performed before the CMF EEPROM can be used reliably.

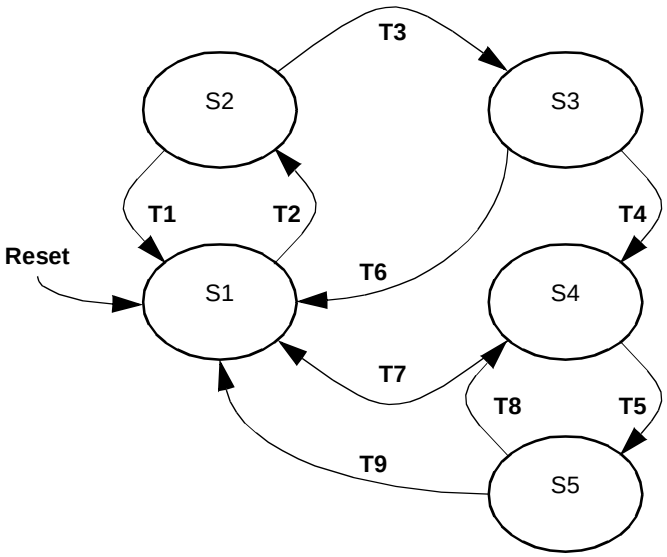


Figure 10-6 Program State Diagram

Table 10-13 Program Interlock State Descriptions

State	Mode	Next State	Transition Requirement	
S1	Normal Operation: Normal array reads and register accesses. The Block protect information and pulse width timing control can be modified.	S2	T2	Write PE = 0, SES = 1
S2	First Program Hardware Interlock Write: Normal read operation still occurs. The array will accept programming writes Accesses to the registers are normal register accesses. A write to CMFICTL2 cannot change EHV at this time. If the write is to a register no data will be stored in the program page buffers and the CMFI shall remain in state S2.	S1	T1	Write SES = 0 or master reset
		S3	T3	Hardware Interlock Write to any CMFI array location This programming write will latch the selected word of data into the programming page buffer and the address shall be latched to select the location that will be programmed. Once a bit has been written then it shall remain in the program buffer until another write to the word or a write of SES = 0 or a program margin read determines that the state of the bit needs no further modification by the program operation If the write is to a register no data will be stored in the program page buffers and the CMFI shall remain in state S2
S3	Expanded Program Hardware Interlock Operation: Program margin reads will occur. Programming writes are accepted so that all program pages may be programmed. These writes may be to any CMFI array location. The program page buffers will be updated using only the data, the lower address (IADDR[5:2]) and the block address. Accesses to the registers are normal register accesses. A write to CMFICTL2 can change EHV. If the write is to a register no data will be stored in the program page buffer.	S1	T6	Write SES = 0 or master reset
		S4	T4	Write EHV = 1.

Table 10-13 Program Interlock State Descriptions (Continued)

State	Mode	Next State	Transition Requirement	
S4	Program Operation: High voltage is applied to the array or shadow information to program the CMFI bit cells. The pulse width timer is active if SCLKR[2:0] ≠ 0 and HVS can be polled to time the program pulse. No further programming writes will be accepted. During programming the CMFI will not generate an address acknowledge for any array access. Accesses to the registers are normal register accesses. A write to CMFICTL2 can change EHV only.	S1	T7	Master reset
		S5	T5	Write EHV = 0, write STOP = 1 or system reset
S5	Program Margin Read Operation: These reads shall determine if the state of the bits on the selected page needs further modification by the program operation. Once a bit is fully programmed, the data stored in the program page shall be updated so no further programming occurs for that bit and the value read is a 0.	S4	T8	Write EHV = 1.
		S1	T9	Write SES = 0 or master reset

10.6.6.2 Program Margin Reads

The CMFI EEPROM provides a program margin read with electrical margin for the program state. Program margin reads provide sufficient margin to assure specified data retention. The program margin read is enabled when SES = 1 and a programming write has occurred. To increase the access time of the program margin read, the burst buffer access time shall be 16 clocks instead of the usual number of clocks as determined by WAIT[1:0] for the first read access. The program margin read and subsequent verify reads will return a 1 for any bit that has not completely programmed. Bits that the programming write left in the non-programmed state will read as a 0. Bits that have completed programming will read as a 0 and update the data in the programming page buffer so that no further programming of those bits will occur. The program margin read occurs whenever the burst buffer data is invalid. See section **10.6.5 Array Read Operation** for information on when the burst buffer is invalid. A program margin read must be done for all pages that are being programmed after each program pulse. This requires two program margin reads for each program buffer. The first required program margin read should be to an address in either the lower or upper 32 bytes of the program buffer while the second should be to an address in the other 32 bytes.

Table 10-14 Results of Programming Margin Read

Current Data in the Program Page Buffer[1]	Current State of Bit	Data Read During Margin Read[2]	New Data for the Program Page Buffer[1]
0	Programmed (0)	0	1
0	Erased (1)	1	0
1	Programmed (0)	0	1
1	Erased (1)	0	1

NOTES:
1. 0 = bit needs to further programming
   1 = bit does not need further programming
2. A “0” read during the margin read means that the bit does NOT need further programming. A “1” means the bit needs to be programmed further.

Failure to read each page that is being programmed after each program pulse may result in the loss of information in the CMFI EEPROM array.

While this will not physically damage the array a full erase of all blocks being programmed must be done before the CMFI EEPROM can be used reliably.

For more information see section **10.6.6.4 Over Programming**.

10.6.6.3 Programming Shadow Information

Programming the shadow information uses the same procedure as programming the array except that only the program page buffer associated with the lowest array block will be used to program the shadow information. Before starting the program sequence SIE must be a 1.

The first 32 bytes (IADDR[7:0] = 0x00 to 0x1F) of the 256 bytes of shadow locations are withheld by Motorola for the shadow information words and future applications. The remaining 224 bytes are available for general use. Programming of these 32 bytes affects the reset configuration of the CMFI EEPROM. The register shadow information word and location within the shadow row is presented in the following table. Registers not listed in this table do not require shadow information for reset configuration but their shadow locations are withheld from general use by Motorola for future applications.

Table 10-15 Register Shadow Information

Register Name	Register Address	Shadow Row Address (addr[7:0])
CMFIMCR	0x00 0000	0x00
CMFIBAR	0x00 0008	0x08
CMFICTL1	0x00 000C	0x0C
CMFIBS0	0x00 0010	0x10
CMFIBS1	0x00 0012	0x12
CMFIBS2	0x00 0014	0x14
CMFIBS3	0x00 0016	0x16

10.6.6.4 Over Programming

Programming a CMFI bit without a program margin read after each program pulse or exceeding the specified program times or voltages will result in an over programmed state. Once a CMFI bit has been over programmed, data in the array block that is located upon the same column shall be lost as the over programmed bit causes the entire column to appear programmed. To restore an array block with an over programmed bit the block must be erased and reprogrammed.

10.6.7 Erase

To modify the charge stored in the isolated element of the CMFI bit from a logic 0 state to a logic 1 state, an erase operation is required. The erase operation cannot change the logic 1 state to a logic 0 state; this transition must be done by the program operation. In the CMFI EEPROM, erase is a bulk operation that shall affect the stored charge of all the isolated elements in an array block. To make the CMFI module block-erasable, the array is divided into blocks that are physically isolated from each other. Each of the array blocks may be erased in isolation or in any combination. The CMFI array block size is fixed for all blocks in the module at 32 Kbytes and the module is comprised of 4, 6 or 8 blocks. If the CMFI EEPROM array is protected (PROTECT = 1), the array will not be erased. Also, if PEEM = 0 no erase voltages will be applied to the array and if B0EM = 0, no programming voltages will be applied to block 0.

The array blocks selected for erase operation are determined by BLOCK[7:0].

10.6.7.1 Erase Sequence

The CMFI EEPROM module requires a sequence of writes to the high voltage control registers (CMFICTL1 and CMFICTL2) and an erase interlock write in order to enable the high voltage to the array and shadow information for erase operation. The erase sequence follows.

1. Write PROTECT = 0 to disable protection on the array.
2. Set the initial pulse width bit settings per **Table 10-7**.
3. Using section **10.4.9 A Technique to Determine SCLKR, CLKPE, and CLKPM**, write the pulse width timing control fields for an erase pulse in the CMFICTL1 register. Write the BLOCK[7:0] to select the blocks to be erased, PE = 1 and SES = 1 in the CMFICTL2 register.
4. Execute an erase interlock write to any CMFI array location.
5. Write EHV = 1 in the CMFICTL2 register.
6. Read the CMFICTL1 register until HVS = 0.
7. Write EHV = 0 in the CMFICTL2 register.
8. To verify the erase operation, read all locations that are being erased, including the shadow information if the block containing it is erased. Off-page reads are erase margin reads that update the read page buffer. (See section **10.6.7.2 Erase Margin Reads**.) If all the locations read as erased, go to step 9.

NOTE

Do not perform erase margin reads until reaching the condition PAWS=0b111, NVR=0 and GDB=1.

9. To reduce the time used for erase margin reads, upon the first read of a zero, do the following:

   a. Write new pulse width parameters, SCLKR, CLKPE, and CLKPM (if required per **Table 10-7**).

   b. Write newPAWS value (if required per **Table 10-7**).

   c. Write new values for NVR and GDB (if required per **Table 10-7**).

   d. Go back to step 5 to apply additional erase pulses.

NOTE

After a location has been verified (all bits erased), it is not necessary to verify the location after subsequent erase pulses.

10. Write SES = 0 in the CMFICTL2 register. The CMFI requires 16 clocks after writing SES = 0 prior to a normal CMFI EEPROM array read. To meet this requirement the CMFI shall allow all access to start any time after writing SES = 0 but the access shall not be completed until after the 16 clock period. This time shall be extended by the number of clocks defined by WAIT[1:0].

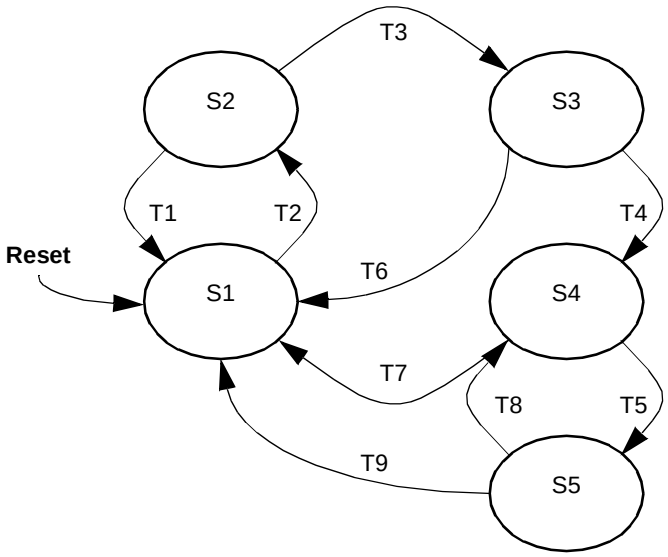


Figure 10-7 Erase State Diagram

Table 10-16 Erase Interlock State Descriptions

State	Mode	Next State	Transition Requirement	
S1	Normal Operation: Normal array reads and register accesses The Block protect information and pulse width timing control can be modified.	S2	T2	Write PE = 1, SES = 1
S2	Erase Hardware Interlock Write: Normal read operation still occurs The CMFI will accept the erase hardware interlock write. This write may be to any CMFI array location Accesses to the registers are normal register accesses. A write to CMFICTL2 cannot set EHV at this time A write to the register is not an erase hardware interlock write and the CMFI shall remain in state S2.	S1	T1	Write SES = 0 or master reset
		S3	T3	Hardware Interlock Write to any CMFI array location is the erase interlock write. If the write is to a register the erase hardware interlock write has not been done and the CMFI shall remain in state S2.
S3	High voltage write enable Erase margin reads will occur Accesses to the registers are normal register accesses. A write to CMFICTL2 can change EHV.	S1	T6	Write SES = 0 or master reset
		S4	T4	Write EHV = 1
S4	Erase Operation: High voltage is applied to the array blocks to erase the CMFI bit cells. The pulse width timer is active if SCLKR[2:0] $\neq$ 0 and HVS can be polled to time the erase pulse During erase the array will not respond to any address. Accesses to the registers are normal register accesses. A write to CMFICTL2 can change EHV only.	S1	T7	Master reset
		S5	T5	Write EHV = 0, write STOP = 1 or system reset
S5	Erase Margin Read Operation: These reads shall determine if the state of the bits on the selected blocks needs further modification by the erase operation. Once a bit is fully erased it shall read as a 1. All words within the blocks being erased must be read to determine if erase is completed.	S4	T8	Write EHV = 1
		S1	T9	Write SES = 0 or master reset

10.6.7.2 Erase Margin Reads

The CMFI EEPROM provides an erase margin read with electrical margin for the erase state. Erase margin reads provide sufficient margin to assure specified data retention. The erase margin read is enabled when SES = 1 and the erase write has occurred. The erase margin read and subsequent on page erase verify reads will return a 0 for any bit that has not completely erased. Bits that have completed erasing will read as a 1. To increase the access time of the erase margin read the access time shall be 16 clocks instead of the usual number of clocks as determined by WAIT[1:0] for the first read access. The erase margin read occurs while doing the transfer from the array to the burst buffer. All locations within the block(s) that are being erased must read as a 1 to determine that no more erase pulses are required.

10.6.7.3 Erasing Shadow Information Words

The shadow information words are erased with either CMFI array block 0 or block 2 depending upon the array configuration. To verify that the shadow information words are erased the SIE bit in CMFIMCR should be set to 1 during the erase margin read while the shadow information is read. For the erase operation to be completed block 0 or 2 must also be fully verified.

Setting SIE = 1 will disable normal array access and should be cleared after verifying the shadow information.

10.6.8 Stop Operation

The CMFI EEPROM goes into a low power operation, or stop operation, while STOP = 1. Setting STOP to 1 will clear EHV to a 0. When the STOP bit is set only the control registers can be accessed on the CMFI EEPROM. The CMFI EEPROM array may not be programmed, erased or read while STOP = 1. With STOP = 1 and $\overline{\text{LOCK}}$ = 1 the array may be mapped to another location in the memory map, and the array Address Space may be changed. During stop operation the CMFI may enter low power stop clock operation.

10.6.8.1 Low Power Stop Clock Operation

The low power stop clock operation is the lowest power configuration of the CMFI EEPROM, disabling the internal clock. This operation is entered when STOP = 1 and the correct system clock disable (ICLKDIS[N]) is asserted and HVS = 0. The BIU will disable the system clock to the state machine at the completion of required functions to protect the CMFI EEPROM after the STOP bit is set and ICLKDIS[N] is asserted. Once the CMFI EEPROM is in low power stop clock operation, all accesses by the IMB3 will be ignored, and AACKB will not be asserted until ICLKDIS[N] is cleared by the IM. When ICLKDIS[N] is cleared, the internal clocks will be enabled and the CMFI EEPROM register control block will respond to IMB3 accesses.

10.6.8.2 STOP Recovery

The CMFI requires 16 clocks after writing STOP = 0 prior to a normal CMFI EEPROM array read. The access time of the first CMFI array read shall be 16 clocks instead of the usual number of clocks as determined by WAIT[1:0]. To meet this requirement the

CMFI shall allow all access to start any time after writing STOP = 0 but the access shall not be completed until after the 16 clock period.

10.6.9 Background Debug Mode or Freeze Operation

While in background debug mode (IFREEZEB = 0) the CMFI should respond normally to accesses except that $\overline{\text{LOCK}}$ is writable.

SECTION 11
STATIC RANDOM ACCESS MEMORY (SRAM)

11.1 Introduction

This SRAM module is a fast access (two clocks) general purpose 8K (8,192 bytes) static RAM (SRAM) for the MCU and is accessed via the IMB3. In addition there is 2K (configured as four blocks of 512 bytes each) of patch static RAM. These modules are fast access (two clocks) general purpose static RAM (SRAM) for the MCU with a patch option which provides a method to overlay the internal CMFI memory for emulation. As an additional feature, the 512-byte arrays can be used as additional SRAM. A register map showing the SRAM and overlay configuration registers and memory blocks is shown in **Figure 11-1**.

The SRAM module is powered by V_{DDL} in normal operation and may be used as standby SRAM if standby power is supplied via the V_{STBY} pin of the MCU. Switching between V_{DDL} and V_{STBY} will occur automatically.

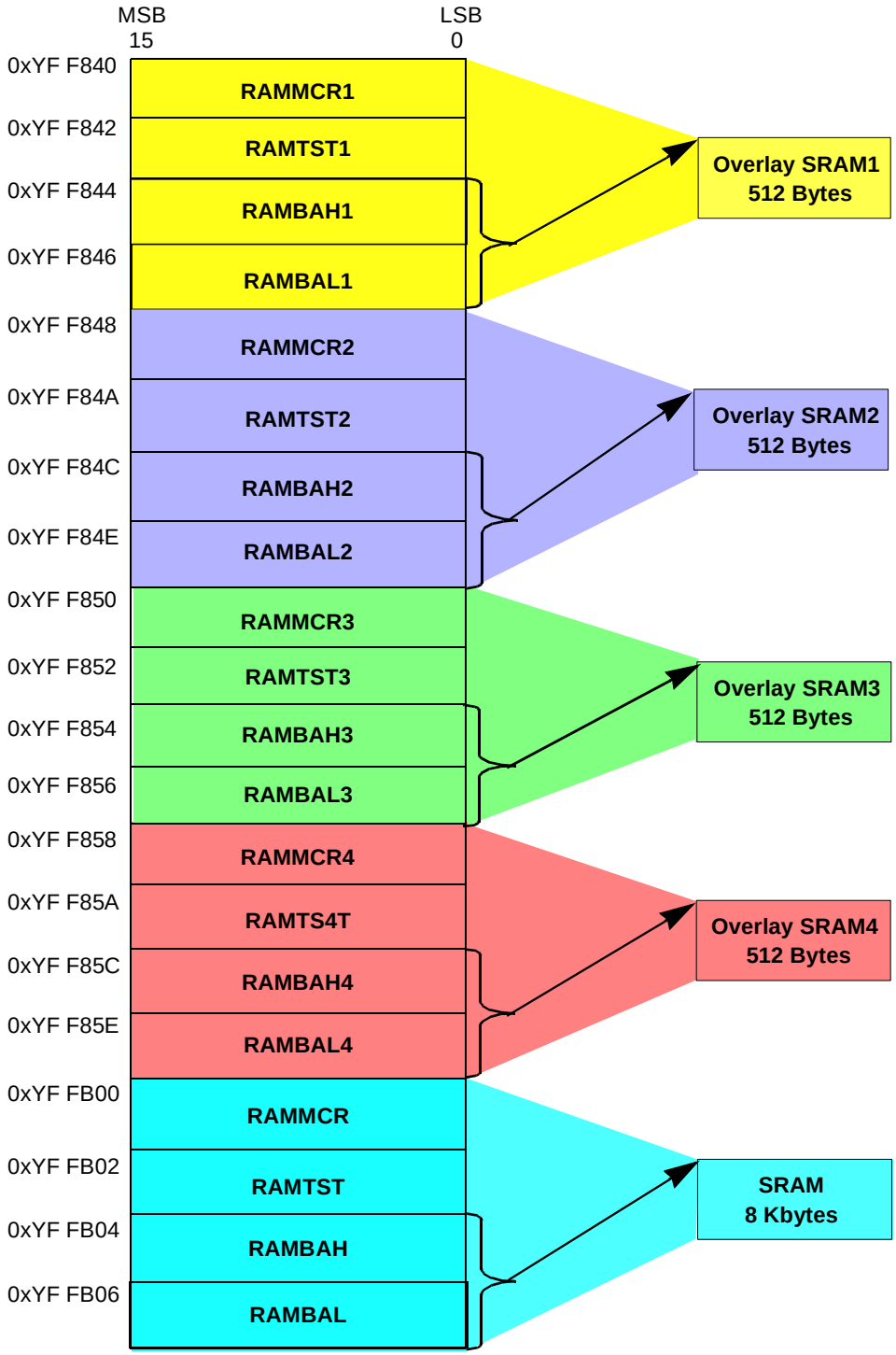
When used as general purpose SRAM, this module is accessed via the IMB3. The SRAM may be read or written as either bytes or words. Access for aligned long-word operations is supported by back-to-back IMB3 accesses (four clocks) to accommodate 32-bit operations.

11.2 Programmer's Model

Each SRAM module consists of two separately addressable sections. The first is a set of memory mapped control and status registers used for configuration and testing of the SRAM array. The second section is the array itself.

11.2.1 SRAM Control Block

There are four registers provided for configuration and control of each SRAM module: SRAM module configuration register (RAMMCR), a factory test register (RAMTST), and the array base address registers (RAMBAH, RAMBAL). In order to offer the maximum protection for the SRAM array, the SRAM module control registers are located in supervisor data space.



Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIM2E configuration register (SCIMMCR).

Figure 11-1 SRAM Module Configuration

11.2.2 SRAM Array

The SRAM array itself can be placed anywhere in the address map of the MCU by means of the array base address registers. The high order address lines (IADDR[31:N] N=13 and 9 for 8K and 512-byte arrays) are compared with the value in RAMBAH and RAMBAL registers for a base address match. This value points to the lowest address that SRAM data may be located and is always on a 8K or 512-byte boundary. The SRAM array top is on a 256-byte boundary. The only restrictions on the base address is that it must be on an address boundary greater than or equal to the array size. If the SRAM array base address is located to overlap the SRAM control block then access to the 8 bytes in the SRAM array located at the same address as the SRAM control block are ignored allowing the control block to be accessed. Note this is only the 8 bytes in the SRAM control block; this mapping may have unknown results for other modules.

11.2.2.1 SRAM Array Addressing

The BIU of the SRAM module compares IADDR[31:N] (N = 13 for 8K and 9 for 512-byte arrays) of the IMB3 with the value of the array base address registers. If they match then the low address and ISIZ[1:0] are used to access the SRAM location in the array. Addresses in the array that are not implemented will be ignored by the SRAM module allowing an external device to respond to the address. Function codes are also checked for the correct access rights. If the array is placed in supervisor space, user accesses will be ignored allowing an external device to respond to the address. If the array is placed in unrestricted space, it will respond to both user and supervisor accesses. The array may also be placed in program space, or program/data space. Program space allows the SRAM array to contain only program instructions for execution, or program counter relative addressing modes for operand fetches from the array. Program/data space allows both program and data may be stored in the SRAM array.

11.3 SRAM Module Control and Status Registers

The SRAM module control and status registers are used to control and monitor the operation of the SRAM array. The following sections describe the operation of each register in the SRAM control block. Since all the registers are in supervisor data space, the only way to change the state of the registers is through a supervisor program. Any other restrictions for making changes to the registers will be noted in the description of the register. Unless otherwise noted, any source of master reset will cause register bits to be forced to their reset state; while, system resets have no effect on the registers. Reads to unimplemented bits will return “0”; while, writes have no effect.

11.3.1 Module Configuration Register (RAMMCR)

All modules on the MCU contain a module configuration register. The RAMMCR register bits configure the SRAM module for stop operation and for proper access rights to the array.

RAMMCR1 — SRAM Module Configuration Register **0xYF F840**
RAMMCR2 **0xYF F848**
RAMMCR3 **0xYF F850**
RAMMCR4 **0xYF F858**
RAMMCR **0xYF FB00**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	RESERVED		PDS	RLCK	0	RASP[1:0]		RESERVED							
RESET:															
1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Table 11-1 RAMMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Stop control. The assertion of the STOP control bit in the RAMMCR register by a bus master signals the SRAM module to enter into the STOP state. When STOP is asserted, SRAM array accesses are ignored. When the SRAM module is in normal mode of operation the array base address registers are write protected. 0 = SRAM module normal operation. 1 = Causes SRAM module to enter low power stop mode.
14:13	—	Reserved
12	PDS	Power down status. PDS is a optional status bit in the RAMMCR that enables a power monitor for the SRAM array. The power monitor circuit will clear the PDS bit (PDS = “0”) if the array standby power is lost. If the PDS bit is unimplemented reads will return “0”. 0 = Power monitor for the SRAM array is disabled. SRAM array standby power has failed. 1 = Power monitor for the SRAM array is enabled. SRAM array standby power has not failed.
11	RLCK	Base address lock. 0 = SRAM base address registers are writable from the IMB3. 1 = SRAM base address registers are write locked.
10	—	Reserved
9:8	RASP[1:0]	Array space. The RASP field limits access to the SRAM array to one of four CPU32 address spaces. See **Table 11-2**. 0 = Only the module configuration register, test register, and interrupt register are designated as supervisor-only data space. Access to all other locations is unrestricted. 1 = All module registers and tables are designated as supervisor-only data space.
7:0	—	Reserved

Table 11-2 RASP Encoding

RASP[1:0]	Space
00	Unrestricted program and data
01	Unrestricted program
10	Supervisor program and data
11	Supervisor program

11.3.2 Array Base Address Registers (RAMBAH, RAMBAL)

The array base address registers are provided to allow the flexibility of placing the SRAM array anywhere in the memory map. RAMBAH and RAMBAL contains an address field used to specify the most significant bits of the lowest address value in

the SRAM array address block. The SRAM array base address is placed on a 512, 1K, 2K or 4-Kbyte block boundary (block size greater than or equal to the size of the array). RAMBAH and RAMBAL may only be written while the SRAM is in STOP mode STOP = “1” and the lock bit is not set RLCK = “0”. Once the RLCK bit is set, writes will have no effect on RAMBAH and RAMBAL. This will prevent runaway software from inadvertently re-mapping the array. The SRAM must be in STOP mode while RAMBAH or RAMBAL are written this prevents inadvertent intermediate array mapping to be acknowledged.

RAMBAH1, RAMBA1L — SRAM Base Address Registers **0xYF F844, 0xYF F846**
RAMBAH2, RAMBAL2 **0xYF F84C, 0xYF F84E**
RAMBAH3, RAMBAL3 **0xYF F854, 0xYF F856**
RAMBAH4, RAMBAL4 **0xYF F85C, 0xYF F85E**
RAMBAH, RAM1BAL **0xYF FB04, 0xYF FB06**

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	LSB 16
RAMBAH															
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RAMBAL							RESERVED								
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 11-3 RAMBAH, RAMBAL Bit Settings

Bit(s)	Name	Description
31:16	RAMBAH	Array base address high. With STOP asserted the base Address field of RAMBAH may be changed so that the array may be placed at the desired address in the memory map. This must be done by a supervisor program since the register is in supervisor data space. To lock the base address field, RLCK in the RAMMCR should be set. This will prevent the base address field from being changed until the next master reset.
15:9	RAMBAL	Array base address low. With STOP asserted the base Address field of RAMBAL may be changed so that the array may be placed at the desired address in the memory map. This must be done by a supervisor program since the register is in supervisor data space. To lock the base address field, RLCK RAMMCR should be set. This will prevent the base address field from being changed until the next master reset.
8:0	—	Reserved

11.4 Operation

The SRAM module has several modes of operation. The following sections describe SRAM module operation in each of these modes.

11.4.1 Normal Operation

Normal operation is when the SRAM may be accessed via the IMB3 by a bus master and is being powered by V_{DDL}. The array may be accessed as byte or word. Access may be either read or write.

11.4.1.1 Read/Write

The SRAM module allows a byte or aligned word read/write in one IMB3 bus cycle. Long word read/write will require an additional bus cycle. An IMB3 bus cycle requires 2 system clocks.

Table 11-4 SRAM Array Read/Write Minimum Access Times

TYPE	Bus Cycles Required for Read or Write	Number of System Clocks
Byte	1	2
Aligned Word	1	2
Aligned Long Word	2	4

11.4.2 Standby Operation

A separate supply pin is used by the standby SRAM module to maintain the contents of the SRAM array during a power down phase. The external supply pin of the MCU is known as V_{STBY}. Data in the standby SRAM will be retained down to the lowest supply voltage, either V_{DDL} or V_{STBY}, see **APPENDIX E ELECTRICAL CHARACTERISTICS**. Circuitry within the standby SRAM module will automatically switch between V_{DDL} and V_{STBY}. The SRAM module will switch to standby power when $V_{DDL} < V_{STBY} - V_{SWITCH}$.

When the SRAM array is powered by the V_{STBY} pin of the MCU, access to the SRAM array is blocked. Data read from the SRAM array during this condition will not be valid. Data written to the SRAM may be corrupted if switching occurs during a write operation. For the module to function correctly as general purpose SRAM, the maximum value for $V_{STBY} \leq V_{DDL}$.

11.4.2.1 Power Down

In order to guarantee valid standby SRAM data during power down, external low voltage inhibit circuitry, (external to the MCU), must be designed to force the RESET pin into the active state before V_{DDL} drops below its normal limit. This is necessary to inhibit a write cycle to the SRAM during power down.

11.4.3 RESET Operation

When a synchronous reset occurs, a bus master will be allowed, as a result of internal synchronization, to complete the current access. Thus, a write bus cycle, byte or word, that is in progress when a synchronous reset occurs will be completed without error. During the RESET state, once an in-progress write has been completed, further writes to the standby SRAM array will be inhibited.

Note that a long word write will be completed coherently only if the reset occurs during the second write bus cycle. If reset occurs during the first write bus cycle, only the first word will be written to the SRAM array and the second write will not be allowed to occur. In this case, the long word data contained in the SRAM will not be coherent. The first word will contain the most significant half of the new long word information and the second word will contain the least significant half of the old long word information.

If a reset is generated by an asynchronous reset such as the loss of clocks or software watchdog time-out, the contents of the standby SRAM array are not guaranteed.

11.4.4 STOP Operation

The assertion of the STOP control bit of the RAMMCR (see **11.3.1 Module Configuration Register (RAMMCR)**) causes the SRAM module to enter its lowest power consuming state. The register block may still be accessed to allow the STOP control bit to be cleared and the array base address registers to be updated, see **11.3.2 Array Base Address Registers (RAMBAH, RAMBAL)**.

When in stop mode, the SRAM array can not be read or written. All data in the array will be retained. Switching to V_{STBY} will occur as normal if V_{DDL} drops below its specified value when the SRAM module is in stop mode.

11.4.5 Overlay Operation

The four 512-byte SRAM blocks can be used independently of the main 8K SRAM array. The blocks can be initialized to be continuous with the main array or can be used to overlay the flash module. The overlay feature is enabled whenever an overlay SRAM module base address is mapped over the flash array address space. The 512-byte SRAM block should be placed on a 512-byte boundary and will respond to any access to the overlayed section of the flash and will disable the flash contents from being read.

SECTION 12
MASK ROM MODULE

12.1 Introduction

The mask ROM module for the Modular Embedded Controller Family is designed to be used with the family's inter-module bus (IMB3) and consequently any CPU capable of operating on it. The mask ROM module implementation for the MC68F375 is 8,192 (8K) bytes.

The array is arranged in a 16-bit configuration and is accessed via the device's internal bus. It may be read as either bytes, aligned words or misaligned words. Access times depend on the number of WAIT states specified at mask programming time, but can be as fast as 2 system clocks for byte and aligned word access. It is also capable of responding to back-to-back IMB3 accesses to provide 2 bus cycle (4 system clocks) access for aligned long word or misaligned word operations, and 3 bus cycles for misaligned long words. The ROM module may be used to contain program information only, or both program and data information.

The ROM module can be used as fast access memory to contain program code which must execute at high speed, or which gets executed often. Operating system kernels and standard subroutines benefit from this fast access time. It can also be programmed to insert WAIT states to accommodate migration from slower external development memory to on-chip ROM, without the need for retiming the system. The ROM module may be configured to generate bootstrap information on RESET, without the array being mapped to location 0x000000.

The ROM module can also operate in a special emulation mode, which simplifies emulation of the internal ROM by an external device, when used with a system integration module which makes use of the ICSMB IMB3 line.

12.2 Mask Programmable Options

Along with the contents of the ROM array, several configuration options must be specified by the customer. These options are mask programmed on the same mask layer as the contents of the array. The options comprise:

- Default reset state of the base address of the array.
- Default reset state of the $\overline{\text{BOOT}}$ control bit which determines if the ROM responds to bootstrap addresses.
- Default reset state of the LOCK control bit which controls write access to configuration registers.
- Default reset state of the WAIT field which controls the number of clocks for ROM accesses.
- Default reset state of the ASPC field which specifies the address space of the ROM array.

- The value of SIGHI and SIGLO which is the ROM signature pattern.
- The default values for the ROM bootstrap information words, ROMBS[0:3]."

12.3 Programmer's Model

The ROM module consists of two separately addressable sections. The first is a memory mapped control register block used for control and configuration information of the ROM module. The second section is the array itself.

12.3.1 ROM Control Block

A 32-byte control block contains registers which are used to control ROM module operation and provide configuration information about the ROM pattern contained in the module. Configuration information is specified and programmed at the same time as the contents of the ROM array and on the same mask layer.

The configuration information contained in this block includes array base address information, bootstrap information and ROM verification information. Control bits are provided to control array operation. A 19-bit field in the control block section contains a signature used for identification and verification of the particular ROM pattern contained in the ROM array, see **Table 12-1**.

12.3.1.1 ROM Module Control Block Addressing

The control block is restricted to supervisor data space. Unimplemented or reserved addresses will return 0's for read accesses. Write accesses to unimplemented or reserved control block addresses will have no effect. Accesses to unimplemented or reserved locations will result in bus error (IBERR) being asserted.

12.3.2 ROM Array

The base address of the memory block which the ROM array resides in is specified in the array base address registers.

The default reset address of the ROM array in the address map of the system is specified by the customer at ROM programming time. The only restrictions on the base address is that it must be on a 8-Kbyte boundary and not overlap the module control register block in the data space memory map. Accesses to unimplemented locations in the address block will be ignored, allowing another internal module or external device to respond.

If the base address is set such that the ROM array overlaps the control register block of the ROM, accesses to the 32 bytes in the array that overlap will be ignored, allowing the control block to remain accessible.

Note that this is only true with respect to the ROM module. If the control register blocks of other modules are overlapped by the ROM array, accesses to the overlapped addresses of other modules will be indeterminate.

12.3.2.1 ROM Array Addressing

The array may be specified to reside in supervisor space to restrict access to supervisor only, or it may be specified to exist in unrestricted space to allow access by both user and supervisor programs. The array may also be configured to respond to program space accesses only or to both data and program space accesses.

The BIU of the ROM module compares internal address [23:13] of the IMB3 with ROMBAH, ROMBAL[23:13]. If they match, the remaining address bits and ISIZ[1:0] are used to access the ROM location in the array. Function codes are also checked for the correct access rights. If the array is specified to exist in supervisor space, user accesses will be ignored allowing an external device to respond to the address. If the array is specified to exist in unrestricted space, it will respond to both user and supervisor accesses. If the array is specified to exist in program space only, data space accesses to the array will be ignored allowing implementation of separate data and program space address maps."

12.4 ROM Module Control and Configuration Registers

This section describes the ROM control and configuration registers. Each register field is described in detail with regard to operation. The ROM module register map is shown in **Table 12-1**.

Table 12-1 ROM Module Register Map

Access	Address[1]	15 8	7 0
S	0xYF F820	ROM Module Configuration Register (ROMMCR) See **Table 12-2** for bit descriptions.	
S	0xYF F824	ROM Base Address High Register (ROMBAH) See **Table 12-3** for bit descriptions.	
S	0xYF F826	ROM Base Address Low Register (ROMBAL) See **Table 12-3** for bit descriptions.	
S	0xYF F828	ROM Signature High Register (SIGHI) See **Table 12-3** for bit descriptions.	
S	0xYF F82A	ROM Signature Low Register (SIGLO) See **Table 12-3** for bit descriptions.	
S	0xYF F830, 0xYF F832, 0xYF F834, 0xYF F836	ROM Bootstrap Information Words (ROMBS[0:4])	

NOTES:
1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SCIM2E configuration register (SCIMMCR).

12.4.1 Module Configuration Register (ROMMCR)

The ROM module configuration register is used to control the operation of the ROM array and provide status information. It also provides the production tester and the customer with configuration information.

ROMMCR — ROM Module Configuration Register **0xYF F820**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	RESERVED		BOOT	LOCK	EMUL	ASPC[1:0][2]		WAIT[1:0][3]		RESERVED					
RESET:															
0	0	0	U[1]	U[1]	0	U[1]	U[1]	U[1]	U[1]	0	0	0	0	0	0

NOTES:
1. The default state of this bit is defined by customer specified options.
2. Indicates bits protected by LOCK and STOP.
3. Indicates bits protected by LOCK only.

Table 12-2 ROMMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Inverted state of D[14]. If STOP is set to a 1 by the inverted D[14] during master reset, the array may be re-enabled by clearing STOP after master reset. If bootstrap mode is enabled, ($\overline{BOOT}$ = 0), clearing STOP will not cause the ROM to enter bootstrap mode. The STOP bit must be set in order to change the value of the array base address (ROMBAH, ROMBAL), the state of the EMUL control bit, or the value of the ASPC field in ROMMCR. Clearing the STOP bit also deactivates emulation mode, but does not disable it. 0 = The ROM module is in normal mode of operation. 1 = Causes the ROM module to enter STOP mode.
14:13	—	Reserved
12	$\overline{BOOT}$	Bootstrap enable. At mask programming time, the ROM module may be specified to function as a bootstrap ROM after RESET or only as a ROM array at a specified base address. The $\overline{BOOT}$ bit is forced to its default reset state by master reset. The default reset state of the $\overline{BOOT}$ bit is specified by the user at mask programming time and is programmed on the same mask layer as the contents of the array. 0 = ROM module will respond to the bootstrap addresses after RESET. 1 = ROM module will not respond to the bootstrap addresses after RESET.
11	LOCK	Lock registers. Once the LOCK bit is set, via an IMB3 write, it cannot be cleared again until after a master reset. ***If the default reset state is 1, all registers and bits protected by the LOCK bit can never be changed.*** The LOCK bit is forced to its default reset state by master reset. The default reset state of the LOCK bit is specified by the user at mask programming time and is programmed on the same mask layer as the contents of the array. To ensure that inadvertent reconfiguration of the ROM cannot occur, the user's initialization program should always write this bit to a one to invoke the write lock mechanism, if it's default reset state is 0. 0 = Write lock is disabled. 1 = Write-locked registers are protected.
10	EMUL	Emulation mode. Emulation mode allows the ROM array to be emulated externally, with access controlled by the ROM module BIU. Bootstrap operation is not affected by emulation mode, nor is it emulated. Emulation mode may be entered by writing EMUL to a 1. The STOP bit in the ROMMCR must be a 1 in order to change the state of the EMUL bit by writing to it via the IMB3. In emulation mode the ROM will not respond to an array address with IAACKB, however, it will respond to control register accesses. Instead of responding with IAACKB to array addresses, it will assert the ICSMB line on the IMB3, allowing the device's external bus interface to assert an external chip select and run a special external bus cycle which is terminated by the ROM asserting IDTACKB on the IMB3. The ROM will assert IDTACKB after the proper number of WAIT states, as specified in the WAIT field of the ROMMCR register. The ROM module will not drive the IMB3 data lines however. Data must be provided by an external device at the proper time. See **4.7.8.6 Emulation Mode Selection**. 0 = The ROM module is in normal mode of operation. 1 = Causes the ROM module to enter emulation mode.

Table 12-2 ROMMCR Bit Settings (Continued)

Bit(s)	Name	Description
9:8	ASPC[1:0]	ROM array space. The ASPC[1:0] field is forced to the default reset state by master reset. The default reset state of the ASPC[1:0] field is specified by the user at mask programming time and is programmed on the same mask layer as the contents of the array. IFC[2:0] is checked by the ROM module BIU to determine if a user or supervisor is requesting array access. If a supervisor program is accessing the array, normal read operation will occur. If a user program is attempting to access the array, the access will be ignored and the address may be decoded externally. 00 = Unrestricted program and data space IFC[2:0] = x01, x10. 01 = Unrestricted program space only. IFC[2:0] = x10. 10 = Supervisor data and program space only. IFC[2:0] = 101, 110. 11 = Supervisor program space only. IFC[2:0] = 110.
7:6	WAIT[1:0]	Wait states. The WAIT field is used to specify the number of WAIT states inserted by the ROM BIU during accesses to the ROM module before asserting IDTACKB. A WAIT state has a duration of one system clock cycle. This affects both control block access and array access. This feature allows the migration of storage space from a slower emulation or development system memory to the onboard ROM module without the need for retiming the system. The default reset state of the WAIT field is specified by the user at mask programming time and is programmed on the same mask layer as the contents of the array. The WAIT field may be written any time the LOCK bit is 0. 00 = 3 clocks per transfer. 01 = 4 clocks per transfer 10 = 5 clocks per transfer 11 = 2 clocks per transfer
5:0	—	Reserved

12.4.2 ROM Base Address Register (ROMBAH, ROMBAL)

The default reset ROM base address fields are specified by the user along with the contents of the array and are programmed on the same mask layer as the contents of the array.

ROMBAH — ROM Base Address High Register **0xYF F824**

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	LSB 16
RESERVED								ROMBAH[1]							
RESET:															
0	0	0	0	0	0	0	0	U[2]	U[2]	U[2]	U[2]	U[2]	U[2]	U[2]	U[2]

NOTES:
1. Indicates bits protected by LOCK and STOP. The default state of these bits is defined by customer-specified options.
2. The default state of these bits is defined by customer specified options.

ROMBAL — ROM Base Address Low Register **0xYF F826**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
ROMBAL[1]				RESERVED											
RESET:															
U[2]	U[2]	U[2]	U[2]	0	0	0	0	0	0	0	0	0	0	0	0

NOTES:
1. Indicates bits protected by LOCK and STOP. The default state of these bits is defined by customer-specified options.

2. The default state of these bits is defined by customer specified options.

Table 12-3 BAR (ROMBAH, ROMBAL) Bit Settings

Bit(s)	Name	Description
31:24	—	Reserved
23:16	ROMBAH	The 32-bit base address of the ROM array memory address block is contained in the ROMBAH and ROMBAL array base address registers. The base address register (BAR) is formed by concatenating the contents of ROMBAH and ROMBAL. ROMBAH contains the high order 16 bits of the address (A[31:16]) and ROMBAL contains the next lower order bits. The value of ROMBAH and ROMBAL are forced to the user defined value programmed in ROM shadow registers on master RESET. ROMBAH and ROMBAL can be written to relocate the ROM array to an alternate block of memory only when the LOCK bit is 0 and the ROM module is in STOP mode..
15:12	ROMBAL	
11:0	—	Reserved

12.4.3 ROM Signature High (SIGHI)

SIGHI — ROM Signature High Register **0xYF F828**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RESERVED													RSP18	RSP17	RSP16
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	U[1]	U[1]	U[1]

NOTES:
1. The default state of these bits is defined by customer specified options.

Table 12-4 SIGHI Bit Settings

Bit(s)	Name	Description
15:3	—	Reserved
2:0	RSP[18:16]	ROM signature pattern. These 3 bits, when concatenated with the 16 bits contained in SIGLO, form a 19-bit unique signature used to verify the contents of the ROM array. This information is programmed along with the contents of the array, on the same mask layer.

12.4.4 ROM Signature Low (SIGLO)

SIGLO — ROM Signature Low Register **0xYF F82A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
RSP15	RSP14	RSP13	RSP12	RSP11	RSP10	RSP9	RSP8	RSP7	RSP6	RSP5	RSP4	RSP3	RSP2	RSP1	RSP0
RESET:															
U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]

NOTES:
1. The default state of these bits is defined by customer specified options.

Table 12-5 SIGLO Bit Settings

Bit(s)	Name	Description
15:0	RSP[15:0]	ROM signature pattern. These 16 bits, when concatenated with the 3 bits contained in SIGHI, form a 19-bit unique signature used to verify the contents of the ROM array. This information is programmed along with the contents of the array, on the same mask layer.

12.5 Bootstrap Information Words (ROMBS0–ROMBS3)

Typically, reset vectors for the system CPU are contained in non-volatile memory and are only fetched when the CPU comes out of reset. The bootstrap information for the processor controlling the system are contained in the ROM module in the four words ROMBS0–ROMBS3. ROMBS0 responds to address 0x000000, ROMBS1 responds to 0x000002, ROMBS2 to 0x000004 and ROMBS3 to 0x000006 on the IMB3. The bootstrap information is specified by the user along with the contents of the array and is programmed along with the contents of the array, on the same mask layer. These registers are read only from the IMB3 in normal mode or bootstrap mode. IMB3 writes do not affect the contents of these registers. In bootstrap mode, ROMBS0–ROMBS3 only respond to supervisor program space accesses. In normal mode, they only respond to supervisor data space accesses.

ROMBS0 — ROM Bootstrap Word 0 **0xYF F830**

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	LSB 16
SP[31:16]															
RESET:															
U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]

NOTES:
1. The default state of these bits is defined by customer-specified options.

ROMBS1 — ROM Bootstrap Word 1 **0xYF F832**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
SP[15:0]															
RESET:															
U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]

NOTES:
1. The default state of these bits is defined by customer-specified options.

ROMBS2 — ROM Bootstrap Word 2 **0xYF F834**

MSB 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	LSB 16
PC[31:16]															
RESET:															
U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]

NOTES:
1. The default state of these bits is defined by customer-specified options.

ROMBS3 — ROM Bootstrap Word 3 **0xYF F836**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
PC[15:0]															
RESET:															
U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]	U[1]

NOTES:
1. The default state of these bits is defined by customer-specified options.

12.6 Operation

The ROM module is accessed via the IMB3 by a bus master. It can be used to contain either program information only or both data and program information. On master reset, it can operate as a bootstrap ROM to provide CPU internal initialization information during the CPU's reset sequence or can be configured to never respond to the bootstrap addresses.

12.6.1 RESET Operation

The ROM uses master RESET to initialize all register bits to their reset values. The LOCK bit will also be cleared if it's default reset state is 0. During master reset, the ROM BIU will monitor three inputs to the module (STOPIN, EMULIN, and EMULEN) to determine if it should respond normally after reset or disable itself for testing purposes, and if emulation mode is enabled.

The value of STOPIN is determined by the state of D[14] during master reset. If the state of STOPIN is 1, the STOP bit in the ROMMCR register will be cleared to 0 and the array will respond normally to the bootstrap address range and the ROM array base address. If STOPIN is 0, the STOP bit will be set and the ROM array will be disabled until the STOP bit is cleared either by an IMB3 write or until the next master reset which occurs with STOPIN = 1. It will not respond to the bootstrap address range or the ROM array base address in BAR (ROMBAH and ROMBAL), allowing an external device to respond to the ROM array's address space, and/or provide bootstrap information. This allows the ROM to be disabled from outside of the device if necessary.

The value of EMULIN is determined by the state of D[10] and the value of EMULEN is determined by the state of D[13] during master reset. If the state of either EMULIN or EMULEN is 1, the EMUL bit in the ROMMCR register will be cleared to 0, ROM emulation mode will not be enabled, and the array will respond normally to valid accesses. If EMULIN and EMULEN are both 0, the EMUL bit in ROMMCR will be set and ROM emulation mode will be enabled until the EMUL bit is cleared by either an IMB3 write or the next master reset occurs with either EMULIN or EMULEN =1.

STOPIN, EMULIN and EMULEN are forced to the value of external pins during master reset. These pins may be data pins for devices that have an external data bus, in which case STOPIN, EMULIN and EMULEN will be driven by corresponding IMB3 lines. This function is performed by the SCIM2E.

Bootstrap mode will be disabled if STOP is set during master reset, regardless of the default reset state of $\overline{\text{BOOT}}$.

12.6.2 Bootstrap Operation

If $\overline{\text{BOOT}}$ = 0 in the ROMMCR, the ROM module may be used as a bootstrap ROM by the system. The ROM module will only respond to the bootstrap addresses if the STOP control bit in the ROMMCR register is a 0. If STOP is a 1, bootstrap addresses will be ignored. Subsequently clearing the STOP bit will not cause the ROM to enter bootstrap mode.

In bootstrap mode, the ROM module will respond only to the bootstrap addresses in supervisor program space, and provide the information contained in the ROMBS0–ROMBS3 registers. Bootstrap mode will terminate automatically after the last word of bootstrap information, ROMBS1 or ROMBS3 is fetched. On the next system clock, the ROM module will begin responding to control block and array addresses only. If $\overline{\text{BOOT}}$ = 1, the ROM module will only respond to its control block address and the array base address specified in ROMBAH and ROMBAL.

12.6.3 Normal Operation

The control registers and the array may be accessed via the IMB3 as byte, and aligned or misaligned word. The ROM array will respond to read operations only. Write operations will be ignored. **12.6.5 Emulation Mode Operation** describes the read/write operation in emulation mode.) If ASPC[1] =1, the ROM will only respond to supervisor mode reads. If ASPC[1] = 0, the ROM will respond to both supervisor and user reads. If ASPC[0] = 1, the array will respond only to program space accesses. If ASPC[0] = 0, the ROM array will respond to both program and data space accesses.

Access to any address which falls between the last byte of the array and the end of the address block containing the array (specified by ROMBAH and ROMBAL), will be ignored, allowing external devices or other modules to adjoin the ROM array in the address map.

Accesses to the ROM array are only allowed if the STOP control bit in the ROMMCR register is a 0. If STOP is a 1, ROM array accesses and bootstrap accesses will be ignored. This allows an external device, or another internal memory module, to respond to the array address range.

12.6.4 Read/Write Access

The ROM module allows a byte or aligned word read/write in one IMB3 bus cycle. Long word read/write or misaligned word read/write will require an additional bus cycle. Misaligned long word read/write will require a total of 3 bus cycles.

An IMB3 bus cycle requires 2 system clocks minimum. See **SECTION 4 SINGLE-CHIP INTEGRATION MODULE 2 (SCIM2E)** for detailed timing of read/write accesses.

Write operations are only allowed to the control block. Write accesses to the array will be ignored (IAACKB will not be asserted) unless the ROM is in emulation mode and the device is in FREEZE mode. **12.6.5 Emulation Mode Operation** describes array read/write operation in emulation mode.)

12.6.5 Emulation Mode Operation

The ROM module has a special emulation mode which allows external emulation of the ROM array, with the module itself performing address decode and bus cycle termination for the external memory which replaces the array. Bootstrap operation is not affected by emulation mode, nor is it emulated.

NOTE

To emulate bootstrap operation, the emulation system should monitor another chip select which is programmed to assert CSBOOT on the SCIM2E, or the emulation system must monitor the external bus to determine when a bootstrap vector is being requested by the CPU.

STOP affects array operation in emulation mode, the same way it does in normal operation. Accesses to the ROM module control registers are unaffected by emulation mode.

Emulation mode is enabled by the EMUL bit in ROMMCR. EMUL can be written via the IMB3 if STOP = 1, and it's reset state is controlled by the state of the EMULIN and EMULEN inputs to the module during master reset. EMULIN and EMULEN are driven from external pins during RESET. If both EMULIN and EMULEN are low during reset, the EMUL bit will be set in ROMMCR and ROM emulation mode will be enabled. If either input is high during reset, EMUL is cleared and ROM emulation mode is disabled.

EMULIN should be connected to a signal which is used to select whether or not the ROM module should be placed in emulation mode, along with the external bus interface. EMULEN should be connected to the same signal used to enable emulation mode for the external bus interface. In this way, the ROM module cannot be placed in emulation mode, unless the external bus interface is also in emulation mode. It also allows the external bus interface to be placed in emulation without placing the ROM module in emulation mode.

In emulation mode, the ROM will assert the ICSMB line on the IMB3 whenever an access is made to a valid ROM array location, instead of asserting IAACKB. Valid array access corresponds to an array location within the address range indicated by the value in the ROMBAH and ROMBAL registers, and FC[2:0] indicate address space requirements specified in ASPC[1:0] field of the ROMMCR register. The ROM will assert ICSMB on all read accesses and will also assert ICSMB on write accesses if the IFREEZE line on the IMB3 is asserted. The ROM will not assert the IAACKB line if it asserts ICSMB. In response to ICSMB, the external bus interface of the device will assert CSM on the SCIM2E, indicating an emulation mode access to the array, and will run an special external bus cycle which will be terminated by the internal signal IDTACKB instead of DSACKx. The ROM module will assert IDTACKB to terminate the

cycle at the proper time, based on the number of WAIT states programmed into the WAIT field in the ROMMCR register. The ROM will not drive the IMB3 data bus, since the data for the cycle will come from the external data pins of the device.

Table 12-6 Minimum ROM Module Access Times

Type of Access	Bus Cycles for Read[1]	Number of System Clocks
Byte	1	2
Aligned Word	1	2
Misaligned Word	2	4
Aligned Long Word	2	4
Misaligned Long Word	3	6

NOTES:

1. Access time is shown for 2 clock access, (WAIT[1:0] = 11). An additional clock must be added for each additional WAIT state programmed in WAIT.

12.6.6 Stop Operation

If the Stop bit is asserted, the ROM module will not respond with IAACKB to any attempts to access the array or the bootstrap information in bootstrap mode. Only the control register block may be accessed at its normal address. The ROM module must be in STOP mode in order to allow the EMUL control bit to be changed via an IMB3 write. The ROM module must be in STOP mode and LOCK = 0 in order to allow the ROMBAH and ROMBAL registers, and the ASPC field of ROMMCR to be written. STOP also disables bootstrap mode if it set during master reset.

12.6.7 FREEZE Operation

Although the ROMMCR register does not contain a FREEZE mode control bit, FREEZE mode will affect ROM emulation mode operation. The ROM module monitors the IFREEZE line on the IMB3 when the ROM is in emulation mode. If IFREEZE is in its asserted state when the ROM is in emulation mode, the ROM module will respond to write accesses to the array as well as read accesses, see **12.6.5 Emulation Mode Operation**

SECTION 13
CONFIGURABLE TIMER MODULE (CTM9)

13.1 Introduction

The configurable timer module (CTM9) is a family of timer modules for the Motorola modular microcontroller family (MMF), including the MC68300 (CPU32) and the MC68HC16 (CPU16) families of microcontrollers (MCUs). The timer architecture is modular relative to the number of time-bases (counter submodules), channels (action submodules) and other available general purpose functions (real time clock, RAM, I/O ports, etc.) that can be included.

Please refer to the ***CTM Reference Manual (CTMRM/D)*** for more information.

13.1.1 CTM9 Configuration

The CTM9 is composed of the following submodules:

- 1 free-running counter submodule (FCSM).
- 2 modulus counter submodule (MCSM).
- 4 single action submodule (SASM).
- 4 double action submodule (DASM).
- 4 dedicated PWM submodule (PWMSM).
- 1 bus interface unit submodule (BIUSM).
- 1 counter prescaler submodule (CPSM).

Figure 13-1 and **Table 13-1** show respectively a block diagram and a table representation of the CTM9 configuration.

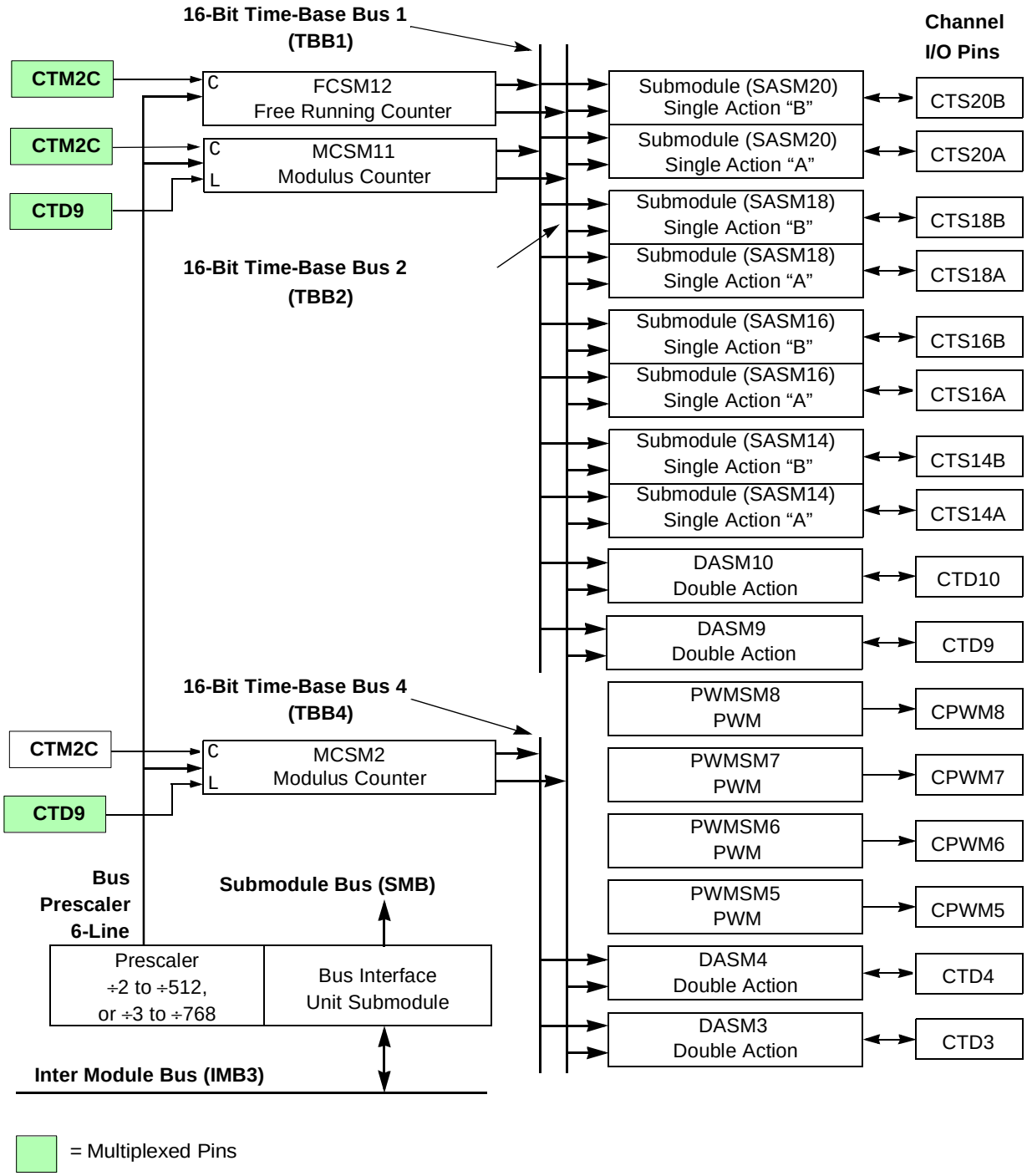


Figure 13-1 Configurable Timer Module, CTM9 Block Diagram

Table 13-1 CTM9 Configuration Description

Submodule Type	Submodule Number	Connected to: tbb 1 (A)	tbb 2 (B)	tbb 3 (B)	tbb 4 (A)	Binary Interrupt Vector Number	Submodule Base Address	Pin Function	Input Pin Name	Output Pin Name
BIUSM+CPSM	0						0xYF F200			
MCSM	2[1]	0	1	0	1	0bxx000010[2]	0xYF F210[3]	Clock	CTM2C	
								Load	CTD9	
DASM	3	0	1	0	1	0bxx000011	0xYF F218	Channel	CTD3	CTD3
DASM	4	0	1	0	1	0bxx000100	0xYF F220	Channel	CTD4	CTD4
PWMSM	5					0bxx000101	0xYF F228	Channel		CPWM 5
PWMSM	6					0bxx000110	0xYF F230	Channel		CPWM 6
PWMSM	7					0bxx000111	0xYF F238	Channel		CPWM 7
PWMSM	8					0bxx001000	0xYF F240	Channel		CPWM 8
DASM	9	1	1	0	0	0bxx001001	0xYF F248	Channel	CTD9	CTD9
DASM	10	1	1	0	0	0bxx001010	0xYF F250	Channel	CTD10	CTD10
MCSM	11	1	1	0	0	0bxx001011	0xYF F258	Clock	CTM2C	
								Load	CTD9	
FCSM	12	1	1	0	0	0bxx001100	0xYF F260	Clock	CTM2C	
SASM	14	1	1	0	0	0bxx001110	0xYF F270	Channel A	CTS14 A	CTS14 A
						0bxx001111		Channel B	CTS14 B	CTS14 B
SASM	16	1	1	0	0	0bxx010000	0xYF F280	Channel A	CTS16 A	CTS16 A
						0bxx010001		Channel B	CTS16 B	CTS16 B
SASM	18	1	1	0	0	0bxx010010	0xYF F290	Channel A	CTS18 A	CTS18 A
						0bxx010011		Channel B	CTS18 B	CTS18 B
SASM	20	1	1	0	0	0bxx010100	0xYF F2A0	Channel A	CTS20 A	CTS20 A
						0bxx010101		Channel B	CTS20 B	CTS20 B

NOTES:

1. Interrupt arbitration priority goes in descending order with submodule #2 having the highest priority and the last interrupting submodule in the table having the lowest priority.
2. xx are the two VECT[7:6] bits contained in the BIUSM.
3. Y=m111, where m is the state of the modmap bit of the MCR of the SIM (Y=0x7 or 0xF).

13.1.2 CTM9 Pins and Naming Convention

The CTM9 uses 17 pins. The usage of these pins is shown in **Figure 13-1** and **Table 13-1**. The CTM9 digital input and output pin names are composed of three sections according to the following convention:

<submodule prefix><submodule number><submodule suffix (optional)>

The pin prefix and suffix for the different submodules used in the CTM9 are as follows:

- FCSM pin name:
  <submodule prefix>: “CTF”
  <submodule suffix>: none
  For example an FCSM placed as submodule number n would have its corresponding input clock pin called: CTFn
- MCSM pin name:
  <submodule prefix>: “CTM”
  <submodule suffix>: C for the Clock pin
  <submodule suffix>: L for the Load pin
  For example an MCSM placed as submodule number n would have its corresponding input clock pin named: CTMnC and its input load pin called CTMnL.
- SASM pin name:
  <submodule prefix>: “CTS”
  <submodule suffix>: A for channel A I/O
  <submodule suffix>: B for channel B I/O
  For example an SASM placed as submodule number n would have its corresponding channel A I/O pin named: CTSnA and its channel B I/O pin called: CTSnB.
- DASM pin name:
  <submodule prefix>: “CTD”
  <submodule suffix>: none
  For example a DASM placed as submodule number n would have its corresponding I/O pin called: CTDn
- PWMSM pin name:
  <submodule prefix>: “CPWM”
  <submodule suffix>: none
  For example a PWMSM placed as submodule number n would have its corresponding output pin called: CPWMn

In the CTM9, some pins are multiplexed between submodules. **Table 13-1** shows using the same pin names for the inputs or outputs which are connected together.

13.2 Free Running Counter Submodule (FCSM)

The free-running counter submodule (FCSM) has a 16-bit up counter with an associated clock source selector, selectable time-base bus drivers, software writable control registers, software readable status bits, and interrupt logic. When the 16-bit up counter overflows from 0xFFFF to 0x0000, an optional overflow interrupt is available to the software. The current state of the 16-bit counter is the primary output of the counter submodules. The software selects which, if any, time-base bus is to be driven by the

16-bit counter. A software control register selects whether the clock input to the counter is one of the taps from the prescaler or an input pin. The polarity of the external input pin is also programmable. The free-running counter submodule operation is comparable to the MC68HC11 counter.

A block diagram of the FCSM is shown in **Figure 13-2**. The main components of the FCSM are a 16-bit loadable free-running up-counter, a clock selector, a time base bus driver and an interrupt interface.

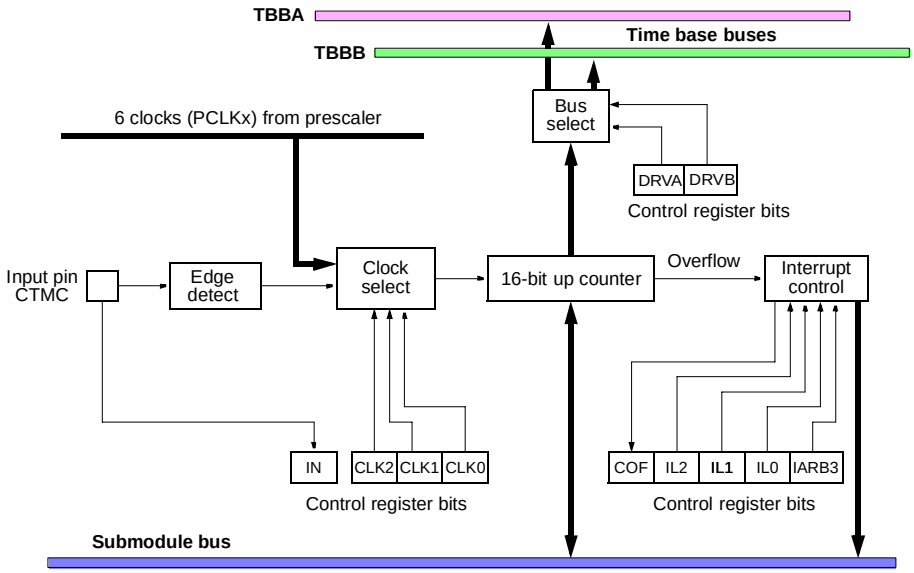


Figure 13-2 FCSM Block Diagram

NOTE

In order to be able to count, the FCSM requires the CPSM clock signals to be present. On coming out of reset, the FCSM will not count internal or external events until the prescaler in the CPSM starts running (when the software sets the PRUN bit). This allows all counters in the CTM submodules to be synchronized.

13.2.1 The FCSM Counter

The FCSM counter section comprises a 16-bit register and a 16-bit up-counter. Reading the register transfers the contents of the counter to the data bus, while a write to the register loads the counter with the new value. Overflow of the counter is defined to be the transition from 0xFFFF to 0x0000. An overflow condition causes the COF flag bit in the FCSMSIC register to be set.

NOTE

Reset presets the counter register to 0x0000. Writing 0x0000 to the counter register while the counter's value is 0xFFFF does not set the

COF flag and does not generate an interrupt request.

13.2.2 FCSM Clock Sources

The user can choose from eight software selectable counter clock sources:

- Six prescaler outputs (PCLKx)
- Input pin rising edge detection on the input pin CTMC
- Input pin falling edge detection on the input pin CTMC

The clock source is selected by the CLK[2:0] bits in the FCSM status, interrupt and control register FCSMSIC (see **13.2.7.1 FCSMSIC — FCSM Status/Interrupt/Control Register**). When the CLK[2:0] bits are being changed, internal circuitry ensures that spurious edges occurring on the CTMC pin do not affect the FCSM.

Note that the read-only IN bit of the FCSMSIC register reflects the state of the input pin CTMC. The input pin is Schmitt triggered and is synchronized with the system clock (f_{SYS}).

13.2.3 FCSM External Event Counting

When an external clock source (on the input pin) is selected, the FCSM is in the event counter mode. The counter can simply count the number of events occurring on the input pin. Alternatively, the FCSM can be programmed to generate an interrupt when a predefined number of events have been counted; this is done by presetting the counter with the two's complement value of the desired number of events. When using the external clock source, the maximum guaranteed external frequency is $f_{SYS}/4$.

13.2.4 The FCSM Time Base Bus Driver

The DRVA and DRVB bits in the FCSMSIC register select the time base buses to be driven (see **13.2.7.1 FCSMSIC — FCSM Status/Interrupt/Control Register**).

WARNING

It is not recommended that the two time base buses be driven at the same time.

13.2.5 FCSM Interrupts

A valid FCSM interrupt can be generated when the COF bit in the FCSMSIC register is set (as a result of the counter overflowing). If the interrupt priority level of the FCSM is non-zero, as defined by the three IL bits in the FCSMSIC register, a valid interrupt request will occur on the IMB.

13.2.6 Freeze Action on the FCSM

When the IMB FREEZE signal is recognized, the FCSM counter stops counting and remains set at its current value. When the FREEZE signal is negated, the counter starts incrementing from its current value, as if nothing had happened. All registers are accessible during freeze.

During freeze, the IN bit in the FCSMSIC register continues to reflect the state of the signal on the input pin CTMC (see **13.2.7.1 FCSMSIC — FCSM Status/Interrupt/ Control Register**).

13.2.7 FCSM Registers

The FCSM register map comprises four 16-bit register locations. As shown in **Table 13-2**, the register block contains two FCSM registers and two reserved registers. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no effect. In CTM implementations featuring multiple FCSMs, each FCSM has its own set of registers.

NOTE

All register addresses in this section are offsets from the base address of the FCSM.

Table 13-2 FCSM Register Map

Address	15 — 8	7 — 0
0xYF F260	Status, interrupt and control register (FCSMSIC)	
0xYF F262	Counter register(FCSMCNT)	

13.2.7.1 FCSMSIC — FCSM Status/Interrupt/Control Register

FMSMSIC — FCSM Status/Interrupt Control Register **0xYF F260**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
COF	IL2	IL1	IL0	IARB3	0	DRVA	DRVB	IN	0	0	0	0	CLK2	CLK1	CLK0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-3 FMSMSIC Bit Settings

Bit(s)	Name	Description
15	COF	Counter overflow flag. This status flag bit indicates whether or not a counter overflow has occurred. An overflow is defined to be the transition of the counter from 0xFFFF to 0x0000. If the IL field is non-zero, an interrupt request is generated when the COF bit is set. The flag clearing mechanism will work only if no flag setting event occurs between the read and write operations; if a COF setting event occurs between the read and write operations, the COF bit will not be cleared. 0 = Counter overflow has not occurred. 1 = Counter overflow has occurred.
14:12	IL[2:0]	Interrupt level. The three interrupt level bits are read/write control bits that select the priority level of interrupt requests made by the FCSM. These bits can be read or written at any time and are cleared by reset. 000 = Interrupt disabled 001 = Interrupt level 1 (lowest) 010 = Interrupt level 2 011 = Interrupt level 3 100 = Interrupt level 4 101 = Interrupt level 5 110 = Interrupt level 6 111 = Interrupt level 7 (highest)
11	IARB3	The read/write IARB3 bit works in conjunction with the IARB[2:0] field in the BIUSM module configuration register. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field (IARB). This interrupt arbitration identification number is used to arbitrate for the IMB when modules generate simultaneous interrupts of the same priority.
10	—	Reserved
9:8	DRV[A:B]	Drive time base bus. DRVA and DRVB are read/write bits that control the connection of the FCSM to the time base buses A and B. These bits are cleared by reset. It is recommended that the two time base buses **not** be driven at the same time. 00 = Neither time base bus A nor time base bus B is driven. 01 = Time base bus B is driven 10 = Time base bus A is driven 11 = Both time base bus A and time base bus B are driven
7	IN	Input pin status. This read-only status bit reflects the logic state of the FCSM input pin CTMC. Writing a ‘zero’ or a ‘one’ to this bit has no effect. Reset has no effect on this bit.
6:3	—	Reserved
2:0	CLK[2:0]	Counter clock select. These read/write control bits select one of six internal clock signals (PCLKx) or one of two external conditions on the input pin (rising edge or falling edge). The maximum frequency of the external clock signals is $f_{SYS}/4$. 000 = Prescaler output 1 (/2 or /3) 001 = Prescaler output 2 (/4 or /6) 010 = Prescaler output 3 (/8 or /12) 011 = Prescaler output 4 (/16 or /24) 100 = Prescaler output 5 (/32 or /48) 101 = Prescaler output 6 (/64 to /512 or /96 to /768) 110 = CTMC pin input, negative edge 111 = CTMC pin input, positive edge

13.2.7.2 FCSMCNT — FCSM Counter Register

FCSMCNT — FCSM Counter Register **0xYF F262**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The FCSM counter register is a read/write register; it is cleared by reset.

13.3 Modulus Counter Submodule (MCSM)

The modulus counter submodule (MCSM) is an enhancement of the free-running counter. A modulus register gives the additional flexibility of recycling the counter at a count other than 64K clock cycles. The state of the modulus register is transferred to the counter under three conditions:

1. When an overflow occurs.
2. When an appropriate transition occurs on the external load pin.
3. When the program writes to the counter register. In this case, the value is first written into the modulus register and immediately transferred to the counter.

Software can also write a value to the modulus register for later loading into the counter with one of the two first criteria. A block diagram of the MCSM is shown in **Figure 13-3**.

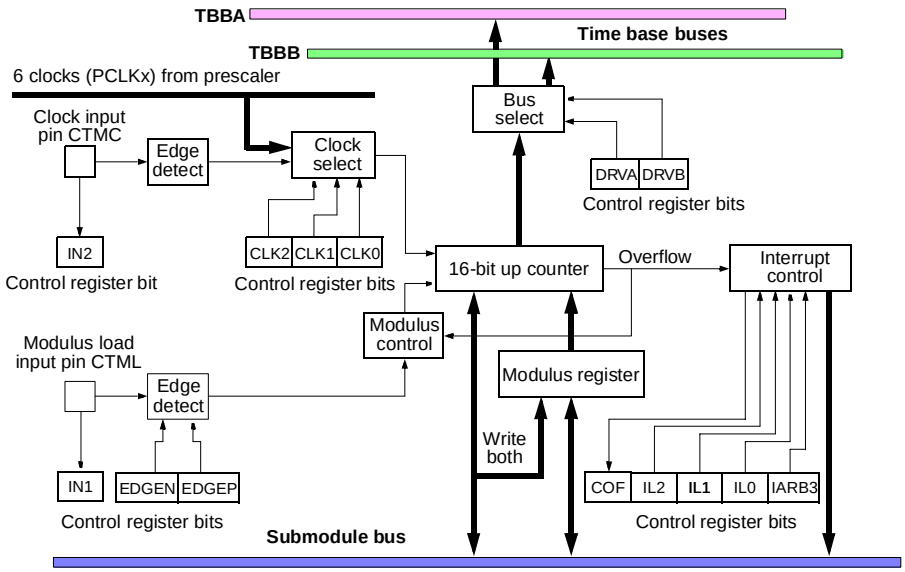


Figure 13-3 MCSM Block Diagram

The main components of the MCSM are a 16-bit modulus latch, a 16-bit loadable up-counter, counter loading logic, a clock selector, a time base bus driver and an interrupt interface.

NOTE

In order to be able to count, the MCSM requires the CPSM clock signals to be present. On coming out of reset, the MCSM will not count internal or external events until the prescaler in the CPSM starts running (when the software sets the PRUN bit). This allows all counters in the CTM submodules to be synchronized.

13.3.1 The MCSM Modulus Latch

The 16-bit modulus latch is a read/write register that is used to reload the counter automatically with a predetermined value. The contents of the modulus latch register can be read at any time. Writing to the register loads the modulus latch with the new value. This value is then transferred to the counter register on the next hardware load of that counter. However, writing to the corresponding counter register loads the modulus latch and the counter register immediately with the new value. The modulus latch register is cleared to 0x0000 by reset.

13.3.2 The MCSM Counter

The counter is composed of a 16-bit read/write register associated with a 16-bit incrementer. Reading the counter transfers the contents of the counter register to the data bus; writing to the counter loads the modulus latch and the counter register immediately with the new value. The counter can be clocked with different clock sources (see **13.3.3 MCSM Clock Sources**).

NOTE

Reset presets the counter register to 0x0000. Writing 0x0000 to the counter register while its value is 0xFFFF does not set the COF flag and does not generate an interrupt.

13.3.2.1 Loading the MCSM Counter Register

The counter register can be loaded by writing directly to it. The counter register is also loaded from the modulus latch each time a counter overflow occurs and the COF flag bit in the MCSM status/interrupt/control register (MCSMSIC) is set.

NOTE

When the modulus latch is loaded with 0xFFFF, the overflow flag is set on every counter clock pulse.

Loading of the counter register from the modulus register can also be triggered by an external event on the modulus load pin CTML. The edge on the CTML pin that triggers the loading of the counter register is selected by bits EDGEN and EDGEP in the MCSMSIC register. Hardware is provided to prevent the occurrence of spurious edges while changing the EDGEN and EDGEP bits. Reset clears the EDGEN and EDGEP

bits to zero, thereby preventing a signal on the CTML pin from loading the counter register until EDGEN and EDGEP have been initialized by the software. The modulus load input pin CTML is Schmitt triggered and synchronized to the system clock (f_{SYS}).

NOTE

The read-only IN1 bit of the MCSMSIC reflects the state of the input pin CTML.

13.3.2.2 Using the MCSM as a Free-Running Counter

The MCSM is a modulus counter. However it can be made to behave like a free-running counter by loading the modulus register with the value 0x0000.

13.3.3 MCSM Clock Sources

The User can choose from eight software selectable counter clock sources:

- Six prescaler outputs (PCLKx)
- Input pin rising edge detection on the input pin CTMC
- Input pin falling edge detection on the input pin CTMC

The clock source is selected by the CLK[2:0] bits in the MCSM status, interrupt and control register MCSMSIC (see **13.3.9 MCSMSIC — MCSM Status/Interrupt/Control Register**). When the CLK[2:0] bits are being changed, internal circuitry ensures that spurious edges occurring on the CTMC pin do not affect the MCSM. The clock input pin CTMC is Schmitt triggered and is synchronized with the system clock (f_{SYS}).

NOTE

The read-only IN2 bit of the MCSMSIC register reflects the state of the input pin CTMC.

13.3.4 MCSM External Event Counting

When an external clock source (on the CTMC input pin) is selected, the MCSM is in the event counter mode. The counter can simply count the number of events occurring on the input pin. Alternatively, the MCSM can be programmed to generate an interrupt when a predefined number of events have been counted; this is done by presetting the counter with the two's complement value of the desired number of events. When using the external clock source, the maximum external guaranteed frequency is $f_{SYS}/4$.

13.3.5 The MCSM Time Base Bus Driver

The DRVA and DRVB bits in the MCSMSIC register select the time base buses to be driven (see **13.3.9 MCSMSIC — MCSM Status/Interrupt/Control Register**).

13.3.6 MCSM interrupts

A valid MCSM interrupt can be generated when the COF bit in the MCSMSIC register is set as a result of the counter overflowing. If the interrupt priority level of the MCSM is non-zero, as defined by the three IL bits in the MCSMSIC register, a valid interrupt request will occur on the IMB.

13.3.7 Freeze Action on the MCSM

When the IMB FREEZE signal is recognized, the MCSM counter stops counting and remains set at its last value. When the FREEZE signal is negated, the counter starts incrementing from its last value, as if nothing had happened. All registers are accessible during freeze.

During freeze, the IN1 and IN2 bits in the MCSMSIC continue to reflect the states of the signals on the input pins (see **13.3.9 MCSMSIC — MCSM Status/Interrupt/Control Register**).

13.3.8 MCSM Registers

The MCSM register map comprises four 16-bit register locations. As shown in **Table 13-4**, the register block contains three FCSM registers and one reserved register. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no effect. In CTM implementations featuring multiple MCSMs, each MCSM has its own set of registers.

Table 13-4 MCSM Register Map

Address	15 — 8 | 7 — 0
0xYF F210	MCSM2 status/interrupt/control register (MCSM2SIC)
0xYF F212	MCSM2 counter (MCSM2CNT)
0xYF F214	MCSM2 modulus latch (MCSM2ML)
0xYF F258	MCSM11 status/interrupt/control register (MCSM11SIC)
0xYF F25A	MCSM11 counter (MCSM11CNT)
0xYF F25C	MCSM11 modulus latch (MCSM11ML)

13.3.9 MCSMSIC — MCSM Status/Interrupt/Control Register

MCSM2SIC — MCSM Status/Interrupt Control Register **0xYF F210**
MCSM11SIC **0xYF F258**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
COF	IL2	IL1	IL0	IARB3	0	DRVA	DRVB	IN2	IN1	EDGEN	EDGEP	0	CLK 2	CLK 1	CLK 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-5 MCSMSIC Bit Settings

Bit(s)	Name	Description
15	COF	Counter overflow flag. This status flag bit indicates whether or not a counter overflow has occurred. An overflow of the MCSM counter is defined to be the transition of the counter from 0xFFFF to 0xxxxx, where 0xxxxx is the value contained in the modulus latch. If the IL field is non-zero, an interrupt request is generated when the COF bit is set. This flag bit is set only by the hardware and cleared only by the software or by a system reset. To clear the flag, the software must first read the bit (as '1') then write a '0' to the bit. The flag clearing mechanism will work only if no flag setting event occurs between the read and write operations; if a COF setting event occurs between the read and write operations, the COF bit will not be cleared. 0 = Counter overflow has not occurred 1 = Counter overflow has occurred.
14:12	IL[2:0]	Interrupt level. The three interrupt level bits are read/write control bits that select the priority level of interrupt requests made by the MCSM. These bits can be read or written at any time and are cleared by reset. 000 = Interrupt disabled. 001 = Interrupt level 1 (lowest). 010 = Interrupt level 2. 011 = Interrupt level 3. 100 = Interrupt level 4. 101 = Interrupt level 5. 110 = Interrupt level 6. 111 = Interrupt level 7 (highest).
11	IARB3	Interrupt arbitration bit 3. The read/write IARB3 bit works in conjunction with the IARB[2:0] field in the BIUSM module configuration register. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field (IARB). This interrupt arbitration identification number is used to arbitrate for the IMB when modules generate simultaneous interrupts of the same priority.
10	—	Reserved
9:8	DRV{A:B]	Drive time base bus. DRVA and DRVB are read/write bits that control the connection of the MCSM to the time base buses A and B. 00 = Neither time base bus A nor time base bus B is driven. 01 = Time base bus B is driven. 10 = Time base bus A is driven. 11 = Both time base bus A and time base bus B are driven.
7	IN2	Clock input pin status. This read-only status bit reflects the logic state of the clock input pin CTMC. Writing a 0 or 1 to this bit has no effect. Reset has no effect on this bit.
6	IN1	Modulus load input pin status. This read-only status bit reflects the logic state of the modulus load input pin CTML. Writing a 0 or 1 to this bit has no effect. Reset has no effect on this bit.
5:4	EDGEN, EDGEP	Modulus load edge sensitivity. These read/write bits select the sensitivity of the edge detection circuitry on the modulus load pin CTML. 00 = None 01 = Positive edge only. 10 = Negative edge only. 11 = Positive and negative edge.
3	—	Reserved
2:0	CLK[2:0]	Counter clock select. These read/write control bits select one of six internal clock signals (PCLKx) or one of two external conditions on the input pin (rising edges or falling edges). The maximum frequency of the external clock signals is $f_{SYS}/4$. 000 = Prescaler output 1 (/2 or /3). 001 = Prescaler output 2 (/4 or /6). 010 = Prescaler output 3 (/8 or /12). 011 = Prescaler output 4 (/16 or /24). 100 = Prescaler output 5 (/32 or /48). 101 = Prescaler output 6 (/64 to /768). 110 = CTMC pin input, negative edge. 111 = CTMC pin input, positive edge.

13.3.10 MCSMCNT — MCSM Counter Register

MCSM2CNT — MCSM Counter Register **0xYF F212**
MCSM11CNT **0xYF F25A**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13.3.11 MCSMML — MCSM Modulus Latch Register

MCSM2ML — MCSM Modulus Latch Register **0xYF F214**
MCSM11ML **0xYF F25C**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13.4 Single-Action Submodule (SASM)

The single action submodule provides an input capture and an output compare for each of two bidirectional pins. All of the functions associated with one pin comprise a SASM channel. Each channel includes a 16-bit equality comparator and one 16-bit register for saving an input capture value or for holding an output compare value. The input edge detector associated with each pin is programmable to cause the capture function to occur on the rising or falling edge. The output flip flop is set by the software to either toggle when an output compare occurs or to transfer a software provided bit value to the output pin. In either the input capture mode or the output compare mode, a software interrupt may be programmed to occur for each channel. Software selection is provided to select which of the two incoming time-base buses is used for input captures or output compares on each channel. Each channel operates independently. However, interrupt and interrupt priority logic are shared by both SASM channels. See **Figure 13-4** for a SASM block diagram.

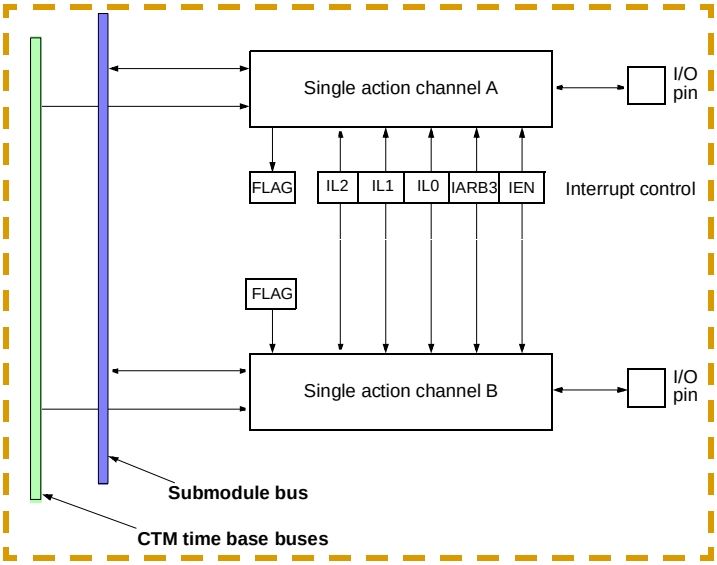


Figure 13-4 SASM Block Diagram

Each SASM channel comprises:

- A time base bus selector (which selects the time base bus to be used by that channel for all timing functions),
- A 16-bit data register (which can be read by the software at any time and which is used for both input capture and output compare functions),
- A 16-bit comparator (which continuously compares the 16-bit value in the data register with the time base bus),
- An output flip-flop (which holds the logic level to be sent to the output pin when a successful output compare occurs),
- An input edge detector (which detects the rising or falling edge that will trigger the input capture function),
- Several status and control bits in the status/interrupt/control register SICA or SICB,
- An interrupt section.

NOTE

During reset the output of the output flip-flop is cleared (i.e., to ‘0’).

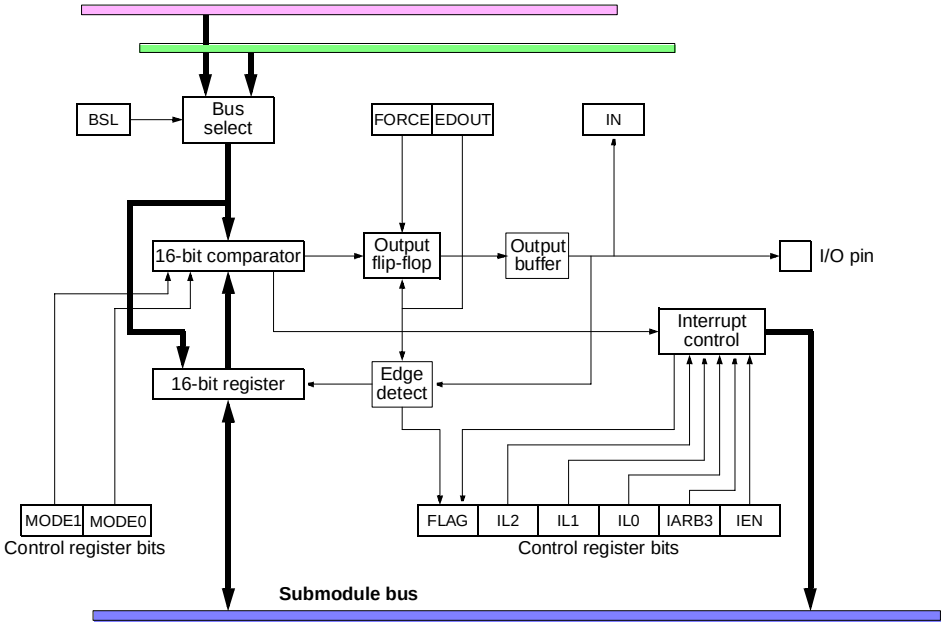


Figure 13-5 SASM Block Diagram (Channel A)

13.4.1 SASM Modes of Operation

Each SASM channel can operate in four different modes:

1. Input capture (IC) (i.e. either as input capture on a rising or falling edge or as a read-only input port)
2. Output compare (OC)
3. Output compare and toggle (OCT)
4. Output port (OP)

NOTE

For a channel operating in IC mode, the IN bit in the SIC register reflects the logic state of the corresponding input pin (after being Schmitt triggered and synchronized). When a channel is operating in OC, OCT or OP mode, the IN bit in the SIC register reflects the logic state of the output of the output flip-flop.

13.4.1.1 Clearing and Using the FLAG Bits

To clear a FLAG bit, the software must first read the channel's SIC register, then write a zero to the FLAG bit. These two steps do not have to be done on consecutive instructions. This clearing sequence must be used in every mode of operation. Writing a one to the FLAG bit has no effect.

WARNING

To avoid spurious interrupts, and to make sure that the FLAG bit is set according to the newly selected mode, the following sequence of operations should be adopted when changing mode:

1. Disable SASM interrupts
2. Change mode
3. Reset the corresponding FLAG bit
4. Re-enable SASM interrupts (if desired)

NOTE

When changing between output modes (OP, OC or OCT), it is not necessary to follow this procedure, as in these modes the FLAG bit merely indicates to the software that the compare value may be updated.

13.4.1.2 Input Capture (IC) Mode

In IC mode, the 16-bit counter value on the selected time base bus is ‘captured’ when a triggering event occurs on the channel’s input pin. Triggering of the input capture circuitry is done by a rising or falling edge on the input pin; the polarity of the triggering edge is selected by the EDOUT bit. The logic level on the input pin can be read by software via the IN bit in the channel’s SIC register.

In IC mode, the input pin is Schmitt triggered and the input signal is synchronized to the system clock (f_{SYS}). The IN bit reflects the state present on the input pin (after being Schmitt triggered and synchronized).

When an input capture occurs, the count value on the selected time base bus is latched into the channel’s 16-bit data register. At the same time, the FLAG bit in the SIC register is set to indicate that an input capture has occurred.

The FLAG bit must be reset by software (see **13.4.1.1 Clearing and Using the FLAG Bits**). If the interrupt is serviced, the FLAG bit should be cleared by the servicing routine before returning from that routine. If a subsequent input capture event occurs while the FLAG bit is set, the new captured counter value is latched, and the FLAG bit remains unchanged.

In IC mode, the value of the EDOUT bit is permanently transferred to the output flip-flop. This value will be output on the pin when the mode is changed to one of the output modes.

13.4.1.3 Output Compare (OC) Mode

In OC mode, the state of an output pin is changed when a successful output compare occurs; an interrupt may also be generated. The output compare circuitry performs a comparison between the 16-bit register and the selected time base bus. When a match is found, the EDOUT bit value is transferred to the output flip-flop. At the same time,

the FLAG bit is set to indicate to the processor that a match has occurred. Depending on the state of the IEN bit, an interrupt can be generated when the FLAG bit is set. The FLAG bit must be reset by software (see **13.4.1.1 Clearing and Using the FLAG Bits**). If the interrupt is serviced, the FLAG bit should be cleared by the servicing routine before returning from that routine. If a subsequent output compare occurs while the FLAG bit is set, the output compare function occurs normally, and the FLAG bit remains set.

An output compare match can be simulated in software by writing a one to the FORCE bit. Setting the FORCE bit forces the EDOUT bit value onto the pin as if an output compare had occurred. In this case, the FLAG bit is not affected. Only if a genuine output compare occurs while doing a force, will the FLAG bit be set to signify that the compare has occurred.

In OC mode, the IN bit value reflects the logic state on the output of the output flip-flop.

13.4.1.4 Output Compare and Toggle (OCT) Mode

In OCT mode, the state of an output pin is toggled each time a successful output compare occurs; an interrupt may also be generated. The output compare circuitry performs a comparison between the 16-bit register and the selected time base bus. When a match is found, the output flip-flop is toggled to the opposite state. At the same time, the FLAG bit is set to indicate to the processor that the output compare has occurred. Depending on the state of the IEN bit, an interrupt can be generated when the FLAG bit is set. The FLAG bit must be reset by software (see **13.4.1.1 Clearing and Using the FLAG Bits**). If the interrupt is serviced, the FLAG bit should be cleared by the servicing routine before returning from that routine. If a subsequent output compare occurs while the FLAG bit is set, the output toggles, and the FLAG bit remains set.

An output compare match can be simulated in software by writing a one to the FORCE bit. Setting the FORCE bit forces the output flip flop to toggle as if an output compare had occurred. In this case, the FLAG bit is not affected. Only if a genuine output compare occurs while doing a force, will the FLAG bit be set to signify that the compare has occurred.

In OCT mode, the IN bit reflects the logic state on the output of the output flip-flop.

13.4.1.5 Output Port (OP) mode

In OP mode the channel's input/output pin is used as a single output port pin. The output compare function is still available, but for internal operation only, and does not affect the state of the output pin. An interrupt may also be generated when a compare occurs. The state of the output pin always reflects the value of the EDOUT bit in the channel's SIC register. Reading the EDOUT bit returns the last value written to it.

The internal compare feature compares the 16-bit register with the selected time base bus. The output compare circuitry performs a comparison between the 16-bit register and the selected time base bus. When a match is found, the FLAG bit is set to indicate to the processor that the output compare has occurred. Depending on the state of the IEN bit, an interrupt can be generated when the FLAG bit is set. The FLAG bit must be

reset by software (see **13.4.1.1 Clearing and Using the FLAG Bits**). If the interrupt is serviced, the FLAG bit should be cleared by the servicing routine before returning from that routine. If a subsequent output compare occurs while the FLAG bit is set, the internal output compare functions normally, and the FLAG bit remains set.

In OP mode, the IN bit value reflects the logic state on the output of the output flip-flop.

13.4.2 SASM Interrupts

Each channel in the dual-channel SASM has separately enabled and initiated interrupts and they each have their own unique vector number and address. However, they are both assigned to the same interrupt level and arbitration priority by the IL[2:0] and IARB3 bits in the SICA register.

A valid SASM interrupt is recognized when the FLAG bit is set, the corresponding IEN bit is set and the interrupt level defined by bits IL[2:0] is not equal to zero.

The FLAG bit is a status bit that indicates, when set, that an input capture or output compare has occurred on the corresponding single action channel.

The relative priority of these sources of interrupt is fixed and channel A has a higher priority than channel B.

13.4.3 Freeze Action on the SASM

When the IMB FREEZE signal is recognized, the SASM input capture and output compare functions are halted. As soon as the FREEZE signal is negated, SASM actions resume as if nothing had happened. During freeze, the IN bits of the SIC registers (SICA and SICB) are readable and return the levels present at the input pins if an input mode is in operation, or the output value if an output mode is in operation (see **13.4.4.1 SICA — SASM Status/Interrupt Control Register A** and **13.4.4.3 SICB — SASM Status/Interrupt Control Register B**). When one of the output modes is in operation, the force output function remains available, allowing the software to output the desired level (a useful feature for debugging). All SASM registers are accessible during freeze.

13.4.4 SASM Registers

The SASM register map comprises eight 16-bit register locations. As shown in **Table 13-6**, the register block contains two SASM registers for each channel and four reserved registers. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no meaning nor effect. All register addresses in this section are specified as offsets from the base address of the SASM. In CTM implementations featuring multiple SASMs, each SASM has its own set of registers.

Table 13-6 SASM Register Map

Address	15 — 8 | 7 — 0
0xYF F270	SASM14 status/interrupt/control register A (S14ICA)
0xYF F272	SASM14 data register A (S14DATA)
0xYF F274	SASM14 status/interrupt/control register B (S14ICB)
0xYF F276	SASM14 data register B (S14DATB)
0xYF F280	SASM16 status/interrupt/control register A (S16ICA)
0xYF F282	SASM16 data register A (S16DATA)
0xYF F284,	SASM16 status/interrupt/control register B (S16ICB)
0xYF F286	SASM16 data register B (S16DATB)
0xYF F290	SASM18 status/interrupt/control register A (S18ICA)
0xYF F292	SASM18 data register A (S18DATA)
0xYF F294	SASM18 status/interrupt/control register B (S18ICB)
0xYF F296	SASM18 data register B (S18DATB)
0xYF F2A0	SASM20 status/interrupt/control register A (S20ICA)
0xYF F2A2	SASM20 data register A (S20DATA)
0xYF F2A4	SASM20 status/interrupt/control register B (S20ICB)
0xYF F2A6	SASM20 data register B (S20DATB)

13.4.4.1 SICA — SASM Status/Interrupt Control Register A

This register contains the control, interrupt enable and status bits for SASM channel A. It also contains the interrupt priority level bits IL[2:0] and the arbitration priority bit IARB3 for the whole SASM (i.e. common to channels A and B).

S14ICA — SASM Status/Interrupt Control Register A **0xYF F270**
S16ICA **0xYF F280**
S18ICA **0xYF F290**
S20ICA **0xYF F2A0**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
FLAG	IL2	IL1	IL0	IARB3	IEN	0	BSL	IN	0	FORCE	EDOUT	0	0	MODE1	MODE0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-7 SICA Bit Settings

Bit(s)	Name	Description
15	FLAG	Event flag. The FLAG bit is set whenever an input capture or output compare event occurs. This flag bit is set only by the hardware and cleared only by the software or by a system reset. If the IL field is non-zero, and the IEN bit is set, an interrupt request is generated when the FLAG bit is set. In IC mode, if a subsequent input capture event occurs while the FLAG bit is set, the new value is latched and the FLAG bit remains set. In OC mode, if a subsequent output compare event occurs while the FLAG bit is set, the compare occurs normally and the FLAG bit remains set. In OCT mode, if a subsequent output compare event occurs while the FLAG bit is set, the toggle of the output signal occurs as normal and the FLAG bit remains set. In OP mode, if a subsequent internal compare event occurs while the FLAG bit is set, the compare occurs normally and the FLAG bit remains set. To clear the flag, the software must first read the bit (as ‘1’) then write a ‘0’ to the bit. The flag clearing mechanism will work only if no flag setting event occurs between the read and write operations; if a FLAG setting event occurs between the read and write operations, the FLAG bit will not be cleared. 0 = An input capture or output compare event has not occurred. 1 = An input capture or output compare event has occurred.
14:12	IL[2:0]	Interrupt level. The three interrupt level bits are read/write control bits that select the priority level of interrupt requests made by the SASM. These bits can be read or written at any time and are cleared by reset. These bits affect both SASM channels, not just channel A. 000 = Interrupt disabled. 001 = Interrupt level 1 (lowest). 010 = Interrupt level 2. 011 = Interrupt level 3. 100 = Interrupt level 4. 101 = Interrupt level 5. 110 = Interrupt level 6. 111 = Interrupt level 7 (highest).
11	IARB3	Interrupt arbitration bit 3. The read/write IARB3 bit works in conjunction with the IARB[2:0] field in the BIUSM module configuration register. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field (IARB). This interrupt arbitration identification number is used to arbitrate for the IMB when modules generate simultaneous interrupts of the same priority. This bit affects both SASM channels, not just channel A.
10	IEN	Interrupt enable. This control bit enables interrupts on channel A when the FLAG bit is set and the IL[2:0] field is non-zero. This bit is cleared by reset 0 = Interrupts disabled. 1 = Interrupts enabled.
9	—	Reserved
8	BSL	Time base bus select. This control bit selects the time base bus to be connected to SASM channel A. This bit is cleared by reset. 0 = Time base bus A selected. 1 = Time base bus B selected.
7	IN	Input pin status. In input mode (IC), the IN bit reflects the logic state present on the corresponding input pin (after being Schmitt triggered and synchronized). In the output modes (OC, OCT and OP), the IN bit value reflects the state of the output of the output flip-flop. The IN bit is a read-only bit; writing to it has no effect. Reset has no effect on this bit.
6	—	Reserved
5	FORCE	Supervisor/user data space. The SUPV bit places the SCIM2E global registers in either supervisor or user data space. The FLAG bit is not affected by the use of the FORCE bit. 0 = No action 1 = Force output flip-flop to behave as if an output compare has just occurred.

Table 13-7 SICA Bit Settings (Continued)

Bit(s)	Name	Description
4	EDOUT	Edge detect and output level. In IC mode, the EDOUT bit is used to select the edge that will trigger the input capture circuitry. In OC mode, the EDOUT bit is used to latch the value to be output to the pin on the next output compare match or when the FORCE bit is set. Internal synchronization ensures that the correct level appears on the output pin when a new value is written to EDOUT and FORCE is set at the same time. Reading EDOUT returns the previous value written. In OCT mode, the EDOUT bit has no effect. However, the force function is still available and will force the value of the EDOUT bit to appear on the output pin. In OP mode, the value of the EDOUT bit is output to the corresponding pin. Reading EDOUT returns the previous value written. 0 = Input capture on falling edge. 1 = Input capture on rising edge.
3:2	—	Reserved
1:0	MODE1, MODE0	SASM operating mode select. These control bits select the mode of operation of the SASM channel, as shown in the following table. MODE1 and MODE0 are cleared by reset. 00 = Input capture (IC). 01 = Output port (OP). 10 = Output compare (OC 11 = Output compare and toggle (OCT)

13.4.4.2 SDATA — SASM Data Register A

SDATA is the 16-bit read-write register associated with channel A. In IC mode, SDATA contains the last captured value. In the OC, OCT and OP modes, it is loaded with the value of the next output compare. SDATA is not affected by reset.

S14DATA — SASM Data Register A **0xYF F272,**
S16DATA **0xYF F282**
S18DATA **0xYF F292**
S20DATA **0xYF F2A2**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

13.4.4.3 SICB — SASM Status/Interrupt Control Register B

This register contains the control and status bits for SASM channel B. The bits it contains are identical to those in SICA, with the exception of the IL[2:0], IARB3 and IEN which apply to both channels simultaneously and which are included only in SICA. For descriptions of the bits, please refer to **13.4.4.1 SICA — SASM Status/Interrupt Control Register A**).

S14ICB — SASM Status/Interrupt Control Register B **0xYF F274**
S16ICB **0xYF F284**
S18ICB **0xYF F294**
S20ICB **0xYF F2A4**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
FLAG	0	0	0	0	0	0	BSL	IN	0	FORCE	EDOUT	0	0	MOD E1	MOD E0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13.4.4.4 SDATB — SASM Data Register B

SDATB is the 16-bit read-write register associated with channel A. In the IC mode, SDATB contains the last captured value. In the OC, OCT and OP modes, it is loaded with the value of the next output compare. SDATB is not affected by reset.

S14DATB — SASM Data Register B **0xYF F276**
S16DATB **0xYF F286**
S18DATB **0xYF F296**
S20DATB **0xYF F2A6**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

13.5 Double-Action Submodule (DASM)

The double-action submodule (DASM) provides two 16-bit input captures or two 16-bit output compare functions that can occur automatically without software intervention. The input edge detector is programmable to cause the capture function to occur on the desired edges. The output flip-flop is set by one of the output compares and is reset by the other one. In either the input capture modes or the output compare modes, an optional interrupt is available to the software. Software selection is provided for which of two incoming time-base buses is used for input captures or output compares.

The DASM can work in six different modes: disable mode, pulse length measurement, period measurement, input capture mode, single pulse generation, and continuous pulse width generation.

The DASM has three data registers that are accessible to the software from the various modes. For some of the modes, two of the registers are cascaded together to provide double buffering. The value in one register is transferred to another register automatically at the correct time so that the minimum pulse (measurement or generation) is just one time-base bus count. See **Figure 13-6** for a DASM block diagram.

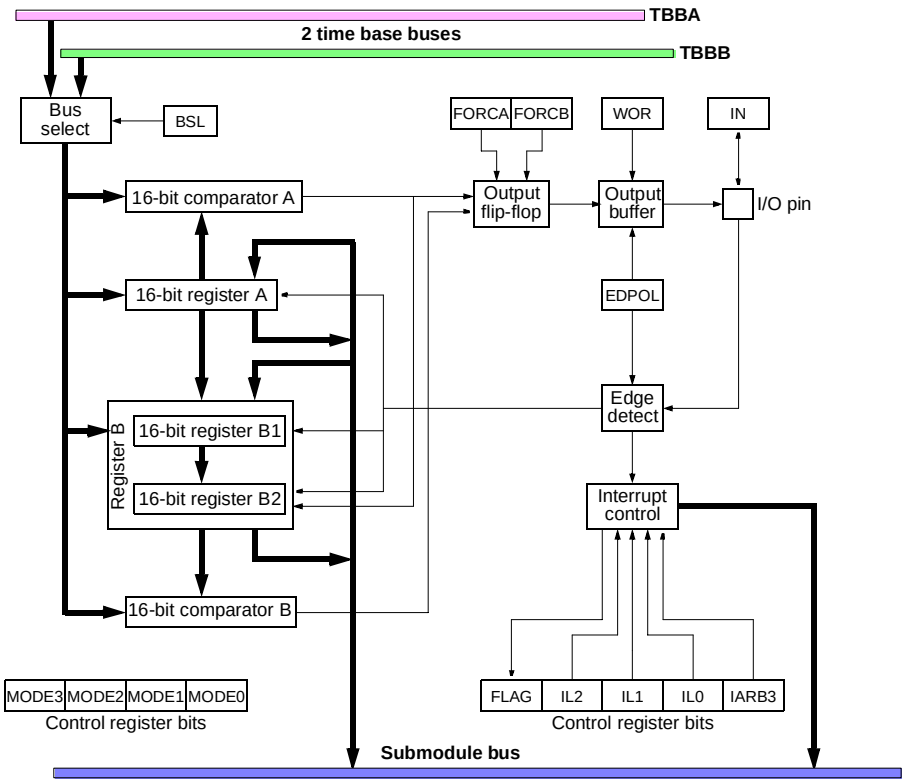


Figure 13-6 DASM Block Diagram

Channel A comprises one 16-bit data register and one 16-bit comparator. Channel B also appears to the user to consist of one 16-bit data register and one 16-bit comparator, however, internally, channel B has two data registers B1 and B2, and the operating mode determines which register is accessed by the software:

- In the input capture modes (IPWM, IPM and IC), registers A and B2 are used to hold the captured values; in these modes, the B1 register is used as a temporary latch for channel B.
- In the output compare modes (OCA and OCAB), registers A and B2 are used to define the output pulse; register B1 is not used in these modes.
- In the output pulse width modulation mode (OPWM), registers A and B1 are used as primary registers and hidden register B2 is used as a double buffer for channel B.

Register contents are always transferred automatically at the correct time so that the minimum pulse (measurement or generation) is just one time base bus count. The A and B data registers are always read/write registers, accessible via the CTM's submodule bus.

In the input capture modes, the edge detect circuitry triggers a capture whenever a rising or falling edge (as defined by the EDPOL bit) is applied to the input pin. The signal on the input pin is Schmitt triggered and synchronized with the system clock (f_{SYS}).

In the disabled mode (DIS) and in the input modes, the IN bit reflects the state present on the input pin (after being Schmitt triggered and synchronized). In the output modes the IN bit reflects the value present at the output of the output flip-flop. The output flip-flop is used in output modes to hold the logic level applied to the output pin.

The time base bus selector is common to all input and output functions; it connects the DASM to time base bus A or B and is controlled in software by the bus select bit BSL in the DASMSIC register.

13.5.1 32-Bit Coherent Access

In the IPWM and IPM modes, 32-bit coherent access of the data registers is supported. A 32-bit coherent access consists of doing a long word aligned access of data register A. In this case, register A is accessed first, immediately followed (on the next cycle) by a register B access. During this time, any flag setting or data transfer from the hidden B register is deferred until coherent access has ended. When the 32-bit access has ended, the DASM finishes any pending B action and resumes normal operation.

13.5.2 DASM Modes of Operation

The mode of operation of the DASM is determined by the mode select bits MODE[3:0] in the DASMSIC register (see **Table 13-8**).

Table 13-8 DASM Modes of Operation

MODE[3:0]	Mode	Description of mode
0000	DIS	Disabled — Input pin is high impedance; IN gives state of the input pin.
0001	IPWM	Input pulse width measurement — Capture on the leading edge and the trailing edge of an input pulse.
0010	IPM	Input period measurement — Capture two consecutive rising/falling edges.
0011	IC	Input capture — Capture when the designated edge is detected.
0100	OCB	Output compare, flag set on B compare — Generate leading and trailing edges of an output pulse and set the flag.
0101	OCAB	Output compare, flag on A and B compare — Generate leading and trailing edges of an output pulse and set the flag.
1xxx	OPWM	Output pulse width modulation — Generate continuous PWM output with 7, 9, 11, 12, 13, 14, 15 or 16 bits of resolution.

WARNING

To avoid spurious interrupts, and to make sure that the FLAG bit is set according to the newly selected mode, the following sequence of operations should be adopted when changing mode:

1. Disable DASM interrupts
2. Change mode
3. Reset the corresponding FLAG bit

4. Re-enable DASM interrupts (if desired)

NOTE

When changing between output modes (OP, OC or OCT), it is not necessary to follow this procedure, as in these modes the FLAG bit merely indicates to the software that the compare value can be updated.

13.5.2.1 Disable (DIS) mode

DIS mode is selected by making MODE[3:0] = 0000.

In this mode, all input capture and output compare functions of the DASM are disabled and the FLAG bit is maintained in its reset state, but the input port pin function remains available. The associated pin becomes a high impedance input and the input level on this pin is reflected by the state of the IN bit in the DASMSIC register. All control and interrupt bits remain accessible, allowing the software to prepare for future mode selection. Data registers A and B are accessible at consecutive addresses. Writing to data register B stores the same value in registers B1 and B2.

WARNING

When changing modes, it is imperative to go through the DIS mode in order to reset the DASM's internal functions properly. Failure to do this could lead to invalid and unexpected output compare or input capture results, and to flags being set incorrectly.

13.5.2.2 Input Pulse Width Measurement (IPWM) Mode

IPWM mode is selected by making MODE[3:0] = 0001.

This mode allows the width of a positive or negative pulse to be determined by capturing the leading edge of the pulse on channel B and the trailing edge of the pulse on channel A; successive captures are done on consecutive edges of opposite polarity. The edge sensitivity is selected by the EDPOL bit in the DASMSIC register.

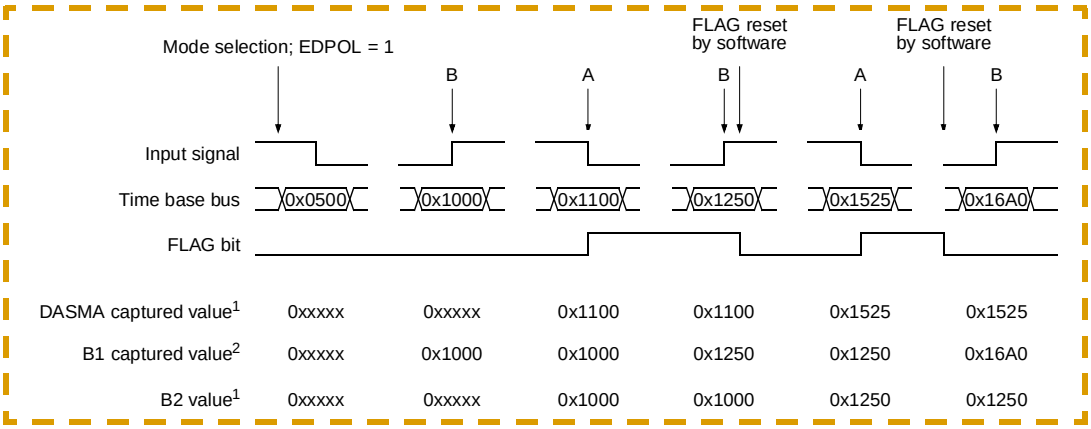
This mode also allows the software to determine the logic level on the input pin at any time by reading the IN bit in the DASMSIC register.

The channel A input capture function remains disabled until the first rising edge triggers the first input capture on channel B. When this rising edge is detected, the count value of the time base bus selected by the BSL bit is latched in the 16-bit data register B1; the FLAG bit is not affected. When the next falling edge is detected, the count value of the time base bus is latched into the 16-bit data register A and, at the same time, the FLAG bit is set and the contents of register B1 are transferred to register B2. Reading data register B returns the value in register B2. If subsequent input capture events occur while the FLAG bit is set, data registers A and B will be updated with the latest captured values and the FLAG bit will remain set.

If a 32-bit coherent operation is in progress when the falling edge is detected, the transfer from B1 to B2 is deferred until the coherent operation is completed. Operation of the DASM then continues on channels B and A as previously described.

The input pulse width is calculated by subtracting the value in data register B from the value in data register A.

Figure 13-7 provides an example of how the DASM can be used for input pulse width measurement.



Notes: 1. These values are accessible to the software.
2. These values are internal and are not accessible.

Figure 13-7 Input Pulse Width Measurement Example

13.5.2.3 Input Period Measurement (IPM) Mode

IPM mode is selected by making MODE[3:0] = 0010.

This mode allows the period of an input signal to be determined by capturing two consecutive rising edges or two consecutive falling edges; successive input captures are done on consecutive edges of the same polarity. The edge polarity is defined by the EDPOL bit in the DASMSIC register.

This mode also allows the software to determine the logic level on the input pin at any time by reading the IN bit in the DASMSIC register.

When the first edge having the selected polarity is detected, the time base bus value is latched into the 16-bit data register A, the data in register B1 is transferred to data register B2 and finally the data in register A is transferred to register B1. On this first capture the FLAG bit is not set. On the second and subsequent captures, the FLAG bit is set immediately before the data in register A is transferred to register B1.

When the second edge of the same polarity is detected, the time base bus value is latched into data register A, the data in register B1 is transferred to data register B2, the FLAG bit is set to signify that the beginning and end points of a complete period have been captured, and finally data register A is transferred to register B1. This

sequence of events is repeated for each subsequent capture. Reading data register B returns the value in register B2.

If a 32-bit coherent operation is in progress when an edge is detected, the transfer of data from B1 to B2 is deferred until the coherent operation is completed. At any time, the input level present on the input pin can be read on the IN bit.

The input pulse period is calculated by subtracting the value in data register B from the value in data register A.

Figure 13-8 provides an example of how the DASM can be used for input period measurement.

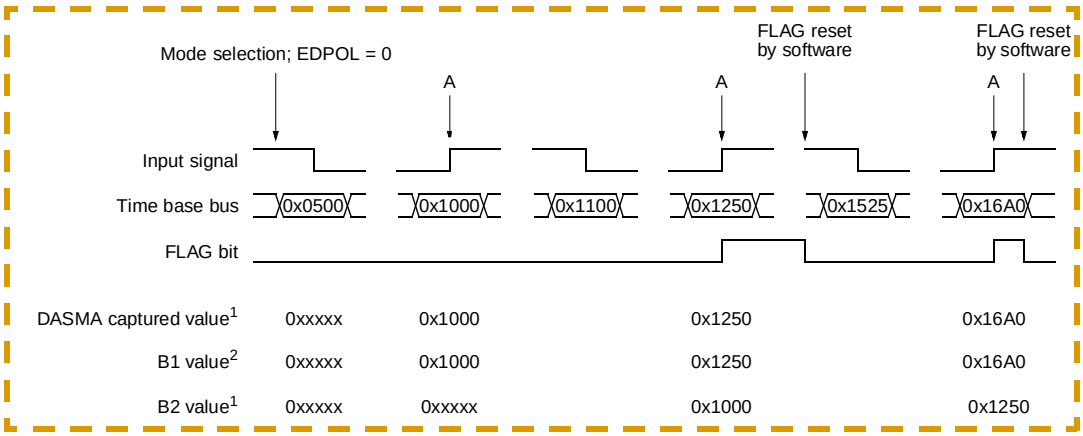


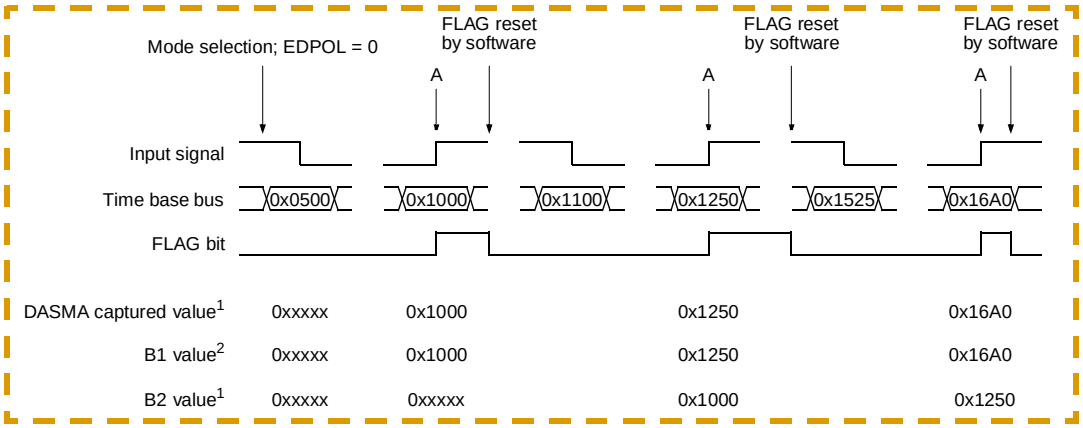
Figure 13-8 Input Period Measurement Example

13.5.2.4 Input Capture (IC) Mode

IC mode is selected by making MODE[3:0] = 0011.

This mode is identical to the input period measurement mode (IPM) described above, with the exception that the FLAG bit is also set at the occurrence of the first detected edge of the selected polarity. In this mode the DASM functions as a standard input capture function in a similar way to the M68HC11 family timers. In this case the value latched in channel B can be ignored.

Figure 13-9 provides an example of how the DASM can be used for input capture.



Notes: 1. These values are accessible to the software.
2. These values are internal and are not accessible.

Figure 13-9 DASM Input Capture Example

13.5.2.5 Output Compare (OCB and OCAB) Modes

OC mode is selected by making MODE[3:0] = 010x. The MODE0 bit controls the setting criteria for the FLAG bit, i.e. when a compare occurs only on channel B or when a compare occurs on either channel (see **13.5.5.1 DASMSIC — DASM Status/Interrupt Control Register**).

This mode allows the DASM to perform four different output functions:

- Single-shot output pulse (two edges), with FLAG set on the second edge.
- Single-shot output pulse (two edges), with FLAG set on both edges.
- Single-shot output transition (one edge).
- Output port pin, with output compare function disabled.

In this mode the leading and trailing edges of variable width output pulses are generated by calculated output compare events occurring on channels A and B, respectively. OC mode may also be used to perform a single output compare function, similar to the M68HC11 timer, or may be used as an output port bit.

In this mode, channel B is accessed via register B2. Register B1 is not used and is not accessible to the user. Both channels work together to generate one ‘single shot’ output pulse signal. Channel A defines the leading edge of the output pulse, while channel B defines the trailing edge of the pulse. FLAG setting can be done when a compare occurs on channel B only or when a compare occurs on either channel (as defined by the MODE0 bit in the DASMSIC register).

When this mode is first selected, both comparators are disabled. Each comparator is enabled by writing to its data register; it remains enabled until the next successful comparison is made on that channel, whereupon it is disabled. The values stored in registers A and B are compared with the count value on the selected time base bus when their corresponding comparators are enabled.

The output flip-flop is set when a match occurs on channel A. The output flip-flop is reset when a match occurs on channel B. The polarity of the output signal is selected by the EDPOL bit. The output flip-flop level can be obtained at any time by reading the IN bit.

If subsequent enabled output compares occur on channels A and B, the output pulses continue to be output, regardless of the state of the FLAG bit.

At any time, the FORCA and FORCB bits allow the software to force the output flip-flop to the level corresponding to a comparison on channel A or B, respectively. Note that the FLAG bit is not affected by these 'force' operations.

Totem pole or open-drain output circuit configurations can be selected using the WOR bit in the DASMSIC register.

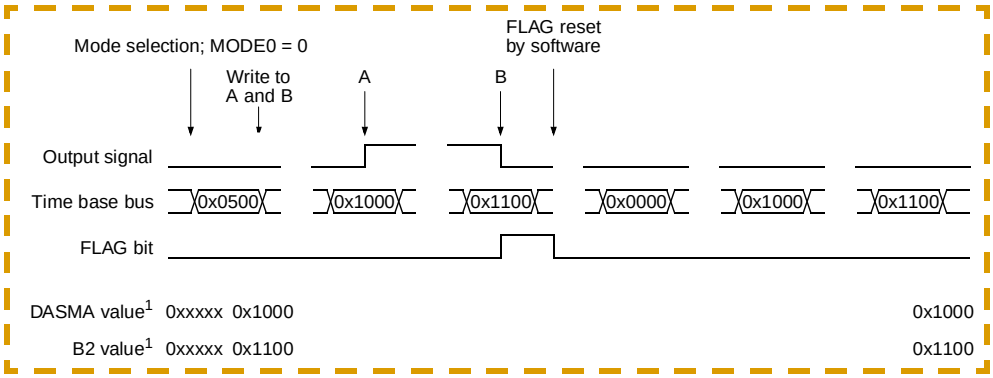
WARNING

There is no hardware protection to disable comparator B while comparator A is enabled. It is the user's responsibility to load data registers A and B with the values needed to produce the desired output pulse.

NOTE

If both channels are loaded with the same value they will try to force different levels on the output flip-flop. Hardware protection circuitry ensures that no contention occurs and the output flip-flop provides a logic zero level output.

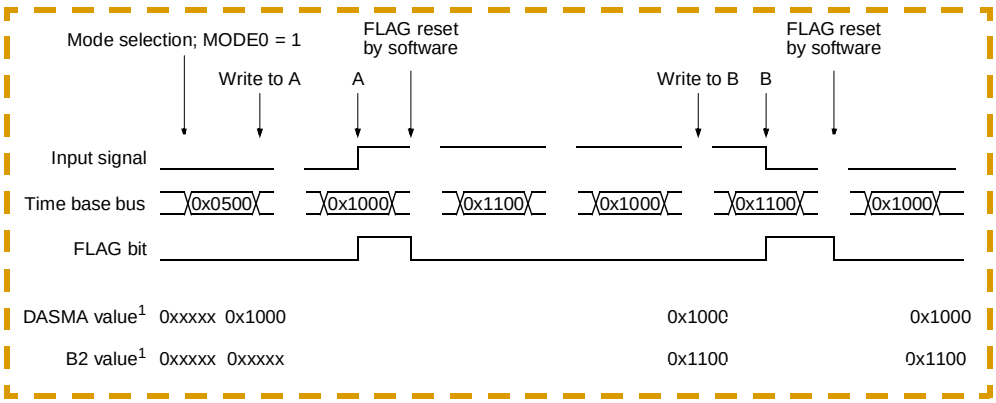
Single Shot Output Pulse Operation — The single shot output pulse operation is selected by writing the leading edge value of the desired pulse to data register A and the trailing edge value to data register B. A single pulse will be output at the desired time, thereby disabling the comparators until new values are written to the data registers. In this mode, registers A and B2 are accessible to the user software (at consecutive addresses). **Figure 13-10** provides an example of how the DASM can be used to generate a single output pulse.



Note: 1. These values are accessible to the software.

Figure 13-10 Single Shot Output Pulse Example

Single Output Compare Operation — The single output compare operation is selected by writing to only one of the two data registers (A or B), thus enabling only one of the comparators. Following the first successful match on the enabled channel, the output level is fixed and remains at the same level indefinitely with no further software intervention being required. In this mode, registers A and B2 are accessible to the user software (at consecutive addresses). **Figure 13-11** provides an example of how the DASM can be used to perform a single output compare.



Note: 1. These values are accessible to the software.

Figure 13-11 Single Shot Output Transition Example

Output Port Bit Operation — The output port bit operation is selected by leaving both channels disabled, i.e. by writing to neither register A nor B. The EDPOL bit alone controls the output value. The same result can be achieved by keeping EDPOL at zero and using the FORCA and FORCB bits to obtain the desired output level.

13.5.2.6 Output Pulse Width Modulation (OPWM) Mode

OPWM mode is selected by making MODE[3:0] = 1xxx. The MODE[2:0] bits allow some of the comparator bits to be masked.

This mode allows pulse width modulated output waveforms to be generated, with eight selectable frequencies (for a given time base). Both channels (A and B) are used to generate one PWM output signal on the DASM pin.

Channel B is accessed via register B1. Register B2 is not accessible to the user. Channels A and B define the leading and trailing edges, respectively, of the PWM output pulse. The value in register B1 is continuously transferred to register B2 in the time between each trailing edge and the following leading edge.

The value loaded in register A is continuously compared with the value on the time base bus. When a match on A occurs, the FLAG bit is set and the output flip-flop is set. The value loaded in register B2 is continually compared with the value on the time base bus. When a match occurs on B, the output flip-flop is reset.

The polarity of the PWM output signal is selected by the EDPOL bit. The output flip-flop level can be obtained at any time by reading the IN bit.

If subsequent compares occur on channels A and B, the PWM pulses continue to be output, regardless of the state of the FLAG bit.

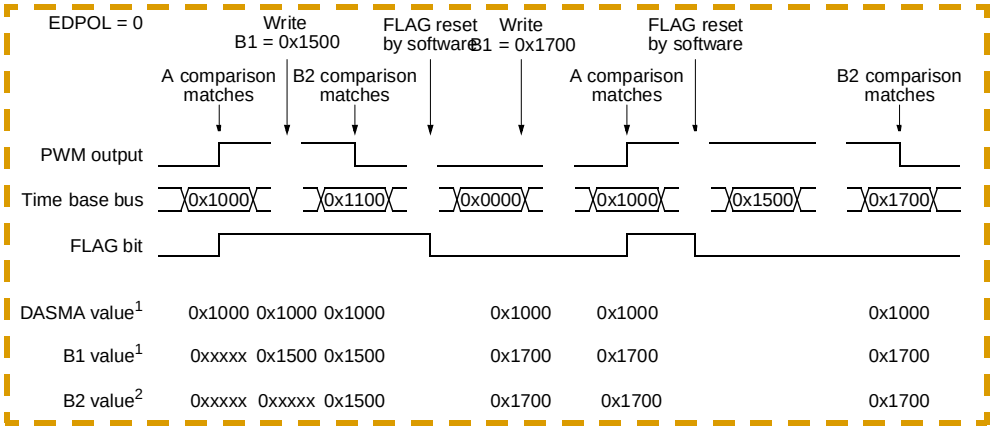
At any time, the FORCA and FORCB bits allow the software to force the output flip-flop to the level corresponding to comparison on A or B respectively. Note that the FLAG bit is not affected by the FORCA and FORCB operations.

WARNING

There is no hardware protection to disable comparator B while comparator A is enabled. It is the user's responsibility to load data registers A and B with the values needed to produce the desired PWM output pulse.

If both channels are loaded with the same value they will try to force different levels on the output flip-flop. Hardware protection circuitry ensures that no contention occurs and the output flip-flop provides a logic zero level output.

Figure 13-12 provides an example of how the DASM can be used for pulse width modulation.



Notes: 1. These values are accessible to the software.
2. These values are internal and are not accessible.

Figure 13-12 DASM Output Pulse Width Modulation Example

To generate PWM output pulses of different frequencies, the 16-bit comparator can have some of its bits masked. This is controlled by bits MODE2, MODE1 and MODE0. The frequency of the PWM output (f_{PWM}) is given by Equation 1 (assuming the DASM is connected to a free running counter):

$$f_{PWM} = \frac{f_{SYS}}{N_{CPSM} \bullet N_{DASM}}$$

where N_{CPSM} is the overall CPSM clock divide ratio (÷2 to ÷512 or ÷3 to ÷768) and N_{DASM} is the DASM divide ratio.

A few examples of frequencies and resolutions that can be obtained are shown in **Table 13-9**.

Table 13-9 DASM PWM Example Output Frequencies/Resolutions at f_{SYS} = 16 MHz

N_{CPSM}	N_{DASM}[1]	PWM output frequency (Hz)	Resolution (bits)
512	65536	0.48	16
2	65536	122.07	16
512	32768	0.95	15
2	32768	244.14	15
512	16384	1.91	14
2	16384	488.28	14
512	8192	3.81	13
2	8192	976.56	13
512	4096	7.63	12
2	4096	1953.13	12
512	2048	15.26	11
2	2048	3906.25	11
512	512	31.04	9
2	512	15625.00	9
512	128	244.14	7
2	128	62500.00	7

NOTES:
1. This table is valid only if the DASM is connected to a free-running counter.

When using 16 bits of resolution on the comparator (MODE[2:0] = 000), the output can vary from a 0% duty cycle up to a duty cycle of 65535/65536. In this case it is not possible to have a 100% duty cycle. In cases where 16-bit resolution is not needed, it is possible to have a duty cycle ranging from 0% to 100%. Setting bit 15 of the value stored in register B to ‘1’ results in the output being ‘always set’. Clearing bit 15 (to ‘0’) allows normal comparisons to occur and the normal output waveform is obtained. Changes to and from the 100% duty cycle are done synchronously, as are all other width changes.

In the OPWM mode, the WOR bit selects whether the output is totem pole driven or open-drain.

13.5.3 DASM interrupts

When the FLAG bit is set, an interrupt request is generated on one of eight levels as defined by the interrupt level bits (IL[2:0]) in the DASMSIC register. If the interrupt level is set to zero, interrupts are disabled.

13.5.4 Freeze Action on the DASM

When the IMB FREEZE signal is recognized, the DASM capture and compare functions are halted. As soon as the FREEZE signal is negated, DASM actions resume as if nothing had happened. During freeze, the IN bit of the DASMSIC register is readable

and returns the level present at the input pin if an input mode is selected, or the output value if an output mode is in operation. When one of the output modes is in operation, the force output function remains available, allowing the software to output the desired level and simplifying debugging. All DASM registers are accessible during freeze.

13.5.5 DASM Registers

The DASM register map comprises four 16-bit register locations. As shown in **Table 13-10**, the register block contains three DASM registers and one reserved register. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no meaning or effect. All register addresses in this section are specified as offsets from the base address of the DASM. In CTM implementations featuring multiple DASMs, each DASM has its own set of registers.

Table 13-10 DASM Register Map

Address	15 — 8 | 7 — 0
0xYF F218	DASM3 status/interrupt/control register (DASM3SIC)
0xYF F21A	DASM3 register A (DASM3A)
0xYF F21C	DASM3 register B (DASM3B)
0xYF F220	DASM4 status/interrupt/control register (DASM4SIC)
0xYF F222	DASM4 register A (DASM4A)
0xYF F224	DASM4 register B (DASM4B)
0xYF F248	DASM9 status/interrupt/control register (DASM9SIC)
0xYF F24A	DASM9 register A (DASM9A)
0xYF F24C	DASM9 register B (DASM9)
0xYF F250	DASM10 status/interrupt/control register (DASM10SIC)
0xYF F252	DASM10 register A (DASM10A)
0xYF F254	DASM10 register B (DASM10)

13.5.5.1 DASMSIC — DASM Status/Interrupt Control Register

DASM3SIC — DASM Status/Interrupt Control Register **0xYF F218**
DASM4SIC **0xYF F220**
DASM9SIC **0xYF F248**
DASM10SIC **0xYF F250**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
FLAG	IL2	IL1	IL0	IARB3	0	WOR	BSL	IN	FORC A	FORC B	ED-POL	MODE 3	MODE 2	MODE 1	MODE 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-11 DASMSIC Bit Settings

Bit(s)	Name	Description
15	FLAG	Flag status. This status bit indicates whether or not an input capture or output compare event has occurred. If the IL field is non-zero, an interrupt request is generated when the FLAG bit is set. The flag clearing mechanism will work only if no flag setting event occurs between the read and write operations; if a FLAG setting event occurs between the read and write operations, the FLAG bit will not be cleared. – In the DIS mode, the FLAG bit is cleared. – In the IPWM mode, the FLAG bit is set each time there is a capture on channel A. – In the IPM mode, the FLAG bit is set each time there is a capture on channel A, except for the first time. – In the IC mode, the FLAG bit is set each time there is a capture on channel A. – In the OCB mode (i.e. when MODE0 = 0), the FLAG bit is only set each time there is a successful comparison on channel B. In the OCAB mode (i.e. when MODE0 = 1), the FLAG bit is set each time there is a successful comparison on either channel A or B. – In the OPWM mode, the FLAG bit is set whenever there is a successful comparison on channel A. 0 = An input capture or output compare event has not occurred. 1 = An input capture or output compare event has occurred.
14:12	IL[2:0]	Interrupt level. The three interrupt level bits are read/write control bits that select the priority level of interrupt requests made by the DASM. These bits can be read or written at any time and are cleared by reset. 000 = Interrupt disabled 001 = Interrupt level 1 (lowest) 010 = Interrupt level 2 011 = Interrupt level 3 100 = Interrupt level 4 101 = Interrupt level 5 110 = Interrupt level 6 111 = Interrupt level 7 (highest)
11	IARB3	Interrupt arbitration bit 3. The read/write IARB3 bit works in conjunction with the IARB[2:0] field in the BIUSM module configuration register. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field (IARB). This interrupt arbitration identification number is used to arbitrate for the IMB when modules generate simultaneous interrupts of the same priority.
10	—	Reserved
9	WOR	Wired-OR. In the DIS, IPWM, IPM and IC modes, the WOR bit is not used; reading this bit returns the value that was previously written. In the OCB, OCAB and OPWM modes, the WOR bit selects whether the output buffer is configured for open-drain or totem pole operation. 0 = Output buffer is totem pole. 1 = Output buffer is open-drain.
8	BSL	Bus select. This control bit selects the time base bus to be connected to the DASM. 0 = The DASM is connected to time base bus A. 1 = The DASM is connected to time base bus B.
7	IN	Input pin status. In the DIS, IPWM, IPM and IC modes, this read-only status bit reflects the logic level on the input pin. In the OCB, OCAB and OPWM modes, reading this bit returns the value latched on the output flip-flop, after EDPOL polarity selection. Writing to this bit has no effect.
6:5	FORCA, FORCB	Force A, B. In the OCB, OCAB and OPWM modes, the FORCA, B bit allows the software to force the output flip-flop to behave as if a successful comparison had occurred on channel A, B (except that the FLAG bit is not set). Writing a one to FORCA, B sets the output flip-flop; writing a zero to it has no effect. In the DIS, IPWM, IPM and IC modes, FORCA and FORCB are not used and writing to them has no effect. Writing a one to both FORCA and FORCB simultaneously resets the output flip-flop

Table 13-11 DASMSIC Bit Settings (Continued)

Bit(s)	Name	Description
4	EDPOL	Edge polarity. In the DIS mode, this bit is not used; reading it returns the last value written. In the IPWM mode, this bit is used to select the capture edge sensitivity of channels A and B. 0 = Channel A captures on a rising edge. Channel B captures on a falling edge. 1 = Channel A captures on a falling edge. Channel B captures on a rising edge. In the IPM and IC modes, the EDPOL bit is used to select the input capture edge sensitivity of channel A. 0 = Channel A captures on a rising edge. 1 = Channel A captures on a rising edge. In the OCB, OCAB and OPWM modes, the EDPOL bit is used to select the voltage level on the output pin. 0 = The output flip-flop logic level appears on the output pin: a compare on channel A sets the output pin, a compare on channel B resets the output pin. 1 = The complement of the output flip-flop logic level appears on the output pin: a compare on channel A resets the output pin; a compare on channel B sets the output pin.
3:0	MODE[3:0]	Mode select. The four mode select bits select the mode of operation of the DASM. To avoid spurious interrupts, it is recommended that DASM interrupts are disabled before changing the operating mode.

13.5.5.2 DASMA — DASM Data Register A

DASM3A — DASM Data Register A **0xYF F21A**
DASM4A **0xYF F222**
DASM9A **0xYF F24A**
DASM10A **0xYF F252**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DASMA is the data register associated with channel A; its use varies with the different modes of operation:

- In the DIS mode, DASMA can be accessed to prepare a value for a subsequent mode selection.
- In the IPWM mode, DASMA contains the captured value corresponding to the trailing edge of the measured pulse.
- In the IPM and IC modes, DASMA contains the captured value corresponding to the most recently detected dedicated edge (rising or falling edge).
- In the OCB and OCAB modes, DASMA is loaded with the value corresponding to the leading edge of the pulse to be generated. Writing to DASMA in the OCB and OCAB modes also enables the corresponding channel A comparator until the next successful comparison.
- In the OPWM mode, DASMA is loaded with the value corresponding to the leading edge of the PWM pulse to be generated.

13.5.5.3 DASMB — DASM Data Register B

DASM3B — DASM Data Register B **0xYF F21C**
DASM4B **0xYF F224**
DASM9B **0xYF F24C**
DASM10B **0xYF F254**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DASMB is the data register associated with channel B; its use varies with the different modes of operation. Depending on the mode selected, software access is to register B1 or register B2.

In the DIS mode, DASMB can be accessed to prepare a value for a subsequent mode selection. In this mode, register B1 is accessed in order to prepare a value for the OPWM mode. Unused register B2 is hidden and cannot be read, but is written with the same value when register B1 is written.

In the IPWM mode, DASMB contains the captured value corresponding to the leading edge of the measured pulse. In this mode, register B2 is accessed; buffer register B1 is hidden and cannot be accessed.

In the IPM and IC modes, DASMB contains the captured value corresponding to the most recently detected period edge (rising or falling edge). In this mode, register B2 is accessed; buffer register B1 is hidden and cannot be accessed.

In the OCB and OCAB modes, DASMB is loaded with the value corresponding to the trailing edge of the pulse to be generated. Writing to DASMB in the OCB and OCAB modes also enables the corresponding channel B comparator until the next successful comparison. In this mode, register B2 is accessed; buffer register B1 is hidden and cannot be accessed.

In the OPWM mode, DASMB is loaded with the value corresponding to the trailing edge of the PWM pulse to be generated. In this mode, register B1 is accessed; buffer register B2 is hidden and cannot be accessed.

13.6 Pulse Width Modulation Submodule (PWMSM)

The purpose of the pulse width modulation submodule (PWMSM) is to create a variable pulse width output signal at a wide range of frequencies, independent of other CTM9 output signals. The PWMSM includes its own counter, and thus does not use the CTM9 time-base buses. The PWMSM pulse width can vary from 0.0 percent to 100.0 percent, with up to 16 bits of resolution. The finest output resolution is the MCU system clock time divided by two (for a system clock of 16.78 MHz, the finest output pulse width resolution is 119 nanoseconds). With the full 16 bits of resolution and the first stage prescaler divide-by-2 clock selection, the period of the PWM output can

range from 7.8 milliseconds to 2.0 seconds (assuming a 16.78 MHz MCU clock). By reducing the counting value, the output signal period can be reduced. The period can be as fast as 488 microseconds (2.048 KHz) with 12 bits of resolution, as fast as 30.5 microseconds (32.768 KHz) with 8 bits of resolution, and as fast as 7.6 microseconds (131.072 KHz) with 6 bits of resolution (still assuming a 16.78 MHz system clock and a first stage prescaler divide-by-2 clock selection). A block diagram of the PWMSM is shown in **Figure 13-13**.

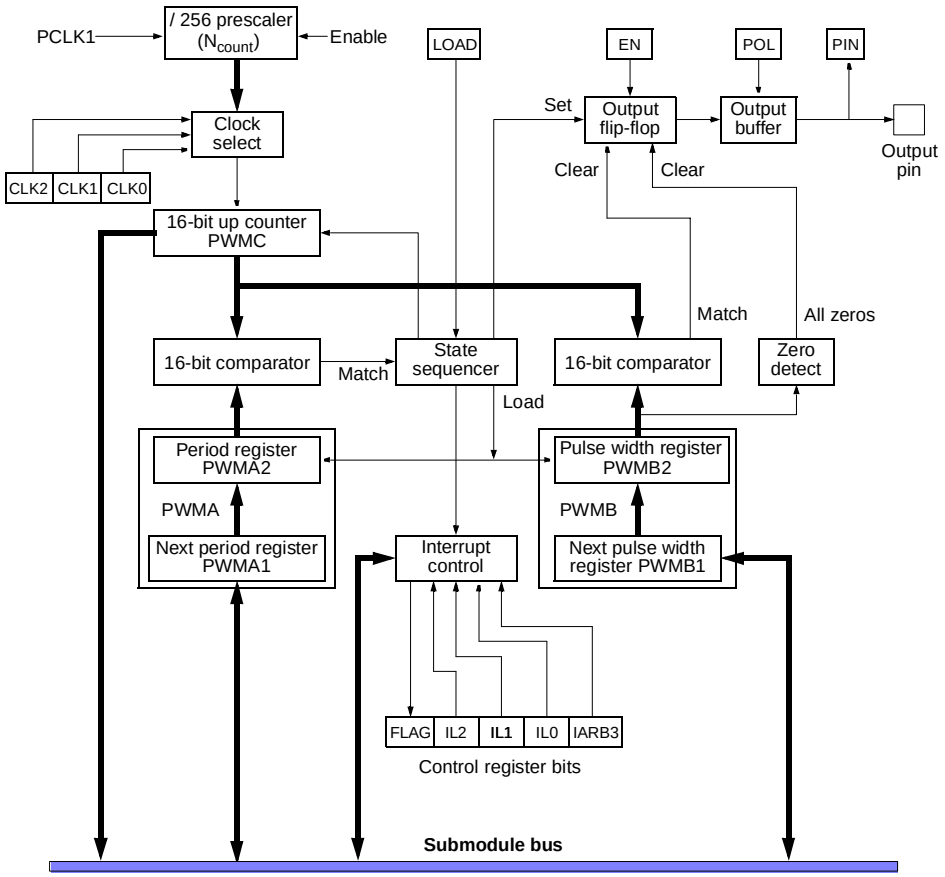


Figure 13-13 Pulse Width Modulation Submodule Block Diagram

13.6.1 Output Flip-Flop and Pin

The output flip-flop is the basic output mechanism of the PWMSM. Except when the required pulse width is 0% or 100%, the output flip-flop is set at the beginning of each period and is cleared at the end of the designated pulse width. The polarity of the output pulse can be selected in software. The output of the PWMSM is connected to an external, output-only pin. When the PWMSM is not required, and is disabled by clearing the EN bit in the PWMSIC register, this pin serves as a digital output-only port pin. When the PWMSM is disabled, the POL bit in the SIC register serves as an output port bit.

13.7 Time Base Bus System

The time base bus (TBB) system makes it possible to freely configure connections between counter submodules and action submodules. However the PWMSM submodules are independent of the time base bus system. The CTM9 configuration is shown in **Figure 13-1**.

13.7.1 Clock Selection

The PWMSM contains an 8-bit prescaler that is clocked by the PCLK1 signal from the CPSM (i.e. the MCU system clock divided by 2 or by 3). A 3-bit field (CLK[2:0]) in the PWMSM status, interrupt and control register (PWMSIC) allows the software to select which of the 8 prescaler outputs drives the PWMSM counter. The prescaler outputs are the main MCU clock divided by: 2, 4, 8, 16, 32, 64, 128 and 512 (or 3, 6, 12, 24, 48, 96, 192 and 768, if the divide-by-3 option is used in the CPSM to generate PCLK1).

13.7.2 The PWMSM Counter (PWMC)

The 16-bit up-counter in the PWMSM provides the time base for the PWM output signal. The counter is held in the 0x0001 state on reset or when the PWMSM is disabled. When the PWMSM is enabled, the counter begins counting at the rate defined by the clock selection. Each time the counter matches the contents of the period register, the counter is preset to 0x0001 and starts to count from that value. The counter can be read at any time without affecting its value. Writing to the counter has no effect.

13.7.3 PWMSM Period Registers and Comparator

The period section of the PWMSM consists of two 16-bit period registers (PWMA1 and PWMA2) and one 16-bit comparator. PWMA2 holds the current PWM period value and PWMA1 holds the next PWM period value. The software establishes the next period of the output PWM signal by writing a value into PWMA1. PWMA2 acts as a double buffer of PWMA1, allowing the contents of PWMA1 to be changed at any time without affecting the current period of the output signal; it cannot be accessed directly by the software. PWMA1 can be read or written at any time. The new value in the PWMA1 register is transferred to PWMA2 on the next full cycle of the output or when a ‘1’ is written to the LOAD bit in the PWMSIC register.

The comparator continuously compares the contents of the PWMA2 register with the value in the PWMSM counter. When a match occurs, the state sequencer sets the output flip-flop and resets the counter to 0x0001.

Period values 0x0000 and 0x0001 are special cases. When PWMA2 contains 0x0000, an output period of 65536 PWM clock periods is generated.

When PWMA2 contains 0x0001, a period match occurs on every PWM clock period: the counter never increments beyond 0x0001 and the output level never changes.

NOTE

A value of 0x0002 in the period register and a value of 0x0001 in the pulse register are the conditions necessary to obtain the maximum possible output frequency for a given PWM clock period.

13.7.4 PWMSM Pulse Width Registers and Comparator

The pulse width section of the PWMSM consists of two 16-bit pulse width registers (PWMB1 and PWMB2) and one 16-bit comparator. PWMB2 holds the current PWM pulse width value and PWMB1 holds the next PWM pulse width value. The software establishes the next pulse width of the output PWM signal by writing a value into PWMB1. Software may write a new pulse width value into PWMB1 at any time and this new value will take effect at the start of the next PWM period (or when the LOAD bit in the PWMSIC register is written to a ‘1’). The PWMSM hardware does not modify the contents of PWMB1 at any time.

PWMB2 acts as a double buffer of PWMB1, allowing the contents of PWMB1 to be changed at any time without affecting the current pulse width of the output signal; it cannot be accessed directly by the software. PWMB1 can be read or written at any time. The new value in the PWMB1 register is transferred to PWMB2 on the next full cycle of the output or when a ‘1’ is written to the LOAD bit in the PWMSIC register.

The pulse width comparator is a 16-bit ‘ones-equality’ comparator that compares the contents of the PWMB2 register with the 16-bit PWM counter. When the counter reaches the value in PWMB2, a match occurs and the output flip-flop is cleared. This pulse width match completes the pulse width; it does not affect the counter. Since a ‘ones-equality’ comparator is used, subsequent comparisons can occur, but will have no effect on the output signal as the output flip-flop has already been cleared.

The PWM output pulse may be as short as one PWM clock period (PWMB2 = 0x0001). It may be as long as one PWM clock period less than the PWM period; for example, the pulse width equal to 65535 PWM clock periods can be obtained by setting PWMB2 = 0xFFFF and PWMA2 = 0x0000.

13.7.5 0% and 100% ‘Pulses’

The 0% and 100% ‘pulses’ are special limiting cases (zero width and infinite width) that are defined by the ‘always clear’ and ‘always set’ states of the output flip-flop.

The 0% pulse is generated by making the pulse width value in PWMB2 equal to 0x0000. The output is a true steady state signal with no glitches.

The 100% pulse is created by making the pulse width value in PWMB2 equal to or greater than the period value in PWMA2. The output is a true steady state signal with no glitches.

It is not possible to have a 100% duty cycle when the output period is selected to be 65536 PWM clock periods (by setting PWMB2 = 0x0000); in this case the maximum duty cycle is 99.998% (100 x 65535/65536).

When using the PWM output signal to generate analog levels, the 0% and 100% pulses provide the full scale values.

Even when 0% or 100% pulses are being generated, the 16-bit PWM counter continues to count and output changes to or from these limit values are done synchronously with the selected period.

13.7.6 PWMSM Coherency

Byte access of registers is discussed in **13.5.1 32-Bit Coherent Access**, however, it should be noted that byte writes to the double buffered registers PWMA1 and PWMB1 are not recommended as the transfer from the primary registers to the secondary registers is done on a word basis.

For most PWMSM operations, 16-bit accesses are sufficient and long word accesses are treated as two word accesses, with one exception — a long word write to the period/pulse width registers. In this case, if the long word write is done within the PWM period, there is no visible effect on the output signal and the new values are stored in PWMA1 and PWMB1 ready to be loaded into the buffer registers at the start of the next period. If, however, the long word write coincides with the end of the period, then the transfer of values from the primary registers to the secondary registers is suppressed until the end of the next PWM period; during this period, the current values in the secondary registers are used for the period and the pulse width.

13.7.7 PWMSM Interrupts

The FLAG bit in the PWMSIC register is set when a new period begins and indicates that the period and pulse width registers (PWMA1 and PWMB1) may be updated with values for the next output period and pulse width. When the FLAG bit is set, an interrupt request is generated on one of eight levels as defined by the interrupt level bits (IL[2:0]) in the PWMSIC register. If the interrupt level is set to zero, interrupts are disabled.

13.7.8 Freeze Action on the PWMSM

When the IMB FREEZE signal is recognized, the PWMSM counter stops incrementing and remains set at its last value. When the FREEZE signal is negated, the counter starts incrementing from its last value, as if nothing had happened.

13.7.9 PWM frequency, Pulse Width and Resolution

Table 13-12 and **Table 13-13** shows the pulse widths and frequencies that can be achieved using the /2 and /3 options and a clock frequency of 16.78 MHz.

Table 13-12 PWM Pulse and Frequency Ranges (in Hz) Using /2 Option (16.78 MHz)

Minimum Pulse Width	Bits of Resolution															
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0.119µs/2	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k	262k	524k	1049k	2097k	4195k
0.238µs/4	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k	262k	524k	1049k	2097k
0.477µs/8	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k	262k	524k	1049k
0.954µs/16	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k	262k	524k
1.91µs/32	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k	262k
3.81µs/64	4.0	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k	131k
7.63µs/128	2.0	4.0	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768	65.5k
30.5µs/512	0.5	1.0	2.0	4.0	8.0	16	32	64	128	256	512	1024	2048	4096	8192	16384

Table 13-13 PWM Pulse and FrequencyRanges (in Hz) Using /3 Option (16.78 MHz)

Minimum Pulse Width	Bits of Resolution															
	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
0.179µs/3	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k	174.8k	349.5k	699.1k	1398k	2796k
0.358µs/6	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k	174.8k	349.5k	699.1k	1398k
0.715µs/12	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k	174.8k	349.5k	699.1k
1.431µs/24	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k	174.8k	349.5k
2.861µs/48	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k	174.8k
5.722µs/96	2.667	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k	87.38k
11.44µs/192	1.333	2.667	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923	21845	43.69k
45.78µs/768	0.333	0.667	1.333	2.667	5.333	10.67	21.33	42.67	85.33	170.7	341.3	682.7	1365	2731	5461	10923

13.7.10 PWM Frequency

The relationship between the PWM output frequency (f_{PWMO}) and the MCU system clock frequency (f_{SYS}) is given by Equation .

$$f_{PWMO} = \frac{f_{SYS}}{N_{CLOCK} \bullet N_{COUNTER}}$$

where N_{CLOCK} is the CPSM clock divide ratio (2 or 3) and $N_{COUNTER}$ is the PWMSM counter divide ratio.

13.7.11 PWM Pulse Width

The minimum output pulse width (t_{PWMIN}) and the MCU system clock frequency (f_{SYS}) is given by Equation .

$$t_{PWMIN} = \frac{N_{CLOCK}}{f_{SYS}}$$

13.7.12 PWM Period and Pulse Width Register Values

The value to be loaded into the PWM period register (PWMA1) to obtain a given period is given by Equation .

$$PWMA1 = \frac{f_{SYS}}{N_{CLOCK} \bullet f_{PWMO}}$$

The value to be loaded into the PWM pulse width register (PWMB1) to obtain a given period is given by Equation .

$$PWMB1 = \frac{t_{PWMO}}{t_{PWMIN}} = \frac{\text{Duty cycle \%}}{100} \cdot PWMA1$$

where $t_{(PWMO)}$ is the actual output pulse width.

13.7.13 PWMSM Register Map and Registers

The PWMSM register map comprises four 16-bit registers as shown in **Table 13-14**. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no meaning nor effect. All register addresses in this section are specified as offsets from the base address of the PWMSM.

Table 13-14 PWMSM Register Map

Address	15 — 8 | 7 — 0
0xYF F228	PWM5 Status, interrupt and control register (PWM5SIC)
0xYF F22A	PWM5 period register(PWM5A)
0xYF F22C	PWM5 pulse width register (PWM5B)
0xYF F22E,	PWM5 counter register (PWM5C)
0xYF F230	PWM6 Status, interrupt and control register (PWM6SIC)
0xYF F232	PWM6 period register(PWM6A)
0xYF F234	PWM6 pulse width register (PWM6B)
0xYF F236	PWM6 counter register (PWM6C)
0xYF F238	PWM7 Status, interrupt and control register (PWM7SIC)
0xYF F23A	PWM7 period register(PWM7A)
0xYF F23C	PWM7 pulse width register (PWM7B)
0xYF F23E	PWM7 counter register (PWM7C)
0xYF F240	PWM8 Status, interrupt and control register (PWM8SIC)
0xYF F242	PWM8 period register(PWM8A)
0xYF F244	PWM8 pulse width register (PWM8B)
0xYF F246	PWM8 counter register (PWM8C)

13.7.13.1 PWMSIC — Status, Interrupt and Control Register

The PWMSIC register contains status, interrupt enable and control bits for the PWMSM. It also contains interrupt level and arbitration bits.

PWM5SIC — PWM Status/Interrupt Control Register **0xYF F228**
PWM6SIC **0xYF F230**
PWM7SIC **0xYF F238**
PWM8SIC **0xYF F240**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
FLAG	IL2	IL1	IL0	IARB3	0	0	0	PIN	0	LOAD	POL	EN	CLK2	CLK1	CLK0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-15 PWMSIC Bit Settings

Bit(s)	Name	Description
15	FLAG	Period completion status. The FLAG bit is a status bit that indicates when the PWM output period has been completed. The FLAG bit is set by the hardware each time a PWM period is completed. Whenever the PWM is enabled, the FLAG bit is set immediately to indicate that the contents of the buffer registers PWMA2 and PWMB2 have been updated, and that the period using these new values has started. It also indicates that the user accessible period and pulse width registers PWMA1 and PWMB1 can be loaded with values for the next PWM period. Once set, the FLAG bit will remain set and will not be affected by any subsequent period completions, until it is cleared by the software. The FLAG bit can only be cleared by software. To clear the flag, the software must first read the bit (as 'one') then write a 'zero' to the bit. Writing a one to the FLAG bit has no effect. When the PWM is disabled the FLAG bit remains in the cleared state. The flag clearing mechanism will work only if no flag setting event occurs between the read and write operations; if a FLAG setting event occurs between the read and write operations, the FLAG bit will not be cleared. When the interrupt level set by the interrupt level bit IL[2:0] is not equal to zero, an interrupt request is generated when the FLAG bit is set. Before returning from the interrupt service routine, the FLAG bit should be cleared by software to prevent the PWMSM from immediately generating another interrupt request on the IMB. 0 = PWM period not completed. 1 = PWM period completed.
14:12	IL[2:0]	Interrupt level. The three interrupt level bits select the interrupt level of requests made by the PWMSMt. 000 = Interrupt disabled 001 = Interrupt level 1 (lowest) 010 = Interrupt level 2 011 = Interrupt level 3 100 = Interrupt level 4 101 = Interrupt level 5 110 = Interrupt level 6 111 = Interrupt level 7 (highest)
11	IARB3	Interrupt arbitration. The read/write IARB3 bit works in conjunction with the IARB[2:0] field in the BIUSM module configuration register. Each module that generates interrupt requests on the IMB must have a unique value in the arbitration field (IARB). This interrupt arbitration identification number is used to arbitrate for the IMB when modules generate simultaneous interrupts of the same priority. The IARB3 bit is cleared by reset.
10:8	—	Reserved
7	PIN	Output pin status. The PIN bit is a status bit that indicates the logic state present on the output pin. The software can thus monitor the waveform being created on the output pin. PIN is a read-only bit; writing to it has no effect. 0 = Logic zero state present on the output pin. 1 = Logic one state present on the output pin.
6	—	Reserved
5	LOAD	Load control. The LOAD bit is a control bit that allows the software to reinitialize the PWMSM and start a new PWM period without causing a glitch on the PWM output signal. 0 = No action. 1 = Load period and pulse width registers. This bit is always read as a zero. Writing a one to this bit results in the following immediate actions: – The contents of PWMA1 (period value) are transferred to PWMA2, – The contents of PWMB1 (pulse width value) are transferred to PWMB2, – The counter register (PWMC) is initialized to 0x0001, – The control logic and state sequencer are reset, – The FLAG bit is set, and – The output flip-flop is set if the new value in PWMB2 is different from 0x0000.

Table 13-15 PWMSIC Bit Settings (Continued)

Bit(s)	Name	Description
4	POL	Output pin polarity control. The POL bit is a control bit that allows the software to set the polarity of the PWM output signal. It works in conjunction with the EN bit and controls whether the PWMSM drives the output pin with the true or inverted value of the output flip-flop, see **Table 13-16**.
3	EN	Enable control. The EN bit is a control bit that allows the software to enable and disable the PWMSM as required. 0 = Disable the PWMSM and stop generation of PWM output pulses. 1 = Enable the PWMSM and start generation of PWM output pulses. While the PWMSM is disabled (EN = 0): – The output flip-flop is held reset and the level on the output pin is set to one or zero according to the state of the POL bit, – The PWMSM's divide-by-256 prescaler is held in reset, – The counter stops incrementing and is held equal to 0x0001, – The comparators are disabled, – And the PWMA1 and PWMB1 registers permanently transfer their contents to the buffer registers (PWMA2 and PWMB2, respectively). When the EN bit is changed from zero to one: – The output flip-flop is set to start the first pulse, – The PWMSM's divide-by-256 prescaler is released, – The counter is released and starts to increment from 0x0001, – And the FLAG bit is set (to indicate that PWMA1 and PWMB1 can be updated with new values of period and pulse width. While EN is set, the PWMSM generates continuously a pulse width modulated output signal based on the data in PWMA2 and PWMB2 (which are updated via PWMA1 and PWMB2 each time a period is completed). To prevent unwanted glitches on the output waveform when disabling the PWMSM, the EN bit should not be cleared by the software until one period has been output as a 0% pulse (PWMB2 = 0x0000)
2:0	CLK[2:0]	Clock rate selection. The CLK bits are control bits that allow the software to select one of the eight counter clock sources coming from the PWMSM prescaler. These bits can be changed by the software at any time. **Table 13-17** shows the counter clock sources and rates in detail.

Table 13-16 PWMSM Output Pin Polarity Selection

Control Bits		Output Pin State	Periodic Edge	Variable Edge	Optional Interrupt On
POL	EN				
0	0	Always low	—	—	—
1	0	Always high	—	—	—
0	1	High pulse	Rising edge	Falling edge	Rising edge
1	1	Low pulse	Falling edge	Rising edge	Falling edge

Table 13-17 PWMSM Clock Rate Selection

PWMSM CLK Bits			CPSM Bit DIV23	PWMSM Clock	Clock Source
CLK2	CLK1	CLK0			
0	0	0	0	f_{SYS} / 2	PCLK1
0	0	1	0	f_{SYS} / 4	Prescaler (/2)
0	1	0	0	f_{SYS} / 8	Prescaler (/4)
0	1	1	0	f_{SYS} / 16	Prescaler (/8)
1	0	0	0	f_{SYS} / 32	Prescaler (/16)
1	0	1	0	f_{SYS} / 64	Prescaler (/32)
1	1	0	0	f_{SYS} / 128	Prescaler (/64)
1	1	1	0	f_{SYS} / 512	Prescaler (/256)
0	0	0	1	f_{SYS} / 3	PCLK1
0	0	1	1	f_{SYS} / 6	Prescaler (/2)
0	1	0	1	f_{SYS} / 12	Prescaler (/4)
0	1	1	1	f_{SYS} / 24	Prescaler (/8)
1	0	0	1	f_{SYS} / 48	Prescaler (/16)
1	0	1	1	f_{SYS} / 96	Prescaler (/32)
1	1	0	1	f_{SYS} / 192	Prescaler (/64)
1	1	1	1	f_{SYS} / 768	Prescaler (/256)

13.7.13.2 PWMA — PWM Period Register

The PWMA register contains the period value for the next cycle of the PWM output waveform. In normal usage, with the PWMSM enabled, the software writes a period value into PWMA1 and this value is then loaded into the PWMA2 register at the end of the current period. If the PWMSM is disabled, a period value written to PWMA1 is loaded into PWMA2 on the next tic (of the MCU system clock). PWMA2 is a temporary register that is used for smoothly updating the PWM period value; it cannot be read or written directly by software.

Software may write a new period value into PWMA1 at any time and this new value will take effect at the start of the next PWM period (or when the LOAD bit in the PWM-SIC register is written to a ‘1’). The PWMSM hardware does not modify the contents of PWMA1 at any time.

PWM5A — PWM Period Register **0xYF F22A**
PWM6A **0xYF F232**
PWM7A **0xYF F23A**
PWM8A **0xYF F242**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

13.7.13.3 PWMB — PWM Pulse Width Register

The PWMB register contains the pulse width value for the next cycle of the PWM output waveform. In normal usage, with the PWMSM enabled, the software writes a pulse width value into PWMB1 and this value is then loaded into the PWMB2 register at the end of the current period. If the PWMSM is disabled, a pulse width value written to PWMB1 is loaded into PWMB2 on the next tic (of the MCU system clock). PWMB2 is a temporary register that is used for smoothly updating the PWM pulse width value; it cannot be read or written directly by software.

Software may write a new pulse width value into PWMB at any time and this new value will take effect at the start of the next PWM period (or when the LOAD bit in the PWMSIC register is written to a ‘1’). The PWMSM hardware does not modify the contents of PWMB1 at any time.

PWM5B — PWM Pulse Width Register **0xYF F22C**
PWM6B **0xYF F234**
PWM7B **0xYF F23C**
PWM8B **0xYF F244**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U

13.7.13.4 PWMC — PWM Counter Register

The counter (register PWMC) is read-only: software may read the counter register at any time; writing to it has no effect. PWMC is loaded with the value 0x0001 on reset and is set to that value and held whenever the PWMSM is disabled (EN = 0).

PWM5C — PWM Counter Register **0xYF F22E**
PWM6C **0xYF F236**
PWM7C **0xYF F23E**
PWM8C **0xYF F246**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

13.8 Bus Interface Unit Submodule (BIUSM)

The bus interface unit submodule (BIUSM) allows all the CTM9 submodules to communicate to the IMB3 via the SMB (sub module bus).

13.8.1 Freeze Action on the BIUSM

When the IMB freeze condition is detected, the FRZ bit in the BIUSM module configuration register determines whether or not the freeze condition is passed on to the other CTM submodules. If FRZ = 0, the freeze condition is ignored; if FRZ = 1, the BIUSM passes the FREEZE signal from the IMB through to the CTM submodules. Each CTM submodule then reacts to the FREEZE signal as defined by its own internal circuitry and control bits.

13.8.2 LPSTOP Action on the BIUSM

When the CPU is stopped by an LPSTOP instruction (from CPU32 or CPU16), the system clock (f_{SYS}) is stopped, thereby shutting down all dependent modules, including the CTM, until the low-power STOP mode is exited.

13.8.3 STOP and WAIT Action on the BIUSM

When the STOP instruction on CPU32 or the WAIT instruction on CPU16 is executed, only the CPU is stopped; the CTM continues to operate as normal. (To stop the CTM operation selectively, refer to the description of the STOP bit in **13.8.4.1 BIUMCR — BIUSM Module Configuration Register**).

13.8.4 BIUSM Registers

The BIUSM register map comprises four 16-bit register locations. As shown in **Table 13-18**, the register block contains the three BIUSM registers and one reserved register. The BIUSM register block always occupies the first four register locations in the CTM register space and cannot be relocated within the CTM structure. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no effect.

Table 13-18 BIUSM Register Map

Address	15 — 8 | 7 — 0
0xYF F200	BIUSM module configuration register (BIUMCR)
0xYF F202	BIUSM test register (BIUTST)
0xYF F204	BIUSM time base register (BIUTBR)

13.8.4.1 BIUMCR — BIUSM Module Configuration Register

The BIUMCR register contains nine defined bits that allow the software to control five functions of the CTM: enabling/disabling of the module, response to FREEZE, vector base address, interrupt arbitration number and access to the time base buses (via the time base register).

BIUMCR — BIUSM Module Configuration Register **0xYF F200**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
STOP	FRZ	0	VECT7	VECT6	IARB2	IARB1	IARB0	0	0	TBRS1	0	0	0	0	TBRS0
RESET:															
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Table 13-19 BIUMCR Bit Settings

Bit(s)	Name	Description
15	STOP	Stop enable. The STOP bit, while asserted, activates the FREEZE signal on the SMB regardless of the state of the FREEZE signal on the IMB. This completely stops the operation of the CTM. Note that some submodules may validate this signal with internal enable bits. The BIUSM continues to operate to allow the CPU access to the submodule's registers. The SMB FREEZE signal remains active until reset or until the STOP bit is negated by the CPU (via the IMB). 0 = Allows operation of the CTM. 1 = Stops operation of the CTM.
14	FRZ	Freeze enable. The FRZ bit, while asserted, activates the FREEZE signal on the SMB when the FREEZE signal on the IMB is active. This completely stops the operation of the CTM. Note that some submodules may validate this signal with internal enable bits. The BIUSM continues to operate to allow the CPU access to the submodule's registers. The SMB FREEZE signal remains active until the FRZ bit is cleared or the IMB FREEZE signal is negated. 0 = Ignores the FREEZE signal on the IMB. 1 = Halts the CTM sub module when the FREEZE signal appears on the IMB.
13	—	Reserved
12:11	VECT[7:8]	Interrupt vector base number. The interrupt vector base number bits select the interrupt vector base number for the CTM. Of the 8 bits necessary for vector number definition, the six least significant bits are programmed by hardware on a submodule basis, while the two remaining bits are provided by VECT7 and VECT6. 00 = Vector base number 0x00. 01 = Vector base number 0x40. 10 = Vector base number 0x80. 11 = Vector base number 0xC0.

Table 13-19 BIUMCR Bit Settings (Continued)

Bit(s)	Name	Description
10:8	IARG[2:0]	Interrupt arbitration identification. The interrupt arbitration bit field (IARB), composed of IARB[2:0] in the BIUMCR and the IARB3 bit within each submodule, provides fifteen different arbitration identification numbers that can be used to arbitrate between interrupt requests occurring on the IMB with the same interrupt priority level. The IARB field defaults to zero on reset, thus preventing the module from arbitrating during an interrupt arbitration acknowledge cycle (IACK). If no IMB arbitration takes place during the IACK cycle the spurious interrupt vector is generated by the SIM (system integration module). This tells the system that the interrupt arbitration number has not been initialized. The seven levels of interrupt are the primary means by which interrupt priority is established. The 4-bit interrupt arbitration number is the secondary priority, allowing up to 15 requests at each primary level. During the IACK cycle the request with the highest arbitration number gets serviced (binary 1111 is the highest priority and binary 0001 is the lowest). Many IMB modules have one software assignable arbitration number for the whole module. The CTM allows two different arbitration numbers to be used by providing each submodule with its own IARB3 bit (which can be set or cleared in software). Once IARB[2:0] are assigned in the BIUSM, they apply to all CTM interrupt requests. Therefore, CTM submodule interrupts can be interleaved in priority with requests from other modules at the same interrupt level.
7:6	—	Reserved
5,0	TBRS1, TBRS0	Time base register bus select. These bits specify which time base bus is accessed when the time base register (BIUTBR) is read. 00 = Time base bus TBB1 01 = Time base bus TBB2 10 = Time base bus TBB3 11 = Time base bus TBB4
4:1	—	Reserved

13.8.4.2 BIUTBR — BIUSM Time Base Register

In normal operation, the BIUTBR is a read-only register used to read the value present on one of the time base buses. The time base bus being accessed is determined by TBRS1 and TBRS0 in the BIUMCR. Writing to the BIUTBR has no effect, except in certain test modes.

BIUTBR — BIUSM Time Base Register **0xYF F204**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
MSB								LSB							
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13.9 Counter Prescaler Submodule (CPSM)

The counter prescaler submodule (CPSM) generates six different clock frequencies which can be used by any counter submodule. Five of these frequencies are derived from a fixed divider. The divide ratio of the last clock frequency is software selectable from a choice of four divide ratios. Note that this submodule is contained within the BIUSM. A block diagram of the CPSM is given in **Figure 13-14**. The clock division ratios available on PCLKx are also shown in the table in **13.9.2.1 CPCR — CPSM Control Register**. These clock signals are provided on the SMB and may be used by any or all CTM submodules.

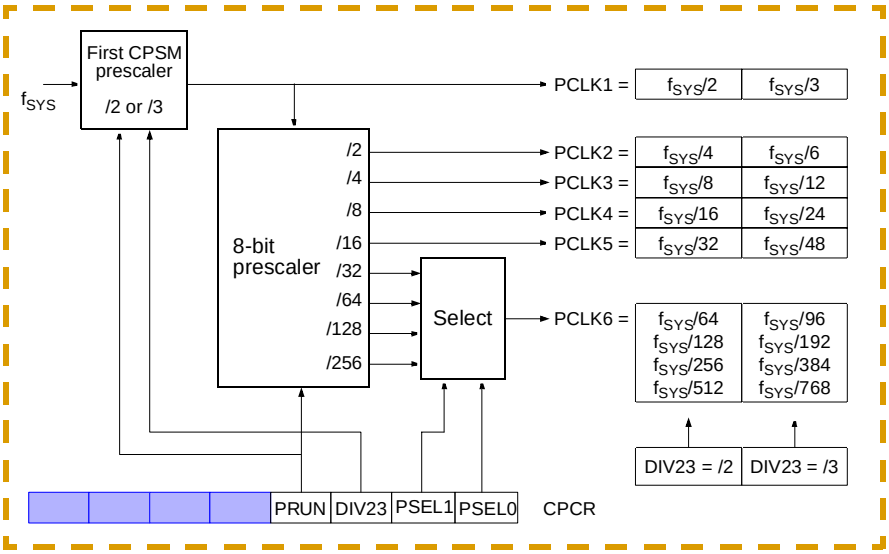


Figure 13-14 CPSM Block Diagram

13.9.1 Freeze Action on the CPSM

When the IMB FREEZE signal is recognized, the CPSM counters stop counting and remain set at their current values. When the FREEZE signal is negated, the counters start incrementing from their current values, as if nothing had happened. All registers are accessible during freeze.

13.9.2 CPSM Registers

The CPSM register map comprises four 16-bit register locations. As shown in **Table 13-20**, the register block contains two CPSM registers and two reserved registers. The CPSM register block always immediately follows the BIUSM register block in the CPSM register map. All unused bits and reserved address locations return zero when read by the software. Writing to unused bits and reserved address locations has no effect.

Table 13-20 CPSM Register Map

Address	15 — 8	7 — 0
0xYF F208	CPSM control register (CPCR)	
0xYF F20A	CPSM test register (CPTR)	

13.9.2.1 CPCR — CPSM Control Register

CPCR — CPSM Control Register **0xYF F208**

MSB 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB 0
0	0	0	0	0	0	0	0	0	0	0	0	PRUN	DIV23	PSEL 1	PSEL 0
RESET:															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-21 CPCR Bit Settings

Bit(s)	Name	Description
15:4	—	Reserved
3	PRUN	Prescaler running. The PRUN bit is a read/write control bit that allows the software to switch the prescaler counter on and off. This bit allows the counters in various CTM submodules to be synchronized. 0 = Prescaler divider is held in reset and is not running. 1 = Prescaler is running.
2	DIV23	Divide by 2 or divide by 3 . The DIV23 bit is a read/write control bit that selects the division ratio of the first prescaler counter. It may be changed by the software at any time and is cleared on reset. 0 = First prescaler stage divides by 2. 1 = First prescaler stage divides by 3.
1:0	PSEL[1:0]	Prescaler division ratio select. These control bits select the division ratio of the programmable prescaler output signal, PCLK6, See **Table 13-22**.

Table 13-22 Prescaler Division Ratio Select

Prescaler Control Register Bits				Prescaler Division Ratio					
PRUN	DIV23	PSEL1	PSEL0	PCLK1	PCLK2	PCLK3	PCLK4	PCLK5	PCLK6
0	X	X	X	0	0	0	0	0	0
1	0	0	0	2	4	8	16	32	64
1	0	0	1	2	4	8	16	32	128
1	0	1	0	2	4	8	16	32	256
1	0	1	1	2	4	8	16	32	512
1	1	0	0	3	6	12	24	48	96
1	1	0	1	3	6	12	24	48	192
1	1	1	0	3	6	12	24	48	384
1	1	1	1	3	6	12	24	48	768

13.9.3 Clock Sources for the Counter Submodules

The software chooses one of seven clock sources for each counter. Six of them are prescaler taps derived from the on-chip oscillator. The highest frequency available to the counter is the MCU system clock divided by 2. Four of the other five taps are binary divisible from the system clock cycle — divide by 4, 8, 16, and 32. Another input clock to the counter is a software defined divide by 64, 128, 256, or 512 from the MCU clock. There is an alternate prescaler option where the MCU clock is divided by 3, 6, 12, 24,

48, 96, 192, 384, and 768. The seventh selectable clock source is an external pin, which may trigger on the rising or falling edge of the input signal. The external input allows the counter to use a frequency not based on the microcontroller oscillator. An alternate use for the external clock source is for event or pulse counting.

13.10 CTM9 Interrupts

The CTM9 is able to generate a diverse set of interrupts on the IMB3. Each interrupting submodule is capable of requesting an interrupt on any of seven levels. A 3-bit level number and a 1-bit arbitration number included in each submodule are initialized by the software. The 3-bit level number selects which of the seven interrupt signals on the IMB are driven by that submodule to create an interrupt request. Of the four priority bits provided on the IMB3 during arbitration among the modules, one of them comes from the interrupting submodule and the CTM9 BIUSM provides the other three. Thus, the CTM9 may respond to two of the possible fifteen arbitration numbers.

During the IMB3 arbitration process, the CTM9 BIUSM manages the separate arbitration among the CTM9 submodules to determine which submodule will respond. Of the submodules which have an interrupt request pending at the level being arbitrated on the IMB, the submodule which has the lowest address is given the highest priority to respond.

Following the interrupt arbitration process, the CTM9 provides an 8-bit vector number. Six of the eight bits are provided by the interrupting submodule. Of the submodules produced to date, a submodule can identify up to two separate interrupt causes, each with unique interrupt vectors. The high-order two bits of the 8-bit vector are provided by the CTM9 BIUSM. The low order six vector bits identify the highest priority interrupt request pending in the CTM9 at the beginning of the arbitration cycle.

13.11 CTM9 Function Examples

The versatility of the CTM9 timer architecture is based on multiple counters and capture/compare channel units interconnected on time-base buses. Rather than present block diagrams of each submodule, this section includes some typical application examples — to show how the submodules can be interconnected to form timing functions. The diagrams used to illustrate these examples show only the blocks utilized for that function.

To illustrate the timing range of the CTM9 in different applications, many of the following paragraphs include time intervals quoted in microseconds and seconds. The assumptions used are that the microcontroller system clock is at 16.78 MHz with minimum prescaling (0.119 microsecond cycle) and with the maximum prescaling (48.0 microsecond cycle). For other system clock cycle rates and prescaler choices, the times mentioned in these paragraphs scale appropriately.

13.11.1 CTM9 Single Input Capture

The CTM9 single-action submodule (SASM) has an input capture register to latch the current state of a time-base bus when an external input edge is detected. The SASM is software programmable to latch on the rising or falling edge of the input signal. The

software also selects one of two time-base buses, each originating at a counter submodule. The software can also enable an interrupt to occur when the input edge is detected to notify the software that new edge capture information is available.

Figure 13-15 shows an example of an MCSM a counter submodule for an SASM configured for input capturing. To measure the period of an incoming signal, the software reads and saves the latched value in register A for one edge, then when the next edge arrives, the software subtracts the new captured value in register A from the previously saved value to obtain the period interval. The maximum period that can be measured is the worst case software response time to a newly captured value.

The software measures the width of a pulse in a very similar way, the only difference is that after each edge, the edge detector is reprogrammed to trigger on the next opposite edge.

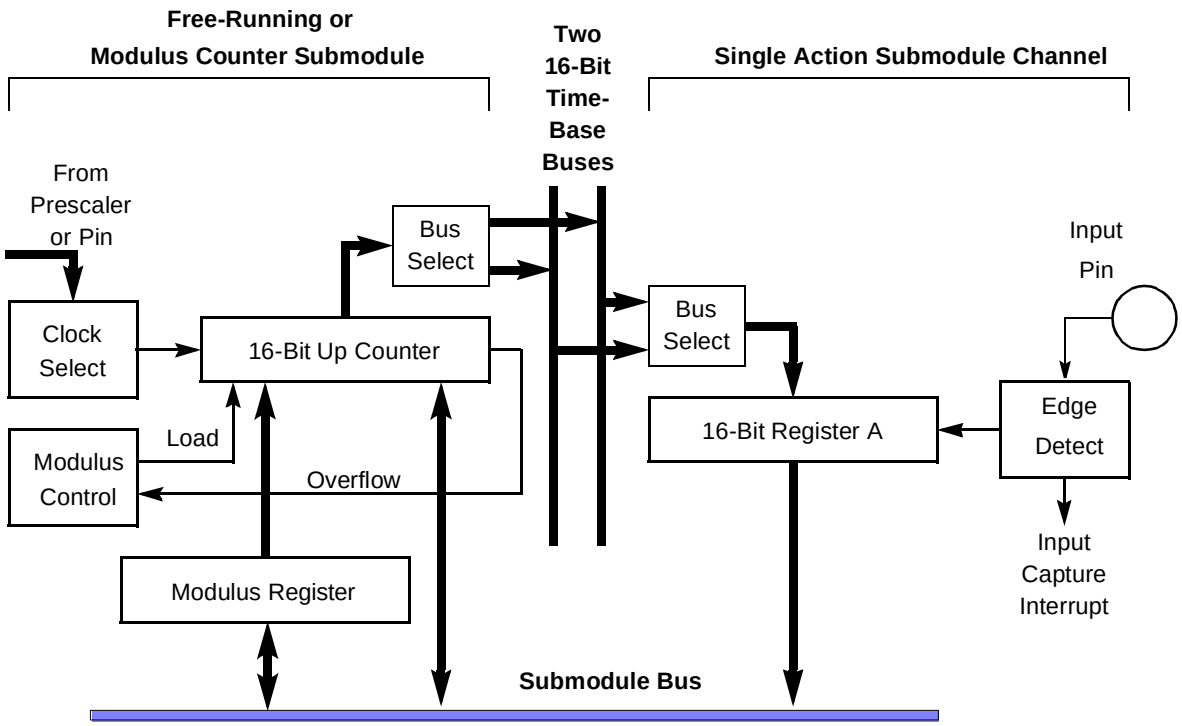


Figure 13-15 CTM9 Example — Single Edge Input Capture

13.11.2 CTM9 Input Double Edge Pulse Width Measurement

To measure the width of an input pulse, the CTM9 double-action submodule (DASM) has two capture registers so that only one interrupt is needed after the second edge. The software can read both edge samples and subtract them to get the pulse width. The leading edge sample is double latched so that the software has the time of one full period of the input signal to read the samples to be sure that nothing is lost. Depending on the prescaler divide ratio, pulses width from 0.119 microseconds to 3

seconds can be measured. Note that a software option is provided to also generate an interrupt after the first edge.

In the example shown in **Figure 13-16**, a counter submodule is used as the time-base for a DASM configured in the input pulse width measurement mode. When the leading edge (programmed for either rising or falling edge) of the input signal occurs, the state of the time-base bus is saved in register B1. When the trailing edge occurs, the time-base bus is latched into register A, and the content of register B1 is transferred to register B2. This operation leaves register B1 free for the next leading edge to occur, as soon as on the next clock cycle. When enabled, an interrupt is provided after the trailing edge, to notify the software that pulse width measurement data is available for a new pulse. After the trailing edge, the software has one cycle time of the input signal to obtain the values for each edge. When software attention is not needed for every pulse, the interrupt can be disabled. The software can at any time read registers A and B2 coherently (using a 32-bit read instruction) to get the latest edge measurements. Since the measurement resolution is 16 bits, signals with pulse duty cycles from 0.0015% to 99.9985% can be measured. The software work is less than half that needed with a timer that requires the software to read one edge and save the value, and then wait for the second edge.

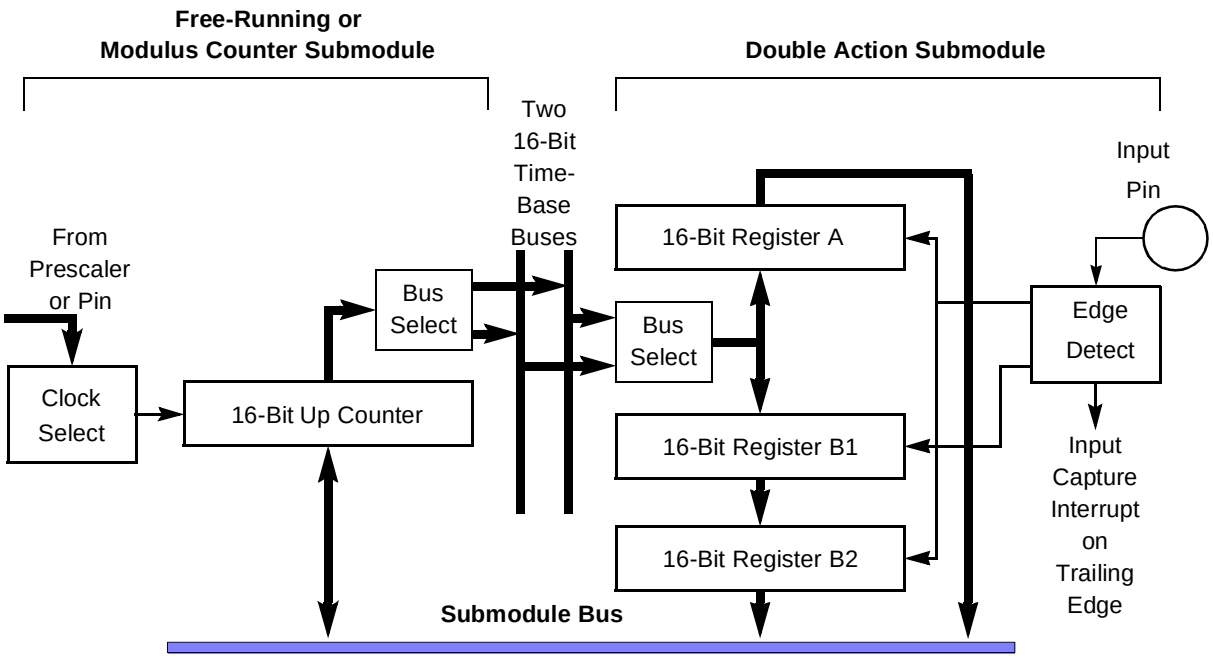


Figure 13-16 CTM9 Example — Double Capture Pulse Width Measurement

13.11.3 CTM9 Input Double Edge Period Measurement

Two samples are also available to the software from a double-action submodule for period measurement. The software can read the previous and the current edge samples and subtract them. As with pulse width measurement, the software can be sure

of not missing samples by insuring that the interrupt response time is faster than the fastest input period. Alternately, when the software is just interested in the latest period measurement, one 32-bit coherent read instruction can get both the current and the previous samples. Depending on the prescaler divide ratio, period times can be measured from 0.119 microseconds to 3 seconds.

Figure 13-17 shows a counter submodule and a DASM combination as an example of period measurement. The software designates whether the rising or falling edge of the input signal is to be used for the measurements. When the edge is detected, the state of the time-base bus is stored in register A, and the content of register B1 is transferred into register B2. After register B2 is safely latched, the content of register A is transferred to register B1. This procedure gives the software coherent current and previous samples in registers A and B2 at all times. An interrupt is available for the cases where the software needs to be aware of each new sample. Note that a software option is provided to also generate an interrupt after the first edge.

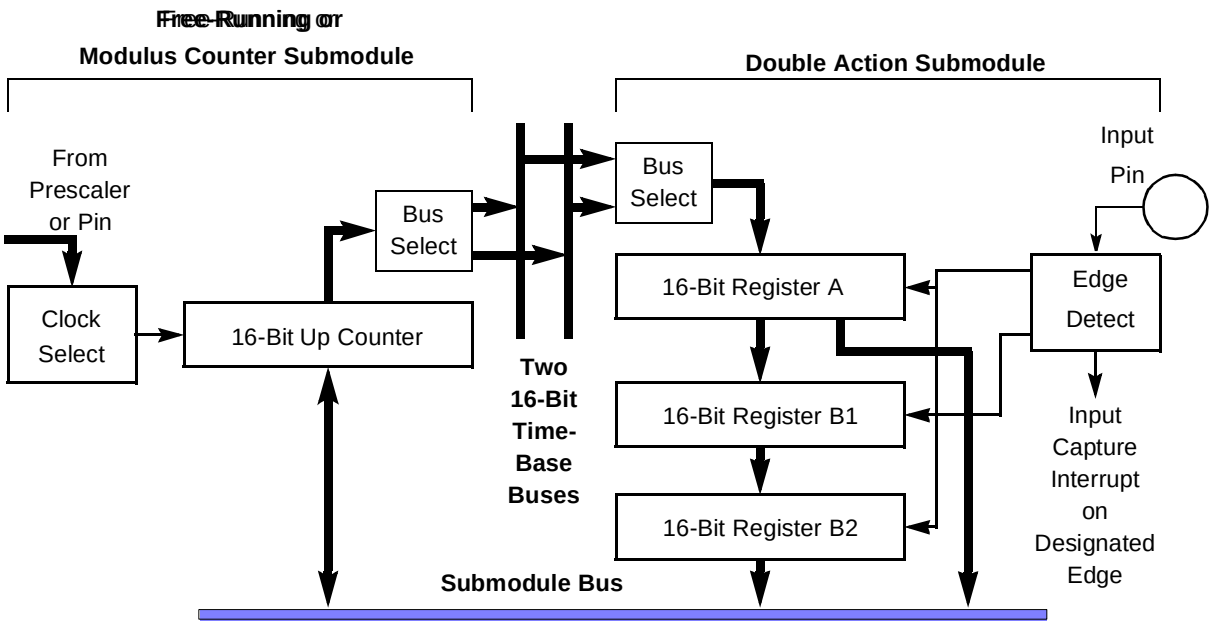


Figure 13-17 CTM9 Example — Double Capture Period Measurement

13.11.4 CTM9 Single Output Compare

To create one output edge, the software can use an SASM channel. The software provides a compare value in a register and the SASM compares that value to the incrementing value seen on one of the time-base buses. When a comparison is detected, the state of the output pin is changed.

The example shown in **Figure 13-18** uses a counter submodule with one channel of an SASM to create an output signal. The software can read the current state of the counter submodule. That, or some other criteria, is used to determine the time-base

value when the output is to change. The software thus writes the compare value into register A. In the SASM control register, the software establishes whether the output flip-flop is to toggle to the opposite state, or is to go to a high or a low level. The output compare interrupt is typically used to notify the software that the previous compare is complete and the SASM is available for a new compare value in register A.

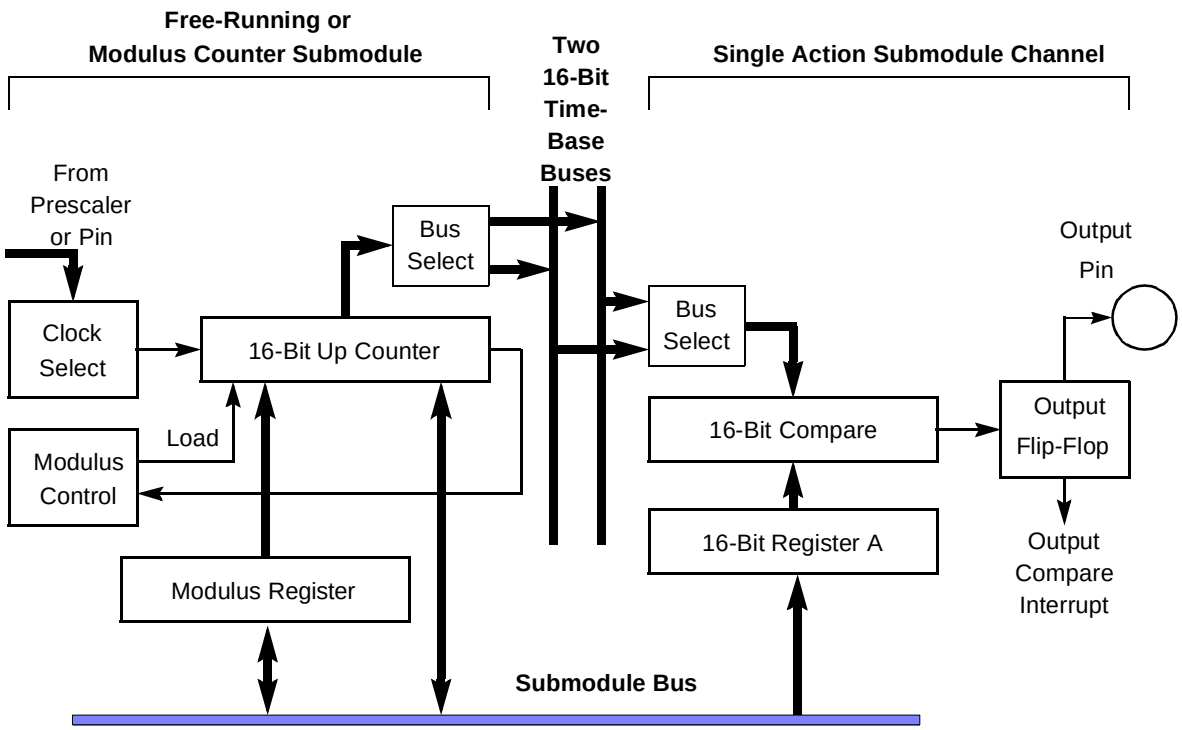


Figure 13-18 CTM9 Example — Single Edge Output Compare

13.11.5 CTM9 Double Edge Single Output Pulse Generation

Software can initialize the CTM9 to generate both the rising and the falling edge of an output pulse. With a DASM, pulses as narrow as 0.119 microseconds can be generated since software action is not needed between the edges. Pulses as long as 3 seconds can be generated. When an interrupt is desired, it can be selected to occur on every edge or only after the second edge.

Figure 13-19 shows how a counter submodule and a DASM can be used to generate both edges of a single output pulse. The software puts the compare value for one edge in register A and the other one in register B2. The DASM automatically creates both edges, and the pulse can be selected by software to be a high-going or a low-going. After the trailing edge, the DASM stops to await further commands from the software. Note that a single edge output can be generated by writing to only one register.

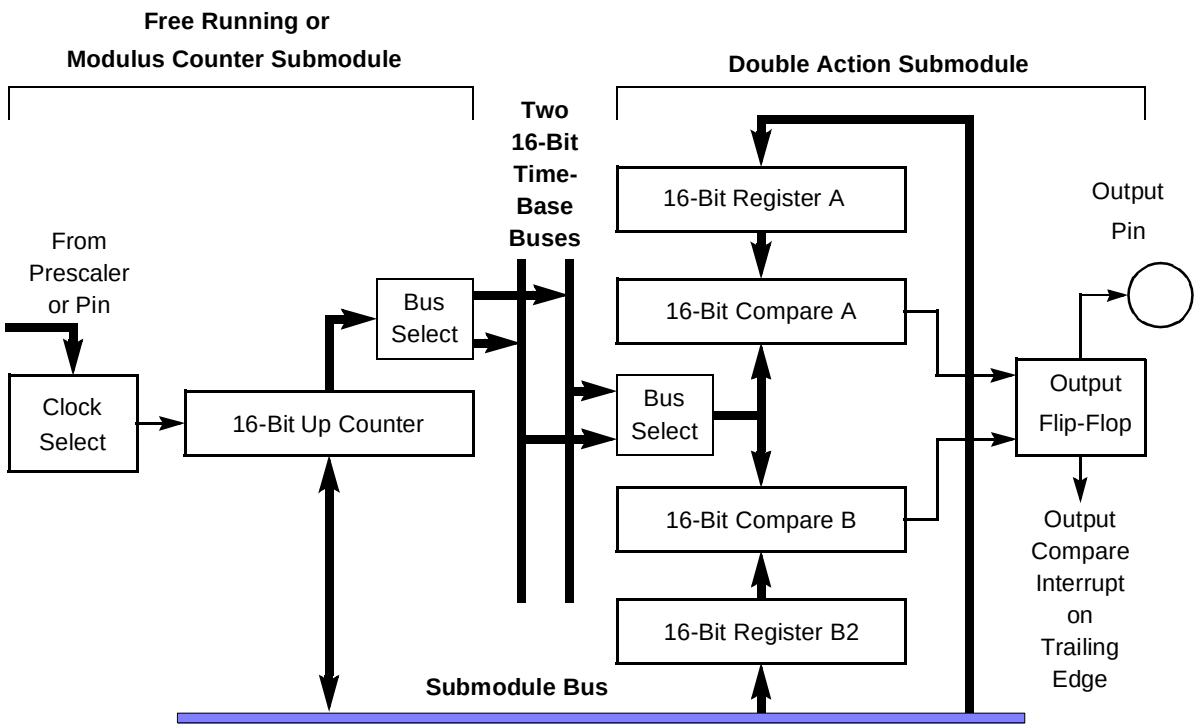


Figure 13-19 CTM9 Example — Double Edge Output Compare

13.11.6 CTM9 Output Pulse Width Modulation With DASM

Output waveforms can be generated with any duty cycle without software involvement. The software sets up a DASM with the compare times for the rising and falling edges, and they are automatically repeated. The software does not need to respond to interrupts to generate continuous pulses. The period may be selected as the period of a free-running counter submodule time-base, times a binary multiplier selected in the DASM. Multiple PWM outputs can be created from multiple DASMs and share one counter submodule, provided that the periods of all of the output signals are a binary multiple of the time-base, and that the counter submodule is operating in a free-running mode. Each DASM has a software selectable “don’t care” on high-order bits of the time-base comparison so that the period of one output can be a binary multiple of another signal. Masking the time-base serves to multiply the period of the time-base by a binary number to form the period of the output waveform. The duty cycle can vary from one cycle to 64K cycles. The frequency can range from 0.3 Hz to 62.5 KHz, though the resolution decreases at the higher frequencies to as low as 7 bits. The generation of output square wave signals is of course the special case where the high and low time are equal.

When an MCSM is used to drive the time-base, the modulus value is the period of the output PWM signal. **Figure 13-20** shows such an example. The polarity of the leading edge of an output waveform is programmable for a rising or a falling edge. The software selects the period of the output signal by programming the MCSM with a modulus

value. The leading edge compare value is written into register A by software, and the trailing edge time is written into register B1. When the leading edge value is reached, the content of register B1 is transferred to register B2, to form the next trailing edge value. Subsequent changes to the output pulse width are made by writing a new time into register B1. Updates to the pulse width are always synchronized to the leading edge of the waveform.

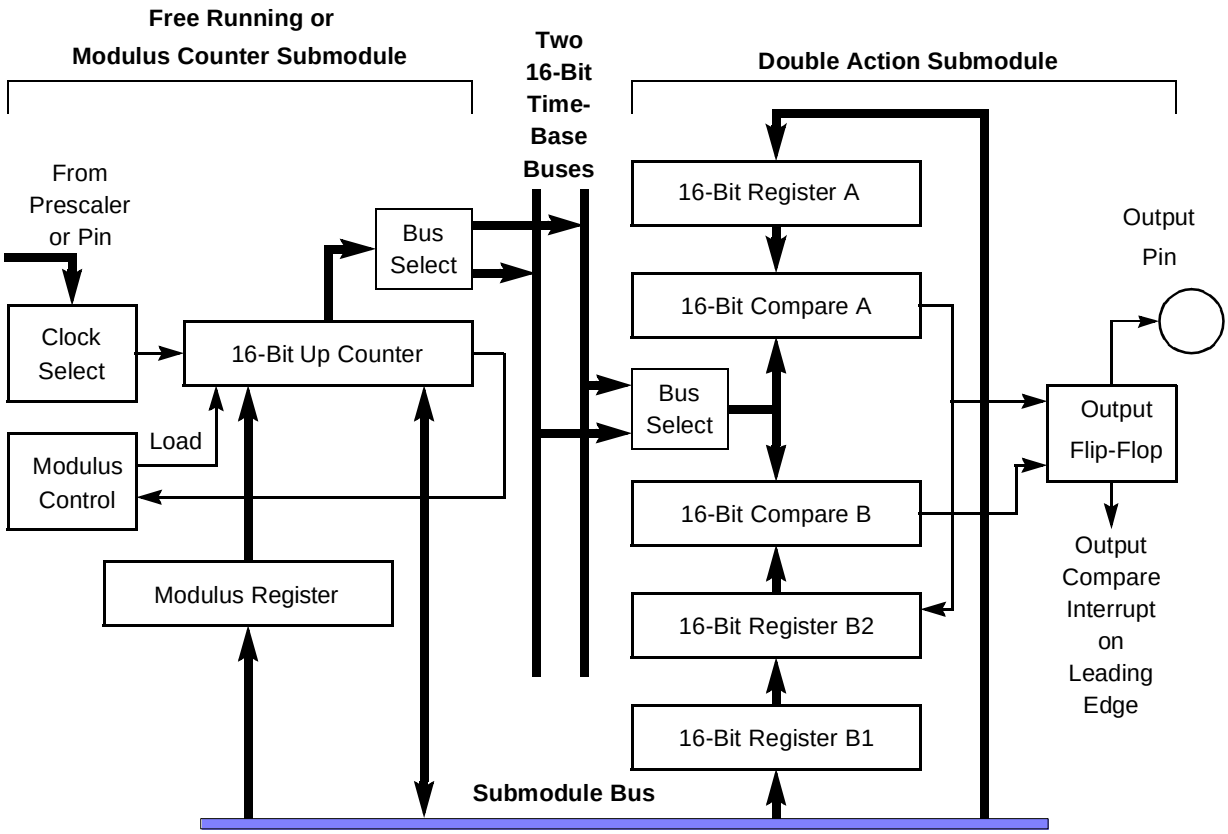


Figure 13-20 CTM9 Example — Pulse Width Modulation Output

It is typical to use the pulse width modulation mode of the DASM without interrupts, though an interrupt can be enabled to occur on the leading edge. When the output is an unchanging repetitive waveform, the DASM continuously generates the signal without any software intervention. When the software needs to change the pulse width, a new trailing edge time is written to the DASM. The output is changed on the next full pulse. When the software needs to change the output at a regular rate, such as an acceleration curve, the leading edge interrupt gives the software one period time to update the new trailing edge time.

13.11.7 CTM9 Input Pulse Accumulation

Counting the number of pulses on an input signal is another capability of the CTM9. Pulse accumulation uses either an FCSM or an MCSM. Since the counters in the

counter submodules are software readable, pulse accumulation does not require the use of an action submodule. The pulse accumulation can operate continuously, interrupting only on binary overflow of the 16-bit counter. When an MCSM is used, an interrupt can instead be created when the pulse accumulation reaches a preprogrammed value. To do that, the two's complement of the value is put in the modulus register and the interrupt occurs when the counter overflows. A similar function can be accomplished with the free-running counter submodule by writing a value into the counter.

APPENDIX A
INTERNAL MEMORY MAP

The tables below use the following notations.

In the Access column:

S = Supervisor Access Only
U = User Access
T = Test Access

In the Reset column:

A = Affected by $\overline{\text{RESET}}$
U = Unchanged
X = Unknown

The codes in the Reset column indicate which reset has an effect on register values.

INDEX of MEMORY MAP TABLES

Table A-1 TouCAN (CAN 2.0B Controller)

Table A-2 CTM9 (Configurable Timer Module)

Table A-3 QADC64 (Queued Analog-to-Digital Converter)

Table A-4 CMFI (CDR MoneT FLASH FOR THE IMB3)

Table A-5 ROM Module

Table A-6 Overlay SRAM Modules (Static Random Access Memory)

Table A-7 DPTRAM (Dual-Port TPU RAM)

Table A-8 SCIM2E (Single-Chip Integration Module)

Table A-9 SRAM Module (Static Random Access Memory)

Table A-10 QSMCM (Queued Serial Multi-Channel Module)

Table A-11 TPU3 (Time Processor Unit)

Table A-1 TouCAN (CAN 2.0B Controller)

Address	Access	Symbol	Register	Size	Reset
0xYF F000 – 0xYF F07F	—	—	Reserved	—	—
0xYF F080	S	TCNMCR	TouCAN Module Configuration Register. See **Table 7-11** for bit descriptions.	16	X
0xYF F082	T	TTR	TouCAN Test Register	16	X
0xYF F084	S	CANICR	TouCAN Interrupt Configuration Register. See **Table 7-12** for bit descriptions.	16	X
0xYF F086	S	CANCTRL0/ CANCTRL1	TouCAN Control Register 0/ TouCAN Control Register 1. See **Table 7-13** and **Table 7-16** for bit descriptions.	16	X
0xYF F088	S	PRESDIV/ CANCTRL2	TouCAN Control and Prescaler Divider Register/ TouCAN Control Register 2. See **Table 7-17** and **Table 7-18** for bit descriptions.	16	X
0xYF F08A	S	TIMER	TouCAN Free-Running Timer Register. See **Table 7-19** for bit descriptions.	16	X
0xYF F090	S	RXGMASKHI	TouCAN Receive Global Mask High. See **Table 7-20** for bit descriptions.	16	X
0xYF F092	S	RXGMASKLO	TouCAN Receive Global Mask Low. See **Table 7-20** for bit descriptions.	16	X
0xYF F094	S	RX14MASKHI	TouCAN Receive Buffer 14 Mask High. See **7.8.9 Receive Buffer 14 Mask Registers** for bit descriptions.	16	X
0xYF F096	S	RX14MASKLO	TouCAN Receive Buffer 14 Mask Low. See **7.8.9 Receive Buffer 14 Mask Registers** for bit descriptions.	16	X
0xYF F098	S	RX15MASKHI	TouCAN Receive Buffer 15 Mask High. See **7.8.10 Receive Buffer 15 Mask Registers** for bit descriptions.	16	X
0xYF F09A	S	RX15MASKLO	TouCAN Receive Buffer 15 Mask Low. See **7.8.10 Receive Buffer 15 Mask Registers** for bit descriptions.	16	X
0xYF F0A0	S	ESTAT	TouCAN Error and Status Register. See **Table 7-21** for bit descriptions.	16	X
0xYF F0A2	S	IMASK	TouCAN Interrupt Masks. See **Table 7-24** for bit descriptions.	16	X
0xYF F0A4	S	IFLAG	TouCAN Interrupt Flags. See **Table 7-25** for bit descriptions.	16	X
0xYF F0A6	S	RXECTR/ TXECTR	TouCAN Receive Error Counter/ TouCAN Transmit Error Counter. See **Table 7-26** for bit descriptions.	16	X
0xYF F100 — 0xYF F10F	S/U	MBUFF0	TouCAN Message Buffer 0. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F110 — 0xYF F11F	S/U	MBUFF1	TouCAN Message Buffer 1. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F120 — 0xYF F12F	S/U	MBUFF2	TouCAN Message Buffer 2. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U

Table A-1 TouCAN (CAN 2.0B Controller) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF F130 — 0xYF F13F	S/U	MBUFF3	TouCAN Message Buffer 3. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F140 — 0xYF F14F	S/U	MBUFF4	TouCAN Message Buffer 4. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F150 — 0xYF F15F	S/U	MBUFF5	TouCAN Message Buffer 5. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F160 — 0xYF F16F	S/U	MBUFF6	TouCAN Message Buffer 6. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F170 — 0xYF F17F	S/U	MBUFF7	TouCAN Message Buffer 7. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F180 — 0xYF F18F	S/U	MBUFF8	TouCAN Message Buffer 8. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F190 — 0xYF F19F	S/U	MBUFF9	TouCAN Message Buffer 9. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1A0 — 0xYF F1AF	S/U	MBUFF10	TouCAN Message Buffer 10. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1B0 — 0xYF F1BF	S/U	MBUFF11	TouCAN Message Buffer 11. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1C0 — 0xYF F1CF	S/U	MBUFF12	TouCAN Message Buffer 12. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1D0 — 0xYF F1DF	S/U	MBUFF13	TouCAN Message Buffer 13. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1E0 — 0xYF F1EF	S/U	MBUFF14	TouCAN Message Buffer 14. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U
0xYF F1F0 — 0xYF F1FF	S/U	MBUFF15	TouCAN Message Buffer 15. See **Figure 7-3** and **Figure 7-4** for message buffer definitions.	—	U

Table A-2 CTM9 (Configurable Timer Module)

Address	Access	Symbol	Register	Size	Reset
BIUSM (CTM9 Bus Interface Unit Submodule)					
0xYF F200	S	**BIUMCR**	**BIUSM Module Configuration** Register. See **Table 13-19** for bit descriptions.	16	X
0xYF F202	T	**BIUTEST**	**BIUSM Test** Register.	16	X
0xYF F204	S	**BIUTBR**	**BIUSM Time Base** Register. See **13.8.4.2 BIUTBR — BIUSM Time Base Register** for bit descriptions.	16	X
CPSM (CTM9 Counter Prescaler Submodule)					
0xYF F208	S	CPCR	CPSM Control Register. See **Table 13-21** for bit descriptions.	16	X
0xYF F20A	T	CPTR	CPSM Test Register.	16	X
MCSM2 (CTM9 Modulus Counter Submodule 2)					
0xYF F210	S	MCSM2SIC	MCSM2 Status/Interrupt/Control Register. See **Table 13-5** for bit descriptions.	16	X
0xYF F212	S	MCSM2CNT	MCSM2 Counter Register. See **13.3.10 MCSMCNT — MCSM Counter Register** for bit descriptions.	16	X
0xYF F214	S	MCSM2ML	MCSM2 Modulus Latch Register. See **13.3.11 MCSMML — MCSM Modulus Latch Register** for bit descriptions.	16	X
DASM3 (CTM9 Double-Action Submodule 3)					
0xYF F218	S	DASM3SIC	DASM3 Status/Interrupt/Control Register. See **Table 13-11** for bit descriptions.	16	X
0xYF F21A	S	DASM3A	DASM3 Register A. See **13.5.5.2 DASMA — DASM Data Register A** for bit descriptions.	16	X
0xYF F21C	S	DASM3B	DASM3 Register B. See **13.5.5.3 DASMB — DASM Data Register B** for bit descriptions.	16	X
DASM4 (CTM9 Double-Action Submodule 4)					
0xYF F220	S	DASM4SIC	DASM4 Status/Interrupt/Control Register. See **Table 13-11** for bit descriptions.	16	X
0xYF F222	S	DASM4A	DASM4 Register A. See **13.5.5.2 DASMA — DASM Data Register A** for bit descriptions.	16	X
0xYF F224	S	DASM4B	DASM4 Register B. See **13.5.5.3 DASMB — DASM Data Register B** for bit descriptions.	16	X
PWM5 (CTM9 Pulse Width Modulation Submodule 5)					
0xYF F228	S	PWM5SIC	PWM5 Status, Interrupt and Control Register. See **Table 13-15** for bit descriptions.	16	X
0xYF F22A	S	PWM5A	PWM5 Period Register. See **13.7.13.2 PWMA — PWM Period Register** for bit descriptions.	16	X
0xYF F22C	S	PWM5B	PWM5 Pulse Width Register. See **13.7.13.3 PWMB — PWM Pulse Width Register** for bit descriptions.	16	X
0xYF F22E	S	PWM5C	PWM5 Counter Register. See **13.7.13.4 PWMC — PWM Counter Register** for bit descriptions.	16	X

Table A-2 CTM9 (Configurable Timer Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
PWM6 (CTM9 Pulse Width Modulation Submodule 6)					
0xYF F230	S	PWM6SIC	PWM6 Status, Interrupt and Control Register. See **Table 13-15** for bit descriptions.	16	X
0xYF F232	S	PWM6A	PWM6 Period Register. See **13.7.13.2 PWMA — PWM Period Register** for bit descriptions.	16	X
0xYF F234	S	PWM6B	PWM6 Pulse Width Register. See **13.7.13.3 PWMB — PWM Pulse Width Register** for bit descriptions.	16	X
0xYF F236	S	PWM6C	PWM6 Counter Register. See **13.7.13.4 PWMC — PWM Counter Register** for bit descriptions.	16	X
PWM7 (CTM9 Pulse Width Modulation Submodule 7)					
0xYF F238	S	PWM7SIC	PWM7 Status, Interrupt and Control Register. See **Table 13-15** for bit descriptions.	16	X
0xYF F23A	S	PWM7A	PWM7 Period Register. See **13.7.13.2 PWMA — PWM Period Register** for bit descriptions.	16	X
0xYF F23C	S	PWM7B	PWM7 Pulse Width Register. See **13.7.13.3 PWMB — PWM Pulse Width Register** for bit descriptions.	16	X
0xYF F23E	S	PWM7C	PWM7 Counter Register. See **13.7.13.4 PWMC — PWM Counter Register** for bit descriptions.	16	X
PWM8 (CTM9 Pulse Width Modulation Submodule 8)					
0xYF F240	S	PWM8SIC	PWM8 Status, Interrupt and Control Register. See **Table 13-15** for bit descriptions.	16	X
0xYF F242	S	PWM8A	PWM8 Period Register. See **13.7.13.2 PWMA — PWM Period Register** for bit descriptions.	16	X
0xYF F244	S	PWM8B	PWM8 Pulse Width Register. See **13.7.13.3 PWMB — PWM Pulse Width Register** for bit descriptions.	16	X
0xYF F246	S	PWM8C	PWM8 Counter Register. See **13.7.13.4 PWMC — PWM Counter Register** for bit descriptions.	16	X
DASM9 (CTM9 Double-Action Submodule 9)					
0xYF F248	S	DASM9SIC	DASM9 Status/Interrupt/Control Register. See **Table 13-11** for bit descriptions.	16	X
0xYF F24A	S	DASM9A	DASM9 Register A. See **13.5.5.2 DASMA — DASM Data Register A** for bit descriptions.	16	X
0xYF F24C	S	DASM9B	DASM9 Register B. See **13.5.5.3 DASMB — DASM Data Register B** for bit descriptions.	16	X
DASM10 (CTM9 Double-Action Submodule 10)					
0xYF F250	S	DASM10SIC	DASM10 Status/Interrupt/Control Register. See **Table 13-11** for bit descriptions.	16	X
0xYF F252	S	DASM10A	DASM10 Register A. See **13.5.5.2 DASMA — DASM Data Register A** for bit descriptions.	16	X

Table A-2 CTM9 (Configurable Timer Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF F254	S	DASM10B	DASM10 Register B. See **13.5.5.3 DASMB — DASM Data Register B** for bit descriptions.	16	X
MCSM11 (CTM9 Modulus Counter Submodule 11)					
0xYF F258	S	MCSM11SIC	MCSM11 Status/Interrupt/Control Register. See **Table 13-5** for bit descriptions.	16	X
0xYF F25A	S	MCSM11CNT	MCSM11 Counter Register. See **13.3.10 MCSMCNT — MCSM Counter Register** for bit descriptions.	16	X
0xYF F25C	S	MCSM11ML	MCSM11 Modulus Latch Register. See **13.3.11 MCSMML — MCSM Modulus Latch Register** for bit descriptions.	16	X
FCSM12 (CTM9 Free Running Counter Submodule 12)					
0xYF F260	S	FCSM12SIC	FCSM12 Status, Interrupt and Control Register. See **Table 13-3** for bit descriptions.	16	X
0xYF F262	S	FCSM12CNT	FCSM12 Counter Register. See **13.2.7.2 FCSMCNT — FCSM Counter Register** for bit descriptions.	16	X
SASM14 (CTM9 Single-Action Submodule 14)					
0xYF F270	S	S14ICA	SASM14 Status/Interrupt/Control Register A. See **Table 13-7** for bit descriptions.	16	X
0xYF F272	S	S14DATA	SASM14 Data Register. See **13.4.4.2 SDATA — SASM Data Register A** for bit descriptions.	16	X
0xYF F274	S	S14ICB	SASM14 Status/Interrupt/Control Register B. See **13.4.4.3 SICB — SASM Status/Interrupt Control Register B** for bit descriptions.	16	X
0xYF F276	S	S14DATB	SASM14 Data Register B. See **13.4.4.4 SDATB — SASM Data Register B** for bit descriptions.	16	X
SASM16 (CTM9 Single-Action Submodule 16)					
0xYF F280	S	S16ICA	SASM16 Status/Interrupt/Control Register. See **Table 13-7** for bit descriptions.	16	X
0xYF F282	S	S16DATA	SASM16 Data Register. See **13.4.4.2 SDATA — SASM Data Register A** for bit descriptions.	16	X
0xYF F284	S	S16ICB	SASM16 Status/Interrupt/Control Register B. See **13.4.4.3 SICB — SASM Status/Interrupt Control Register B** for bit descriptions.	16	X
0xYF F286	S	S16DATB	SASM16 Data Register B. See **13.4.4.4 SDATB — SASM Data Register B** for bit descriptions.	16	X
SASM18 (CTM9 Single-Action Submodule 18)					
0xYF F290	S	S18ICA	SASM18 Status/Interrupt/Control Register. See **Table 13-7** for bit descriptions.	16	X
0xYF F292	S	S18DATA	SASM18 Data Register. See **13.4.4.2 SDATA — SASM Data Register A** for bit descriptions.	16	X
0xYF F294	S	S18ICB	SASM18 Status/Interrupt/Control Register B. See **13.4.4.3 SICB — SASM Status/Interrupt Control Register B** for bit descriptions.	16	X

Table A-2 CTM9 (Configurable Timer Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF F296	S	S18DATB	SASM18 Data Register B. See **13.4.4.4 SDATB — SASM Data Register B** for bit descriptions.	16	X
SASM20 (CTM9 Single-Action Submodule 20)					
0xYF F2A0	S	S20ICA	SASM20 Status/Interrupt/Control Register. See **Table 13-7** for bit descriptions.	16	X
0xYF F2A2	S	S20DATA	SASM20 Data Register. See **13.4.4.2 SDATA — SASM Data Register A** for bit descriptions.	16	X
0xYF F2A4	S	S20ICB	SASM20 Status/Interrupt/Control Register B. See **13.4.4.3 SICB — SASM Status/Interrupt Control Register B** for bit descriptions.	16	X
0xYF F2A6	S	S20DATB	SASM20 Data Register B. See **13.4.4.4 SDATB — SASM Data Register B** for bit descriptions.	16	X

Table A-3 QADC64 (Queued Analog-to-Digital Converter)

Address	Access	Symbol	Register	Size	Reset
0xYF F400	S	QADC64MCR	QADC64 Module Configuration Register. See **Table 5-7** for bit descriptions.	16	X
0xYF F402	T	QADC64TEST	QADC64 Test Register	16	X
0xYF F404	S	QADC64INT	Interrupt Register. See **Table 5-8** for bit descriptions.	16	X
0xYF F406	S	PORTQA/ PORTQB	Port A and Port B Data. See **Table 5-9** for bit descriptions.	16	X
0xYF F408	S	DDRQA	Port A Data Direction Register. See **Table 5-10** for bit descriptions.	16	X
0xYF F40A	S	QACR0	QADC64 Control Register 0. See **Table 5-11** for bit descriptions.	16	X
0xYF F40C	S	QACR1	QADC64 Control Register 1. See **Table 5-12** for bit descriptions.	16	X
0xYF F40E	S	QACR2	QADC64 Control Register 2. See **Table 5-14** for bit descriptions.	16	X
0xYF F410	S	QASR0	QADC64 Status Register 0. See **Table 5-16** for bit descriptions.	16	X
0xYF F412	S	QASR1	QADC64 Status Register 1. See **Table 5-18** for bit descriptions.	16	X
0xYF F414 – 0xYF F5FE	—	—	Reserved	—	—
0xYF F600 – 0xYF F67F	S	CCW	QADC64 Conversion Command Word Table. See **Table 5-19** for bit descriptions.	16	X
0xYF F680 – 0xYF F6FE	S	RJURR	QADC64 Result Word Table Right-Justified, Unsigned Result Register. See **5.12.11 Result Word Table** for bit descriptions.	16	X

Table A-3 QADC64 (Queued Analog-to-Digital Converter) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF F700 – 0xYF F77E	S	LJSRR	QADC64 Result Word Table Left-Justified, Signed Result Register. See **5.12.11 Result Word Table** for bit descriptions.	16	X
0xYF F780 – 0xYF F7FE	S	LJURR	QADC64 Result Word Table Left-Justified, Unsigned Result Register. See **5.12.11 Result Word Table** for bit descriptions.	16	X

Table A-4 CMFI (CDR MoneT FLASH FOR THE IMB3)

Address	Access	Symbol	Register	Size	Reset
0xYF F800	S	CMFIMCR	CMFI Module Configuration Register. See **Table 10-4** for bit descriptions.	16	X[1]
0xYF F802	—	—	Reserved	—	—
0xYF F804	S	CMFITST	CMFI Module Test Register. See **Table 10-5** for bit descriptions.	16	X
0xYF F806	—	—	Reserved	—	—
0xYF F808	S	CMFIBAH	CMFI Base Address High Register. See **Table 10-8** for bit descriptions.	16	X[1]
0xYF F80A	S	CMFIBAL	CMFI Base Address Low Register. See **Table 10-8** for bit descriptions.	16	X[1]
0xYF F80C	S	CMFICTL1	CMFI High Voltage Control Register 1. See **Table 10-9** for bit descriptions.	16	X[1]
0xYF F80E	S	CMFICTL2	CMFI High Voltage Control Register 2. See **Table 10-10** for bit descriptions.	16	X[1]
0xYF F810 – 0xYF F816	S	CMFIBS[3:0]	CMFI Shadow Block Registers. See **10.6.6.3 Programming Shadow Information** for bit descriptions.	16	X

NOTES:
1. Reset state determined by contents of the shadow row.

Table A-5 ROM Module

Address	Access	Symbol	Register	Size	Reset
0xYF F820	S	ROMMCR	ROM Module Configuration Register. See **Table 12-2** for bit descriptions.	16	X
0xYF F824	S	ROMBAH	ROM Module Base Address High Register. See **Table 12-3** for bit descriptions.	16	X
0xYF F826	S	ROMBAL	ROM Module Base Address Low Register. See **Table 12-3** for bit descriptions.	16	X
0xYF F828	S	SIGHI	ROM Module Signature High Register. See **Table 12-4** for bit descriptions.	16	X
0xYF F82A	S	SIGLO	ROM Module Signature Low Register. See **Table 12-5** for bit descriptions.	16	X
0xYF F830 – 0xYF F836	S	ROMBS[0:4]	ROM Module Bootstrap Information Words. See **12.5 Bootstrap Information Words (ROMBS0–ROMBS3)** for bit descriptions.	16	X

Table A-6 Overlay SRAM Modules (Static Random Access Memory)

Address	Access	Symbol	Register	Size	Reset
SRAM1					
0xYF F840	S	RAMMCR1	SRAM1 Module Configuration Register. See **Table 11-1** for bit descriptions.	16	X
0xYF F842	T	RAMTST1	SRAM1 Test Register.	16	X
0xYF F844	S	RAMBAH1	SRAM1 Base Address High Register. See **Table 11-3** for bit descriptions.	16	X
0xYF F846	S	RAMBAL1	SRAM1 Base Address Low Register. See **Table 11-3** for bit descriptions.	16	X
SRAM2					
0xYF F848	S	RAMMCR2	SRAM2 Module Configuration Register. See **Table 11-1** for bit descriptions.	16	X
0xYF F84A	T	RAMTST2	SRAM2 Test Register.	16	X
0xYF F84C	S	RAMBAH2	SRAM2 Base Address High Register. See **Table 11-3** for bit descriptions.	16	X
0xYF F84E	S	RAMBAL2	SRAM2 Base Address Low Register. See **Table 11-3** for bit descriptions.	16	X
SRAM3					
0xYF F848	S	RAMMCR3	SRAM3 Module Configuration Register. See **Table 11-1** for bit descriptions.	16	X
0xYF F84A	T	RAMTST3	SRAM3 Test Register.	16	X
0xYF F84C	S	RAMBAH3	SRAM3 Base Address High Register. See **Table 11-3** for bit descriptions.	16	X
0xYF F84E	S	RAMBAL3	SRAM3 Base Address Low Register. See **Table 11-3** for bit descriptions.	16	X
SRAM4					
0xYF F848	S	RAMMCR4	SRAM4 Module Configuration Register. See **Table 11-1** for bit descriptions.	16	X
0xYF F84A	T	RAMTST4	SRAM4 Test Register.	16	X
0xYF F84C	S	RAMBAH4	SRAM4 Base Address High Register. See **Table 11-3** for bit descriptions.	16	X
0xYF F84E	S	RAMBAL4	SRAM4 Base Address Low Register. See **Table 11-3** for bit descriptions.	16	X

Table A-7 DPTRAM (Dual-Port TPU RAM)

Address	Access	Symbol	Register	Size	Reset
0xYF F880	S	DPTMCR	DPT Module Configuration Register. See **Table 9-2** for bit descriptions.	16	X
0xYF F882	T	DPTTCR	DPT Test Register.	16	X
0xYF F884	S	DPTBAR	DPT Array Address Register. See **Table 9-3** for bit descriptions.	16	X
0xYF F886	S	MISRH	DPT Multiple Input Signature Register High. See **9.4.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	16	X

Table A-7 DPTRAM (Dual-Port TPU RAM) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF F888	S	MISRL	DPT Multiple Input Signature Register Low. See **9.4.4 MISR High (MISRH) and MISR Low (MISRL)** for bit descriptions.	16	X
0xYF F88A	S	MISCNT	DPT Multiple Input Signature Counter. See **9.4.5 MISC Counter (MISCNT)** for bit descriptions.	16	X

Table A-8 SCIM2E (Single-Chip Integration Module)

Address	Access	Symbol	Register	Size	Reset
0xYF FA00	S	SCIMMCR	SCIM2E Module Configuration Register. See **Table 4-1** for bit descriptions.	16	X
0xYF FA04	S	SYNCR	SCIM2E Synthesizer Control Register. See **4.3.6 Clock Synthesizer Control Register** for bit descriptions.	16	X
0xYF FA06	S	RSR	SCIM2E Rest Status Register. See **Table 4-23** for bit descriptions.	16	X
0xYF FA0A	S	PORTA	SCIM2E Port A Data Register. See **4.10.2 Port A and B Data Registers** for bit descriptions.	16	X
0xYF FA0B	S	PORTB	SCIM2E Port B Data Register. See **4.10.2 Port A and B Data Registers** for bit descriptions.	16	X
0xYF FA0C	S	PORTG	SCIM2E Port G Data Register. See **4.10.5.1 Port G and H Data Registers** for bit descriptions.	16	X
0xYF FA0D	S	PORTH	SCIM2E Port HData Register. See **4.10.5.1 Port G and H Data Registers** for bit descriptions.	16	X
0xYF FA0E	S	DDRG	SCIM2E Port G Data Direction Register. See **4.10.5.2 Port G and H Data Direction Registers** for bit descriptions.	16	X
0xYF FA0F	S	DDRH	SCIM2E Port H Data Direction Register. See **4.10.5.2 Port G and H Data Direction Registers** for bit descriptions.	16	X
0xYF FA11	S	PORTE0	SCIM2E Port E0 Data Register. See **4.10.3.1 Port E Data Register** for bit descriptions.	8	X
0xYF FA13	S	PORTE1	SCIM2E Port E1 Data Register. See **4.10.3.1 Port E Data Register** for bit descriptions.	8	X
0xYF FA14	S	DDRAB	SCIM2E Port A/B Data Direction Register. See **4.10.3.2 Port E Data Direction Register** for bit descriptions.	16	X
0xYF FA15	S	DDRE	SCIM2E Port E Data Direction Register. See **4.10.3.2 Port E Data Direction Register** for bit descriptions.	16	X
0xYF FA17	S	PEPAR	SCIM2E Port E Pin Assignment Register. See **4.10.3.3 Port E Pin Assignment Register** for bit descriptions.	8	X
0xYF FA19	S	PORTF0	SCIM2E Port F Data Register 0. See **4.10.4.1 Port F Data Register** for bit descriptions.	8	X

Table A-8 SCIM2E (Single-Chip Integration Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF FA1B	S	PORTF1	SCIM2E Port F Data Register 1. See **4.10.4.1 Port F Data Register** for bit descriptions.	8	X
0xYF FA1F	S	PFPAR	SCIM2E Port F Pin Assignment Register 1. See **4.10.4.3 Port F Pin Assignment Register** for bit descriptions.	8	X
0xYF FA1D	S	DDRF	SCIM2E Port F Data Direction Register 1. See **4.10.4.2 Port F Data Direction Register** for bit descriptions.	8	X
0xYF FA21	S	SYPCR	SCIM2E System Protection Control Register. See **Table 4-15** for bit descriptions.	8	X
0xYF FA22	S	PICR	SCIM2E Periodic Interrupt Control Register. See **Table 4-9** for bit descriptions.	16	X
0xYF FA24	S	PITR	SCIM2E Periodic Interrupt Timer Register. See **Table 4-9** for bit descriptions.	16	X
0xYF FA27	S	SWSR	SCIM2E Software Watchdog Service Register. See **4.4.6.1 Software Watchdog Service Register** for bit descriptions.	8	X
0xYF FA29	S	PORTFE	SCIM2E Port F Edge-Detect Flag Register. See **4.10.4.4 Port F Edge-Detect Flag Register** for bit descriptions.	8	X
0xYF FA2B	S	PFIVR	SCIM2E Port F Edge-Detect Interrupt Vector Register. See **4.10.4.5 Port F Edge-Detect Interrupt Vector** for bit descriptions.	8	X
0xYF FA2D	S	PFLVR	SCIM2E Port F Edge-Detect Interrupt Level Register. See **4.10.4.6 Port F Edge-Detect Interrupt Level** for bit descriptions.	8	X
0xYF FA41	S	PORTC	SCIM2E PORTC Data Register. See **4.9.1.1 Port C Data Register** for bit descriptions.	8	X
0xYF FA44	S	CSPAR0	SCIM2E Chip-Select Pin Assignment Register 0. See **4.9.1 Chip-Select Pin Assignment Register** for bit descriptions.	16	X
0xYF FA46	S	CSPAR1	SCIM2E Chip-Select Pin Assignment Register 1. See **4.9.1 Chip-Select Pin Assignment Register** for bit descriptions.	16	X
0xYF FA48	S	CSBARBT	SCIM2E Chip-Select Base Address Register Boot. See **Table 4-36** for bit descriptions.	16	X
0xYF FA4A	S	CSORBT	SCIM2E Chip-Select Option Register Boot. See **Table 4-37** for bit descriptions.	16	X
0xYF FA4C	S	CSBAR0	SCIM2E Chip-Select Base Address Register 0. See **Table 4-36** for bit descriptions.	16	X
0xYF FA4E	S	CSOR0	SCIM2E Chip-Select Option Register 0. See **Table 4-37** for bit descriptions.	16	X
0xYF FA50 – 0xYF FA56	S	—	Reserved	—	X
0xYF FA58	S	CSBAR3	SCIM2E Chip-Select Base Address Register 3. See **Table 4-36** for bit descriptions.	16	X
0xYF FA5A	S	CSOR3	SCIM2E Chip-Select Option Register 3. See **Table 4-37** for bit descriptions.	16	X
0xYF FA5C – 0xYF FA5E	S	—	Reserved	—	X

Table A-8 SCIM2E (Single-Chip Integration Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF FA60	S	CSBAR5	SCIM2E Chip-Select Base Address Register 5. See **Table 4-36** for bit descriptions.	16	X
0xYF FA62	S	CSOR5	SCIM2E Chip-Select Option Register 5. See **Table 4-37** for bit descriptions.	16	X
0xYF FA64	S	CSBAR6	SCIM2E Chip-Select Base Address Register 6. See **Table 4-36** for bit descriptions.	16	X
0xYF FA66	S	CSOR6	SCIM2E Chip-Select Option Register 6. See **Table 4-37** for bit descriptions.	16	X
0xYF FA68	S	CSBAR7	SCIM2E Chip-Select Base Address Register 7. See **Table 4-36** for bit descriptions.	16	X
0xYF FA6A	S	CSOR7	SCIM2E Chip-Select Option Register 7. See **Table 4-37** for bit descriptions.	16	X
0xYF FA6C	S	CSBAR8	SCIM2E Chip-Select Base Address Register 8. See **Table 4-36** for bit descriptions.	16	X
0xYF FA6E	S	CSOR8	SCIM2E Chip-Select Option Register 8. See **Table 4-37** for bit descriptions.	16	X
0xYF FA70	S	CSBAR9	SCIM2E Chip-Select Base Address Register 9. See **Table 4-36** for bit descriptions.	16	X
0xYF FA72	S	CSOR9	SCIM2E Chip-Select Option Register 9. See **Table 4-37** for bit descriptions.	16	X
0xYF FA74	S	CSBAR10	SCIM2E Chip-Select Base Address Register 10. See **Table 4-36** for bit descriptions.	16	X
0xYF FA76	S	CSOR10	SCIM2E Chip-Select Option Register 10. See **Table 4-37** for bit descriptions.	16	X

Table A-9 SRAM Module (Static Random Access Memory)

Address	Access	Symbol	Register	Size	Reset
0xYF FB00	S	RAMMCR	RAM Module Configuration Register. See **Table 11-1** for bit descriptions.	16	X
0xYF FB02	S	RAMTST	RAM Module Configuration Register.	16	X
0xYF FB04	S	RAMBAH	RAM Module Base Address High Register. See **Table 11-3** for bit descriptions.	16	X
0xYF FB06	S	RAMBAL	RAM Module Base Address Low Register. See **Table 11-3** for bit descriptions.	16	X

Table A-10 QSMCM (Queued Serial Multi-Channel Module)

Address	Access	Symbol	Register	Size	Reset
0xYF FC00	S	QSMCMMCR	QSMCM Module Configuration Register. See **Table 6-3** for bit descriptions.	16	X
0xYF FC02	T	QTEST	QSMCM Test Register.	16	X
0xYF FC04	S	QILR	QSMCM Interrupt Level, Interrupt Vector Register See **Table 6-4** for bit descriptions.	8	X
0xYF FC05	S	QIVR	QSMCM Interrupt Level, Interrupt Vector Register See **Table 6-5** for bit descriptions.	8	X
0xYF FC06	S	—	Reserved	—	X
0xYF FC07	S	QSPI_IL	QSMCM Interrupt Level, QSPI interrupt level. See **Table 6-6** for bit descriptions.	8	X

Table A-10 QSMCM (Queued Serial Multi-Channel Module) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF FC08	S	SCC1R0	SCI1Control Register 0. See **Table 6-23** for bit descriptions.	16	X
0xYF FC0A	S	SCC1R1	SCI1Control Register 1. See **Table 6-24** for bit descriptions.	16	X
0xYF FC0C	S	SC1SR	SCI1 Status Register. See **Table 6-25** for bit descriptions.	16	X
0xYF FC0E	S	SC1DR	SCI1 Data Register. See **Table 6-26** for bit descriptions.	16	X
0xYF FC10 – 0xYF FC12	S	—	Reserved	—	X
0xYF FC14	S	PORTQS	QSMCM Port QS Data Register. See **6.6.1 Port QS Data Register (PORTQS)** for bit descriptions.	16	X
0xYF FC16	S	PQSPAR/ DDRQST	QSMCM Port QS PIn Assignment Register/ QSMCM Port QS Data Direction Register. See **Table 6-10** and **Table 6-11** for bit descriptions.	16	X
0xYF FC18	S	SPCR0	QSPI Control Register 0. See **Table 6-13** for bit descriptions.	16	X
0xYF FC1A	S	SPCR1	QSPI Control Register 1. See **Table 6-15** for bit descriptions.	16	X
0xYF FC1C	S	SPCR2	QSPI Control Register 2. See **Table 6-16** for bit descriptions.	16	X
0xYF FC1E	S	SPCR3	QSPI Control Register 3. See **Table 6-17** for bit descriptions.	16	X
0xYF FC1F	S	SPSR	QSPI Status Register 3. See **Table 6-18** for bit descriptions.	8	X
0xYF FC20	S	SCC2R0	SCI2 Control Register 0. See **Table 6-23** for bit descriptions.	16	X
0xYF FC22	S	SCC2R1	SCI2 Control Register 1. See **Table 6-24** for bit descriptions.	16	X
0xYF FC24	S	SC2SR	SCI2 Status Register. See **Table 6-24** for bit descriptions.	16	X
0xYF FC26	S	SC2DR	SCI2 Data Register. See **Table 6-26** for bit descriptions.	16	X
0xYF FC28	S	QSCI1CR	QSCI1 Control Register. See **Table 6-31** for bit descriptions.	16	X
0xYF FC2A	S	QSCI1SR	QSCI1 Status Register. See **Table 6-32** for bit descriptions.	16	X
0xYF FC2C – 0xYF FC4A	S	SCTQ	Transmit Queue Locations. See **6.8.7.5 Transmitter Operation** for bit descriptions.	16	X
0xYF FC4C – 0xYF FC6A	S	SCRQ	Receive Queue Locations. See **6.8.7.6 Receiver Operation** for bit descriptions.	16	X
0xYF FD40 – 0xYF FD7F	S/U	REC.RAM	Receive Data RAM See **6.7.2.1 Receive RAM** for bit descriptions.		
0xYF FD80 – 0xYF FDBF	S/U	TRAN.RAM	Transmit Data RAM See **6.7.2.2 Transmit RAM** for bit descriptions.		
0xYF FDC0 – 0xYF FDDF	S/U	COMD.RAM	Command RAM See **Table 6-19** for bit descriptions.		

Table A-11 TPU3 (Time Processor Unit)

Address	Access	Symbol	Register	Size	Reset
0xYF FE00	S	TPUMCR	TPU3 Module Configuration Register See **Table 8-6** for bit descriptions.	16	X
0xYF FE02	T	TCR	TPU3 Test Configuration Register	16	X
0xYF FE04	S	DSCR	Development Support Control Register See **Table 8-7** for bit descriptions.	16	X
0xYF FE06	S	DSSR	Development Support Status Register See **Table 8-8** for bit descriptions.	16	X
0xYF FE08	S	TICR	TPU3 Interrupt Configuration Register See **Table 8-9** for bit descriptions.	16	X
0xYF FE0A	S	CIER	Channel Interrupt Enable Register See **Table 8-10** for bit descriptions.	16	X
0xYF FE0C	S	CFSR0	Channel Function Selection Register 0 See **Table 8-11** for bit descriptions.	16	X
0xYF FE0E	S	CFSR1	Channel Function Selection Register 1 See **Table 8-11** for bit descriptions.	16	X
0xYF FE10	S	CFSR2	Channel Function Selection Register 2 See **Table 8-11** for bit descriptions.	16	X
0xYF FE12	S	CFSR3	Channel Function Selection Register 3 See **Table 8-11** for bit descriptions.	16	X
0xYF FE14	S	HSQR0	Host Sequence Register 0 See **Table 8-12** for bit descriptions.	16	X
0xYF FE16	S	HSQR1	Host Sequence Register 1 See **Table 8-12** for bit descriptions.	16	X
0xYF FE18	S	HSRR0	Host Service Request Register 0 See **Table 8-13** for bit descriptions.	16	X
0xYF FE1A	S	HSRR1	Host Service Request Register 1 See **Table 8-13** for bit descriptions.	16	X
0xYF FE1C	S	CPR0	Channel Priority Register 0 See **Table 8-14** for bit descriptions.	16	X
0xYF FE1E	S	CPR1	Channel Priority Register 1 See **Table 8-14** for bit descriptions.	16	X
0xYF FE20	S	CISR	Channel Interrupt Status Register See **Table 8-16** for bit descriptions.	16	X
0xYF FE22	T	LR	Link Register	16	X
0xYF FE24	T	SGLR	Service Grant Latch Register	16	X
0xYF FE26	T	DCNR	Decoded Channel Number Register	16	X
0xYF FE28	S	TPUMCR2	TPU Module Configuration Register 2 See **Table 8-17** for bit descriptions.	16	X
0xYF FE2A	S	TPUMCR3	TPU Module Configuration 3 See **Table 8-20** for bit descriptions.	16	X
0xYF FE2C	T	ISDR	Internal Scan Data Register	16	X
0xYF FE2E	T	ISCR	Internal Scan Control Register	16	X
0xYF FF00 – 0xYF FF0F	S	—	Channel 0 Parameter Registers	16	X
0xYF FF10 – 0xYF FF1F	S	—	Channel 1 Parameter Registers	16	X
0xYF FF20 – 0xYF FF2F	S	—	Channel 2 Parameter Registers	16	X

Table A-11 TPU3 (Time Processor Unit) (Continued)

Address	Access	Symbol	Register	Size	Reset
0xYF FF30 – 0xYF FF3F	S	—	Channel 3 Parameter Registers	16	X
0xYF FF40 – 0xYF FF4F	S	—	Channel 4 Parameter Registers	16	X
0xYF FF50 – 0xYF FF5F	S	—	Channel 5 Parameter Registers	16	X
0xYF FF60 – 0xYF FF6F	S	—	Channel 6 Parameter Registers	16	X
0xYF FF70 – 0xYF FF7F	S	—	Channel 7 Parameter Registers	16	X
0xYF FF80 – 0xYF FF8F	S	—	Channel 8 Parameter Registers	16	X
0xYF FF90 – 0xYF FF9F	S	—	Channel 9 Parameter Registers	16	X
0xYF FFA0 – 0xYF FFAF	S	—	Channel 10 Parameter Registers	16	X
0xYF FFB0 – 0xYF FFBF	S	—	Channel 11 Parameter Registers	16	X
0xYF FFC0 – 0xYF FFCF	S	—	Channel 12 Parameter Registers	16	X
0xYF FFD0 – 0xYF FFDF	S	—	Channel 13 Parameter Registers	16	X
0xYF FFE0 – 0xYF FFEF	S	—	Channel 14 Parameter Registers	16	X
0xYF FFF0 – 0xYF FFFF	S	—	Channel 15 Parameter Registers	16	X

APPENDIX B
REGISTER GENERAL INDEX

–B–

BAR (ROMBAH, ROMBAL)
    base address register (BAR — ROMBAH, ROMBAL) 12-5
BIUMCR (BIUSM module configuration register) 13-51
BIUSM
    module configuration register (BIUMCR) 13-51, 13-54
    time base register (BIUTBR) 13-52
BIUTBR (BIUSM time base register) 13-52

–C–

CANCTRL0 (control register 0) 7-26
CANCTRL1 (control register 1) 7-27
CANCTRL2 (control register 2) 7-29
CANICR (TouCAN interrupt configuration register) 7-25
CANMCR (TouCAN module configuration register) 7-23
CCW (conversion command word table) 5-48
CFSR0 (TPU3 channel function select register 0) 8-15
CFSR1 (TPU3 channel function select register 1) 8-15
CFSR2 (TPU3 channel function select register 2) 8-15
CFSR3 (TPU3 channel function select register 3) 8-15
CIER (TPU3 channel interrupt enable register) 8-14
CISR (TPU3 channel interrupt status register) 8-18
CMFI
    CMFI EEPROM configuration register (CMFIMCR) 10-9
    CMFI high voltage control register (CMFICTLx) 10-16
CMFIBAH (CMFI base address high register) 10-15
CMFIBAL (CMFI base address low register) 10-15
CMFIBS0 (CMFI bootstrap information words) 10-19
CMFIBS01 (CMFI bootstrap information words) 10-19
CMFIBS2 (CMFI bootstrap information words) 10-19
CMFIBS3 (CMFI bootstrap information words) 10-20
CMFICTL1 (CMFI high voltage control register 1) 10-16
CMFICTL2 (CMFI high voltage control register 2) 10-12, 10-17
CMFIMCR (CMFI EEPROM configuration register) 10-10
CPCR (CPSM control register) 13-54
CPR0 (TPU3 channel priority register 0) 8-17
CPR1 (TPU3 channel priority register 1) 8-17
CSBAR (SCIM2 chip-select base address register) 4-80
CSBARBT (SCIM2 chip-select base boot register) 4-79
CSOR (SCIM2 chip select option register) 4-81
CSORBT (SCIM2 chip-select option boot register) 4-80
CSPAR0 (SCIM2 chip-select pin assignment register 0) 4-76
CSPAR1 (SCIM2 chip-select pin assignment register 1) 4-76

–D–

DASM
    data register A (DASMA) 13-37
    data register B (DASMB) 13-38
    status/interrupt control register (DASMSIC) 13-35
DASMA (DASM data register A) 13-37, 13-38

DASMSIC (DASM status/interrupt control register) 13-35
DDRAB, DDRE (SCIM2 port E data direction registers) 4-89
DDRAB, PEPAR (SCIM2 port E pin assignment register) 4-89
DDRF (SCIM2 port F data direction register) 4-92
DDRG, DDRH (SCIM2 port G,H data direction registers) 4-94
DDRQA (PORTQA data direction register) 5-37
DDRQS (PORTQS data direction register) 6-12
DPTBAR (RAM array base address register) 9-5
DPTMCR (DPTRAM module configuration register) 9-3
DPTRAM
    module configuration register (DPTMCR) 9-3
    RAMbase address register (DPTBAR) 9-4
    test register 9-4
DSCR (TPU3 development support control register) 8-12
DSSR (TPU3 development support status register) 8-13

–E–

ESTAT (error and status register) 7-31

–F–

FCSM
    status/interrupt control register (FMSMSIC) 13-7
FCSMCNT (FCSM counter register) 13-9
FCSMSIC (FCSM status/interrupt control register) 13-7

–H–

HSQR0 (TPU3 host sequence register 0) 8-16
HSQR1 (TPU3 host sequence register 1) 8-16
HSSR0 (TPU3 host service request register 0) 8-16
HSSR1 (TPU3 host service request register 1) 8-16

–I–

IFLAG (interrupt flag register) 7-33
IMASK (interrupt mask register) 7-33

–L–

LJSRR (left justified, signed result register) 5-51
LJURR (left justified, unsigned result register) 5-51

–M–

MCSM
    counter register (MCSMCNT) 13-14
    status/interrupt control register (MCSMSIC) 13-12
MCSMCNT (MCSM counter register) 13-14
MCSMML (MCSM modulus latch register) 13-14
MCSMSIC (MCSM status/interrupt control register) 13-12
MISCNT (MISC counter) 9-6
MISRH (multiple input signature register high) 9-5
MISRL (multiple input signature register low) 9-5

–P–

PFIVR (SCIM2 port F edge-detect interrupt vector register) 4-93
PFLVR (SCIM2 port F edge-detect interrupt level register) 4-93
PFPAR (SCIM2 port F pin assignment register) 4-92
PICR (SCIM2 periodic interrupt control register) 4-31
PITR (SCIM2 periodic interrupt timer register) 4-31

PORTA-B (SCIM2 port A and B data registers) 4-88
PORTC (SCIM2 PORTC data register) 4-78
PORTE (SCIM2 port E data registers) 4-89
PORTF (SCIM2 port F data registers) 4-91
PORTFE (SCIM2 port F edge-detect flag register) 4-93
PORTG, PORTH (SCIM2 port G,H data registers) 4-94
PORTQA (port QA data register) 5-36
PORTQB (port QB data register) 5-36
PORTQS (port QS data register) 6-10
PQSPAR (PORTQS pin assignment register) 6-11
PRESDIV (prescaler divide register) 7-28
PWM
    counter register (PWMC) 13-49
    period register (PWMA) 13-48
    pulse width register (PWMB) 13-49
PWMA (PWM period register) 13-49
PWMB (PWM pulse width register) 13-49
PWMC (PWM counter register) 13-50
PWMSIC (PWSM status/interrupt control register) 13-45

–Q–

QACR0 (QADC64 control register 0) 5-37
QACR1 (QADC64 control register 1) 5-38
QACR2 (QADC64 control register 2) 5-40
QADC64
    control register 0 (QACR0) 5-37
    control register 1 (QACR1) 5-38
    control register 2 (QACR2) 5-40
    interrupt register (QADC64INT) 5-35
    port A/B data register (PORTQA/B) 5-36
    PORTQA data direction register (DDRQA) 5-37
    status register 0 (QASR0) 5-42
    status register 1 (QASR1) 5-44
    successive approximation register (SAR) 5-14
QADC64INT (QADC64 interrupt register) 5-36
QADC64MCR (QADC64 module configuration register) 5-35
QASR0 (QADC64 status register 0) 5-43, 5-44
QILR (QSMCM interrupt level register) 6-7
QIVR (QSMCM interrupt vector register) 6-8
QSCI1CR (QSCI1 control register) 6-57
QSCI1SR (QSCI1 status register) 6-58
QSMCM
    configuration register (QMCR) 6-6
    interrupt level registers (QILR, QIVR) 6-7
    port QS data register (PORTQS) 6-10
    PORTQS data direction register (DDRQS) 6-12
    PORTQS pin assignment register (PQSPAR) 6-10
    QSCI1 control register (QSCI1CR) 6-57
    QSCI1 status register (QSCI1SR) 6-58
    QSPI command RAM (CRx) 6-22
    QSPI control register 0 (SPCR0) 6-16
    QSPI control register 1 (SPCR1) 6-18
    QSPI control register 2 (SPCR2) 6-18
    QSPI control register 3 (SPCR3) 6-19
    QSPI registers 6-15
    QSPI status register (SPSR) 6-20
    queued SCI1 status and control registers 6-57
    SCI control register 0 (SCCxR0) 6-45
    SCI control register 1 (SCCxR1) 6-45
    SCI data register (SCxDR) 6-49

SCI registers 6-44
SCI status register (SCxSR) 6-47
test register (QTEST) 6-7
QSMCMMCR (QSMCM module configuration register) 6-7
QSPI_IL (QSMCM queued SPI interrupt level register) 6-8

–R–

RAMMCR (SRAM module configuration register) 11-4
RAMMCR RAMBAH, RAMBAL (SRAM array base address register) 11-5
RJURR (right justified, unsigned result register) 5-51
ROM
module configuration register (ROMMCR) 12-3
ROMBAH (ROM base address high register) 12-5
ROMBAL (ROM base address low register) 12-5
ROMBS0–ROMBS3
bootstrap information words (ROMBS0–ROMBS3) 12-7
ROMBS0–ROMBS3 (ROM bootstrap information words) 12-7, 12-8
ROMMCR (ROM module configuration registers) 12-4
RSR
reset status register (RSR) 4-56
RSR (SCIM2 reset status register) 4-56
RXECTR (receive error counter) 7-34
RXGMSKHI (receive global mask register high) 7-30

–S–

SASM
data register A (SDATA) 13-22
data register B (SDATB) 13-23
status/interrupt control register A (SICA) 13-20
status/interrupt control register B (SICB) 13-22
SCCxR0 (QSMCM SCI control register 0) 6-45
SCCxR1 (QSMCM SCI control register 1) 6-46
SCDR (QSMCM SCI data register) 6-49
SCIM
clock synthesizer control register (SYNCR) 4-14
SCIM2
chip-select base address registers (CSBARBT) 4-78
module configuration register (SCIMMCR) 4-2
periodic interrupt control register (PICR) 4-30
periodic interrupt timer register (PITR) 4-31
port C data register (PORTC) 4-78
port E data direction registers (DDRAB, DDRE) 4-89
port E pin assignment register (PEPAR) 4-89
port F data direction register (DDRF) 4-92
port F edge-detect flag register (PORTFE) 4-93
port F edge-detect interrupt level register (PFLVR) 4-93
port F edge-detect interrupt vector register (PFIVR) 4-93
port F pin assignment register (PFPAR) 4-92
port G,H data direction registers ((DDRG, DDRH) 4-94
port G,H data registers (PORTG, PORTH) 4-94
software watchdog servicel register (SWSR) 4-28
system protection control register (SYPCR) 4-24
SCIMMCR (SCIM2 module configuration register) 4-2
SCxSR (QSMCM SCIx status register) 6-47
SDATA (SASM data register A) 13-22
SDATB (SASM data register B) 13-23
SICA (SASM status/interrupt control register A) 13-20
SICB (SASM status/interrupt control register B) 13-23
SIGHI

ROM signature register (SIGHI) 12-6
SIGHI (ROM signature high register) 12-6
SIGLO
ROM signature register (SIGLO) 12-6
SIGLO (ROM signature low register) 12-6
SPCR0 (QSPI control register 0) 6-16
SPCR1 (QSPI control register 1) 6-18
SPCR2 (QSPI control register 2) 6-19
SPCR3 (QSPI control register 3) 6-19
SPSR (QSPI status register) 6-20
SWSR (SWSR software watchdog service register) 4-28
SYNCR (SYNCR clock synthesizer control register, extenal clock mode) 4-15
SYNCR (SYNCR clock synthesizer control register, fast reference model) 4-14
SYNCR (SYNCR clock synthesizer control register, slow reference model) 4-14
SYPCR (SCIM2 system protection control register) 4-24

–T–

TICR (TPU3 interrupt configuration register) 8-14
TIMER (free running timer register) 7-29
TouCAN
control register 0 (CANCTRL0) 7-26
control register 1 (CANCTRL1) 7-27
control register 2 (CANCTRL2) 7-29
error and status register (ESTAT) 7-31
interrupt configuration register (CANICR) 7-25
interrupt flag register (IFLAG) 7-33
interrupt mask register (IMASK) 7-33
module configuration register (CANMCR) 7-23
prescaler divide register (PRESDIV) 7-28
receive buffer 14 mask registers 7-30
receive buffer 15 mask registers 7-31
receive global mask registers (RXGMSKHI) 7-30
receive mask registers 7-7
TPU3
channel function select registers (CFSRx) 8-15
channel interrupt enable register (CIER) 8-14
channel interrupt status register (CISR) 8-18
channel priority registers (CPRx) 8-17
decoded channel number register (DCNR) 8-18
development support control register (DSCR) 8-12
development support status register (DSSR) 8-13
host sequence registers (HSQRx) 8-15
host service request registers (HSSRx) 8-16
interrupt configuration register (TICR) 8-14
link register (LR) 8-18
module configuration register (TPUMCR) 8-10
module configuration register 2 (TPUMCR2) 8-18
module configuration register 3 (TPUMCR3) 8-20
service grant latch register (SGLR) 8-18
test configuration register (TCR) 8-12
test registers (ISDR, ISCR) 8-20
TPUMCR (TPU3 module configuration register) 8-10
TPUMCR2 (TPU3 module configuration register 2) 8-18
TPUMCR3 (TPU3 module configuration register 3) 8-20

APPENDIX C
REGISTER DIAGRAM INDEX

–B–

BIUMCR (BIUSM module configuration register) 13-51
BIUTBR (BIUSM time base register 13-52

–C–

CANCTRL0 (control register 0) 7-26
CANCTRL1 (control register 1) 7-27
CANCTRL2 (control register 2) 7-29
CANICR (TouCAN interrupt configuration register) 7-25
CANMCR (TouCAN module configuration register) 7-23
CCW (conversion command word table) 5-48
CFSR0 (TPU3 channel function select register 0) 8-15
CFSR1 (TPU3 channel function select register 1) 8-15
CFSR2 (TPU3 channel function select register 2) 8-15
CFSR3 (TPU3 channel function select register 3) 8-15
CIER (TPU3 channel interrupt enable register) 8-14
CISR (TPU3 channel interrupt status register) 8-18
CMFIBAH (CMFI base address high register) 10-15
CMFIBAL (CMFI base address low register) 10-15
CMFIBS0 (CMFI bootstrap information words) 10-19
CMFIBS1 (CMFI bootstrap information words) 10-19
CMFIBS2 (CMFI bootstrap information words) 10-19
CMFIBS3 (CMFI bootstrap information words) 10-20
CMFICTL1 (CMFI high voltage control register 1) 10-16
CMFICTL2 (CMFI high voltage control register 2) 10-12, 10-17
CMFIMCR (CMFI EEPROM configuration register) 10-10
CPCR (CPSM control register) 13-54
CPR0 (TPU3 channel priority register 0) 8-17
CPR1 (TPU3 channel priority register 1) 8-17
CSBAR (SCIM2 chip-select base address register) 4-80
CSBARBT (SCIM2 chip-select base boot register) 4-79
CSOR (SCIM2 chip select option register) 4-81
CSORBT (SCIM2 chip-select option boot register) 4-80
CSPAR0 (SCIM2 chip-select pin assignment register 0) 4-76
CSPAR1 (SCIM2 chip-select pin assignment register 1) 4-76

–D–

DASMA (DASM data register A) 13-37, 13-38
DASMSIC (DASM status/interrupt control register) 13-35
DDRAB, DDRE (SCIM2 port E data direction registers) 4-89
DDRF (SCIM2 port F data direction register) 4-92
DDRG, DDRH (SCIM2 port G,H data direction registers) 4-94
DDRQA (PORTQA data direction register) 5-37
DDRQS (PORTQS data direction register) 6-12
DPTBAR (DPT RAM array base address register) 9-5
DPTMCR (DPTRAM module configuration register) 9-3
DSCR (TPU3 development support control register) 8-12
DSSR (TPU3 development support status register) 8-13

–E–

ESTAT (error and status register) 7-31

–F–

FCSMCNT (FCSM counter register) 13-9
FCSMSIC (FCSM status/interrupt control register) 13-7

–H–

HSQR0 (TPU3 host sequence register 0) 8-16
HSQR1 (TPU3 host sequence register 1) 8-16
HSSR0 (TPU3 host service request register 0) 8-16
HSSR1 (TPU3 host service request register 1) 8-16

–I–

IFLAG (interrupt flag register) 7-33
IMASK (interrupt mask register) 7-33

–L–

LJSRR (left justified, signed result register) 5-51
LJURR (left justified, unsigned result register) 5-51

–M–

MCSMCNT (MCSM counter register) 13-14
MCSMML (MCSM modulus latchl register) 13-14
MCSMSIC (MCSM status/interrupt control register) 13-12
MISCNT (MISC counter) 9-6
MISRH (multiple input signature register high) 9-5
MISRL (multiple input signature register low) 9-5

–P–

PEPAR (SCIM2 port E pin assignment register) 4-89
PFIVR (SCIM2 port F edge-detect interrupt vector register) 4-93
PFLVR (SCIM2 port F edge-detect interrupt level register) 4-93
PFPAR (SCIM2 port F pin assignment register) 4-92
PICR (SCIM2 periodic interrupt control register) 4-31
PITR (SCIM2 periodic interrupt timer register) 4-31
PORTA-B (SCIM2 port A and B data registers) 4-88
PORTC (SCIM2 PORTC data register) 4-78
PORTE (SCIM2 port E data registers) 4-89
PORTF (SCIM2 port F data registers) 4-91
PORTFE (SCIM2 port F edge-detect flag register) 4-93
PORTG, PORTH (SCIM2 port G,H data registers) 4-94
PORTQA (port QA data register) 5-36
PORTQB (port QB data register) 5-36
PORTQS (port QS data register) 6-10
PQSPAR (PORTQS pin assignment register) 6-11
PRESDIV (prescaler divide register) 7-28
PWMA (PWM period register 13-49
PWMB (PWM pulse width register 13-49
PWMC (PWM counter register 13-50
PWMSIC (PWM status/interrupt control register) 13-45

–Q–

QACR0 (QADC64 control register 0) 5-37
QACR1 (QADC64 control register 1) 5-38
QACR2 (QADC64 control register 2) 5-40
QADC64INT (QADC64 interrupt register) 5-36
QADC64MCR (QADC64 module configuration register) 5-35
QASR0 (QADC64 status register 0) 5-43, 5-44
QILR (QSMCM interrupt level register) 6-7
QIVR (QSMCM interrupt vector register) 6-8

QSCI1CR (QSCI1 control register) 6-57
QSCI1SR (QSCI1 status register) 6-58
QSMCMMCR (QSMCM module configuration register) 6-7
QSPI_IL (QSMCM queued SPI interrupt level register) 6-8

–R–

RAMBAH, RAMBAL (SRAM array base address register) 11-5
RAMMCR (SRAM module configuration register) 11-4
RJURR (right justified, unsigned result register) 5-51
ROMBAH {ROM base address high register) 12-5
ROMBAL (ROM base address low register) 12-5
ROMBS0–ROMBS3 (ROM bootstrap information words) 12-7, 12-8
ROMMCR (ROM module configuration registers) 12-4
RSR (SCIM2 reset status register) 4-56
RXECTR (receive error counter) 7-34
RXGMSKHI (receive global mask register high) 7-30

–S–

SCCxR0 (QSMCM SCI control register 0) 6-45
SCCxR1 (QSMCM SCI control register 1) 6-46
SCDR (QSMCM SCI data register) 6-49
SCIMMCR (SCIM2 module configuration register) 4-2
SCxSR (QSMCM SCIx status register) 6-47
SDATA (SASM data register A) 13-22
SDATB (SASM data register B) 13-23
SICA (SASM status/interrupt control register A) 13-20
SICB (SASM status/interrupt control register B) 13-23
SIGHI (ROM signature high register) 12-6
SIGLO (ROM signature low register) 12-6
SPCR0 (QSPI control register 0) 6-16
SPCR1 (QSPI control register 1) 6-18
SPCR2 (QSPI control register 2) 6-19
SPCR3 (QSPI control register 3) 6-19
SPSR (QSPI status register) 6-20
SWSR (SCIM2 software watchdog service register) 4-28
SYNCR (SYNCR clock synthesizer control register, external clock mode) 4-15
SYNCR (SYNCR clock synthesizer control register, fast reference model) 4-14
SYNCR (SYNCR clock synthesizer control register, slow reference model) 4-14
SYPCR (SCIM2 system protection control register) 4-24

–T–

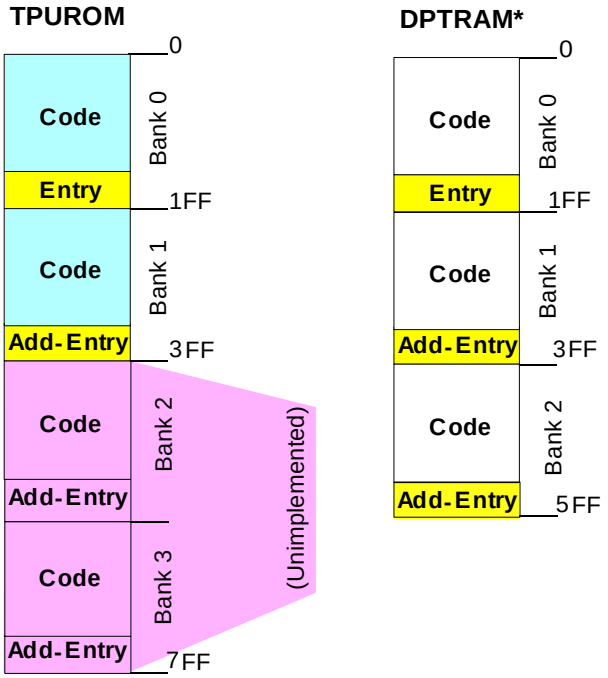
TICR (TPU3 interrupt configuration register) 8-14
TIMER (free running timer register) 7-29
TPUMCR (TPU3 module configuration register) 8-10
TPUMCR2 (TPU3 module configuration register 2) 8-18
TPUMCR3 (TPU3 module configuration register 3) 8-20

APPENDIX D
TPU ROM FUNCTIONS

The following pages provide brief descriptions of the pre-programmed functions in the TPU3. For detailed descriptions, refer to the programming note for the individual function. Motorola Programming Note TPUPN00/D, ***Using the TPU Function Library and TPU Emulation Mode***, provides a list of available programming notes.

D.1 Overview

The TPU3 contains 4 Kbytes of microcode ROM. This appendix defines the functions that are in the standard ROM on the MC68F375. The TPU3 can have up to 8 Kbytes of memory and a maximum of four entry tables (see **Figure D-1**).



*The DPTRAM is located at 0x30 2000 until it is switched to emulation mode. In emulation mode, the DPTRAM is accessible by the TPUs only.

Figure D-1 TPU3 Memory Map

The TPU3 can address up to 8 Kbytes of memory at any one time. It has 4 Kbytes of internal ROM, located in banks zero and one, and 6 Kbytes of dual ported SRAM (DPTRAM), located in banks zero, one and two. As only one type of memory can be used at a time, the TPU3 must either use the internal ROM or the SRAM. Functions from both memory types cannot be used in conjunction.

A new feature of the TPU3 microcode ROM is the existence of two entry tables in the 4 Kbytes of internal ROM. Each entry table has a set of sixteen functions and the user defines which of the two tables the TPU3 will be able to access. Only one table can be used at a time and functions from the two entry tables cannot be mixed. The default entry table is located in bank zero. This table is identical to the standard microcode ROM in the TPU2, so any CPU code written for the TPU2 will work unchanged on the TPU3. The functions in the default entry table in bank zero are listed in Table 1.

Table D-1 Bank 0 Functions

Function Number	Function Nickname	Function Name
$F	PTA	Programmable Time Accumulator
$E	QOM	Queued Output Match
$D	TSM	Table Stepper Motor
$C	FQM	Frequency Measurement
$B	UART	Universal Asynchronous Receiver/Transmitter
$A	NITC	New Input Capture/Input Transition Counter
9	COMM	Multiphase Motor Commutation
8	HALLD	Hall Effect Decode
7	MCPWM	Multi-Channel Pulse Width Modulation
6	FQD	Fast Quadrature Decode
5	PPWA	Period/Pulse Width Accumulator
4	OC	Output Compare
3	PWM	Pulse Width Modulation
2	DIO	Discrete Input/Output
1	SPWM	Synchronized Pulse Width Modulation
0	SIOP	Serial Input/output Port

The CPU selects which entry table to use by setting the ETBANK field in the TPUMCR2 register. This register is write once after reset. Although one entry table is specified at start-up, it is possible, in some cases, to use functions from both tables without resetting the microcontroller. A customer may, for example, wish to use the ID function from bank one to verify the TPU microcode version but then use the MCPWM function from bank zero. As a customer will typically only run the ID function during system configuration, and not again after that, the bank one entry table can be changed to the bank zero entry table using the soft reset feature of the TPU3. The procedure should be:

1. Set ETBANK field in TPUMCR2 to %01 to select the entry table in bank one
2. Run the ID function
3. Stop the TPU3 by setting the STOP bit in the TPUMCR to one.
4. Reset the TPU3 by setting the SOFTRST bit in the TPUMCR2 register
5. Wait at least nine clocks
6. Clear the SOFTRST bit in the TPUMCR2 register

The TPU3 stays in reset until the CPU clears the SOFTRST bit. After the SOFTRST bit has been cleared the TPU3 will be reset and the entry table in bank zero will be

selected by default. To select the bank zero entry table write %00 to the ETBANK field in TPUMCR 2. It is good practice always to initialize any write once register to ensure an incorrect value is not accidentally written.

The descriptions below document the functions listed in **Table D-1** (bank0) of the TPU3 ROM module.

D.2 Programmable Time Accumulator (PTA)

PTA accumulates a 32-bit sum of the total high time, low time, or period of an input signal over a programmable number of periods or pulses. The period accumulation can start on a rising or falling edge. After the specified number of periods or pulses, the PTA generates an interrupt request.

From one to 255 period measurements can be accumulated before the TPU interrupts the CPU, providing instantaneous or average frequency measurement capability.

Figure D-2 shows all of the host interface areas for the PTA function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	PTA FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE	00 — HIGH TIME ACCUMULATE 01 — LOW TIME ACCUMULATE 10 — PERIOD ACCUMULATE, RISING 11 — PERIOD ACCUMULATE, FALLING	$###E14–$###E16
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — NOT USED 10 — NOT USED 11 — INITIALIZE	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
$###FW0		CHANNEL_CONTROL
$###FW2	MAX_COUNT	PERIOD_COUNT
$###FW4	LAST_TIME	
$###FW6	ACCUM	
$###FW8	HW	
$###FWA	LW	
$###FWC		
$###FWE		

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = CHANNEL NUMBER

TPU PTA CHR

Figure D-2 PTA Parameters

D.3 Queued Output Match TPU Function (QOM)

QOM can generate single or multiple output match events from a table of offsets in parameter RAM. Loop modes allow complex pulse trains to be generated once, a specified number of times, or continuously. The function can be triggered by a link from another TPU channel. In addition, the reference time for the sequence of matches can be obtained from another channel. QOM can generate pulse-width modulated waveforms, including waveforms with high times of 0% or 100%. QOM also allows a TPU channel to be used as a discrete output pin.

Figure D-3 shows all of the host interface areas for the QOM function. The bit encodings shown in **Table D-2** describe the corresponding fields in parameter RAM.

Table D-2 QOM Bit Encoding

A	Timebase Selection
0	Use TCR1 as Timebase
1	Use TCR2 as Timebase

	Edge Selection
0	Falling Edge at Match
1	Rising Edge at Match

B:C	Reference for First Match
00	Immediate TCR Value
01	Last Event Time
10	Value Pointed to by REF_ADDR
11	Last Event Time

CONTROL BITS

Bits	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	QOM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE	00 — SINGLE-SHOT MODE 01 — LOOP MODE 10 — CONTINUOUS MODE 11 — CONTINUOUS MODE	$###E14–$###E16
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE, NO PIN CHANGE 10 — INITIALIZE, PIN LOW 11 — INITIALIZE, PIN HIGH	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1	0
$###FW0	REF_ADDR	LAST_OFF_ADDR	A
$###FW2	LOOP_CNT	(LAST_MATCH_TIM) OFF_PTR	C
$###FW4	OFFSET_1		;
$###FW6	OFFSET_2		;
$###FW8	OFFSET_3		;
$###FWA	OFFSET_4		;
$###FWC	OFFSET_5*		;
$###FWE	OFFSET_6*		;
$###F(W + 1)0	OFFSET_7*		;
$###F(W + 1)2	OFFSET_8*		;
⋮	⋮		⋮
$###F(W + 1)14	OFFSET_14*		;

* NOT AVAILABLE ON ALL CHANNELS

= WRITTEN BY CPU
= WRITTEN BY CPU AND TPU
= WRITTEN BY TPU
= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU QOM CHRT

Figure D-3 QOM Parameters

D.4 Table Stepper Motor (TSM)

The TSM function provides for acceleration and deceleration control of a stepper motor with a programmable number of step rates up to 58. TSM uses a table in parameter RAM, rather than an algorithm, to define the stepper motor acceleration profile, allowing the user to fully define the profile. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table. The CPU need only write a desired position, and the TPU accelerates, slews, and decelerates the motor to the required position. Full and half step support is provided for two-phase motors. In addition, a slew rate parameter allows fine control of the terminal running speed of the motor independent of the acceleration table.

Figure D-4 and **Figure D-5** show all of the host interface areas for the TSM function when operating in master and slave mode, respectively.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	TSM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE BITS	x0 — LOCAL MODE ACCELERATION TABLE x1 — SPLIT MODE ACCELERATION TABLE 0x — ROTATE PIN_SEQUENCE ONCE BETWEEN STEPS 1x — ROTATE PIN_SEQUENCE TWICE BETWEEN STEP	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE, PIN LOW 10 — INITIALIZE, PIN HIGH 11 — MOVE REQUEST (MASTER ONLY)	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1	0
$###FW0	DESIRED_POSITION		
$###FW2	CURRENT_POSITION		
$###FW4	TABLE_SIZE	TABLE_INDEX	
$###FW6	SLEW_PERIOD		S
$###FW8	START_PERIOD		A
$###FWA	PIN_SEQUENCE		
$###FWC			
$###FWE			

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU TSM MAS CHRT

Figure D-4 TSM Parameters — Master Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	TSM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE BITS	x0 — LOCAL MODE ACCELERATION TABLE x1 — SPLIT MODE ACCELERATION TABLE 0x — ROTATE PIN_SEQUENCE ONCE BETWEEN STEPS 1x — ROTATE PIN_SEQUENCE TWICE BETWEEN STEP	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE, PIN LOW 10 — INITIALIZE, PIN HIGH 11 — MOVE REQUEST (MASTER ONLY)	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
$###F(W + 1)0	ACCEL_RATIO_2	ACCEL_RATIO_1
$###F(W + 1)2	ACCEL_RATIO_4	ACCEL_RATIO_3
$###F(W + 1)4	ACCEL_RATIO_6	ACCEL_RATIO_5
$###F(W + 1)6	ACCEL_RATIO_8	ACCEL_RATIO_7
$###F(W + 1)8	ACCEL_RATIO_10	ACCEL_RATIO_9
$###F(W + 1)A	ACCEL_RATIO_12	ACCEL_RATIO_11
$###F(W + 1)C *	ACCEL_RATIO_14 *	ACCEL_RATIO_13 *
⋮	⋮	⋮
$###F(W + 3)A *	ACCEL_RATIO_36 *	ACCEL_RATIO_35 *

* OPTIONAL ADDITIONAL PARAMETERS NOT AVAILABLE IN ALL CASES. REFER TO MOTOROLA PROGRAMMING NOTE TPUPN04 ***Table Stepper Motor TPU Function (TSM)*** FOR DETAILS.

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = MASTER CHANNEL NUMBER

TPU TSM SLV CHRT

Figure D-5 TSM Parameters — Slave Mode

D.5 Frequency Measurement (FQM)

FQM counts the number of input pulses to a TPU channel during a user-defined window period. The function has single shot and continuous modes. No pulses are lost between sample windows in continuous mode. The user selects whether to detect pulses on the rising or falling edge. This function is intended for high speed measurement; measurement of slow pulses with noise rejection can be made with PTA.

Figure D-6 shows all of the host interface areas for the FQM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
3 2 1 0	CHANNEL FUNCTION SELECT	xxxx — FQD FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE BITS	00 — BEGIN WITH FALLING EDGE, SINGLE-SHOT MODE 01 — BEGIN WITH FALLING EDGE, CONTINUOUS MODE 10 — BEGIN WITH RISING EDGE, SINGLE-SHOT MODE 11 — BEGIN WITH RISING EDGE, CONTINUOUS MODE	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE (RESET CONDITION) 01 — NOT USED 10 — INITIALIZE 11 — NOT USED	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
$###FW0	
$###FW2	
$###FW4	CHANNEL_CONTROL
$###FW6	WINDOW_SIZE
$###FW8	PULSE_COUNT
$###FWA	IN_WINDOW_ACCUMULATOR

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU FQM CHRT

Figure D-6 FQM Parameters

D.6 Universal Asynchronous Receiver/Transmitter (UART)

The UART function uses one or two TPU channels to provide asynchronous communications. Data word length is programmable from one to 14 bits. The function supports detection or generation of even, odd, and no parity. Baud rate is freely programmable and can be higher than 100 Kbaud. Eight bi-directional UART channels running in excess of 9600 baud could be implemented on the TPU.

Figure D-7 and **Figure D-4** show all of the host interface areas for the UART function in transmitting and receiving modes, respectively.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	UART FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE BITS	00 — NO PARITY 01 — NO PARITY 10 — EVEN PARITY 11 — ODD PARITY	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NOT USED 01 — NOT USED 10 — TRANSMIT 11 — RECEIVE	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM (TRANSMITTER)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$###FW0	PARITY_TEMP															
$###FW2	MATCH_RATE															
$###FW4	TDRE		TRANSMIT_DATA_REG													
$###FW6	DATA_SIZE															
$###FW8	ACTUAL_BIT_COUNT															
$###FWA	SHIFT_REGISTER															

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU UART TRANS

Figure D-7 UART Transmitter Parameters

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	UART FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE BITS	00 — NO PARITY 01 — NO PARITY 10 — EVEN PARITY 11 — ODD PARITY	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NOT USED 01 — NOT USED 10 — TRANSMIT 11 — RECEIVE	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM (RECEIVER)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$###FW0	PARITY_TEMP															
$###FW2	MATCH_RATE															
$###FW4	PE	FE	RECEIVE_DATA_REG													
$###FW6	DATA_SIZE															
$###FW8	ACTUAL_BIT_COUNT															
$###FWA	SHIFT_REGISTER															

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU UART REC CHRT

Figure D-8 UART Receiver Parameters

D.7 New Input Capture/Transition Counter (NITC)

Any channel of the TPU can capture the value of a specified TCR or any specified location in parameter RAM upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the bus master. The times of the most recent two transitions are maintained in parameter RAM. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, ceasing channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to other channels.

Figure D-9 shows all of the host interface areas for the NITC function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	NITC FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE	00 — SINGLE-SHOT MODE, NO LINKS 01 — CONTINUOUS MODE, NO LINKS 10 — SINGLE-SHOT MODE, LINKS 11 — CONTINUOUS MODE, LINKS	$###E14–$###E16
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE TCR MODE 10 — INITIALIZE PARAMETER MODE 11 — NOT USED	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$###FW0								CHANNEL_CONTROL								
$###FW2	START_LINK_CHANNEL				LINK_CHANNEL_COUNT				PARAM_ADDR							0
$###FW4	MAX_COUNT															
$###FW6	TRANS_COUNT															
$###FW8	FINAL_TRANS_TIME															
$###FWA	LAST_TRANS_TIME															

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU NITC CHRT

Figure D-9 NITC Parameters

D.8 Multiphase Motor Commutation (COMM)

The COMM function generates the phase commutation signals for a variety of brushless motors, including three-phase brushless direct current. It derives the commutation state directly from the position decoded in FQD, thus eliminating the need for hall effect sensors.

The state sequence is implemented as a user-configurable state machine, thus providing a flexible approach with other general applications. A CPU offset parameter is provided to allow all the switching angles to be advanced or retarded on the fly by the CPU. This feature is useful for torque maintenance at high speeds.

Figure D-10 and **Figure D-11** show all of the host interface areas for the COMM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
3 2 1 0	CHANNEL FUNCTION SELECT	xxxx — FQD FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE	00 — SENSORLESS MATCH UPDATE MODE 01 — SENSORLESS MATCH UPDATE MODE 10 — SENSORLESS LINK UPDATE MODE 11 — SENSORED MODE	$###E14–$###E16
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — NOT USED 10 — INITIALIZE OR FORCE STATE 11 — INITIALIZE OR FORCE IMMEDIATE STATE TEST	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$###FW0					START_LINK_CHANNEL				COUNTER_ADDR							
$###FW2	NO_OF_STATES								STATE_NO							
$###FW4	OFFSET															
$###FW6	UPDATE_PERIOD															
$###FW8	UPPER															
$###FWA	LOWER															
$###FWC																
$###FWE																

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = MASTER COMM CHANNEL NUMBER

TPU COMM CHRT 1

Figure D-10 COMM Parameters, Part 1 of 2

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
$###F(W + 1)0	LENGTH	STATE 0 / PIN_CONFIG
$###F(W + 1)2	LENGTH	STATE 1 / PIN_CONFIG
$###F(W + 1)4	LENGTH	STATE 2 / PIN_CONFIG
$###F(W + 1)6	LENGTH	STATE 3 / PIN_CONFIG
$###F(W + 1)8	LENGTH	STATE 4 / PIN_CONFIG
$###F(W + 1)A	LENGTH	STATE 5 / PIN_CONFIG
$###F(W + 1)C	LENGTH	STATE 6 * / PIN_CONFIG
$###F(W + 1)E	LENGTH	STATE 7 * / PIN_CONFIG
$###F(W + 2)0	LENGTH	STATE 8 * / PIN_CONFIG
$###F(W + 2)2	LENGTH	STATE 9 * / PIN_CONFIG
⋮		⋮
$###F(W + 3)A	LENGTH	STATE 21 * / PIN_CONFIG

*NOT AVAILABLE ON ALL CHANNELS

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = MASTER COMM CHANNEL NUMBER

TPU COMM CHRT 2

Figure D-11 COMM Parameters, Part 2 of 2

D.9 Hall Effect Decode (HALLD)

The HALLD function decodes the sensor signals from a brushless motor, along with a direction input from the CPU, into a state number. The function supports two- or three-sensor decoding. The decoded state number is written into a COMM channel, which outputs the required commutation drive signals. In addition to brushless motor applications, the function can have more general applications, such as decoding “option” switches.

Figure D-12 shows all of the host interface areas for the HALLD function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
3 2 1 0	CHANNEL FUNCTION SELECT	xxxx — FQD FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE	00 — CHANNEL A 01 — CHANNEL B 10 — CHANNEL B 11 — CHANNEL C (3-CHANNEL MODE ONLY)	$###E14–$###E16
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — NOT USED 10 — INITIALIZE, 2-CHANNEL MODE 11 — INITIALIZE, 3-CHANNEL MODE	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT STATUS	x — NOT USED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
$###FW0		
$###FW2		
$###FW4		
$###FW6	DIRECTION (1)	
$###FW8		STATE_NO_ADDR (2)
$###FWA	PINSTATE	
$###FWC		
$###FWE		

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = CHANNEL NUMBER

NOTES:
1. CHANNEL A ONLY.
2. 1 CHANNEL ONLY (CHANNEL B IN 2-CHANNEL MODE, CHANNEL C IN 3-CHANNEL MODE).

TPU HALLD CHRT

Figure D-12 HALLD Parameters

D.10 Multichannel Pulse-Width Modulation (MCPWM)

MCPWM generates pulse-width modulated outputs with full 0% to 100% duty cycle range independent of other TPU activity. This capability requires two TPU channels plus an external gate for one PWM channel. (A simple one-channel PWM capability is supported by the QOM function.)

Multiple PWMs generated by MCPWM have two types of high time alignment: edge aligned and center aligned. Edge-aligned mode uses n + 1 TPU channels for n PWMs; center-aligned mode uses 2n + 1 channels. Center-aligned mode allows a user-defined “dead time” to be specified so that two PWMs can be used to drive an H-bridge without destructive current spikes. This feature is important for motor control applications.

Figure D-13 through **Figure D-18** show the host interface areas for the MCPWM function in each mode.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
3 2 1 0	CHANNEL FUNCTION SELECT	xxxx — FQD FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE	00 — EDGE ALIGNED MODE 01 — SLAVE A TYPE CENTER ALIGNED MODE 10 — SLAVE B TYPE CENTER ALIGNED MODE 11 — SLAVE C TYPE CENTER ALIGNED MODE	$###E14–$###E16
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE AS SLAVE (INVERTED) 10 — INITIALIZE AS SLAVE (NORMAL) 11 — INITIALIZE AS MASTER	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
$###FW0	PERIOD	
$###FW2	IRQ_RATE	PERIOD_COUNT
$###FW4	LAST_RISE_TIME	
$###FW6	LAST_FALL_TIME	
$###FW8	RISE_TIME_PTR	
$###FWA	FALL_TIME_PTR	
$###FWC		
$###FWE		

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU MCPWM MAS CHRT

Figure D-13 MCPWM Parameters — Master Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	MCPWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE	00 — EDGE ALIGNED MODE 01 — SLAVE A TYPE CENTER ALIGNED MODE 10 — SLAVE B TYPE CENTER ALIGNED MODE 11 — SLAVE C TYPE CENTER ALIGNED MODE	$###E14–$###E16
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE AS SLAVE (INVERTED) 10 — INITIALIZE AS SLAVE (NORMAL) 11 — INITIALIZE AS MASTER	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
$###FW0	PERIOD
$###FW2	HIGH_TIME
$###FW4	
$###FW6	HIGH_TIME_PTR
$###FW8	RISE_TIME_PTR
$###FWA	FALL_TIME_PTR
$###FWC	
$###FWE	

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU MCPWM S EA CHRT

Figure D-14 MCPWM Parameters — Slave Edge-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	MCPWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE	00 — EDGE ALIGNED MODE 01 — SLAVE A TYPE CENTER ALIGNED MODE 10 — SLAVE B TYPE CENTER ALIGNED MODE 11 — SLAVE C TYPE CENTER ALIGNED MODE	$###E14–$###E16
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE AS SLAVE (INVERTED) 10 — INITIALIZE AS SLAVE (NORMAL) 11 — INITIALIZE AS MASTER	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
$###FW0	PERIOD	
$###FW2	NXT_B_RISE_TIME	
$###FW4	NXT_B_FALL_TIME	
$###FW6	DEAD_TIME	HIGH_TIME_PTR
$###FW8	RISE_TIME_PTR	
$###FWA	FALL_TIME_PTR	
$###FWC		
$###FWE		

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU MCPWM SA NIC

Figure D-15 MCPWM Parameters — Slave Ch A Non-Inverted Center-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	MCPWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE	00 — EDGE ALIGNED MODE 01 — SLAVE A TYPE CENTER ALIGNED MODE 10 — SLAVE B TYPE CENTER ALIGNED MODE 11 — SLAVE C TYPE CENTER ALIGNED MODE	$###E14–$###E16
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE AS SLAVE (INVERTED) 10 — INITIALIZE AS SLAVE (NORMAL) 11 — INITIALIZE AS MASTER	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
$###FW0	HIGH_TIME
$###FW2	CURRENT_HIGH_TIME
$###FW4	TEMP_STORAGE
$###FW6	
$###FW8	B_FALL_TIME_PTR
$###FWA	B_RISE_TIME_PTR
$###FWC	
$###FWE	

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU MCPWM SB NIC

Figure D-16 MCPWM Parameters — Slave Ch B Non-Inverted Center-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	MCPWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE	00 — EDGE ALIGNED MODE 01 — SLAVE A TYPE CENTER ALIGNED MODE 10 — SLAVE B TYPE CENTER ALIGNED MODE 11 — SLAVE C TYPE CENTER ALIGNED MODE	$###E14–$###E16
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE AS SLAVE (INVERTED) 10 — INITIALIZE AS SLAVE (NORMAL) 11 — INITIALIZE AS MASTER	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
$###FW0	PERIOD	
$###FW2	NXT_B_FALL_TIME	
$###FW4	NXT_B_RISE_TIME	
$###FW6	DEAD_TIME	HIGH_TIME_PTR
$###FW8	FALL_TIME_PTR	
$###FWA	RISE_TIME_PTR	
$###FWC		
$###FWE		

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU MCPWM SA ICA

Figure D-17 MCPWM Parameters — Slave Ch A Inverted Center-Aligned Mode

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	MCPWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE	00 — EDGE ALIGNED MODE 01 — SLAVE A TYPE CENTER ALIGNED MODE 10 — SLAVE B TYPE CENTER ALIGNED MODE 11 — SLAVE C TYPE CENTER ALIGNED MODE	$###E14–$###E16
1 0	HOST SERVICE REQUEST	00 — NO HOST SERVICE (RESET CONDITION) 01 — INITIALIZE AS SLAVE (INVERTED) 10 — INITIALIZE AS SLAVE (NORMAL) 11 — INITIALIZE AS MASTER	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT ENABLE	0 — CHANNEL INTERRUPTS DISABLED 1 — CHANNEL INTERRUPTS ENABLED	$###E0A
0	CHANNEL INTERRUPT STATUS	0 — CHANNEL INTERRUPT NOT ASSERTED 1 — CHANNEL INTERRUPT ASSERTED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
$###FW0	HIGH_TIME
$###FW2	CURRENT_HIGH_TIME
$###FW4	TEMP_STORAGE
$###FW6	
$###FW8	B_FALL_TIME_PTR
$###FWA	B_RISE_TIME_PTR
$###FWC	
$###FWE	

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU MCPWM SB ICA CHRT

Figure D-18 MCPWM Parameters — Slave Ch B Inverted Center-Aligned Mode

D.11 Fast Quadrature Decode TPU Function (FQD)

FQD is a position feedback function for motor control. It decodes the two signals from a slotted encoder to provide the CPU with a 16-bit free-running position counter. FQD incorporates a “speed switch” which disables one of the channels at high speed, allowing faster signals to be decoded. A time stamp is provided on every counter update to allow position interpolation and better velocity determination at low speed or when low resolution encoders are used. The third index channel provided by some encoders is handled by the ITC function.

Figure D-19 and **Figure D-20** show the host interface areas for the FQD function for primary and secondary channels, respectively.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	CHANNEL INTERRUPT ENABLE	x — NOT USED	$###E0A
3 2 1 0	CHANNEL FUNCTION SELECT	xxxx — FQD FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE BITS	00 — PRIMARY CHANNEL (NORMAL MODE) 01 — SECONDARY CHANNEL (NORMAL MODE) 10 — PRIMARY CHANNEL (FAST MODE) 11 — SECONDARY CHANNEL (FAST MODE)	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE (RESET CONDITION) 01 — NOT USED 10 — READ TCR1 11 — INITIALIZE	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT STATUS	xx — NOT USED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
$###FW0	EDGE_TIME
$###FW2	POSITION_COUNT
$###FW4	TCR1_VALUE
$###FW6	CHAN_PINSTATE
$###FW8	CORR_PINSTATE_ADDR
$###FWA	EDGE_TIME_LSB_ADDR
$###FWC	
$###FWE	

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = CHANNEL NUMBER

TPU FQD PRI CH

Figure D-19 FQD Parameters — Primary Channel

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
0	CHANNEL INTERRUPT ENABLE	x — NOT USED	$###E0A
3 2 1 0	CHANNEL FUNCTION SELECT	xxxx — FQD FUNCTION NUMBER (ASSIGNED DURING MICROCODE ASSEMBLY)	$###E0C–$###E12
1 0	HOST SEQUENCE BITS	00 — PRIMARY CHANNEL (NORMAL MODE) 01 — SECONDARY CHANNEL (NORMAL MODE) 10 — PRIMARY CHANNEL (FAST MODE) 11 — SECONDARY CHANNEL (FAST MODE)	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE (RESET CONDITION) 01 — NOT USED 10 — READ TCR1 11 — INITIALIZE	$###E18–$###E1A
1 0	CHANNEL PRIORITY	00 — DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
0	CHANNEL INTERRUPT STATUS	xx — NOT USED	$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
$###F(W+1)0	
$###F(W+1)2	
$###F(W+1)4	TCR1_VALUE
$###F(W+1)6	CHAN_PINSTATE
$###F(W+1)8	CORR_PINSTATE_ADDR
$###F(W+1)A	EDGE_TIME_LSB_ADDR
$###F(W+1)C	
$###F(W+1)E	

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

TPU FQD SEC CHRT

Figure D-20 FQD Parameters — Secondary Channel

D.12 Period/Pulse-Width Accumulator (PPWA)

The period/pulse-width accumulator (PPWA) algorithm accumulates a 16-bit or 24-bit sum of either the period or the pulse width of an input signal over a programmable number of periods or pulses (from one to 255). After an accumulation period, the algorithm can generate a link to a sequential block of up to eight channels. The user specifies a starting channel of the block and number of channels within the block. Generation of links depends on the mode of operation.

Any channel can be used to measure an accumulated number of periods of an input signal. A maximum of 24 bits can be used for the accumulation parameter. From one to 255 period measurements can be made and summed with the previous measurement(s) before the TPU interrupts the CPU, allowing instantaneous or average frequency measurement, and the latest complete accumulation (over the programmed number of periods).

The pulse width (high-time portion) of an input signal can be measured (up to 24 bits) and added to a previous measurement over a programmable number of periods (one to 255). This provides an instantaneous or average pulse-width measurement capability, allowing the latest complete accumulation (over the specified number of periods) to always be available in a parameter.

By using the output compare function in conjunction with PPWA, an output signal can be generated that is proportional to a specified input signal. The ratio of the input and output frequency is programmable. One or more output signals with different frequencies, yet proportional and synchronized to a single input signal, can be generated on separate channels.

Figure D-21 shows the host interface areas and parameter RAM for the PPWA function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	PPWA FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	CHANNEL PRIORITY	00 — CHANNEL DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
1 0	HOST SEQUENCE BITS	00 — ACCUMULATE 24-BIT PERIODS, NO LINKS 01 — ACCUMULATE 16-BIT PERIODS, LINKS 10 — ACCUMULATE 24-BIT PULSE WIDTHS, NO LINKS 11 — ACCUMULATE 16-BIT PULSE WIDTHS, LINKS	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NOT USED 01 — NOT USED 10 — INITIALIZE 11 — NOT USED	$###E18–$###E1A
0	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	$###E0A
0	INTERRUPT STATUS		$###E20

PARAMETER RAM

	15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0
$###FW0	START_LINK_CHANNEL	LINK_CHANNEL_COUNT	CHANNEL_CONTROL
$###FW2	MAX_COUNT		PERIOD_COUNT
$###FW4	LAST_ACCUM		
$###FW6	ACCUM		
$###FW8	ACCUM_RATE		PPWA_UB
$###FWA	PPWA_LW		
$###FWC			
$###FWE			

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = CHANNEL NUMBER

NOTES:

1. THE TPU DOES NOT CHECK THE VALUE OF LINK_CHANNEL_COUNT. IF THIS PARAMETER IS **NOT** > 0 AND ≤ 8, RESULTS ARE UNPREDICTABLE.
2. MAX_COUNT MAY BE WRITTEN AT ANY TIME BY THE HOST CPU, BUT IF THE VALUE WRITTEN IS ≤ PERIOD_COUNT, A PERIOD OR PULSE-WIDTH ACCUMULATION IS TERMINATED. IF THIS HAPPENS, THE NUMBER OF PERIODS OVER WHICH THE ACCUMULATION IS DONE WILL NOT CORRESPOND TO MAX_COUNT.

1049A

Figure D-21 PPWA Parameters

D.13 Output Compare (OC)

The output compare (OC) function generates a rising edge, falling edge, or a toggle of the previous edge in one of three ways:

1. Immediately upon CPU initiation, thereby generating a pulse with a length equal to a programmable delay time
2. At a programmable delay time from a user-specified time
3. Continuously. Upon receiving a link from a channel, OC references, without CPU interaction, a specifiable period and calculates an offset:

$$\text{Offset} = \text{Period} \times \text{Ratio}$$

where RATIO is a parameter supplied by the user.

This algorithm generates a 50% duty-cycle continuous square wave with each high/low time equal to the calculated OFFSET. Due to offset calculation, there is an initial link time before continuous pulse generation begins.

Figure D-22 shows the host interface areas and parameter RAM for the OC function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	OC FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	CHANNEL PRIORITY	00 — CHANNEL DISABLED 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
1 0	HOST SEQUENCE BITS	0x — MATCHES AND PULSES SCHEDULED 1x — ONLY READ TCR1, TCR2	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE REQUEST 01 — HOST-INITIATED PULSE 10 — NOT USED 11 — INITIALIZE, CONTINUOUS PULSES	$###E18–$###E1A
0	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	$###E0A
0	INTERRUPT STATUS		$###E20

PARAMETER RAM

	15 14 13 12 11 10 9	8	7 6 5 4 3	2 1	0
$###FW0		CHANNEL_CONTROL			
$###FW2	OFFSET				
$###FW4	RATIO		REF_ADDR1		0
$###FW6	REF_ADDR2	0	REF_ADDR3		0
$###FW8	REF_TIME				
$###FWA	ACTUAL_MATCH_TIME				
$###FEC	TCR1				
$###FEE	TCR2				

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = CHANNEL NUMBER

1024A

Figure D-22 OC Parameters

D.14 Pulse-Width Modulation (PWM)

The TPU can generate a pulse-width modulation (PWM) waveform with any duty cycle from zero to 100% (within the resolution and latency capability of the TPU). To define the PWM, the CPU provides one parameter that indicates the period and another parameter that indicates the high time. Updates to one or both of these parameters can direct the waveform change to take effect immediately, or coherently beginning at the next low-to-high transition of the pin.

Figure D-23 shows the host interface areas and parameter RAM for the PWM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	PWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
1 0	HOST SEQUENCE BITS	xx — NOT USED	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NOT USED 01 — IMMEDIATE UPDATE OF PWM 10 — INITIALIZE 11 — NOT USED	$###E18–$###E1A
0	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	$###E0A
0	INTERRUPT STATUS		$###E20

PARAMETER RAM

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
$###FW0		CHANNEL_CONTROL
$###FW2	OLDRIS	
$###FW4	PWMHI (1, 3)	
$###FW6	PWMPER (2,3)	
$###FW8	PWMRIS	
$###FWA		
$###FWC		
$###FWE		

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = CHANNEL NUMBER

NOTES:
1. BEST-CASE MINIMUM FOR PWMHI IS 32 SYSTEM CLOCK CYCLES.
2. BEST-CASE MINIMUM FOR PWMPER IS 48 SYSTEM CLOCK CYCLES.
3. PWMHI AND PWMPER MUST BE ACCESSED COHERENTLY.

1027A

Figure D-23 PWM Parameters

D.15 Discrete Input/Output (DIO)

The DIO function allows a TPU channel to be used as a digital I/O pin.

When a pin is used as a discrete input, a parameter indicates the current input level and the previous 15 levels of a pin. Bit 15, the most significant bit of the parameter, indicates the most recent state. Bit 14 indicates the next most recent state, and so on. The programmer can choose one of the three following conditions to update the parameter:

1. when a transition occurs
2. when the CPU makes a request, or
3. when a rate specified in another parameter is matched

When a pin is used as a discrete output, it is set high or low only upon request by the CPU.

Figure D-24 shows the host interface areas for the DIO function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	DIO FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
1 0	HOST SEQUENCE BITS	00 — UPDATE ON TRANSITION 01 — UPDATE AT MATCH RATE 10 — UPDATE ON HSR 11 11 — NOT USED	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NOT USED 01 — DRIVE PIN HIGH 10 — DRIVE PIN LOW 11 — INITIALIZE	$###E18–$###E1A
0	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	$###E0A
0	INTERRUPT STATUS		$###E20

PARAMETER RAM

	15 14 13 12 11 10 9	8 7 6 5 4 3 2 1 0
$###FW0		CHANNEL_CONTROL
$###FW2	PIN_LEVEL	
$###FW4	MATCH_RATE	
$###FW6		
$###FW8		
$###FWA		
$###FWC		
$###FWE		

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = CHANNEL NUMBER

1017A

Figure D-24 DIO Parameters

D.16 Synchronized Pulse-Width Modulation (SPWM)

The SPWM function generates a pulse-width modulated waveform (PWM). The CPU can change the period or high time of the waveform at any time. Three different operating modes allow the function to maintain complex timing relationships between channels without CPU intervention.

The SPWM output waveform duty cycle excludes 0% and 100%. If a PWM does not need to maintain a time relationship to another PWM, the PWM function should be used instead.

Figure D-25 and **Figure D-26** show all of the host interface areas for the SPWM function.

CONTROL BITS

	NAME	OPTIONS	ADDRESSES
3 2 1 0	CHANNEL FUNCTION SELECT	SPWM FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)	$###E0C–$###E12
1 0	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY	$###E1C–$###E1E
1 0	HOST SEQUENCE BITS	00 — MODE 0 01 — MODE 1 10 — MODE 2 11 — NOT USED	$###E14–$###E16
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE REQUEST 01 — NOT USED 10 — INITIALIZE 11 — IMMEDIATE UPDATE (MODE 1)	$###E18–$###E1A
0	INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED	$###E0A
0	INTERRUPT STATUS		$###E20

PARAMETER RAM (MODE 0)

	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
$###FW0	LASTRISE	CHANNEL_CONTROL
$###FW2	NEXTRISE	
$###FW4	HIGH_TIME	
$###FW6	PERIOD	
$###FW8		REF_ADDR1
$###FWA	DELAY	
$###FWC		
$###FWE		

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = CHANNEL NUMBER

1030A-1

Figure D-25 SPWM Parameters, Part 1 of 2

PARAMETER RAM (MODE 1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$###FW0	LASTRISE							CHANNEL_CONTROL								
$###FW2	NEXTRISE															
$###FW4	HIGH_TIME															
$###FW6	DELAY															
$###FW8	REF_ADDR1								REF_ADDR2							
$###FWA	REF_VALUE															
$###FWC																
$###FWE																

PARAMETER RAM (MODE 2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$###FW0	LASTRISE							CHANNEL_CONTROL								
$###FW2	NEXTRISE															
$###FW4	HIGH_TIME															
$###FW6	PERIOD															
$###FW8	START_LINK_CHANNEL				LINK_CHANNEL_COUNT				REF_ADDR1							
$###FWA	DELAY															
$###FWC																
$###FWE																

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = CHANNEL NUMBER

1030A-2

Figure D-26 SPWM Parameters, Part 2 of 2

D.17 Serial Input/Output Port (SIOP)

The serial input/output port (SIOP) TPU function uses two or three TPU channels to form a uni- or bi-directional synchronous serial port that can be used to communicate with a wide variety of devices. Features such as baud rate and transfer size are user programmable. The function can also produce a clock-only, when it uses just one channel.

The SIOP TPU function has been designed to closely resemble the SIOP hardware port found on some Motorola MCUs and can be used to add serial capabilities to a device without a serial port, or extend the capabilities of one with a hardware synchronous port.

SIOP operates in master mode (i.e., the TPU always generates the clock) and the following features are programmable by the user:

1. Choice of clock-only (one channel), clock + transmit (two channels), clock + receive (two channels) or clock + transmit + receive (three channels) operating modes
2. Baud rate period is freely programmable by the user over a 15-bit range of TCR1 counts
3. Selection of msb or lsb first shift direction
4. Variable transfer size from one to 16 bits
5. Clock polarity is programmable

When a transfer of data is complete the SIOP function notifies the host CPU by issuing an interrupt request. The arrangement of the multiple SIOP channels is fixed: the data out channel is the channel above the clock channel and the data in channel is the channel below the clock channel. In clock-only or uni-directional mode, the unused TPU channels are free to run other TPU functions. Two possible SIOP configurations are show in **Figure D-27**.

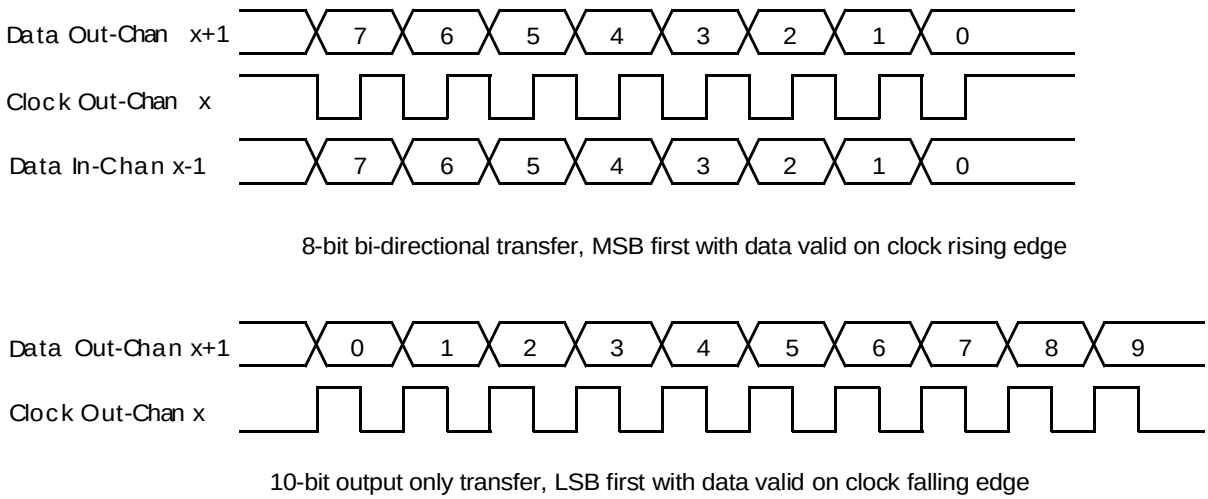


Figure D-27 Two Possible SIOP Configurations

D.17.1 Parameters

Figure D-28 shows the host interface areas and parameter RAM for the SIOP function. The following sections describe these parameters. Note that only the clock channel requires any programming by the user — the data in and out channels are entirely under TPU microcode control.

CONTROL BITS

	NAME	OPTIONS
3 2 1 0	CHANNEL FUNCTION SELECT	SIOP FUNCTION NUMBER (ASSIGNED DURING MICRO-CODE ASSEMBLY)
1 0	CHANNEL PRIORITY	00 — DISABLE 01 — LOW PRIORITY 10 — MEDIUM PRIORITY 11 — HIGH PRIORITY
1 0	HOST SEQUENCE BITS	00 — CLOCK CHANNEL ACTIVE ONLY, NO DATA TRANSFER 01 — D_{OUT} CHANNELS ACTIVE, NO DATA RECEIVE 10 — CLOCK AND D_{IN} CHANNELS ACTIVE, NO DATA TRANSMIT 11 — FULL BIDIRECTIONAL TRANSMIT AND RECEIVE
1 0	HOST SERVICE BITS	00 — NO HOST SERVICE (RESET CONDITION) 01 — NO ACTION 10 — NO ACTION 11 — INITIALIZE CLOCK CHANNEL AND START TRANSFER
0	CHANNEL INTERRUPT ENABLE	0 — INTERRUPT NOT ASSERTED 1 — INTERRUPT ASSERTED

PARAMETER RAM

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
$###FW0	S							CHANNEL_CONTROL								
$###FW2	HALF_PERIOD															
$###FW4													BIT_COUNT			
$###FW6													XFER_SIZE			
$###FW8	DATA															

= WRITTEN BY CPU

= WRITTEN BY CPU AND TPU

= WRITTEN BY TPU

= UNUSED PARAMETERS

W = PRIMARY CHANNEL NUMBER

Figure D-28 SIOP Parameters

D.17.1.1 CHAN_CONTROL

This 9-bit CPU written parameter is used to setup the clock polarity for the SIOP data transfer. The valid values for CHAN_CONTROL for this function are given in the table below. CHAN_CONTROL must be written by the host prior to issuing the host service request (HSR) to initialize the function.

Table D-3 SIOP Function Valid CHAN_Control Options

CHAN_CONTROL[1] 8 7 6 5 4 3 2 1 0	Resulting Action
0 1 0 0 0 1 1 0 1	Data valid on clock Falling edge.
0 1 0 0 0 1 1 1 0	Data valid on clock Rising edge.

NOTES:
1. Other values of CHAN_CONTROL may result in indeterminate operation.

D.17.1.2 BIT_D

BIT_D is a CPU written bit that determines the direction of shift of the SIOP data. If BIT_D is zero then SIOP_DATA is right shifted (lsb first). If BIT_D is one then SIOP_DATA is left shifted (msb first).

D.17.1.3 HALF_PERIOD

This CPU-written parameter defines the baud rate of the SIOP function. The value contained in HALF_PERIOD is the number of TCR1 counts for a half SIOP clock period (e.g., for a 50 KHz baud rate, with a TCR1 period of 240 ns, the value [(1/50 KHz)/2]/240 ns = 42 should be written to HALF_PERIOD. The range for HALF_PERIOD is 1 to $8000, although the minimum value in practice will be limited by other system conditions. See notes on use and performance of SIOP function.

D.17.1.4 BIT_COUNT

This parameter is used by the TPU to count down the number bits remaining while a transfer is in progress. During the SIOP initialization state, BIT_COUNT is loaded with the value contained in XFER_SIZE. It is then decremented as the data is transferred and when it reaches zero, the transfer is complete and the TPU issues an interrupt request to the CPU.

D.17.1.5 XFER_SIZE

This CPU-written parameter determines the number of bits that make up a data transfer. During initialization, XFER_SIZE is copied into BIT_COUNT. XFER_SIZE is shown as a 5-bit parameter to match the maximum size of 16 bits in SIOP_DATA, although the TPU uses the whole word location. For normal use, XFER_SIZE should be in the range 1-to-16.

D.17.1.6 SIOP_DATA

This parameter is the data register for all SIOP transfers. Data is shifted out of one end of SIOP_DATA and shifted in at the other end, the shift direction being determined by the value of BIT_D. In output only mode, zero will be shifted into SIOP_DATA and

in input only mode, the data shifted out is ignored. In clock-only mode SIOP_DATA is still shifted. Note that no ‘justifying’ of SIOP_DATA is performed by the TPU, (e.g., if an 8-bit bi-directional transfer is made, shifting lsb first, then the bottom byte of SIOP_DATA will be shifted out and the input data will be shifted into the upper byte of SIOP_DATA).

NOTE

SIOP_DATA is not buffered. The CPU should only access it between completion of one transfer and the start of the next.

D.17.2 Host CPU Initialization of the SIOP Function

The CPU initializes the SIOP function by:

1. Disabling the channel by clearing the two channel priority bits
2. Selecting the SIOP function on the channel by writing the assigned SIOP function number to the function select bits
3. Writing CHAN_CONTROL in the clock channel parameter RAM
4. Writing HALF_PERIOD, BIT_D and XFER_SIZE in the clock channel parameter RAM to determine the speed, shift direction and size of the transfer
5. Writing SIOP_DATA if the data output is to be used
6. Selecting the required operating mode via the two host sequence bits
7. Issuing a host service request type %11
8. Enabling service by assigning H, M or L priority to the clock channel via the two channel priority bits

The TPU then starts the data transfer, and issues an interrupt request when the transfer is complete.

Once the function has been initialized, the CPU only needs to write SIOP_DATA with the new data and issue a HSR %11 to initiate a new transfer. In input or clock-only modes, just the HSR %11 is required.

D.17.3 SIOP Function Performance

Like all TPU functions, the performance limit of the SIOP function in a given application is dependent to some extent on the service time (latency) associated with other active TPU channels. This is due to the operational nature of the scheduler. Where two channels are being used for a uni-directional system, and no other TPU channels are active, the maximum baud rate is approximately 230 KHz at a bus speed of 16.77 MHz. A three-channel bi-directional system under the same conditions has a maximum baud rate of approximately 200 KHz. When more TPU channels are active, these performance figures will be degraded, however, the scheduler assures that the worst case latency in any TPU application can be closely approximated. It is recommended that the guidelines given in the TPU reference manual be used along with the information given in the SIOP state timing table to perform an analysis on any proposed TPU application that appears to approach the performance limits of the TPU.

Table D-4 SIOP State Timing[1]

State Number and Name	Max. CPU Clock Cycles	Number of RAM Accesses by TPU
S1 SIOP_INIT		
HSQ = X0	28	7
X1	38	7
S2 DATA_OUT		
HSQ = X0	14	4
X1	24	4
S3 DATA_IN		
HSQ = 0X	14	4
1X	28	6

NOTES:
1. Execution times do not include the time slot transition time (TST = 10 or 14 CPU clocks).

D.17.3.1 XFER_SIZE Greater than 16

XFER_SIZE is normally programmed to be in the range 1-to-16 to match the size of SIOP_DATA, and has thus been shown as a 5-bit value in the host interface diagram. However, the TPU actually uses all 16 bits of the XFER_SIZE parameter when loading BIT_COUNT. In some unusual circumstances this can be used to the user's advantage. If an input device is producing a data stream of greater than 16 bits then manipulation of XFER_SIZE will allow selective capturing of the data. In clock-only mode, the extended XFER_SIZE can be used to generate up to $FFFF clocks.

D.17.3.2 Data Positioning

As stated above, no 'justifying' of the data position in SIOP_DATA is performed by the TPU. This means that in the case of a byte transfer, the data output will be sourced from one byte and the data input will shift into the other byte. This rule holds for all data size options except 16 bits when the full SIOP_DATA register is used for both data output and input.

D.17.3.3 Data Timing

In the example given in **Figure D-29**, the data output transitions are shown as being completely synchronous with the relevant clock edge and it is assumed that the data input is latched exactly on the opposite clock edge. This is the simplest way to show the examples, but is not strictly true. Since the TPU is a multi-tasking system, and the data channels are manipulated directly by microcode software while servicing the clock edge, there is a finite delay between the relevant clock edge and the data-out being valid or the data-in being latched. This delay is equivalent to the latency in servicing the clock channel due to other TPU activity and is shown as 'Td' in the timing diagram. Td is the delay between the clock edge and the next output data being valid and also the delay between the opposite clock edge and the input data being read. For the vast majority of applications, the delay Td will not present a problem and can be ignored. Only for a system which heavily loads the TPU should the user calculate the worst case latency for the SIOP clock channel + actual SIOP service time (= Td) and ensure that the baud rate is chosen such that HALF_PERIOD - Td is not less that the

minimum setup time of the receiving device. A transmitting device must also hold data valid for a minimum time of Td after the clock.

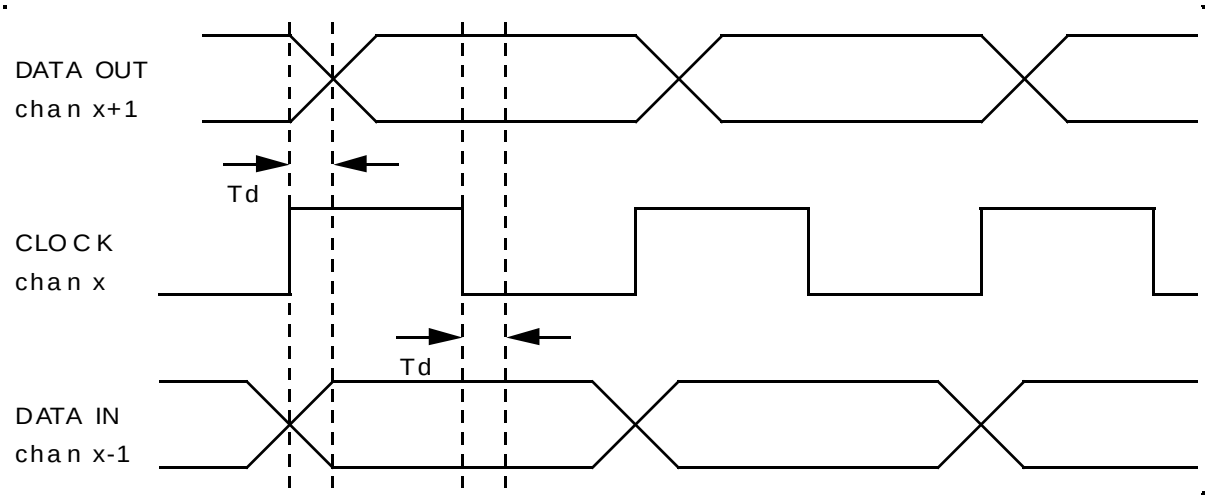


Figure D-29 SIOP Function Data Transition Example

APPENDIX E
ELECTRICAL CHARACTERISTICS

E.1 Absolute Maximum Ratings

Table E-1 Absolute Maximum Ratings

Num	Rating	Symbol	Value	Unit
1	3.3V Supply Voltage[1, 2, 6] 5V Supply Voltage	V_{DDL}, $V_{DDDP-TRAM}$ V_{DDH}	– 0.3 to + 4.0 – 0.3 to + 6.0	V
2	Input Voltage[1, 2, 4, 6]	V_{in}	– 0.3 to + 5.5	V
3	Instantaneous Maximum Current Single pin limit (applies to all pins)[1, 4, 5, 6]	I_D	25	mA
4	Operating Maximum Current Digital Input Disruptive Current [3, 4, 5, 6, 7] $V_{NEGCLAMP} \cong -0.3$ V $V_{POSCLAMP} \cong V_{DD} + 0.3$	I_{ID}	– 500 to 500	µA
5	Operating Temperature Range No Suffix “C” Suffix “V” Suffix “M” Suffix	T_A	T_L to T_H 0 to 70 – 40 to 85 – 40 to 105 – 40 to 125	°C
6	Storage Temperature Range	T_{stg}	– 55 to 150	°C

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.
3. All functional non-supply pins are internally clamped to V_{SS}. All functional pins except EXTAL and XFC are internally clamped to V_{DD}. Does not include QADC64 pins (see **Table E-10**).
4. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values.
5. Power supply must maintain regulation within operating V_{DD} range during instantaneous and operating maximum current conditions except for V_{PP}.
6. This parameter is periodically sampled rather than 100% tested.
7. Total input current for all digital input-only and all digital input/output pins must not exceed 10 mA. Exceeding this limit can cause disruption of normal operation.

E.2 Radiated Emissions

Characterization of device emissions will be performed per SAE J1752/3 issued March 1995.

E.3 Thermal Characteristics

Table E-2 Thermal Characteristics

Num	Characteristic	Symbol	Value	Unit
1	Thermal Resistance Thermal Resistance 217 /P4 PBGA Package	Θ_{JA}	TBD	°C/W

The average chip-junction temperature (T_J) in C can be obtained from:

$$T_J = T_A + (P_D \times \Theta_{JA}) \qquad (1)$$

where:

T_A = Ambient Temperature, °C
Θ_{JA}= Package Thermal Resistance, Junction-to-Ambient, °C/W
$P_D = P_{INT} + P_{I/O}$
$P_{INT}= I_{DD} \times V_{DD}$, Watts — Chip Internal Power
$P_{I/O}$= Power Dissipation on Input and Output Pins — User Determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected. An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \qquad (2)$$

Solving equations 1 and 2 for K gives:

$$K = P_D + (T_A + 273°C) + \Theta_{JA} \times P_{D^2} \qquad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at 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.

E.4 DC Characteristics

Table E-3 DC Characteristics

(V_{DDH} = 5.0 Vdc ± 10%, V_{DDL} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
1a	3v Input High Voltage, EPEB0	V_{IH3}	2.0	V_{DDH} + 0.3	V
1b	5v Input High Voltage[1] Groups 2 and 7 pins, except EPEB0 Groups 1, 5, and 6 pins	V_{IH5}	 0.7 (V_{DDH}) 3.3	 V_{DDH} + 0.3 $V_{DD}H$ + 0.3	 V V
2a	3v Input Low Voltage, EPEB0	V_{IL3}	V_{SS} – 0.3	0.8	V
2b	5v Input Low Voltage[1] Groups 2 and 7 pins, except EPEB0 Groups 1, 5, and 6 pins	V_{IL5}	 V_{SS} – 0.3 V_{SS} – 0.3	 0.2 (V_{DDH}) 2.3	 V V
3	Input Hysteresis $\overline{BKPT}$/$\overline{DSCLK}$, CTM2C, Port CT, $\overline{IPIPE}$/DSO, Port A, Port B, Port E[7:3], Port F, Port G, Port H, $\overline{RESET}$, $\overline{IFETCH}$/DSI, Port TP, T2CLK	V_{HYS}	0.5	—	V
4	3v and 5v Input Leakage Current[1] $V_{in} = V_{DD}$ or V_{SS} all I/O and Input-only pins	I_{in}	–1.0	1.0	µA
5	High Impedance (Off-State) Leakage Current[1] $V_{in} = V_{DD}$ or V_{SS} All I/O and Output-only pins	I_{OZ}	–1.0	1.0	µA

Table E-3 DC Characteristics (Continued)

(V_{DDH} = 5.0 Vdc ± 10%, V_{DDL} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
6	3v Output High Voltage I_{OH} = –.01 mA XTAL	V_{OH3}	2.4	—	V
6a	5v Fast Output High Voltage[1, 2] IOH = –2.0 mA Groups 3 and 4 Output-only and Groups 5, 6, and 7 I/O pins	V_{OH5}	V_{DDH} – 0.7	—	V
6b	5v Slow Output High Voltage[2] I_{OH} = –0.4 mA Ports A, B, C, E, F, G, H	V_{OH5S}	V_{DDH} – 0.7	—	V
7	3v Output Low Voltage I_{OL} = .01 mA XTAL	V_{OL3}	—	0.4	V
7a	5v Fast Output Low Voltage[1] I_{OL}= 4.0 mA Group 4 Output-only and Groups 5 I/O pins I_{OL}= 2.0 mA Group 3 Output-only and Groups 6 and 7 I/O pins	V_{OL5}	—	0.4	V
7b	5v Slow Output Low Voltage I_{OL} = 0.4 mA Ports A, B, C, E, F, G, H	V_{OL5S}	—	0.4	V
8	Data Bus Mode Select Pull-up Current V_{in} = V_{IL} DATA[15:0] V_{in} = V_{IH} DATA[15:0]	I_{MSP}	— –15	–120 —	µA
9	V_{DD} Supply Current[3] RUN[4] RUN, TPU3 emulation mode CMFI Program/Erase Adder[5] LPSTOP, 4.194MHz crystal, VCO Off (STSIM = 0) LPSTOP (External clock input frequency = maximum f_{sys})	I_{DDL} I_{DDH} I_{DDL} S_{IDD} S_{IDD}	— — — — —	190 40 210 100 500 10	mA mA mA mA µA mA
10	Clock Synthesizer Operating Voltage	V_{DDSYN}	3.0	3.6	V
11	V_{DDSYN} Supply Current[3] 4.194 MHz crystal, VCO on, maximum f_{sys} External Clock, maximum f_{sys} LPSTOP, 4.194MHz crystal, VCO off (STSIM = 0) 4.194 MHz crystal, V_{DD} powered down	I_{DDSYN} I_{DDSYN} S_{IDDSYN} I_{DDSYN}	— — — —	10 7 5 5	mA mA mA mA
12	RAM Standby Voltage[6] Specified V_{DD} applied V_{DD} = V_{SS}	V_{SB}	0.0 3.0	3.6 3.6	V
12A	DPTRAM Voltage[7]	$V_{DDDP-TRAM}$	3.0	3.6	V
13	RAM Standby Current[3, 6, 8] Normal RAM operation $V_{DD} > V_{SB} – 0.5$ V Transient condition $V_{SB} – 0.5 V \geq V_{DD} \geq V_{SS} + 0.5$ V Standby operation $V_{DD} < V_{SS} + 0.5$ V	I_{SB}	— — —	10 3 25	µA mA µA

Table E-3 DC Characteristics (Continued)

(V_{DDH} = 5.0 Vdc ± 10%, V_{DDL} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H)

Num	Characteristic	Symbol	Min	Max	Unit
13A	DPTRAM Current[3, 7, 8]	I_{DPTRAM}	—	20	mA
14	Power Dissipation[9]	P_D	—	1.07	W
15	Input Capacitance[1, 10] All Input-only pins All I/O pins	—	 — —	 10 20	pF
16	Load Capacitance[1] Groups 4 and 5 I/O pins Group 3 Output-Only and Groups 6 and 7 I/O pins	C_L	 — —	 70 50	pF

NOTES:

1. Input-Only Pins:
   Group 1:
   $\overline{BKPT}$/DSCLK, TSC, CTM2C, $\overline{BERR}$/$\overline{SCEN}$, $\overline{T2CLK}$
   Group 2:
   Port QS - PQS8/RXD1, PQS10/RXD2
   Other — XFC, EXTAL, ANX[15:0], CNRX, EPEB0
   Output-Only Pins:
   Group 3:
   Port CT — $\overline{CPWM}$[8:5]
   Other — $\overline{CNTX}$
   Group 4:
   Port C — $\overline{ADDR23}$/$\overline{CS10}$/$\overline{ECLK}$, $\overline{ADDR}$[22:19]/$\overline{CS}$[9:6]/$\overline{PC}$[6:3], $\overline{FC2}$/$\overline{CS5}$/$\overline{PC2}$, $\overline{FC1}$/$\overline{PC1}$, $\overline{FC0}$/$\overline{CS3}$/$\overline{PC0}$
   Other — $\overline{IPIPE}$/$\overline{DS0}$, $\overline{CSBOOT}$, $\overline{BG}$/CSM, $\overline{CLKOUT}$, $\overline{FREEZE}$/$\overline{QOUT}$, $\overline{ADDR}$[2:0], R/$\overline{W}$
   Group 8:
   Other — $\overline{XTAL}$
   Input/Output Pins:
   Group 5:
   Port A — PA[7:0]/ADDR[18:11]
   Port B — PB[7:0]/ADDR[10:3]
   Port E — PE[7:6]/SIZ[1:0], PE5/$\overline{AS}$, PE4/$\overline{DS}$, PE[1:0]/$\overline{DSACK[1:0]}$
   Port F — PF[7:5]/$\overline{IRQ[7:5]}$, PF0/FASTREF
   Port G — PG[7:0]/DATA[15:8]
   Port H — PH[7:0]/DATA[7:0]
   Other — $\overline{HALT}$, $\overline{RESET}$, $\overline{BGACK}$/$\overline{CSE}$, $\overline{BR}$/$\overline{CS0}$, $\overline{IFETCH}$/DSI
   Group 6:
   Port CT — $\overline{CTD[10:9]}$/[4:3], $\overline{CTS[20A/B:18A/B:16A/B14A/B]}$
   Port TP — $\overline{TP[15:0]}$
   Group 7:
   Port QS — PQS7/TXD1, PQS9/TXD2, PQS[6:4]/PCS[3:1], PQS3/PCS0/$\overline{SS}$, PQS2/SCK, PQS1/MOSI, PQS0/MISO
   Pin groups do not include QADC64 pins. See **Table E-10** through **Table E-13** for information concerning the QADC64.
2. Use of an active pulldown device is recommended during reset to select operating mode.
3. Total operating current is the sum of the appropriate I_{DDH}, I_{DDL} , I_{DDSYN}, I_{DDA}, I_{DPTRAM}, and I_{SB} values.
4. Current measured at 24 MHz system clock frequency, all modules active.
5. Add current if CMFI is being programed or erased.
6. The SRAM module will not switch into standby mode as long as V_{SB} does not exceed V_{DD} by more than 0.5 volts. The SRAM array cannot be accessed while the module is in standby mode.
7. The DPTRAM module will not switch into standby mode as long as V_{SB} does not exceed V_{DD} by more than 0.5 volts. The DPTRAM array cannot be accessed while the module is in standby mode.
8. When V_{DD} is transitioning during power-up or power down sequence, and V_{SB} is applied, current flows between the V_{STBY} and V_{DD} pins, which causes standby current to increase toward the maximum transient condition specification. System noise on the V_{DD} and V_{STBY} pins can contribute to this condition.
9. Power dissipation measured at 24 MHz system clock frequency, all modules active, not in TPU emulation mode, not programming or erasing the CMFI. Add for TPU emulation and/or CMFI program/erase operation. Power dissipation can be calculated using the following expression:
   P_D = Maximum V_{DDH} (I_{DDH} + I_{DDA}) + Maximum V_{DDL} (I_{DDL} + I_{DDSYN} +I_{DPTRAM} + I_{SB})
10. This parameter is periodically sampled rather than 100% tested.

E.5 AC Characteristics

Table E-4 Clock Control Timing

(V_{DDL} and V_{DDSYN} = 3.3 Vdc ±10%, V_{SS} = 0 Vdc, T_A = T_L to T_H.)

Num	Characteristic	Symbol	Minimum	Maximum	Unit
1	PLL Reference Frequency Range Indirect: Fast reference mode Slow reference mode	f_{ref}	 1 25	 6 50	 MHz KHz
2	System Frequency [1] On-Chip PLL System Frequency: Fast reference mode Slow reference mode External Clock Operation	f_{sys}	DC $(f_{ref})/128$ $4(f_{ref})$ dc	33.6 33.6 33.6 33.6	MHz
3	PLL Lock Time Changing W or Y in SYNCR or exiting from LPSTOP[2] Warm Start-up[3] Cold start (fast mode only)[4]	t_{lpll}	 — — —	 20 50 75	ms
4	VCO frequency[5]	f_{VCO}	—	$2(f_{sys}$ max)	MHz
5	Limp Mode Clock Frequency [6]	f_{limp}	0.1	f_{sys} max/2	MHz
6	CLKOUT Jitter[7,8] Slow Reference Mode (32.768 kHz): Short term (5 µs interval) Long term (500 µs interval) Fast Reference Mode (4.194 MHz): Short term (3 system clocks) Long term (2 ms interval)	J_{CLK}	 -0.5 -0.05 -1.0 -0.01	 0.5 0.05 1.0 0.01	%

NOTES:

1. All internal registers retain data at 0 Hz.
2. Assumes that V_{DDSYN} and V_{DD} are stable, that an external filter is attached to the XFC pin, and that the crystal oscillator is stable.
3. Assumes that V_{DDSYN} is stable, that an external filter is attached to the XFC pin, and that the crystal oscillator is stable, followed by V_{DD} ramp-up. Lock time is measured from VDD at specified minimum to $\overline{RESET}$ negated.
4. Cold start is measured from V_{DDSYN} and V_{DD} at specified minimum to $\overline{RESET}$ negated.
5. Internal VCO frequency (f_{VCO}) is determined by SYNCR W and Y bit values. The SYNCR X bit controls a divide-by-two circuit that is not in the synthesizer feedback loop. When X = 0, the divider is enabled, and $f_{sys} = f_{VCO}/4$. When X = 1, the divider is disabled, and $f_{sys} = f_{VCO}/2$. X must equal one when operating at maximum specified f_{sys}.
6. Determined by internal loss-of-clock oscillator operating frequency.
7. Jitter is the average deviation from programmed frequency measured over the specified interval at maximum f_{sys}. Measurements are made with the device powered by filtered supplies and clocked by a stable external clock signal. Noise injected into the PLL circuitry via power supply pins and variation in crystal oscillator frequency increase the stability percentage for a given interval. The use of reference frequencies and system frequencies very much different than those shown here may require different XFC filter values than shown in **4.3.7.2 Phase Comparator and Filter** to maintain optimum jitter performance.
8. This parameter is periodically sampled rather than 100% tested.

Table E-5 AC Timing

(V_{DDH} = 5.0 Vdc ± 10%, V_{DDL} and V_{DDSYN} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
F1	Frequency of Operation	f_{SLW}	0.13	33.6	MHz
1	Clock Period	t_{cyc}	29.8	—	ns
1A	ECLK Period	t_{Ecyc}	$8*t_{cyc}$	—	ns
1B	External Clock Input Period[2,3]	t_{Xcyc}	14.9	—	ns
2, 3	Clock Pulse Width	t_{CW}	$0.5t_{cyc}$-3	—	ns
2A, 3A	ECLK Pulse Width	t_{ECW}	t_{cyc}-7	—	ns
2B, 3B	External Clock Input High/Low Time[2]	t_{XCHL}	7.45	—	ns
4, 5	Clock Rise and Fall Time	t_{Crf}	—	3	ns
4A, 5A	Rise and Fall Time SCIM2E pins: CSBOOT, CLKOUT, $\overline{BKPT}$, $\overline{IFETCH}$, $\overline{IPIPE}$, $\overline{BG}$, $\overline{BR}$, $\overline{BGACK}$, A[23,2:0], FREEZE, $\overline{BERR}$, R/$\overline{W}$, $\overline{HALT}$, $\overline{RESET}$ FC[2:0], A[22:3], D[15:0], SIZE[1:0], $\overline{AS}$, $\overline{DS}$, RMC, AVEC, $\overline{DSACK}$[1:0][4] Fast[5] Slow[5.] IRQ[7:1], FASTREF[6]	t_{rf}	—	3 3 >200 >200	ns
4B, 5B	External Clock Rise and Fall Time[7]	t_{XCrf}	—	3	ns
6	Clock High to Address, FC, SIZE Valid	t_{CHAV}	0	0.5	t_{cyc}
7	Clock High to Address, Data, FC, SIZE High Impedance	t_{CHAZx}	0	1.0	t_{cyc}
8	Clock High to Address, FC, SIZE Invalid	t_{CHAZn}	0	—	ns
9	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted	t_{CLSA}	2	$0.5t_{cyc}$	ns
9A	$\overline{AS}$, to $\overline{DS}$, or $\overline{CS}$ Asserted (Read)[8]	t_{STSA}	–15	8	ns
9C	Clock Low to $\overline{IFETCH}$, $\overline{IPIPE}$ Asserted	t_{CLIA}	2	11	ns
11	Address, FC, SIZE Valid to $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Asserted	t_{AVSA}	0.25	—	t_{cyc}
12	Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated	t_{CLSN}	1	15	ns
12A	Clock Low to $\overline{IFETCH}$, $\overline{IPIPE}$ Negated	t_{CLIN}	1	11	ns
13	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to Address, FC, SIZE Invalid (Address Hold)	t_{SNAI}	0.25	—	t_{cyc}
14	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Read) Width Asserted	t_{SWA}	50	—	ns
14A	$\overline{DS}$, $\overline{CS}$ Width Asserted (Write)	t_{SWAW}	23	—	ns
14B	$\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted (Fast Write Cycle)	t_{SWDW}	20	—	ns
15	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated[9]	t_{SN}	20	—	ns
16	Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ High Impedance	t_{CHSZ}	—	30	ns
17	$\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to R/$\overline{W}$ Negated	t_{SNRN}	0.25	—	t_{cyc}
18	Clock High to R/$\overline{W}$ High	t_{CHRH}	0	0.5	t_{cyc}

Table E-5 AC Timing (Continued)

(V_{DDH} = 5.0 Vdc ± 10%, V_{DDL} and V_{DDSYN} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
20	Clock High to R/$\overline{W}$ Low	t_{CHRL}	0	0.5	t_{cyc}
21	R/$\overline{W}$ Asserted to $\overline{AS}$, $\overline{CS}$ Asserted	t_{RAAA}	0.25	—	t_{cyc}
22	R/$\overline{W}$ Low to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{RASA}	35	—	ns
23	Clock High to Data Out Valid	t_{CHDO}	—	0.5	t_{cyc}
24	Data Out Valid to Negating Edge of $\overline{AS}$, $\overline{CS}$ (Fast Write Cycle)	t_{DVASN}	0.25	—	t_{cyc}
25	$\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold)	t_{SNDOI}	0.25	—	t_{cyc}
26	Data Out Valid to $\overline{DS}$, $\overline{CS}$ Asserted (Write)	t_{DVSA}	0.25	—	t_{cyc}
27	Data In Valid to Clock Low (Data Setup)	t_{DICL}	3	—	ns
27A	Late $\overline{BERR}$, $\overline{HALT}$ Asserted to Clock Low (Setup Time)	t_{BELCL}	10	—	ns
28	$\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACK[1]}$, $\overline{BERR}$, $\overline{HALT}$ Negated	t_{SNDN}	0	40	ns
29	$\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)[10]	t_{SNDI}	0	—	ns
29A	$\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance [10, 11]	t_{SHDI}	—	28	ns
30	CLKOUT Low to Data In Invalid (Fast Cycle Hold)[10]	t_{CLDI}	0.25	—	t_{cyc}
30A	CLKOUT Low to Data In High Impedance[10]	t_{CLDH}	—	45	ns
31	$\overline{DSACK[1]}$ Asserted to Data In Valid[12]	t_{DADI}	—	25	ns
33	Clock Low to $\overline{BG}$ Asserted/Negated	t_{CLBAN}	—	0.5	t_{cyc}
35	$\overline{BR}$ Asserted to $\overline{BG}$ Asserted[13]	t_{BRAGA}	1	—	t_{cyc}
37	$\overline{BGACK}$ Asserted to $\overline{BG}$ Negated	t_{GAGN}	1	2	t_{cyc}
39	$\overline{BG}$ Width Negated	t_{GH}	1	—	t_{cyc}
39A	$\overline{BG}$ Width Asserted	t_{GA}	1	—	t_{cyc}
46	R/$\overline{W}$ Width Asserted (Write or Read)	t_{RWA}	75	—	ns
46A	R/$\overline{W}$ Width Asserted (Fast Write or Read Cycle)	t_{RWAS}	45	—	ns
47A	Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK[1]}$, $\overline{BERR}$, $\overline{HALT}$	t_{AIST}	3	—	ns
47B	Asynchronous Input Hold Time	t_{AIHT}	8	—	ns
48	$\overline{DSACK[1]}$ Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted[14]	t_{DABA}	—	0.5	t_{cyc}
53	Data Out Hold from Clock High	t_{DOCH}	0	—	ns
54	Clock High to Data Out High Impedance	t_{CHDH}	—	14	ns
55	R/$\overline{W}$ Asserted to Data Bus Impedance Change	t_{RADC}	20	—	ns
56	$\overline{RESET}$ Pulse Width (Reset Instruction)	t_{HRPW}	512	—	t_{cyc}
57	$\overline{BERR}$ Negated to $\overline{HALT}$ Negated (Rerun)	t_{BNHN}	0	—	ns
70	Clock Low to Data Bus Driven (Show)	t_{SCLDD}	0	0.5	t_{cyc}
71	Data Setup Time to Clock Low (Show)	t_{SCLDS}	8	—	ns
72	Data Hold from Clock Low (Show)	t_{SCLDH}	5	—	ns

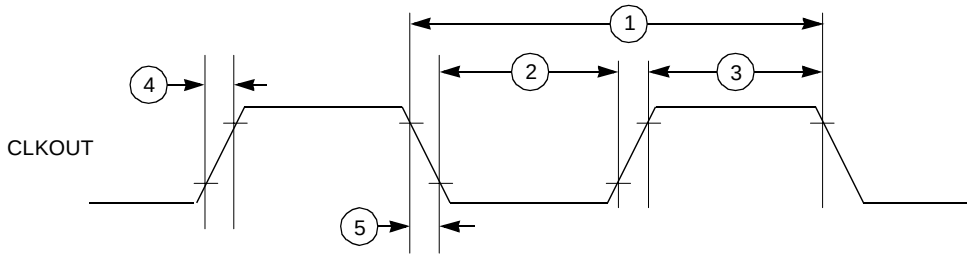
Table E-5 AC Timing (Continued)

(V_{DDH} = 5.0 Vdc ± 10%, V_{DDL} and V_{DDSYN} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
73	$\overline{BKPT}$ Input Setup Time	t_{BKST}	0.25	—	t_{cyc}
74	$\overline{BKPT}$ Input Hold Time	t_{BKHT}	5	—	ns
75	Mode Select Setup Time	t_{MSS}	10	—	t_{cyc}
76	Mode Select Hold Time	t_{MSH}	0	—	ns
77	$\overline{RESET}$ Assertion Time[15]	t_{RSTA}	2	—	t_{cyc}
78	$\overline{RESET}$ Rise Time[16,17]	t_{RSTR}	—	10	t_{cyc}

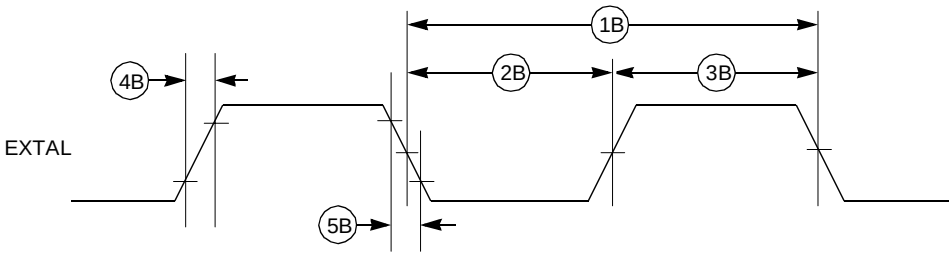
NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
2. When an external clock is used, minimum high and low times are based on a 50% duty cycle. The minimum allowable t_{Xcyc} period is reduced when the duty cycle of the external clock varies. The relationship between external clock input duty cycle and minimum t_{Xcyc} is expressed:
   Minimum t_{Xcyc} period = minimum t_{XCHL} / (50% – external clock input duty cycle tolerance).
3. Due to the requirement by the CMFI module, the SCIM2E on this device requires an external clock equal to 2X system frequency to be driven when in external clock mode.
4. Rev B silicon has these pins forced in FAST mode.
5. Pins are set to FAST mode when they are configured as bus and bus control pins. Pins are set to SLOW mode when they are digital I/O. The SCIM2E has a bit in the MCR to force FAST Mode.
6. These pins have fast and slow rise fall times of 3 and >200, depending on the state of the SLOWE bit in the SCIMMCR.
7. Parameters for an external clock signal applied while the internal PLL is disabled (VDDSYN/MODCLK pin held low during reset). Does not pertain to an external reference clock source while the PLL is enabled (VDDSYN/MODCLK pin held high during reset). When the PLL is enabled, the clock synthesizer detects successive transitions of the reference signal. If transitions occur within the correct clock period, rise/fall times and duty cycle are not critical.
8. The amount of skew depends on the relative loading of these signals.
9. If multiple chip selects are used, $\overline{CS}$ width negated applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select. The $\overline{CS}$ width negated between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
10. Hold times are specified with respect to $\overline{DS}$ or $\overline{CS}$ on asynchronous reads and with respect to CLKOUT on fast cycle reads. The user is free to use either hold time.
11. Maximum value is equal to (t_{cyc} / 2) + 25 ns.
12. If the asynchronous setup time requirements are satisfied, the $\overline{DSACK[1]}$ low to data setup time and $\overline{DSACK[1]}$ low to $\overline{BERR}$ low setup time can be ignored. The data must only satisfy the data-in to clock low setup time for the following clock cycle. $\overline{BERR}$ must satisfy only the late $\overline{BERR}$ low to clock low setup time for the following clock cycle.
13. To ensure coherency during every operand transfer, $\overline{BG}$ is not asserted in response to $\overline{BR}$ until after all cycles of the current operand transfer are complete.
14. In the absence of $\overline{DSACK[1]}$, $\overline{BERR}$ is an asynchronous input using the asynchronous setup time.
    Address access time = (2.5 + WS) t_{cyc} – t_{CHAV} – t_{DICL}
    Chip select access time = (2 + WS) t_{cyc} – t_{CLSA} – t_{DICL}
    Where: WS = number of wait states. When fast termination is used (2 clock bus) WS = –1.
15. After external $\overline{RESET}$ negation is detected, a short transition period (approximately 2 t_{cyc}) elapses, then the Integration Module drives $\overline{RESET}$ low for 512 tcyc.
16. External assertion of the $\overline{RESET}$ input can overlap internally-generated resets. To insure that an external reset is recognized in all cases, $\overline{RESET}$ must be asserted for at least 590 CLKOUT cycles.
17. External logic must pull $\overline{RESET}$ high during this period in order for normal MCU operation to begin.



68300 CLKOUT TIM

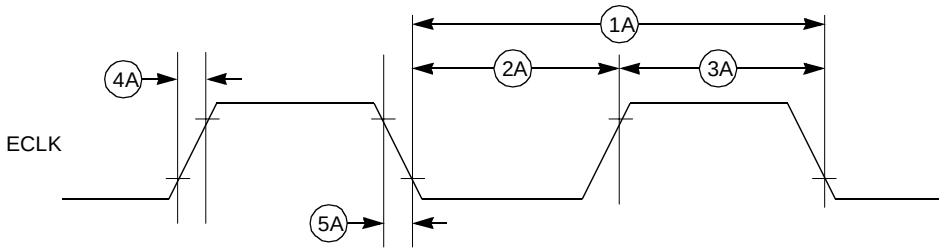
Figure E-1 CLKOUT Output Timing Diagram



NOTE: TIMING SHOWN WITH RESPECT TO 20% AND 70% V_{DD}.
PULSE WIDTH SHOWN WITH RESPECT TO 50% V_{DD}.

68300 EXT CLK INPUT TIM

Figure E-2 External Clock Input Timing Diagram



NOTE: TIMING SHOWN WITH RESPECT TO 20% AND 70% V_{DD}

68300 ECLK OUTPUT TIM

Figure E-3 ECLK Output Timing Diagram

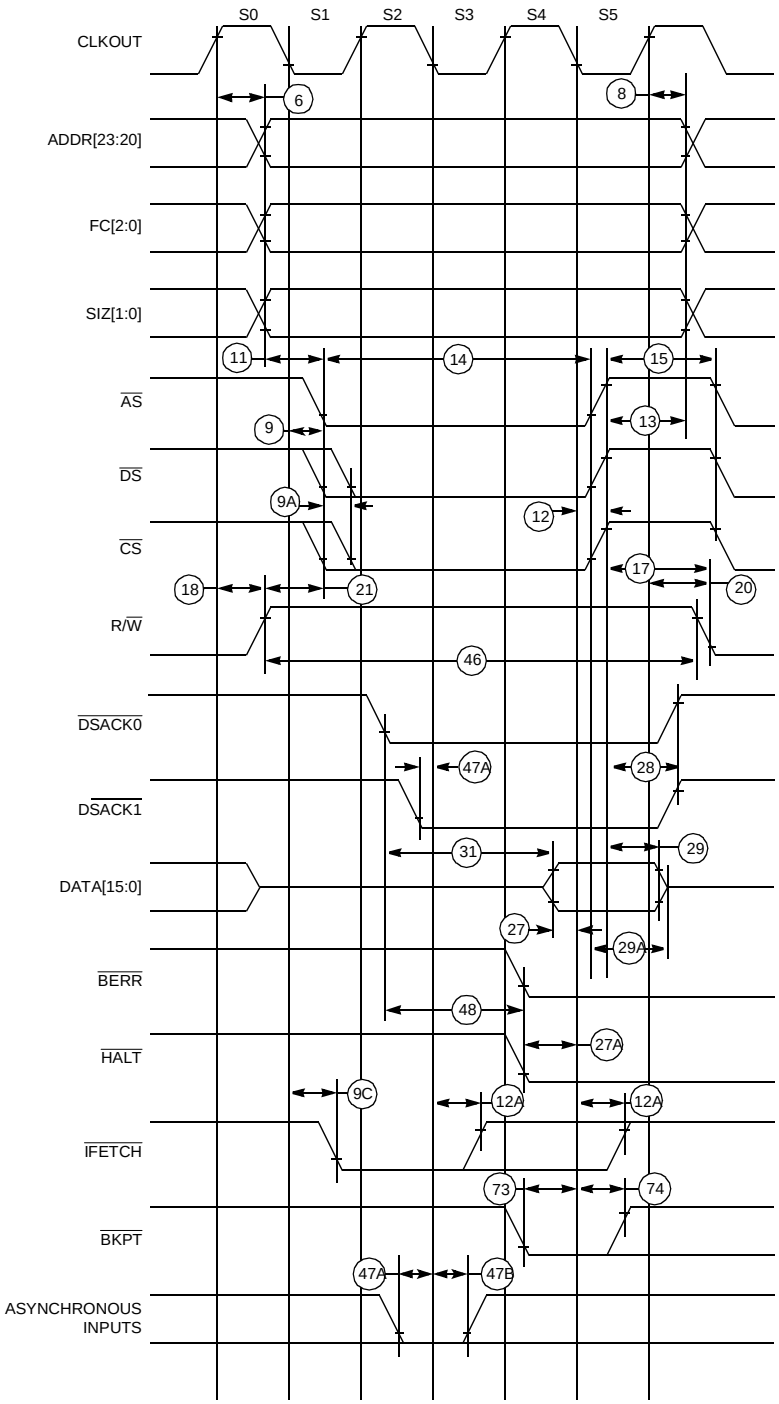
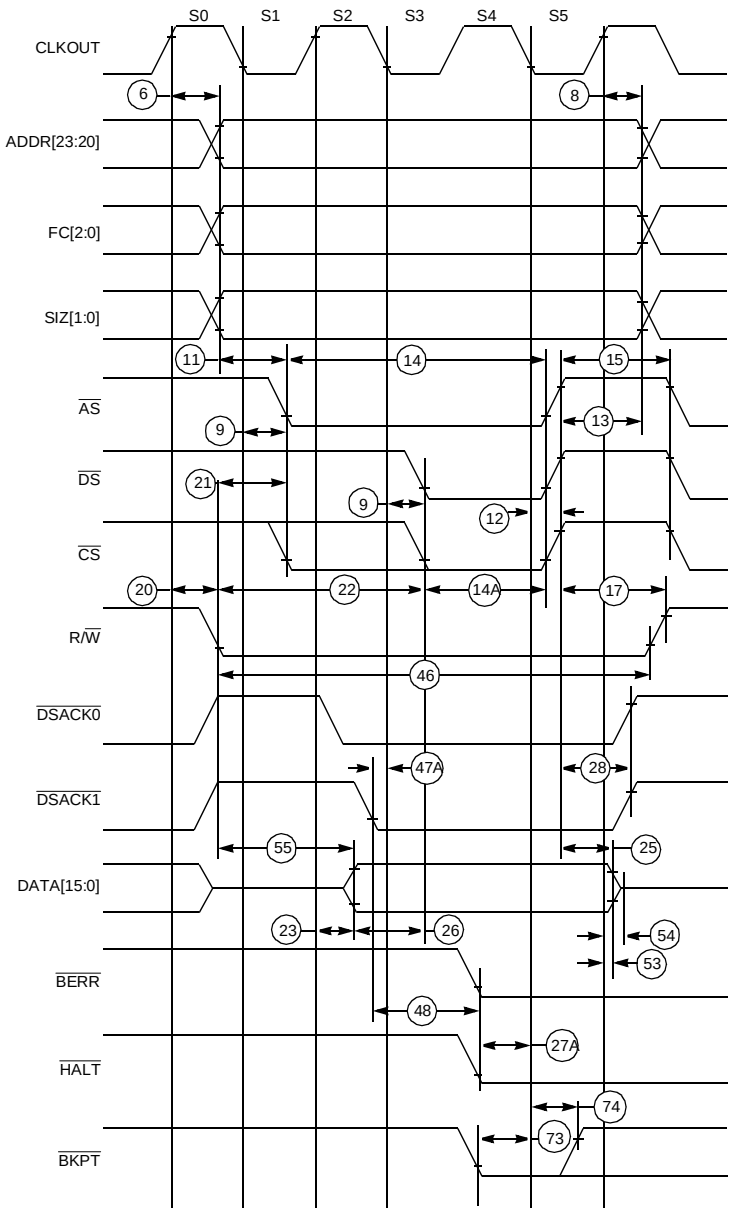


Figure E-4 Read Cycle Timing Diagram



68300 WR CYC TIM

Figure E-5 Write Cycle Timing Diagram

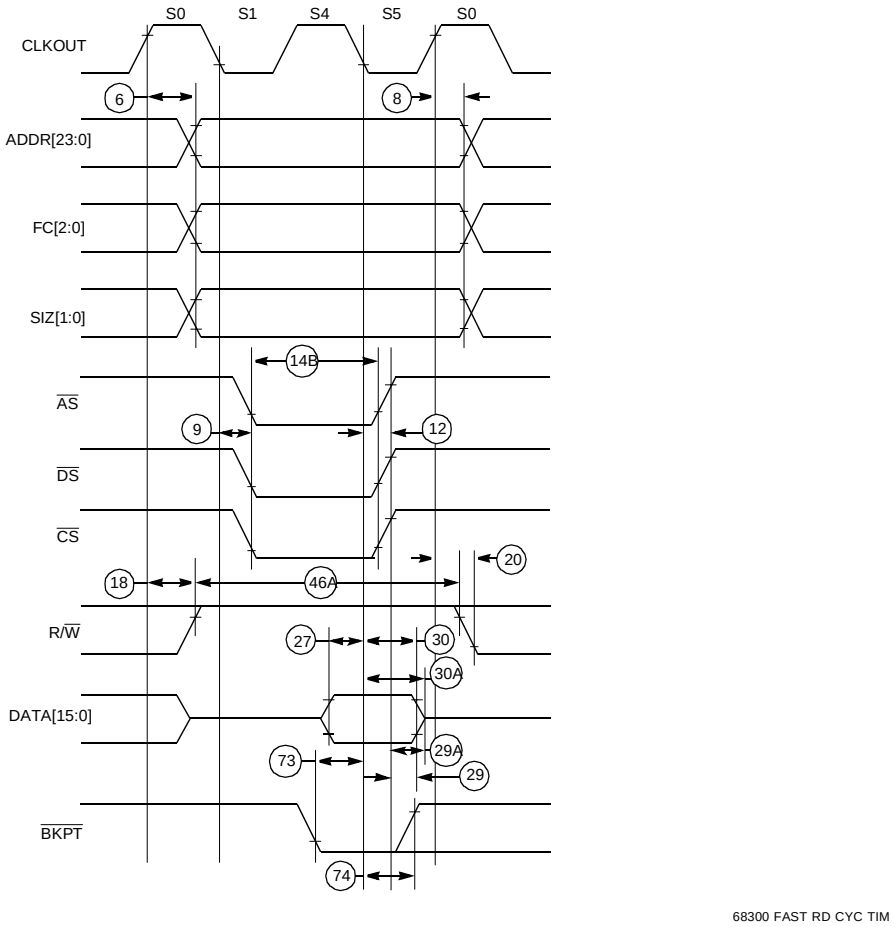


Figure E-6 Fast Termination Read Cycle Timing Diagram

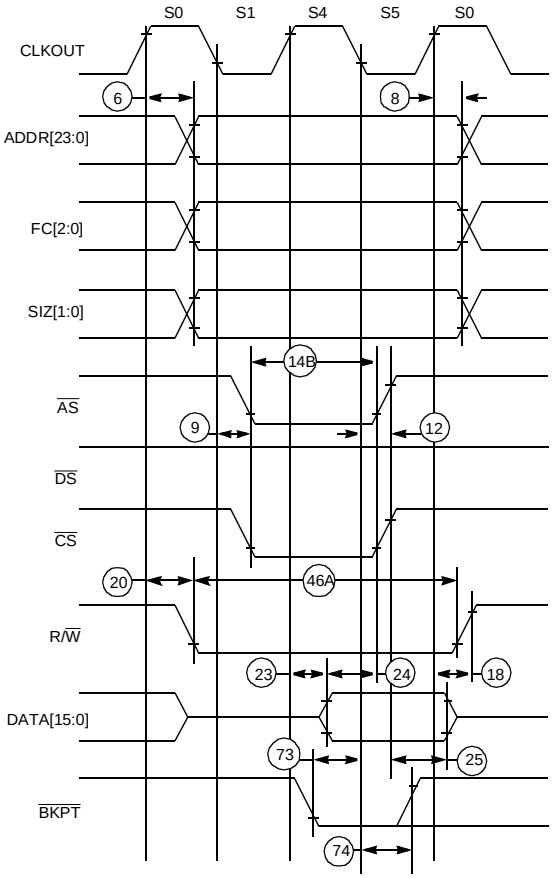


Figure E-7 Fast Termination Write Cycle Timing Diagram

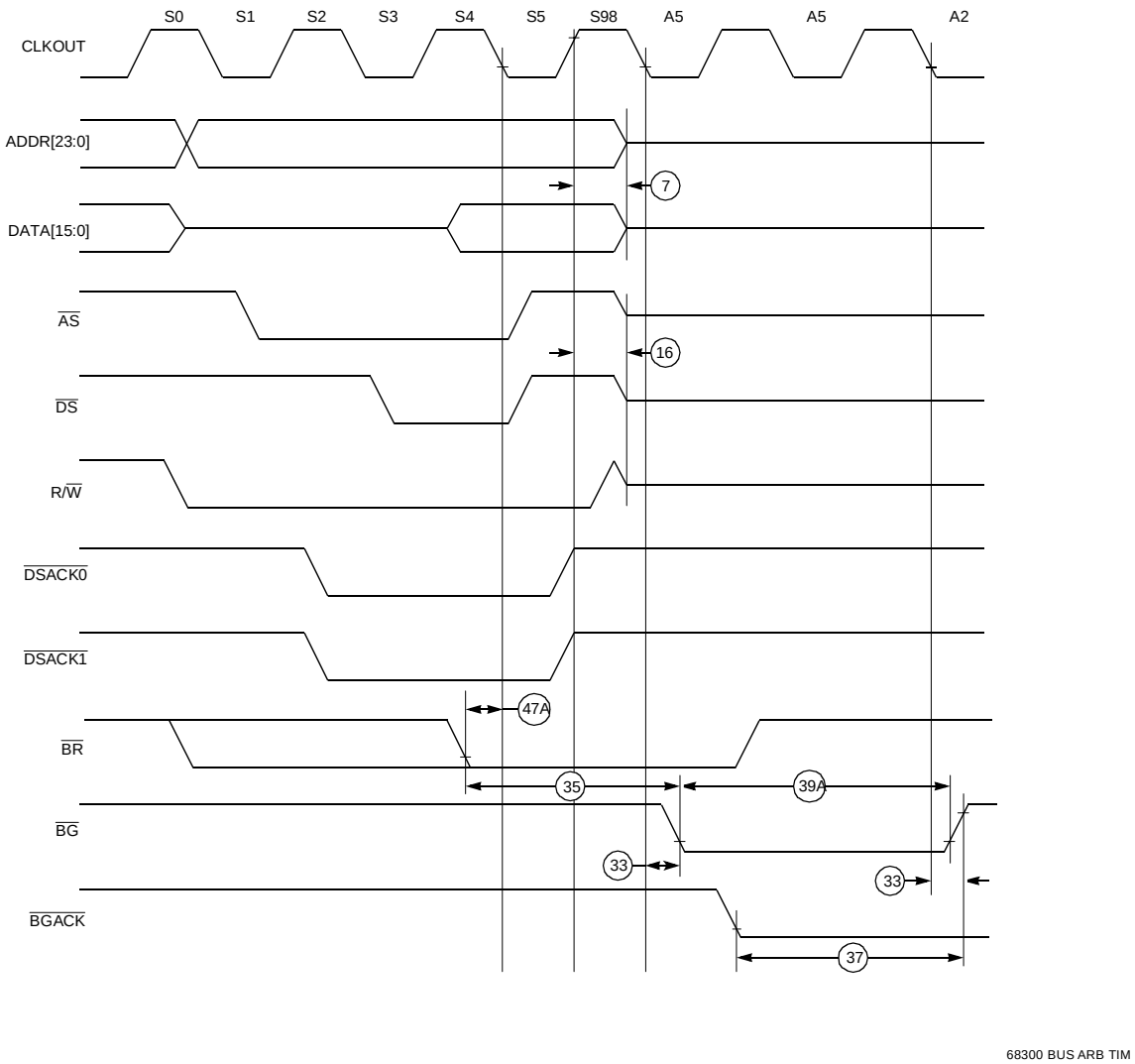


Figure E-8 Bus Arbitration Timing Diagram — Active Bus Case

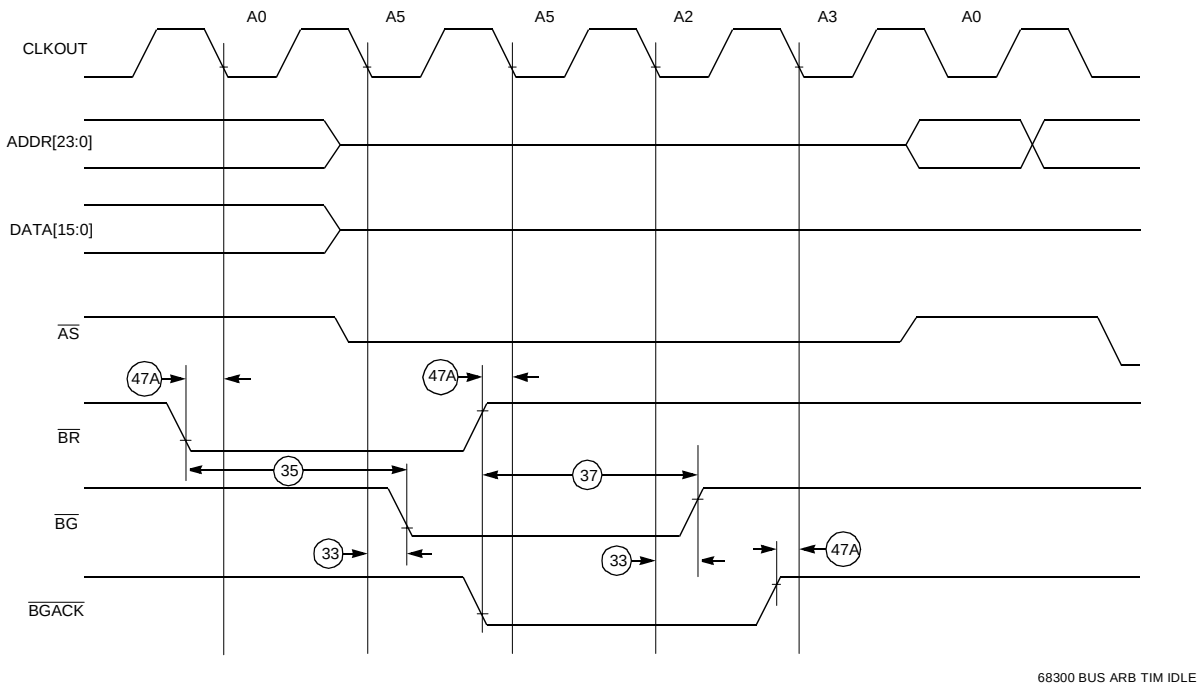
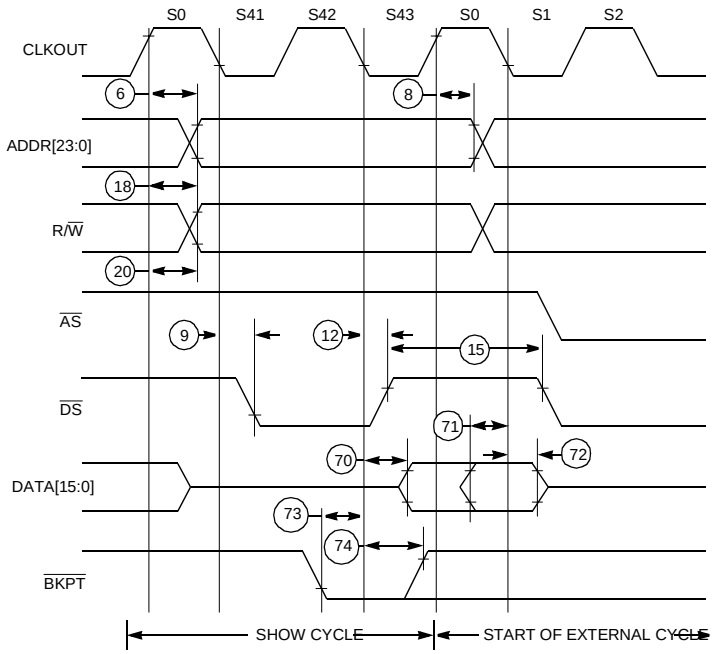


Figure E-9 Bus Arbitration Timing Diagram — Idle Bus Case



NOTE:
Show cycles can stretch during clock phase S42 when bus accesses take longer than two cycles due to IMB module wait-state insertion.

68300 SHW CYC TIM

Figure E-10 Show Cycle Timing Diagram

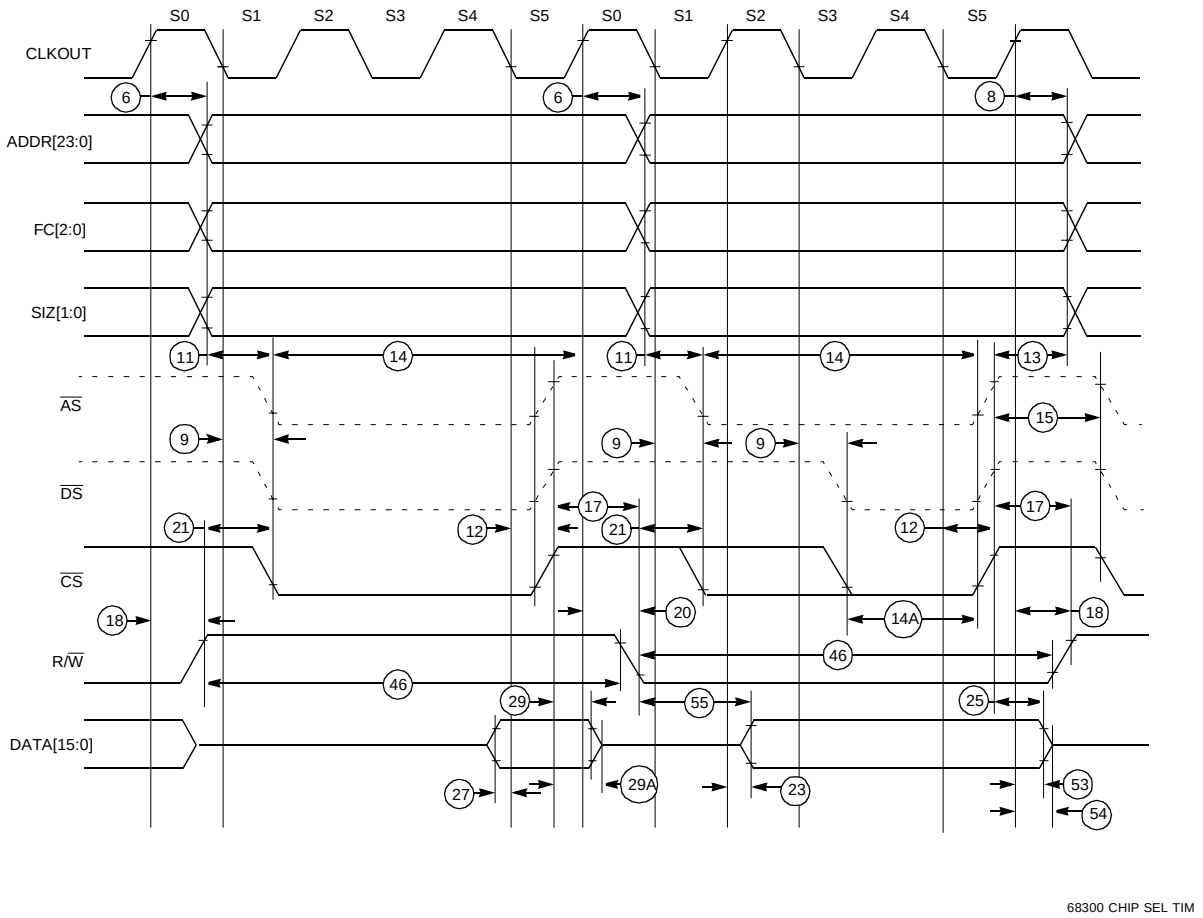


Figure E-11 Chip-Select Timing Diagram

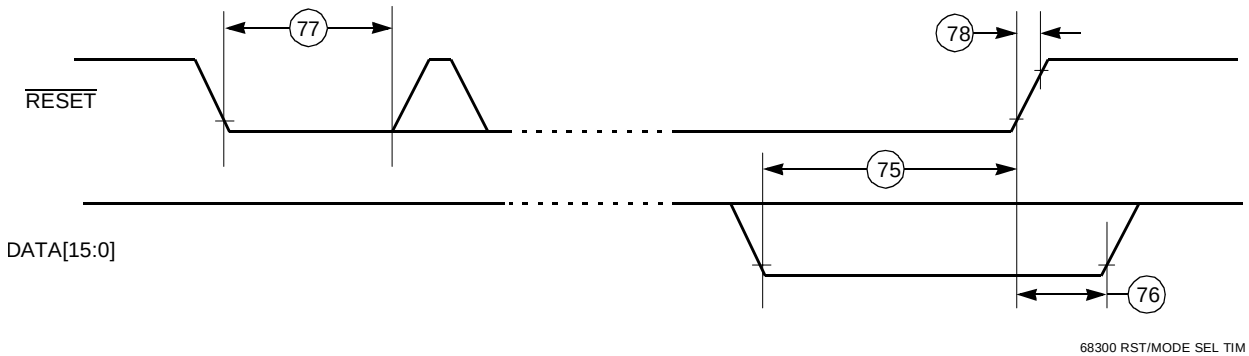


Figure E-12 Reset and Mode Select Timing Diagram

Table E-6 Background Debugging Mode Timing

(V_{DDH} = 5.0 Vdc ± 10%, V_{DDL} and V_{DDSYN} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
B0	DSI Input Setup Time	t_{DSISU}	8	—	ns
B1	DSI Input Hold Time	tDSIH	5	—	ns
B2	DSCLK Setup Time	tDSCSU	8	—	ns
B3	DSCLK Hold Time	tDSCH	5	—	ns
B4	DSO Delay Time	tDSOD	—	13	ns
B5	DSCLK Cycle Time	tDSCCYC	2	—	t_{cyc}
B6	CLKOUT High to FREEZE Asserted/Negated	tFRZAN	—	25	ns
B7	CLKOUT High to $\overline{\text{IFETCH}}$ High Impedance	tIFZ	—	25	ns
B8	CLKOUT High to $\overline{\text{IFETCH}}$ Valid	tIF	—	25	ns
B9	DSCLK Low Time	tDSCLO	1	—	t_{cyc}

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.

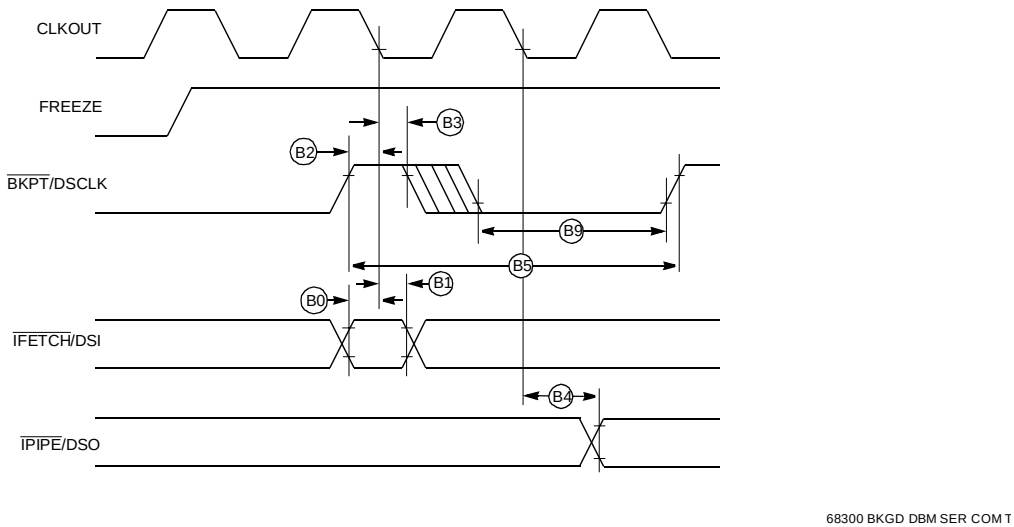


Figure E-13 Background Debugging Mode Timing — Serial Communication

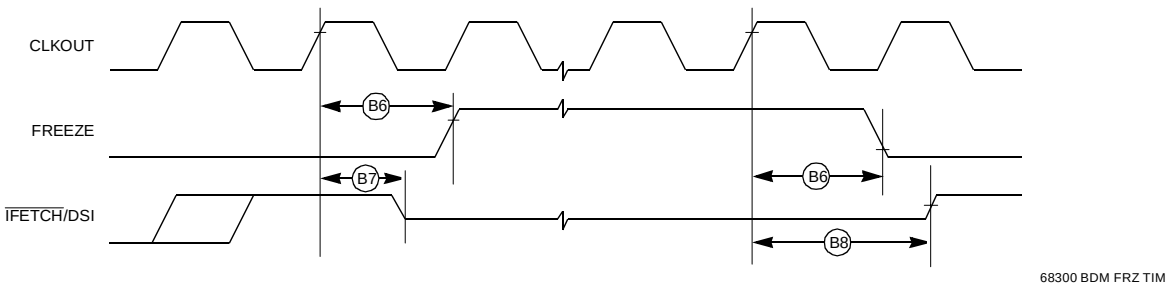


Figure E-14 Background Debugging Mode Timing — Freeze Assertion

Table E-7 ECLK Bus Timing

(V_{DDH} = 5.0 Vdc ± 10%, V_{DDL} and V_{DDSYN} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
E1	ECLK Low to Address Valid[2]	t_{EAD}	—	60	ns
E2	ECLK Low to Address Hold	t_{EAH}	10	—	ns
E3	ECLK Low to $\overline{CS}$ Valid ($\overline{CS}$ Delay)	t_{ECSD}	—	150	ns
E4	ECLK Low to $\overline{CS}$ Hold	t_{ECSH}	15	—	ns
E5	$\overline{CS}$ Negated Width	t_{ECSN}	30	—	ns
E6	Read Data Setup Time	t_{EDSR}	30	—	ns
E7	Read Data Hold Time	t_{EDHR}	15	—	ns
E8	ECLK Low to Data High Impedance	t_{EDHZ}	—	60	ns
E9	$\overline{CS}$ Negated to Data Hold (Read)	t_{ECDH}	0	—	ns
E10	$\overline{CS}$ Negated to Data High Impedance	t_{ECDZ}	—	1	t_{cyc}
E11	ECLK Low to Data Valid (Write)	t_{EDDW}	—	2	t_{cyc}
E12	ECLK Low to Data Hold (Write)	t_{EDHW}	5	—	ns
E13	$\overline{CS}$ Negated to Data Hold (Write)	t_{ECHW}	0	—	ns
E14	Address Access Time (Read)[3]	t_{EACC}	386	—	ns
E15	Chip-Select Access Time (Read)[4]	t_{EACS}	296	—	ns
E16	Address Setup Time	t_{EAS}		1/2	t_{cyc}

NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
2. When previous bus cycle is not an ECLK cycle, the address may be valid before ECLK goes low.
3. Address access time = $t_{ECYC} - t_{EAD} - t_{EDSR}$.
4. Chip-select access time = $t_{ECYC} - t_{ECSD} - t_{EDSR}$.

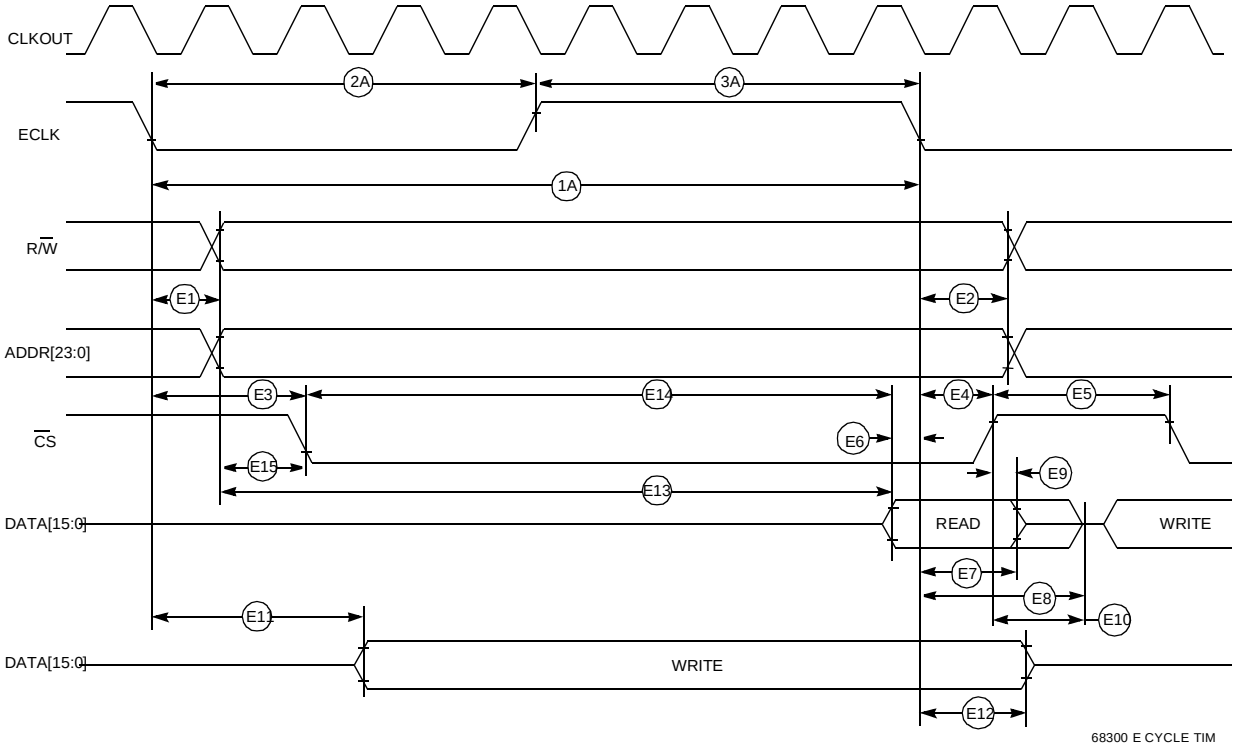


Figure E-15 ECLK Timing Diagram

E.6 QSPI Characteristics

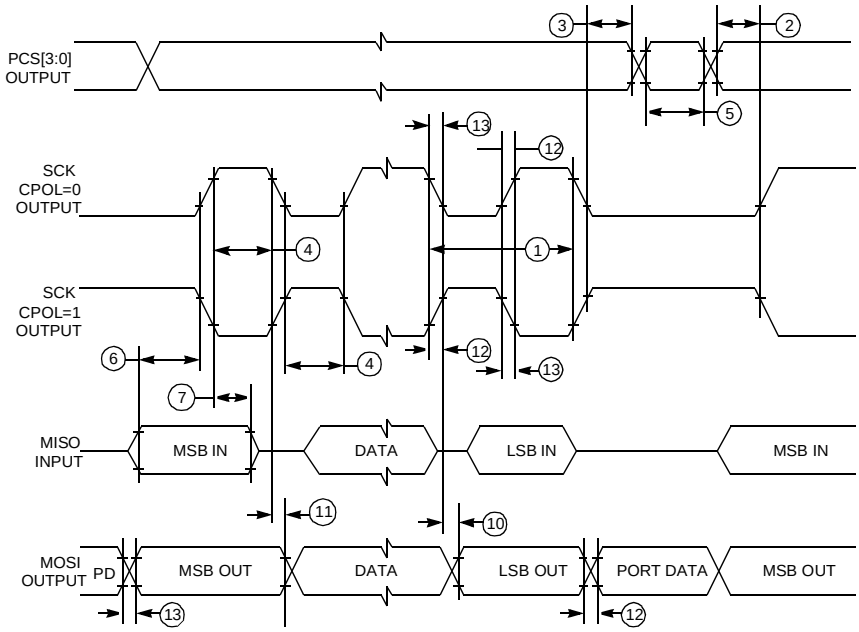
Table E-8 QSPI Timing

(V_{DDH} = 5.0 Vdc ± 10%, V_{DDSYN} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H 50 pF load on all QSPI pins)[1]

Num	Function	Symbol	Min	Max	Unit
0	Operating Frequency Master Slave	f_{op}	 DC DC	 1/4 1/4	 System Clock Frequency System Clock Frequency
1	Cycle Time Master Slave	t_{qcyc}	 4 4	 510 —	 t_{cyc} t_{cyc}
2	Enable Lead Time Master Slave	t_{lead}	 2 2	 128 —	 t_{cyc} t_{cyc}
3	Enable Lag Time Master Slave	t_{lag}	 — 2	 1/2 —	 SCK t_{cyc}
4	Clock (SCK) High or Low Time Master Slave[2]	t_{sw}	 2 tcyc – 60 2 tcyc – n	 255 tcyc —	 ns ns
5	Sequential Transfer Delay Master Slave (Does Not Require Deselect)	t_{td}	 17 13	 8192 —	 t_{cyc} t_{cyc}
6	Data Setup Time (Inputs) Master Slave	t_{su}	 30 20	 — —	 ns ns
7	Data Hold Time (Inputs) Master Slave	t_{hi}	 0 20	 — —	 ns ns
8	Slave Access Time	t_a	—	1	t_{cyc}
9	Slave MISO Disable Time	t_{dis}	—	2	t_{cyc}
10	Data Valid (after SCK Edge) Master Slave	t_v	 — —	 50 50	 ns ns
11	Data Hold Time (Outputs) Master Slave	t_{ho}	 0 0	 — —	 ns ns
12	Rise Time Input Output	 t_{ri} t_{ro}	 — —	 1 50	 µs ns
13	Fall Time Input[3] Output	 t_{fi} t_{fo}	 — —	 1 50	 µs ns

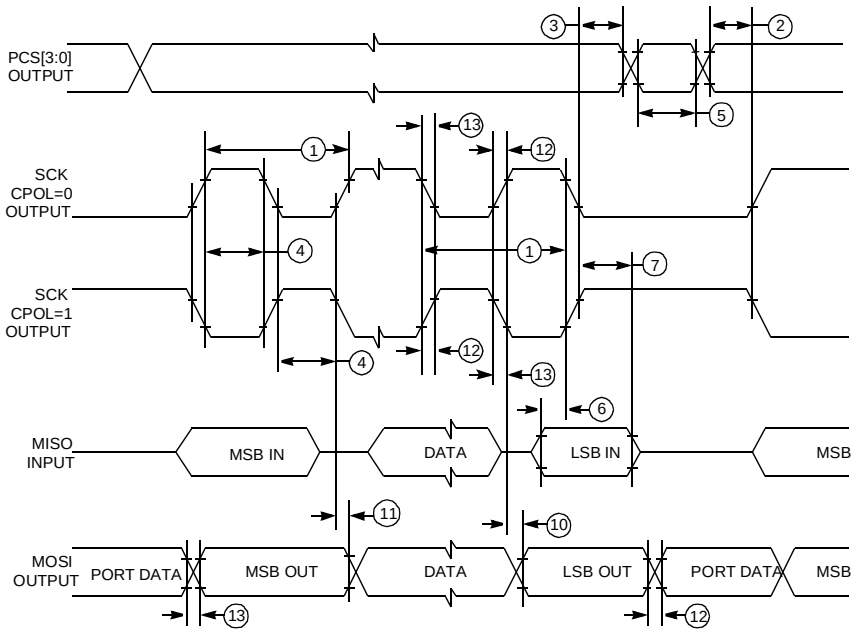
NOTES:

1. All AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels unless otherwise noted.
2. For high time, n = External SCK rise time; for low time, n = External SCK fall time.
3. Data can be recognized properly with longer transition times as long as MOSI/MISO signals from external sources are at valid VOH/VOL prior to SCK transitioning between valid VOL and VOH. Due to process variation, logic decision point voltages of the data and clock signals can differ, which can corrupt data if slower transition times are used.



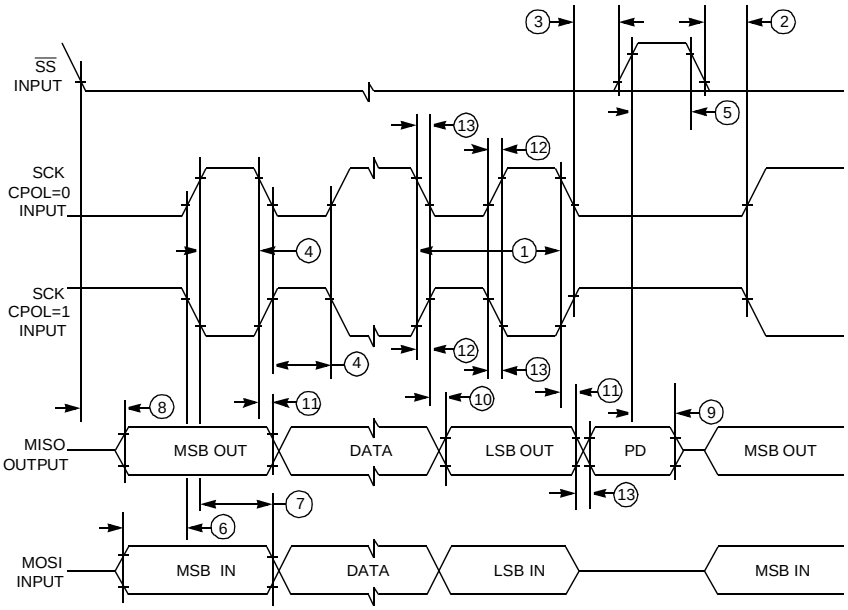
QSPI MAST CPHA0

Figure E-16 QSPI Timing — Master, CPHA = 0



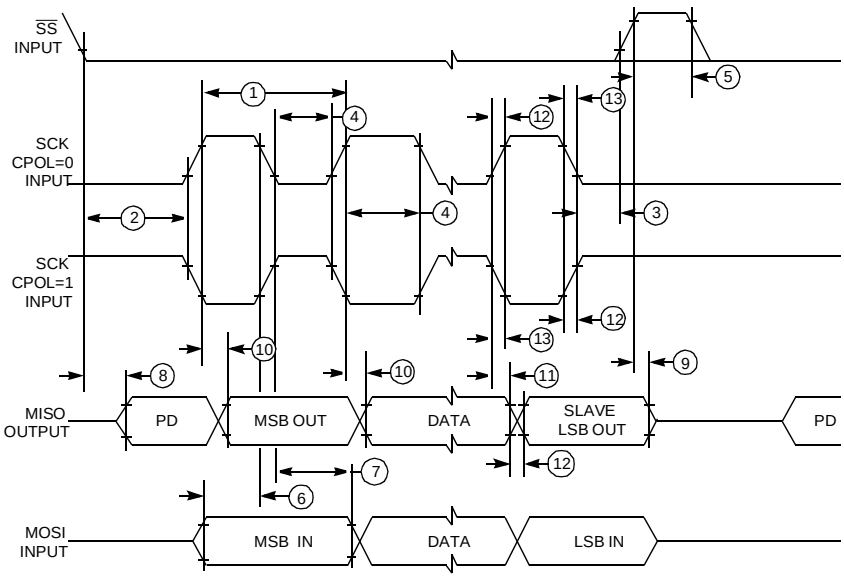
QSPI MAST CPHA1

Figure E-17 QSPI Timing — Master, CPHA = 1



QSPI SLV CPHA0

Figure E-18 QSPI Timing — Slave, CPHA = 0



QSPI SLV CPHA1

Figure E-19 QSPI Timing — Slave, CPHA = 1

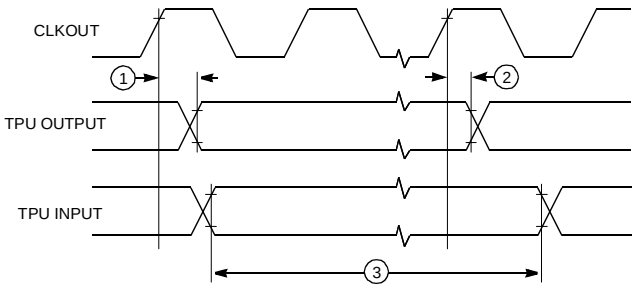
E.7 TPU3 Characteristics

Table E-9 Time Processor Unit (TPU3) Timing

(V_{DDH} = 5.0 Vdc ± 10%, V_{DDSYN} = 3.3 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, f_{sys} = 20.97 MHz)[1, 2]

Num	Rating	Symbol	Min	Max	Unit
1	CLKOUT High to TPU3 Output Channel Valid	t_{CHTOV}	2	23	ns
2	CLKOUT High to TPU3 Output Channel Hold	t_{CHTOH}	0	20	ns
3	TPU3 Input Channel Pulse Width	t_{TIPW}	4	—	t_{cyc}

NOTES:
1. AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels.
2. Timing not valid for external T2CLK input.



TPU I/O TIM

Figure E-20 TPU Timing Diagram

E.8 QADC64 and AMUX Characteristics

Table E-10 QADC64 Maximum Ratings

Num	Parameter	Symbol	Min	Max	Unit
1	Analog Supply, with reference to V_{SSA}	V_{DDA}	– 0.3	6.0	V
2	Digital Supply, with reference to V_{SS}	V_{DDL}	– 0.3	4.0	V
3	Reference Supply, with reference to V_{RL}	V_{RH}	– 0.3	6.0	V
4	V_{SS} Differential Voltage	$V_{SS} – V_{SSA}$	– 0.1	0.1	V
5	V_{DD} Differential Voltage	$V_{DDL} – V_{DDA}$	– 6.0	4.0	V
6	V_{REF} Differential Voltage	$V_{RH} – V_{RL}$	– 0.3	6.0	V
7	V_{RH} to V_{DDA} Differential Voltage	$V_{RH} – V_{DDA}$	– 6.0	6.0	V
8	V_{RL} to V_{SSA} Differential Voltage	$V_{RL} – V_{SSA}$	– 0.3	0.3	V
9	Disruptive Input Current[1, 2, 3, 4, 5, 6, 7] $V_{NEGCLAMP}$ = -0.3 $V_{POSCLAMP}$ = TBD (8 – 12 V)	I_{NA}	– 500	500	µA
10	Positive Overvoltage Current Coupling Ratio[1, 5, 6, 8] PQA PQB	K_P	2000 2000	— —	—
11	Negative Overvoltage Current Coupling Ratio[1, 5, 6, 8] PQA PQB	K_N	125 500	— —	—
12	Maximum Input Current[3, 4, 6] $V_{NEGCLAMP}$ = -0.3 V $V_{POSCLAMP}$ = TBD (8 – 12 V)	I_{MA}	– 25	25	mA

NOTES:

1. Below disruptive current conditions, the channel being stressed has conversion values of $3FF for analog inputs greater than V_{RH} and $000 for values less than V_{RL}. This assumes that $V_{RH} \leq V_{DDA}$ and $V_{RL} \geq V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions.
2. Input signals with large slew rates or high frequency noise components cannot be converted accurately. These signals also affect the conversion accuracy of other channels.
3. Exceeding limit may cause conversion error on stressed channels and on unstressed channels. Transitions within the limit do not affect device reliability or cause permanent damage.
4. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values using positive and negative clamp values, then use the larger of the calculated values.
5. This parameter is periodically sampled rather 100% tested.
6. Condition applies to one pin at a time.
7. Determination of actual maximum disruptive input current, which can affect operation, is related to external system component values.
8. Current coupling is the ratio of the current induced from overvoltage (positive or negative, through an external series coupling resistor), divided by the current induced on adjacent pins. A voltage drop may occur across the external source impedances of the adjacent pins, impacting conversions on these adjacent pins.

Table E-11 QADC64 DC Electrical Characteristics (Operating)

(V_{SS} and V_{SSA} = 0 Vdc, F_{QCLK} = 2.1 MHz, T_A within operating temperature range)

Num	Parameter	Symbol	Min	Max	Unit
1	Analog Supply[1]	V_{DDA}	4.5	5.5	V
2	Digital Supply[1]	V_{DD}	3.0	3.6	V
3	V_{SS} Differential Voltage	$V_{SS} - V_{SSA}$	– 100	100	mV
4	V_{DD} Differential Voltage	$V_{DD} - V_{DDA}$	– 1.0	1.0	V
5	Reference Voltage Low[2]	V_{RL}	V_{SSA}	V_{SSA} +0.1	V
6	Reference Voltage High[2]	V_{RH}	V_{DDA} - 0.1	V_{DDA}	V
7	V_{REF} Differential Voltage[3]	V_{RH} - V_{RL}	4.5	5.5	V
8	Mid-Analog Supply Voltage	V_{DDA} / 2	2.25	2.75	V
9	Input Voltage	V_{INDC}	V_{SSA}-0.3	V_{DDA}+0.3	V
10	Input High Voltage, PQA and PQB	V_{IH}	0.7 (V_{DDA})	V_{DDA} + 0.3	V
11	Input Low Voltage, PQA and PQB	V_{IL}	V_{SSA} – 0.3	0.4(V_{DDA})	V
12	Input Hysteresis[4]	V_{HYS}	0.5	—	V
13	Output Low Voltage, PQA[5] IOL = 5.0 mA IOL = 10.0 µA	V_{OL}	 — —	 0.4 0.2	V
14	Analog Supply Current Normal Operation[6] Low-Power Stop	I_{DDA}	 — —	 15.0 10.0	 mA µA
15	Reference Supply Current	I_{REF}	—	250	µA
16	Load Capacitance, PQA	C_L	—	50	pF
17	Input Current, Channel Off[7] PQA PQB	I_{OFF}	 — —	 200 150	nA
18	Total Input Capacitance[8] PQA Not Sampling PQA Sampling PQB Not Sampling PQB Sampling	C_{IN}	 — — — —	 15 20 10 15	pF

NOTES:

1. Refers to operation over full temperature and frequency range.
2. To obtain full-scale, full-range results, $V_{SSA} \le V_{RL} \le V_{INDC} \le V_{RH} \le V_{DDA}$.
3. Accuracy tested and guaranteed at V_{RH} - V_{RL} = 5.0V ± 10%.
4. Parameter applies to the following pins:
   Port A: PQA[7:0]/AN[59:58]/ETRIG[2:1]
   Port B: PQB[7:0]/AN[3:0]/AN[51:48]/AN[Z:W]
5. Open drain only.
6. Current measured at maximum system clock frequency with QADC64 active.
7. Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each 10° C decrease from maximum temperature.
8. This parameter is periodically sampled rather than 100% tested.

Table E-12 QADC64 AC Electrical Characteristics (Operating)

(V_{DDL} = 3.3 Vdc ± 10%, V_{DDH} and V_{DDA} = 5.0 Vdc ± 10%, V_{SS} and V_{SSA} = 0Vdc, T_A within operating temperature range)

Num	Parameter	Symbol	Min	Max	Unit
1	QADC64 Clock (QCLK) Frequency[1]	F_{QCLK}	0.5	2.1	MHz
2	QADC64 Clock Duty Cycle[2, 3] High Phase Time ($T_{PSL} \leq T_{PSH}$)	T_{PSH}	500	—	ns
3	Conversion Cycles[4]	CC	20	32	QCLK cycles
4	Conversion Time[4, 4, 5] F_{QCLK} = 0.987 MHz[6] Min = CCW/IST =%00 Max = CCW/IST =%11 F_{QCLK} = 2.098 MHz[1, 7] Min = CCW/IST =%00 Max = CCW/IST =%11	T_{CONV}	18.0 8.58	32 15.24	µs
5	Stop Mode Recovery Time	T_{SR}	—	10	µs

NOTES:

1. Conversion characteristics vary with F_{QCLK} rate. Reduced conversion accuracy occurs at max F_{QCLK} rate.
2. Duty cycle must be as close as possible to 75% to achieve optimum performance.
3. Minimum applies to 1.0 MHz operation.
4. Assumes that short input sample time has been selected (IST = 0).
5. Assumes that f_{sys} = 16.78 MHz.
6. Assumes F_{QCLK} = 0.987 MHz, with clock prescaler values of:
   QACR0: PSH =%01100, PSA =%0, PSL = %011
   CCW: BYP =%0
7. Assumes F_{QCLK} = 2.098 MHz, with clock prescaler values of:
   QACR0: PSH =%00101, PSA =%0, PSL = %001
   CCW: BYP =%0

Table E-13 QADC64 Conversion Characteristics (Operating)

(V_{DDL}= 3.3 Vdc ± ± 10%, V_{DDH}, V_{DDA} = 5.0 Vdc ± 10%, V_{SS} and V_{SSA} = 0 Vdc, T_A within operating temperature range0.5 MHz ≤ F_{QCLK} ≤ 2.1 MHz, 2 clock input sample time)

Num	Parameter	Symbol	Min	Typ	Max	Unit
1	Resolution[1]	1 Count	—	5	—	mV
3	Integral nonlinearity	INL	—	—	±2.0	Counts
4	Absolute error[2,3,4] F_{QCLK} = 0.987MHz[5] PQA PQB F_{QCLK} = 2.098 MHz[6] PQA PQB	AE	 — — — —	 — — — —	 ±2.5 ±2.5 ±4.0 ±4.0	Counts
5	Source impedance at input[7]	R_S	—	100	—	KΩ

NOTES:

1. At V_{RH} - V_{RL} = 5.12 V, one count = 5 mV.
2. This parameter is periodically sampled rather that 100% tested.
3. Absolute error includes 1/2 count (2.5mV) of inherent quantization error and circuit (differential, integral, and offset) error. Specification assumes that adequate low-pass filtering is present on analog input pins — capacitive filter with 0.01µF to 0.1 µF capacitor between analog input and analog ground, typical source isolation impedance of 20 KΩ.
4. Assumes f_{sys} = 16.78 MHz.
5. Assumes clock prescaler values of:
   QACR0: PSH =%01100, PSA =%0, PSL = %011
   CCW: BYP =%0
6. Assumes clock prescaler values of:
   QACR0: PSH =%00101, PSA =%0, PSL = %001
   CCW: BYP =%0
7. Maximum source impedance is application-dependent. Error resulting from pin leakage depends on junction leakage into the pin and on leakage due to charge-sharing with internal capacitance.Error from junction leakage is a function of external source impedance and input leakage current. In the following expression, expected error in result value due to junction leakage is expressed in voltage (V_{errj}):

   $$V_{errj} = R_S \times I_{OF}$$

   where IOFF is a function of operating temperature. (See **Table E-11, note 7**).
   Charge-sharing leakage is a function of input source impedance, conversion rate, change in voltage between successive conversions, and the size of the decoupling capacitor used. Error levels are best determined empirically. In general, continuous conversion of the same channel may not be compatible with high source impedance.

Table E-14 AMUX_HV Absolute Maximum Ratings

Num	Parameter	Symbol	Min	Max	Unit
1	Analog Supply, with reference to V_{SSA}	V_{DDA}	– 0.3	6.0	V
2	Internal Digital Supply[1], with reference to V_{SSI}	V_{DDI}	– 0.3	4.0	V
3	V_{SS} Differential Voltage	$V_{SSI} – V_{SSA}$	– 0.1	0.1	V
4	V_{DD} Differential Voltage[2]	$V_{DDI} – V_{DDA}$	– 6.0	4.0	V
5	Maximum Input Current[3,4,5]	I_{MA}	– 25	25	mA

NOTES:

1. For internal digital supply of V_{DDI} = 3.3 V typical.
2. Refers to allowed random sequencing of power supplies.
3. Transitions within the limit do not affect device reliability or cause permanent damage. Exceeding limit may cause permanent conversion error on stressed channels and on unstressed channels.
4. Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values using $V_{POSCLAMP} = V_{DDA}$ + 0.3 V and $V_{NEGCLAMP}$ = – 0.3 V, then use the larger of the calculated values.
5. Condition applies to one pin at a time.

Table E-15 AMUX_HV DC Electrical Characteristics (Operating)

(V_{SSI} and V_{SSA} = 0 Vdc, F_{QCLK} = 2.0 MHz, T_A within operating temperature range)

Num	Parameter	Symbol	Min	Max	Unit
1	Analog Supply[1]	V_{DDA}	4.5	5.5	V
2	Internal Digital Supply[1]	V_{DDI}	3.0	3.6	V
3	V_{SS} Differential Voltage	$V_{SSI} – V_{SSA}$	– 100	100	mV
4	Input Voltage	V_{INDC}	V_{SSA}	V_{DDA}	V
5	Analog Supply Current[2]	I_{DDA}	—	10	µA
6	Input Current, Channel Off[3]	I_{OFF}	– 200nA	200	nA
7	Total Input Capacitance[4] ANXx Not Sampling ANXx Sampling[5]	C_{IN}	— —	15 20	pF
8	Disruptive Input Injection Current[6]	I_{INJ}	– 3	+3	mA

NOTES:

1. Refers to operation over full temperature and frequency range.
2. VDDA is used only for a pin stress protection circuit. It draws DC current only when the pin voltage is above VDDA. It draws a minimal capacitive switching current when the channel is changed.
3. Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each 10° C decrease from maximum temperature.
4. This parameter is periodically sampled rather than 100% tested.
5. Sampling adds an incremental capacitance of roughly 5pF.
6. Below disruptive current conditions, the channel being stressed has conversion values of 0x3FF for analog inputs greater than V_{RH} and 0x000 for values less than V_{RL}. This assumes that $V_{RH} \leq V_{DDA}$ and $V_{RL} \geq V_{SSA}$ due to the presence of the sample amplifier. Other channels are not affected by non-disruptive conditions.

Table E-16 AMUX_HV Conversion Characteristics (Operating)

(V_{DDI} = 3.3 Vdc ± 0.3 V, V_{DDA} = 5.0 Vdc ± 10%, V_{SSI} and V_{SSA} = 0 Vdc, T_A within operating temperature range, 0.5 MHz ≤ F_{QCLK} ≤ 2.1 MHz, 4[1] clock input sample time, AMUX/QADC64 configured for internal multiplexing only.)

Num	Parameter	Symbol	Min	Typ	Max	Unit
1	Source impedance at input[2,3]	R_S	—	7.5	—	KΩ

NOTES:

1. Input sample time was increased from the 2 clocks used on channels connected directly to the QADC64 (AN0–AN4, AN48–AN59) to 4 on extended channels (ANX0–ANX31). This is to compensate for additional series resistance of the AMUX. It is possible that this can change back to 2 after characterization.
2. Maximum source impedance is application-dependent. Error resulting from pin leakage depends on junction leakage into the pin and on leakage due to charge-sharing with internal capacitance.
   Error from junction leakage is a function of external source impedance and input leakage current. In the following expression, expected error in result value due to junction leakage is expressed in voltage (V_{errj}):
   $$V_{errj} = R_S * I_{OFF}$$
   where I_{OFF} is a function of operating temperature. (See **Table E-15**, Note **3.**).
   Charge-sharing leakage is a function of input source impedance, conversion rate, change in voltage between successive conversions, and the size of the filtering capacitor used. Error levels are best determined empirically. Charge-sharing leakage for AMUX inputs will be larger due to the increased internal parasitic capacitances. In general, continuous conversion of the same channel may not be compatible with high source impedance.
3. AMUX/QADC64 configured for internal multiplexing only.

E.9 CTM9 Electrical Characteristics

The electrical and timing characteristics of the CTM9 are the same as those of other MC68300/MC68HC16 family MCUs.

E.9.1 5 V, 16.78 MHz Operating Conditions

Table E-17 CTM9 5 V, 16.78 MHz Operating Conditions

	Condition	Symbol	Min	Typ	Max	Units
Operating Temperature		Temp	-40		+125	°C
Main Supply Voltage		V_{DD}	4.50	5.00	5.50	V
Operating Frequency	Main Clock	F_{clk}	0		16.78	MHz

E.9.2 5 V, 25.0 MHz Operating Conditions

Table E-18 CTM9 5 V, 25 MHz Operating Conditions

	Condition	Symbol	Min	Typ	Max	Units
Operating Temperature		Temp	-40		+125	°C
Main Supply Voltage		V_{DD}	4.50	5.00	5.50	V
Operating Frequency	Main Clock	F_{clk}	0		25.0	MHz

Table E-19 FCSM Timing Characteristics

(V_{DDL} = 3.3 Vdc ± 10%, V_{DDH} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H)

Num	Parameter	Symbol	Min	Max	Unit
1	Input pin frequency[1]	f_{PCNTR}	0	$f_{sys}/4$	MHz
2	Input pin low time[1]	t_{PINL}	$2.0/f_{sys}$	—	µs
3	Input pin high time[1]	t_{PINH}	$2.0/f_{sys}$	—	µs
4	Clock pin to counter increment	t_{PINC}	$4.5/f_{sys}$	$6.5/f_{sys}$	µs
5	Clock pin to new TBB value	t_{PTBB}	$5.0/f_{sys}$	$7.0/f_{sys}$	µs
6	Clock pin to COF set ($FFFF)	t_{PCOF}	$4.5/f_{sys}$	$6.5/f_{sys}$	µs
7	Pin to IN bit delay	t_{PINB}	$1.5/f_{sys}$	$2.5/f_{sys}$	µs
8	Flag to IMB interrupt request	t_{FIRQ}	$1.0/f_{sys}$	$1.0/f_{sys}$	µs
9	Counter resolution[2]	t_{CRES}	—	$2.0/f_{sys}$	µs

NOTES:
1. Value applies when using external clock.
2. Value applies when using internal clock. Minimum counter resolution depends on prescaler divide ratio selection.

Table E-20 MCSM Timing Characteristics

(V_{DDL} = 3.3 Vdc ± 10%, V_{DDH} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H)

Num	Parameter	Symbol	Min	Max	Unit
1	Input pin frequency[1]	f_{PCNTR}	0	$f_{sys}/4$	MHz
2	Input pin low time[1]	t_{PINL}	$2.0/f_{sys}$	—	µs
3	Input pin high time[1]	t_{PINH}	$2.0/f_{sys}$	—	µs
4	Clock pin to counter increment	t_{PINC}	$4.5/f_{sys}$	$6.5/f_{sys}$	µs
5	Clock pin to new TBB value	t_{PTBB}	$5.0/f_{sys}$	$7.0/f_{sys}$	µs
6	Clock pin to COF set ($FFFF)	t_{PCOF}	$4.5/f_{sys}$	$6.5/f_{sys}$	µs
7	Load pin to new counter value	t_{PLOAD}	$2.5/f_{sys}$	$3.5/f_{sys}$	µs
8	Pin to IN bit delay	t_{PINB}	$1.5/f_{sys}$	$2.5/f_{sys}$	µs
9	Flag to IMB interrupt request	t_{FIRQ}	$1.0/f_{sys}$	$1.0/f_{sys}$	µs
10	Counter resolution[2]	t_{CRES}	—	$2.0/f_{sys}$	µs

NOTES:
1. Value applies when using external clock.
2. Value applies when using internal clock. Minimum counter resolution depends on prescaler divide ratio selection.

Table E-21 SASM Timing Characteristics

(V_{DDL} = 3.3 Vdc ± 10%, V_{DDH} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H)

Num	Parameter	Symbol	Min	Max	Unit
1	Input pin low time	t_{PINL}	$2.0/f_{sys}$	—	µs
2	Input pin high time	t_{PINH}	$2.0/f_{sys}$	—	µs
3	Input capture resolution[1]	t_{RESCA}	—	$2.0/f_{sys}$	µs
4	Pin to input capture delay	t_{PCAPT}	$2.5/f_{sys}$	$4.5/f_{sys}$	µs
5	Pin to FLAG set	t_{PFLAG}	$2.5/f_{sys}$	$4.5/f_{sys}$	µs
6	Pin to IN bit delay	t_{PINB}	$1.5/f_{sys}$	$2.5/f_{sys}$	µs
7	OCT output pulse	t_{OCT}	$2.0/f_{sys}$	—	µs
8	Compare resolution[1]	t_{RESCM}		$2.0/f_{sys}$	µs
9	TBB change to FLAG set	t_{CFLAG}	$1.5/f_{sys}$	$1.5/f_{sys}$	µs
10	TBB change to pin change[2]	t_{CPIN}	$1.5/f_{sys}$	$1.5/f_{sys}$	µs
11	Flag to IMB interrupt request[2]	t_{FIRQ}	$1.0/f_{sys}$	$1.0/f_{sys}$	µs

NOTES:
1. Minimum resolution depends on counter and prescaler divide ratio selection.
2. Time given from when new value is stable on time base bus.

Table E-22 DASM Timing Characteristics

(V_{DDL} = 3.3 Vdc ± 10%, V_{DDH} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H)

Num	Parameter	Symbol	Min	Max	Unit
1	Input pin low time	t_{PINL}	$2.0/f_{sys}$	—	µs
2	Input pin high time	t_{PINH}	$2.0/f_{sys}$	—	µs
3	Input capture resolution[1]	t_{RESCA}	—	$2.0/f_{sys}$	µs
4	Pin to input capture delay	t_{PCAPT}	$2.5/f_{sys}$	$4.5/f_{sys}$	µs
5	Pin to FLAG set	t_{PFLAG}	$2.5/f_{sys}$	$4.5/f_{sys}$	µs
6	Pin to IN bit delay	t_{PINB}	$1.5/f_{sys}$	$2.5/f_{sys}$	µs
7	OCT output pulse	t_{OCT}	$2.0/f_{sys}$	—	µs
8	Compare resolution[1]	t_{RESCM}	—	$2.0/f_{sys}$	µs
9	TBB change to FLAG set	t_{CFLAG}	$1.5/f_{sys}$	$1.5/f_{sys}$	µs
10	TBB change to pin change[2]	t_{CPIN}	$1.5/f_{sys}$	$1.5/f_{sys}$	µs
11	Flag to IMB interrupt request[2]	t_{FIRQ}	$1.0/f_{sys}$	$1.0/f_{sys}$	µs

NOTES:
1. Minimum resolution depends on counter and prescaler divide ratio selection.
2. Time given from when new value is stable on time base bus.

Table E-23 PWMSM Timing Characteristics

(V_{DDL} = 3.3 Vdc ± 10%, V_{DDH} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)

Num	Parameter	Symbol	Min	Max	Unit
1	PWMSM output resolution[1]	t_{PWMR}	—	—	µs
2	PWMSM output pulse[2]	t_{PWMO}	$2.0/f_{sys}$	—	µs
3	PWMSM output pulse[3]	t_{PWMO}	$2.0/f_{sys}$	$2.0/f_{sys}$	µs
4	CPSM enable to output set PWMSM enabled before CPSM, DIV23 = 0 PWMSM enabled before CPSM, DIV23 = 1	t_{PWMP}	$3.5/f_{sys}$ $6.5/f_{sys}$	—	µs
5	PWM enable to output set PWMSM enabled before CPSM, DIV23 = 0 PWMSM enabled before CPSM, DIV23 = 1	t_{PWME}	$3.5/f_{sys}$ $5.5/f_{sys}$	$4.5/f_{sys}$ $6.5/f_{sys}$	µs
6	FLAG to IMB interrupt request	t_{FIRQ}	$1.5/f_{sys}$	$2.5/f_{sys}$	µs

NOTES:
1. Minimum output resolution depends on counter and prescaler divide ratio selection.
2. Excluding the case where the output is always zero.
3. Excluding the case where the output is always zero.

E.10 TouCAN Characteristics

Table E-24 TouCAN AC Characteristics[1]

Num	Parameter	Symbol	Value	Unit
1	CNTX0 – Delay from ICLOCK	—	19	ns
2	CNRX0 – Set-up to rise ICLOCK	—	0	ns
3	TOUCAN serial (Tx, Rx pins) max frequency	—	1	MHz

NOTES:
1. Measured at: 4.5V, 150°C

E.11 CMFI FLASH Characteristics

Table E-25 CMFI Program and Erase Characteristics

(VDDF = 3.3V ± 0.3V, VPP= 4.75V to 5.25V, TA = T_L to T_H)[1]

Symbol	Meaning	Value			Units
		Minimum	Typical	Maximum	
E_{PULSE}	Number of erase pulses	8	8	27	Pulses
T_{ERASE}	Erase pulse time	98	100	102	mS
P_{PULSE}	Number of program pulses	—	800	48000[2]	Pulses
T_{PROG}	Program pulse time	48	50	256.5	µS

NOTES:
1. T_L is defined to be -40 °C and T_H is defined to be 125 °C.
2. The worse case condition for programming time is at T_A = -40 °C and VPP = 4.75 volts.

Table E-26 CMFI AC and DC Power Supply Characteristics

Symbol	Meaning	Value
V_{DDF}=V_{DDL}	Operating voltage Read, Program or Erase	3.0 V to 3.6 V
I_{DDF}	Operating current at 40.0MHz, V_{DDF} = 3.3V for a 256K-byte Read, Program or Erase operation Disabled	 50 mA maximum 5 mA maximum
V_{PP}	External Program or Erase voltage Read Program or Erase	 (V_{DDF} - 350 m V) to 5.5 V 5.0 V ± 5%
I_{PP}	External Program and Erase current Read, V_{PP} = 5V Program, V_{PP} = 5V Erase, V_{PP} = 5V	 <100 µA 30 mA maximum 30 mA maximum

Table E-27 CMFI FLASH EEPROM Module Life

Symbol	Meaning	Value
P/E Cycles[1]	Maximum number of Program/ Erase cycles[2] before failure.	100[3,4]
Retention	Data retention at average operating temperature of 85 °C.	10 years

1) A Program/Erase cycle is defined as switching the bits from 1 → 0 → 1.
2) Reprogramming of a CMFI array block prior to erase is not required.
3) Number of Program/Erase cycles to be adjusted pending characterization of production silicon.
4) Target failure rate at specified number of program/erase cycles of 2ppm pending characterization of production silicon.

E.11.1 EPEB0 Pin Timing

The EPEB0 pin uses a two-period clock synchronizer that switches the signal CMF_EPEE after two consecutive values of EPEB0 as shown in **Figure E-21**. At first the value of CMF_EPEE is unknown as the prior information for the EPEB0 pin is not provided. After two clocks with EPEB0 low the signal CMF_EPEE is low. One high or low clock of EPEB0 will not cause CMF_EPEE to switch. After two high or low clocks of EPEB0 CMF_EPEE will switch to the value of EPEB0.

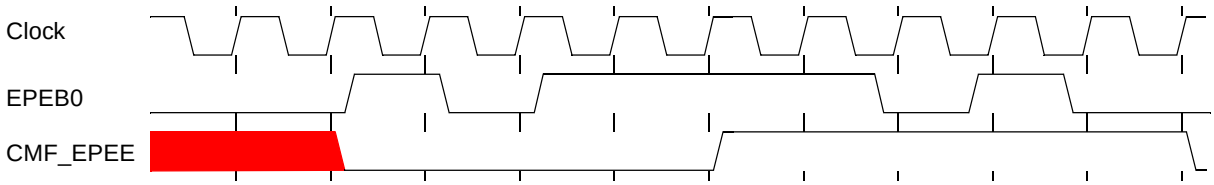


Figure E-21 EPEB0 Pin Timing

E.11.2 FLASH Program/Erase Voltage Conditioning

A voltage of at least V_{DDL} -0.35 V must be applied at all times to the V_{PP} pins or damage to the FLASH module can occur. FLASH modules can be damaged by power on and power off V_{PP} transients. V_{PP} must not rise to programming level while V_{DDL} is below the specified minimum value, and must not fall below the minimum specified value while V_{DDL} is applied. shows the V_{PP} and V_{DDL} operating envelope.

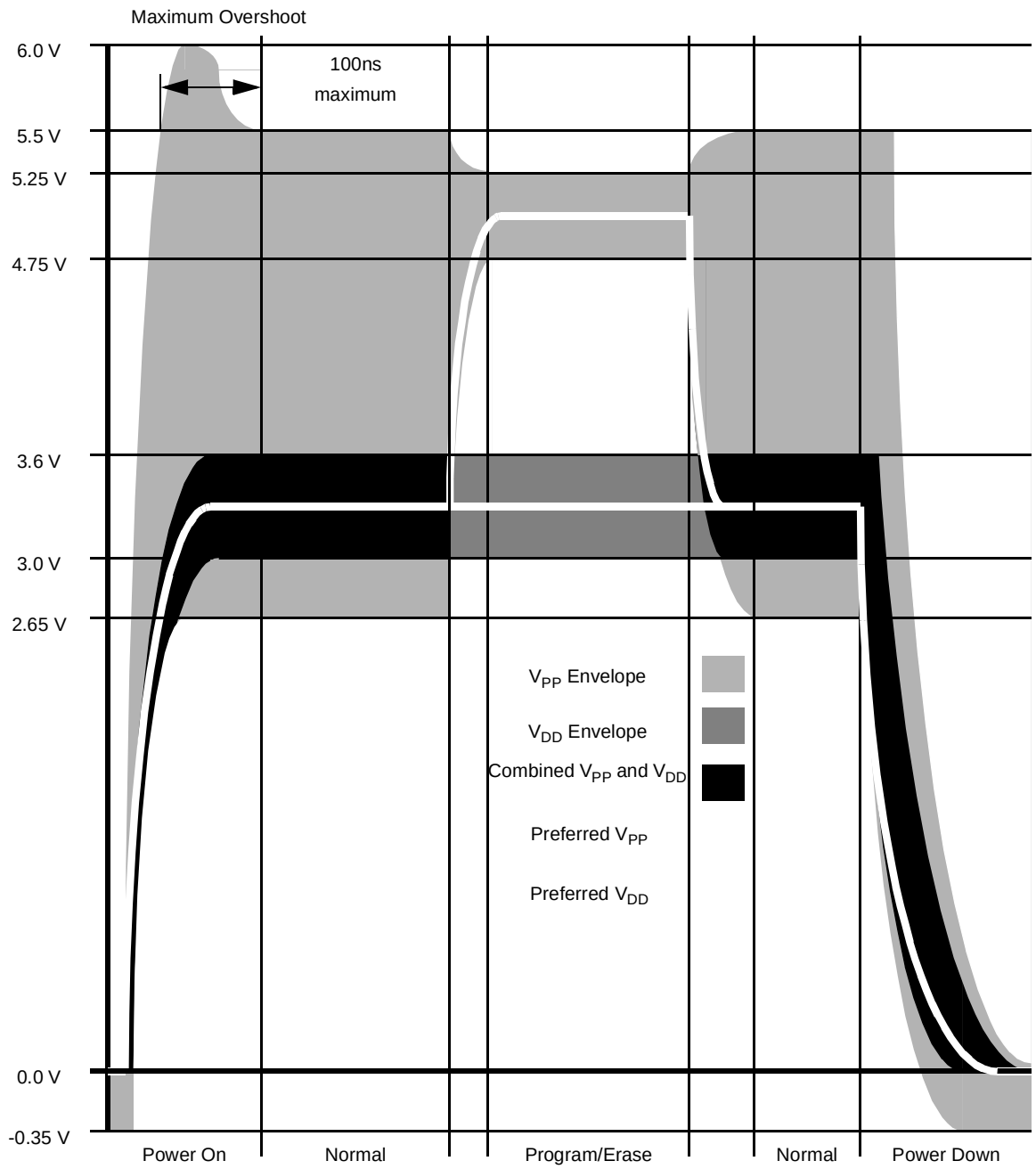


Figure E-22 V_{PP} and V_{DD} Power Sequencing

Use of an external circuit to condition V_{PP} is recommended. **Figure E-23** shows a simple circuit that maintains required voltages and filters transients. V_{PP} is pulled up to V_{DDL} via Schottky diode D2, protecting V_{DDL} from excessive reverse current. D2 also protects the FLASH from damage should the programming voltage go to zero. Programming power supply voltage must be adjusted to compensate for the forward-bias drop across D1. R1 provides a discharge bleed path for C1. Allow for RC charge and discharge time constants when applying and removing power. When using this circuit,

keep leakage from external devices connected to the VPP pin low, to minimize diode voltage drop.

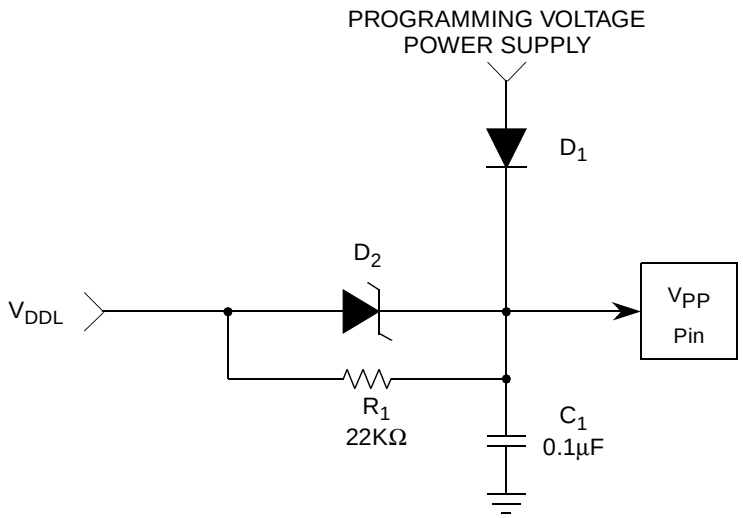


Figure E-23 A Recommended External V_{PP} Pin Conditioning Circuit

E.12 ROM Electrical Specifications

Table E-28 IMB3 ROM AC/DC Characteristics

Normal Operating Specifications		
V_{DDL}	Operating Voltage	+3.3V +/- 0.3V
f_{max}	Maximum Operating Frequency	33 MHz
T	Operating Temperature range	-40° to +125 °C

INDEX

–A–

ABD 4-61, 4-66, 4-67
ACKERR 7-31
Acknowledge error (ACKERR) 7-31
ADDR
  bus signals 4-32
  signal 4-37
Address
  bus (ADDR) 4-32
  bus disable (ABD) 4-61
  -mark wakeup 6-56
  space 5-7
    encoding 4-35
  strobe ($\overline{AS}$). *See* $\overline{AS}$ 4-32
AN 5-4, 5-5
Analog
  front-end multiplexer 5-14
  input
    multiplexed 5-5
    port A 5-4
    port B 5-4
  section contents 5-1
  submodule block diagram 5-12
  supply pins 5-6
Arbitration 6-6
Arbitration priorities 4-71
$\overline{AS}$ 4-42, 4-51
$\overline{AS}$ 4-49, 4-50
Asynchronous
  bus cycles classified as 4-39
  cycles 4-31
  exceptions 4-55
  mode 4-84
  resets 4-55
ATEMP 3-22
Autovector ($\overline{AVEC}$). *See* $\overline{AVEC}$ 4-25
$\overline{AVEC}$ 4-25, 4-36, 4-72, 4-73, 4-75, 4-78, 4-85

–B–

Background
  debug mode 3-19, 4-43
    commands 3-22
    connector pinout 3-26
    enabling 3-20
    entering 3-21
    registers
      fault address register (FAR) 3-23
      instruction program counter (PCC) 3-23
      return program counter (RPC) 3-23
    returning from 3-24
    serial
      data word 3-26
      I/O block diagram 3-25
      interface 3-24
      peripheral interface protocol (SPI) 3-25
    sources 3-20
  debugging mode
    freeze assertion diagram E-17
    serial
      communication diagram E-17
Base ID mask bits 7-30
Basic operand size 4-37
Baud
  clock 6-51
BCD 3-4
Beginning of queue 2 (BQ2) 5-41
Berg connector (male) 3-26
$\overline{BERR}$ 4-35, 4-39, 4-60, 4-64, 4-69, 4-72, 4-73, 4-88
$\overline{BERR}$ 4-43, 4-48, 4-49
  assertion results 4-47
$\overline{BG}$ 4-50, 4-65, 4-78
$\overline{BGACK}$ 4-50, 4-65, 4-78
BGND instruction 3-21
Binary
  -coded decimal (BCD) 3-4
  divider 5-25
  -weighted capacitors 5-14
Bit stuff error (STUFFERR) 7-31
BITERR 7-31
BITS 6-17
Bits per transfer
  enable (BITSE) 6-23
  field (BITS) 6-17
BITSE 6-23, 6-38
Bit-time 6-50
BIUMCR — BIUSM module configuration register 13-51
BIUSM
  BIUMCR — BIUSM module configuration register 13-51
  BIUTBR — BIUSM time base register 13-52
  freeze 13-50
  interrupt vector base number 13-51
  LPSTOP 13-50
  reading the time base bus 13-52
  STOP 13-50
  WAIT 13-50
BIUTBR — BIUSM time base register 13-52
$\overline{BKPT}$ 3-20, 4-68
  external signal 3-21
$\overline{BKPT}$ 4-43
BKPT (TPU asserted) 8-13

BLC 8-12
Block
size (BLKSZ) 4-78
encoding 4-79
block diagrams
CPSM 13-53
DASM 13-24
FCSM 13-5
MCSM 13-9
PWMSM 13-39
SASM 13-15
SASM channel A 13-16
BME 4-25
BMT 4-25
BOFFINT 7-32
Boundary conditions 5-17
BQ2 5-17, 5-41
$\overline{\text{BR}}$ 4-50, 4-65, 4-77, 4-78
$\overline{\text{BR}}$ 4-49
Branch latch control (BLC) 8-12
Break frame 6-51
Breakpoint
acknowledge cycle 4-43
asserted flag (BKPT) 8-13
flag (PCBK) 8-13
hardware breakpoints 4-43
instruction 3-19
mode selection 4-68
operation 4-45
software breakpoints 4-43
BSA 3-20
BSL
bit in SICA 13-21
Bus
arbitration
timing diagrams
active E-14
idle E-15
arbitration for a single device 4-51
cycle
regular 4-39
termination sequences 4-46
error
exception processing 4-48
signal ($\overline{\text{BERR}}$) 4-48
signal ($\overline{\text{BERR}}$). *See* $\overline{\text{BERR}}$ 4-25
timing of 4-48
grant ($\overline{\text{BG}}$). *See* $\overline{\text{BG}}$ 4-50
grant acknowledge ($\overline{\text{BGACK}}$). *See* $\overline{\text{BGACK}}$ 4-50
monitor 4-25
external enable (BME) 4-25
timeout period 4-25
timing (BMT) 4-25
off interrupt
(BOFFINT) 7-32
request ($\overline{\text{BR}}$). *See* $\overline{\text{BR}}$ 4-50
state analyzer (BSA) 3-20
BUSY 7-5, 7-15
BYP 5-13, 5-49
Bypass mode 5-13
BYTE (upper/lower byte option) 4-81, 4-83

–C–

C (carry) flag 3-6
CAN2.0B
system 7-3
CANCTRL0 7-26
CANCTRL1 7-27
CANCTRL2 7-29
CANICR 7-25
CCL 8-12
CCR 3-6, 4-71
CCW 5-1, 5-48
CF1 5-43
CF2 5-43
CFSR 8-15
CH 8-14, 8-17, 8-18
CHAN 5-49
changing mode
DASM 13-25
SASM 13-17
CHANNEL 8-15
Channel
assignments
multiplexed 5-50
nonmultiplexed 5-50
conditions latch (CCL) 8-12
interrupt
base vector (CIBV) 8-14
enable
/disable field (CH) 8-14
request level (CIRL) 8-14
status (CH) 8-18
invalid 5-49
number (CHAN) 5-49
orthogonality 8-3
priority registers 8-17
register breakpoint flag (CHBK) 8-13
reserved 5-49
CHBK 8-13
Chip-select
address compare logic 4-79
assertion 4-73
base address registers (CSBAR) 4-78
reset values 4-86
block (SCIM2) 4-1
circuit block diagram 4-75
operation 4-84
option registers (CSOR) 4-80
reset values 4-86
pin assignment registers (CSPAR) 4-75, 4-77
field encoding 4-77
reset operation 4-85
signals for interrupt acknowledge cycle termination 4-84
timing diagram E-16
CIBV 8-14
CIE1 5-39
CIE2 5-41
CIER 8-14, 8-18

CIRL 8-14
CISR 8-8, 8-18
CLKOUT 4-39, 4-55
    output timing diagram E-9
CLKS 8-12
Clock
    block diagram 5-26
    frequency 5-26
    generation 5-25
    output (CLKOUT) 4-39
    phase (CPHA) 6-17
    polarity (CPOL) 6-17
clocks
    FCSM 13-6
    MCSM 13-11
    PWMSM 13-40, 13-48
CMFI
    registers
        CMFI EEPROM configuration register (CMFIMCR) 10-9
CMFI EEPROM configuration register CMFIMCR 10-9
CNRX/TX pins 7-2
Code 7-5
Coherency 5-6, 5-22, 8-4
coherency
    DASM 13-25–13-28
    PWMSM 13-42
COMM D-17
Command
    RAM 6-22
    word pointer (CWP) 5-44
Common in-circuit emulator 3-20
Comparator 5-14
Completed queue pointer (CPTQP) 6-21
Condition code register (CCR) 3-6, 8-5
CONT 6-23
Continue (CONT) 6-23
Continuous transfer mode 6-15
Control registers
    0 (QACR0) 5-37
    1 (QACR1) 5-38
    2 (QACR2) 5-40
Conversion
    command word table (CCW) 5-1, 5-14
    cycle times 5-12
    stages 5-47
counter
    DASM 13-33
    FCSM 13-5
    MCSM 13-10
counter overflow
    FCSM 13-5
    MCSM 13-13
CPCR — CPSM control register 13-54
CPHA 6-17, 6-34
CPOL 6-17, 6-34
CPR 8-17
CPSM
    block diagram 13-53
    CPCR — CPSM control register 13-54
    freeze 13-53
    prescaler division ratio selection 13-54
CPTQP 6-21, 6-24
CPU space 4-84
    address encoding 4-43
    cycles 4-42, 4-84
    encoding for interrupt acknowledge 4-85
    read cycle 4-71
CPU16 4-86
CPU32 4-20, 4-26, 4-30, 4-37, 4-38, 4-46, 4-55, 4-68, 4-70, 4-79, 4-86
    address registers/address organization in 3-5
    addressing modes 3-9
    block diagram 3-2
    data registers 3-4
        data organization 3-5
    development support 3-18
    exception 4-55
    exception processing 3-16
    generated message encoding 3-26
    instructions 3-10
        LPSTOP 3-15
        MOVEC 3-7
        MOVES 3-7
        special control instructions 3-15
        table lookup and interpolate (TBL) 3-15
        umimplemented MC68020 instructions 3-10
    loop mode 3-15
    memory organization 3-7
    processing states 3-9
    reference manual 3-1
    register
        model 3-3
    registers 3-2
        alternate function code registers (SFC/DFC) 3-7
        condition code register (CCR) 3-6, 4-70
        control registers 3-6
        program counter (PC) 3-1
        stack pointer (SP) 3-1
        status register (SR) 3-6
        vector base register (VBR) 3-7
    virtual memory 3-9
CPUD 4-66, 4-67
CRCERR 7-31
CSBARBT 4-79
$\overline{\text{CSBOOT}}$ 4-65, 4-73, 4-75, 4-78, 4-80, 4-86
    reset values 4-87
$\overline{\text{CSE}}$ 4-65, 4-68, 4-69, 4-88
$\overline{\text{CSM}}$ 4-65, 4-68
CSPAR 4-65
CWP 5-44
Cyclic redundancy check error (CRCERR) 7-31

–D–

DAC 5-1
DASM
    block diagram 13-24
    changing mode 13-25, 13-26
    clock divide ratio 13-33
    coherency 13-25–13-28

configuring the output buffer 13-36
counter 13-33
DASMA — DASM data register A 13-37
DASMB — DASM data register B 13-38
DASMSIC — DASM status/interrupt/control register 13-35
disable (DIS) mode 13-26
effect of reset on output flip-flop 13-30
flag clearing mechanism 13-36
forcing an output compare 13-36
freeze 13-34
input capture (IC) mode 13-28
input period measurement (IPM) mode 13-27
input pin 13-24
input pin logic level 13-36
interrupts 13-34
minimum pulse width 13-24
modes of operation 13-25
multiple DASMs 13-35
output compare (OCA and OCAB) modes 13-29
output frequencies and resolutions 13-34
output pin 13-25
output port bit operation 13-31
output pulse width modulation (OPWM) mode 13-32
output pulse width modulation example 13-33
pulse width measurement (IPWM) mode 13-26
pulse width measurement example 13-27
registers
data register A (DASMA) 13-37
data register B (DASMB) 13-38
status/interrupt
control register (DASMSIC) 13-35
reserved register 13-35
selecting the input capture edge sensitivity 13-37
selecting the mode of operation 13-37
selecting the time base bus 13-36
single output compare 13-31
single shot output pulse 13-30
single-shot output pulse example 13-31
spurious interrupts 13-25
DASMA 13-37
DASMA — DASM data register A 13-37
DASMB 13-38
DASMB — DASM data register B 13-38
DASMSIC 13-35
DASMSIC — DASM status/interrupt/control register 13-35
DATA 4-32
Data
and size acknowledge ($\overline{\text{DSACK}}$). *See* $\overline{\text{DSACK}}$ 4-25
bus
mode select conditioning circuits
alternate 4-64
preferred 4-63
mode selection 4-62
signals (DATA) 4-32
field for RX/TX frames (TOUCAN) 7-5
frame 6-50
multiplexer 4-37
strobe ($\overline{\text{DS}}$). *See* $\overline{\text{DS}}$ 4-34
types 3-4
DBcc 3-16
DCNR 8-18
DDRAB 4-64, 4-89
DDRE 4-89
DDRF 4-92
DDRG 4-94
DDRH 4-94
DDRQA 5-35, 5-37
DDRQS 6-9, 6-33, 6-37
Delay
after transfer (DT) 6-23, 6-35
before SCK (DSCKL) 6-18
Designated CPU space 4-34
DFC 3-7
Digital
control section
contents 5-1, 5-14–5-29
input
/output port (PQA) 5-4
port (PQB) 5-4
to analog converter (DAC) 5-14
DIO D-37
Disable TPU2 pins field (DTPU) 8-19
Disabled mode 5-18
Discrete input/output (DIO) D-37
parameters D-38
DIV2 8-19
DIV8 clock 8-7
Divide by two control field (DIV2) 8-19
Double
-buffered 6-53, 6-54
bus fault 3-21, 4-48
-row header 3-26
double buffer 13-40, 13-41
DRVA, DRVB
bits in MCSMSIC 13-13
$\overline{\text{DS}}$ 4-51, 4-62
$\overline{\text{DS}}$ 4-49, 4-50
$\overline{\text{DSACK}}$ 4-25, 4-32, 4-35, 4-39, 4-72, 4-73, 4-75, 4-78, 4-80, 4-84
external/internal generation 4-42
option fields 4-42
signal effects 4-36
$\overline{\text{DSACK}}$ 4-43
assertion results 4-47
DSCK 6-23
DSCKL 6-18
DSCLK 3-25
DSCR 8-12
DSSR 8-13
DT 6-23
DTL 6-18
DTPU 8-19
Dynamic bus sizing 4-36

–E–

ECLK 4-65
output timing diagram E-9
timing diagram E-19

EDOUT - bit in SICA 13-22
EMPTY 7-5
EMU 8-4, 8-11
Emulation
    control (EMU) 8-11
    support 8-4
Emulation mode
    selection 4-69
Emulator chip select ($\overline{CSE}$) 4-69
Encoded
    one of three channel priority levels (CH) 8-17
    time function for each channel (CHANNEL) 8-15
    type of host service (CH) 8-17
Ending queue pointer (ENDQP) 6-19
End-of-
    frame (EOF) 7-16
End-of-queue condition 5-47
ENDQP 6-19, 6-24
Entry
    table bank select field (ETBANK) 8-19
EOF 7-16
EOQ 5-17
ERRINT 7-32
Error
    conditions 6-55
    counters 7-9
    interrupt (ERRINT) 7-32
ESTAT 7-31
ETBANK 8-19
ETRIG 5-5
event counter
    FCSM 13-6
    MCSM 13-11
Event timing 8-3
Exception
    instruction (RTE) 4-49
    processing 3-16, 4-55
        sequence 3-18
        types of exceptions 3-18
        vectors 3-16
            exception vector assignments 3-17
    vector 4-55, 8-5
EXT 4-56
EXTAL 4-57, 4-69
Extended message format 7-1
    frames 7-5
External
    bus
        arbitration 4-50
        interface (EBI) 4-1, 4-31
            bus control signals 4-32
    clock input timing diagram E-9
    digital supply pin 5-6
    multiplexing 5-9
    reset (EXT) 4-56
    trigger pins 5-5
    trigger single-scan mode 5-20
Externally
    multiplexed mode (MUX) 5-38

–F–

FAR 3-23
Fast
    termination
        read cycle timing diagram E-12
        write cycle timing diagram E-13
    termination cycles 4-38, 4-41
Fast quadrature decode TPU function (FQD) D-28
    parameters
        primary channel D-29
        secondary channel D-30
FASTREF 4-6, 4-60
Fault confinement state (FCS) 7-10, 7-32
FC 4-34
FCS 7-10, 7-32
FCSM
    block diagram 13-5
    clock sources 13-6
    counter 13-5
    counter overflow 13-8
    counter register 13-9
    driving the time base bus 13-8
    effect of reset 13-5
    event counter 13-6
    flag clearing 13-8
    freeze 13-6
    input pin - CTMC 13-6, 13-8
    interrupts 13-6
    maximum external clock frequency 13-6, 13-8
    registers
        status/interrupt
            control register (FCSMSIC) 13-7
    reserved registers 13-7
    selecting the clock source 13-8
    setting the interrupt level 13-8, 13-36
    using multiple FCSMs 13-7
FCSMCNT 13-9
FCSMSIC 13-7
FE 6-49, 6-55
Final sample time 5-12
FLAG
    bit in DASMSIC 13-36
    bit in SICA 13-21
FORCE - bit in SICA 13-21
FORMERR 7-31
FPSCK 8-19
F_{QCLK} 5-25
FQD D-28
FQM D-10
Frame 6-50
    size 6-55
Frames
    overload 7-16
    remote 7-16
Framing error (FE) flag 6-49, 6-55
FREEZ ACK 7-17
FREEZE
    assertion response (FRZ)
        QADC 5-7, 5-35, 11-4

QSM 6-5
SCIM2 4-5
TPU 8-12
bus monitor (FRZBM) 4-3
software enable (FRZSW) 4-3
FREEZE (internal signal) 5-48
f_{ref} 4-6, 4-22, 4-57
Frequency measurement (FQM) D-10
parameters D-11
FRZ 5-7, 5-35, 7-11, 8-12, 11-4
FRZACK 7-11
FRZBM 4-3
FRZSW 4-3
f_{sys} 4-6
F-term encoding 4-42
FULL 7-5
Fully (16-bit) expanded mode 4-66
Function
library for TPU 8-4
Function code (FC) signals 4-34, 4-42

–G–

General-purpose I/O ports 4-87
glitches 13-47
Global registers 5-34

–H–

Hall effect decode (HALLD) D-19
parameters D-20
HALLD D-19
HALT 6-20, 7-11
$\overline{\text{HALT}}$ 4-25, 4-35, 4-39, 4-50
$\overline{\text{HALT}}$ 4-49, 4-50
assertion results 4-47
Halt
acknowledge flag (HALTA) 6-21
monitor 4-25
enable (HME) 4-25, 4-26
reset (HLT) 4-56
operation 4-49
negating/reasserting 4-49
QSPI (HALT) 6-20
HALTA 6-21
HALTA and MODF Interrupt Enable (HMIE) 6-41
HALTA/MODF interrupt enable (HMIE) bit 6-20
Handshaking 4-38
Hang on T4 (HOT4) 8-12
Hardware breakpoints 4-43
HLT 4-56
HME 4-25, 4-26
HMIE 6-20
HOT4 8-12
HSQR 8-15
HSSR 8-16

–I–

I/O port operation 5-8
IACK 4-71, 13-52
IARB
TouCAN 7-25
TPU 8-5, 8-12
IARB3
bit in MCSMSIC 13-13
bit in SICA 13-21
ID
Extended (IDE) field 7-6
HIGH field 7-6
LOW field 7-6
IDE 7-6
Identifier (ID) 7-1
bit field 7-6
IDLE 6-48, 6-55, 7-32
Idle
CAN status (IDLE) 7-32
frame 6-50
-line
detect type (ILT) 6-46
detected (IDLE) 6-48, 6-55
detection process 6-55
interrupt enable (ILIE) 6-46, 6-56
type (ILT) bit 6-55
IEN - bit in SICA 13-21
$\overline{\text{IFETCH}}$ 4-66, 4-67
IFLAG 7-33
IL[2:0]
bits in SICA 13-21
ILIE 6-46, 6-56
ILSCI 6-8
ILT 6-46, 6-55
IMASK 7-33
IMB 5-1, 5-24
IN
bit in SICA 13-21
Information processing time (IPT) 7-9
Initial sample time 5-12
Input
sample time (IST) 5-28, 5-49
input capture
SASM IC mode 13-17
input pins
DASM 13-24
FCSM 13-6, 13-8
MCSM 13-11, 13-13
SASM 13-16
Interchannel communication 8-4
Intermission 7-16
Intermodule bus (IMB) 5-1
Internal
bus
error ($\overline{\text{BERR}}$) 4-25, 4-26
monitor 4-25
Interrupt
acknowledge and arbitration 4-71
arbitration 4-4, 6-6
IARB field
TouCAN 7-25
TPU 8-5, 8-12
exception processing 4-70

priority
and recognition 4-70
mask (IP) field 3-6, 4-70, 6-6, 8-5
processing summary 4-73
vector
number 6-6
interrupt acknowledge - see IACK
interrupt arbitration number 13-52
Interrupt Level of SCI (ILSCI) 6-8
Interrupts
QADC 5-29
SCIM2 4-70
TOUCAN 7-19
TPU 8-5
interrupts
DASM 13-34
FCSM 13-6
MCSM 13-11
PWMSM 13-42
SASM 13-17, 13-18, 13-19
setting the priority level 13-52
spurious interrupt vector 13-52
vector base number 13-51
Inter-transfer delay 6-14
Invalid channel number 5-49
IP 6-6, 8-5
$\overline{\text{IPIPE}}$ 4-66, 4-67
IPT 7-9
$\overline{\text{IRQ}}$ 8-5
$\overline{\text{IRQ7}}$ 4-70, 4-71
IST 5-28, 5-49

–L–

LBUF 7-28
Least significant bit (LSB) 5-14
Length of delay after transfer (DTL) 6-18
Limp status (SLIMP) 4-23
LJSRR 5-51
loading the MCSM counter register 13-10
LOC 4-56
Lock
/release/busy mechanism 7-15
Loop
mode 3-15
(LOOPS) 6-46
instruction sequence 3-16
LOOPQ 6-20
LOOPS 6-46
LOSCD 4-22, 4-23
Loss
of reference signal 4-22
-of-clock oscillator (LOSCD) 4-23
Loss of clock reset (LOC) 4-56
Low power stop (LPSTOP)
CPU32 3-15
QADC 5-6
QSM 6-5
Lowest buffer transmitted first (LBUF) 7-28
Low-power
broadcast cycle 4-46
CPU space cycle 4-46
interrupt mask level 4-46
operation 4-20
stop mode enable (STOP)
QADC 5-35
SCIM2 4-30
TPU 8-11
LPSTOP 3-15, 4-21, 4-30
flowchart 4-22
LR 8-18
LSB 3-4, 5-14

–M–

M 6-46, 6-51
M6800 bus E clock signal 4-65
M68000 family
compatibility 3-15
development support 3-19
Mask
examples for normal/extended messages 7-8
registers (RX) 7-7
Master
/slave mode select (MSTR) 6-17
maximum external clock frequency
FCSM 13-6, 13-8
MCSM 13-11, 13-13
MC68010 3-15
MC68020 3-10, 3-15
MCPWM D-21
MCSM
as free-running counter 13-11
block diagram 13-9
clock input pin - CTMC 13-11, 13-13
clocks 13-11
counter 13-10
counter overflow 13-13
counter register 13-14
driving the time base bus 13-13
effect of reset 13-10
event counter 13-11
flag clearing 13-13
freeze 13-12
input pin - CTML 13-11
interrupts 13-11
loading the counter register 13-10
maximum external clock frequency 13-11, 13-13
MCSMCNT — MCSM counter register 13-14
MCSMML — MCSM modulus latch register 13-14
MCSMSIC — MCSM status/interrupt/control register 13-12
modulus latch 13-10
modulus latch register 13-14
modulus load input pin - CTML 13-13
reserved registers 13-12
selecting CTML edge sensitivity 13-13
selecting the clock source 13-13
selecting the time base bus 13-11
setting the interrupt level 13-13
status/interrupt register 13-12
MCSMCNT 13-14

MCSMCNT — MCSM counter register 13-14
MCSMML 13-14
MCSMML — MCSM modulus latch register 13-14
MCSMSIC 13-12
MCSMSIC — MCSM status/interrupt/control register 13-12
MCU
basic system 4-33
Memory
CPU32 organization 3-7
virtual 3-9
Message
buffer
address map 7-23
code for RX/TX buffers 7-5
deactivation 7-13
structure 7-3
format error (FORMERR) 7-31
Mid-analog supply voltage 5-14
minimum pulse width
DASM 13-24
PWMSM 13-44
Misaligned operand 4-37
MISO 6-33, 6-37
MM 4-3, 4-4
MODE 4-81
Mode
fault flag (MODF) 6-21, 6-26
select (M) 6-46
Mode Fault Flag (MODF) 6-41
MODE1, MODE0 - bit in SICA 13-22
Modes
disabled 5-18
reserved 5-18
scan. See Scan modes
modes of operation
DASM 13-25
DASM disable mode 13-26
DASM period measurement mode 13-27
DASM pulse width measurement mode 13-26
event counting mode 13-6, 13-11
input capture 13-17, 13-28
output compare 13-17, 13-29
output compare and toggle 13-18
output port 13-18
pulse width modulation 13-32
STOP mode 13-50
test mode 13-52
MODF 6-21, 6-26, 6-41
Module
mapping (MM) bit 4-3
Modulus
counter 6-51
modulus latch 13-10
MOSFET 4-64
MOSI 6-33, 6-37
Most significant bit (MSB) 5-14
MQ1 5-39
MQ2 5-41
MSB 3-4, 5-14
MSTR 6-17
Multichannel pulse-width modulation (MCPWM) D-21
parameters
master mode D-22
slave channel A
inverted center aligned mode D-26
non-inverted center aligned mode D-24
slave channel B
inverted center aligned mode D-27
non-inverted center aligned mode D-25
slave edge-aligned mode D-23
Multimaster operation 6-26
Multiphase motor commutation (COMM) D-17
parameters D-18, D-19
multiple DASMs 13-35
Multiple end-of-queue 5-17
multiple FCSMs 13-7
multiple SASMs 13-19
Multiplexed analog inputs 5-5
MUX 5-8, 5-38

–N–

N (negative) flag 3-6
New
queue pointer value (NEWQP) 6-19
New input capture/transistion counter (NITC) D-15
parameters D-16
NEWQP 6-19, 6-24
NF 6-49, 6-55
NITC D-15
Noise
error flag (NF) 6-49
errors 6-55
flag (NF) 6-55
reduction 4-4
Non-maskable interrupt 4-71
NOT ACTIVE 7-5
Not ready (NOTRDY) 7-21
NOTRDY 7-17, 7-21

–O–

OC D-33
On-chip breakpoint hardware 3-27
OP (1 through 3) 4-36
Opcode tracking 3-26
open-drain 13-30, 13-34, 13-36
Operand
alignment 4-37
byte order 4-37
destination 3-4
misaligned 4-37
source 3-4
transfer cases 4-37
Operating
configuration out of reset 4-60
mode selection 4-60
OR 6-48
output compare
SASM OC mode 13-17

Output compare (OC) D-33
parameters D-34
output compare and toggle
SASM OCT mode 13-18
output flip-flop
effect of reset 13-15, 13-30, 13-40
output pins
DASM 13-25
PWMSM 13-39
outputs
SASM OP mode 13-18
Overload frames 7-16
OVERRUN 7-5
Overrun error (OR) 6-48

–P–

P 5-49
Parameter RAM
address map 8-21
Parity
(PF) flag 6-55
checking 6-52
enable (PE) 6-46
error (PF) bit 6-49
errors 6-55
type (PT) 6-46
type (PT) bit 6-52
Pause (P) 5-15, 5-49
PCBK 8-13
PCC 3-23
PCS 6-23
to SCK delay (DSCK) 6-23
PCS0/$\overline{\text{SS}}$ 6-38
PCS3-PCS0/SS 6-41
PE 6-46
PEPAR 4-89
Period
/pulse-width accumulator (PPWA) D-31
parameters D-32
period register values
PWMSM 13-44
Periodic
/interval timer 5-29
interrupt
modulus counter 4-29
priority 4-30
request level (PIRQL) 4-31
timer 4-28
components 4-28
interrupt priority and vectoring 4-29
modulus (PITM field) 4-29, 4-31
PIT period calculation 4-29
vector (PIV) 4-31
timer prescaler (PTP) 4-31
timer prescaler control (PTP) 4-29
Peripheral
breakpoints 3-21
chip-selects (PCS) 6-23, 6-36
Peripheral Chip-Select 3-0/Slave Select (PCS3-PCSO/SS) 6-41
PF 6-49, 6-55
PF1 5-43
PF2 5-43
PFIVR 4-93
PFLVR 4-93
PFPAR 4-92
Phase buffer segment 1/2 (PSEG1/2) bit field 7-29
PICR 4-30
PIE1 5-39
PIE2 5-41
Pin
associations with basic configuration options 4-60
reset states 4-58
PIRQL 4-31
PITM 4-29, 4-31
PITR 4-29, 4-31
PIV 4-31
PLL 4-57
Pointer 6-15
POR 4-57
Port
A 4-88
data register (PORTA) 4-88
B 4-88
data register (PORTB) 4-88
C
data register (PORTC) 4-78
E 4-88
F 4-90
block diagram 4-91
edge-detect interrupt level register (PFLVR) 4-90
edge-detect interrupt vector register (PFIVR) 4-90
G 4-93
data register (PORTG) 4-94
H 4-94
data direction register (DDRH) 4-94
data register (PORTH) 4-94
size 4-77
PORTA 4-88
PORTB 4-88
PORTE 4-89
PORTF0/1 4-91
PORTFE 4-93
PORTG 4-94
PORTH 4-94
PORTQA 5-35, 5-36, 5-51
PORTQB 5-35, 5-36
PORTQS 6-9
POW 4-56
Power
-on reset (POR) 4-57
-up reset (POW) 4-56
PPWA D-31
PQA 5-8
PQB 5-4, 5-9
PQSPAR 6-9, 6-33, 6-37
Prescaler 5-26
clock
(PSCK) 8-11
high time (PSH) 5-38

low time (PSL) 5-38
clock high time (PSH) 5-26
clock low time (PSL) 5-26
control
for TCR1 8-5
for TCR2 8-7
divide
factor field 7-29
register (PRESDIV) 7-9, 7-28
prescaler, switching on and off. 13-54
PRESDIV (bit field) 7-29
PRESDIV (register) 7-9, 7-28
Program counter (PC) 3-1, 3-6
Programmable
channel service priority 8-4
transfer length 6-14
Programmable time accumulator (PTA) D-3
parameters D-4
Propagation segment time (PROPSEG) 7-28
PROPSEG 7-11, 7-28
PSCK 8-11
PSEG1 7-29
PSEG2 7-9, 7-11, 7-29
PSEGS1 7-11
PSH 5-26, 5-38
PSL 5-26, 5-38
PT 6-46, 6-52
PTA D-3
PTP 4-31
Pulse-width modulation (PWM) D-35
parameters D-36, D-43
PWM D-35
registers
counter register (PWMC) 13-49
period register (PWMA) 13-48
pulse width register (PWMB) 13-49
PWMA 13-48
PWMA — PWM period register 13-48
PWMB 13-49
PWMB — PWM pulse width register 13-49
PWMC 13-49
PWMC — PWM counter register 13-49
PWMSIC — Status, interrupt and control register 13-45
PWMSM
block diagram 13-39
clock rate selection 13-48
clock selection 13-40
coherency 13-42
comparators 13-40
counter 13-40
effect of reset on counter 13-40
effect of reset on output flip-flop 13-40
enabling and disabling the PWMSM 13-47
freeze 13-42
frequency 13-42, 13-43
interrupts 13-42
maximum duty cycle 13-41
minimum pulse width 13-44
output flip-flop 13-39, 13-41
output pin 13-39
output pulse width 13-41
period registers 13-40
pulse width 13-42, 13-44
pulse width register values 13-44
pulse width registers 13-41
PWMA — PWM period register 13-48
PWMB — PWM pulse width register 13-49
PWMC — PWM counter register 13-49
PWMSIC — Status, interrupt and control register 13-45
reinitialization 13-46
resolution 13-42
selecting the counter clock source 13-47
selecting the output pin polarity 13-48
setting the polarity of the output signal 13-47

–Q–

QACR0 5-37
QACR1 5-38
QACR2 5-40
QADC
conversion characteristics (operating) E-29
electrical characteristics (operating)
DC E-28
features 5-2
maximum ratings E-28
pin functions diagram 5-2
registers
interrupt register (QADCINT) 5-34
module configuration register (QADCMCR) 5-6, 5-34, 11-4
port
A data register (PORTQA) 5-35
B data register (PORTQB) 5-35
data direction register (DDRQA) 5-35
status register 1 (QASR1) 5-44
QADC64
registers
control register 0 (QACR0) 5-37
control register 1 (QACR1) 5-38
control register 2 (QACR2) 5-40
conversion command word table (CCW) 5-48
interrupt register (QAD64CINT) 5-36
left justified, signed result register (LJSRR) 5-51
module configuration register (QADC64MCR) 5-35
port
A data register (PORTQA) 5-35
B data register (PORTQB) 5-35
QB data register (PORTQB) 5-36
PORTQA data direction register (DDRQA) 5-37
PORTQA data register (PORTQA) 5-36
right justified, unsigned result register (RJURR) 5-51
status register 0 (QASR0) 5-43
QADC64INT 5-36
QADC64MCR 5-35
QADCINT 5-34
QADCMCR 5-6, 5-34, 11-4
QASR 5-42

QASR0 5-43
QASR1 5-44
QCLK 5-24
    frequency 5-25
QDDR 6-12, 6-41
QOM D-5
QPAR 6-10
QPDR 6-10, 6-41
QS 5-44
QSM
    pin function 6-10
    QSPI 6-13
        operating modes 6-26
        operation 6-24
        RAM 6-21
    registers
        pin control registers 6-8
            port QS
                data direction register (DDRQS) 6-9
                data register (PORTQS) 6-9
        QSPI
            control register 0 (SPCR0) 6-16
            control register 1 (SPCR1) 6-18
            control register 2 (SPCR2) 6-18
            control register 3 (SPCR3) 6-19
            status register (SPSR) 6-19
        SCI
            control register 0 (SCCR0) 6-45
            control register 1 (SCCR1) 6-45
            data register (SCDR) 6-49
            status register (SCSR) 6-47
    SCI 6-41
        operation 6-50
        pins 6-50
        registers 6-44
QSM Data Direction Register (QDDR) 6-12, 6-41
QSM Pin Assignment Register (QPAR) 6-10
QSM Port Data Register (QPDR) 6-10, 6-41
QSPI 6-13
    block diagram 6-14
    enable (SPE) 6-18
    finished flag (SPIF) 6-21
    initialization operation 6-27
    loop mode (LOOPQ) 6-20
    master operation flow 6-28
    operating modes 6-26
        master mode 6-26, 6-33
            wraparound mode 6-37
        slave mode 6-26, 6-37
    operation 6-24
    peripheral chip-selects 6-36
    RAM 6-21, 6-22
        command RAM 6-22
        receive RAM 6-22
        transmit RAM 6-22
    timing
        master E-21
        slave E-22
QSPI Enable (SPE) 6-41
QSPI Status Register (SPSR) 6-41
Quad-word data 3-4
Queue 5-15
    pointers
        completed queue pointer (CPTQP) 6-24
        end queue pointer (ENDQP) 6-24
        new queue pointer (NEWQP) 6-24
    status (QS) 5-44
Queue 1
    completion
        flag (CF1) 5-43
        interrupt enable (CIE1) 5-39
    operating mode (MQ1) 5-39
    pause
        flag (PF1) 5-43
        interrupt enable (PIE1) 5-39
    single-scan enable (SSE1) 5-39
    trigger overrun (TOR1) 5-43
Queue 2
    completion
        flag (CF2) 5-43
        interrupt enable (CIE2) 5-41
    operating mode (MQ2) 5-41
    pause
        flag (PF2) 5-43
        interrupt enable (PIE2) 5-41
    resume (RES) 5-41
    single-scan enable bit (SSE2) 5-41
    trigger overrun (TOR2) 5-43
Queued
    analog-to-digital converter. See QADC 5-1
    serial
        peripheral interface (QSPI) 6-13
Queued output match TPU function (QOM) D-5
    parameters D-6

–R–

R/$\overline{W}$ 4-4, 4-5, 4-34, 4-39, 4-42, 4-62, 4-66
    field 4-81
RAF 6-48
RAMBAH, RAMBAL 11-5
RASP
    encoding 11-4
RC
    oscillator 4-22
RDRF 6-48, 6-55
RE 6-44, 6-46, 6-54
Read
    /write signal (R/$\overline{W}$). *See* R/$\overline{W}$ 4-34
    cycle 4-40
        flowchart 4-40
        timing diagram E-10
    system register command (RSREG) 3-22
Receive
    data
        register full (RDRF) 6-48
    error status flag (RXWARN) 7-32
    RAM 6-22
    time sample clock (RT) 6-51, 6-54
Receiver
    active (RAF) 6-48

data register (RDRF) flag 6-55
enable (RE) 6-46, 6-54
interrupt enable (RIE) 6-46
wakeup (RWU) 6-47, 6-56
Receiver Enable (RE) 6-44
Reception of transmitted frames 7-13
Registers 5-51
Remote
frames 7-16
transmission request (RTR) 7-5, 7-6
RES 5-41
Reserved
channel number 5-49
mode 5-18
$\overline{\text{RESET}}$ 3-20, 4-6, 4-52, 4-56, 4-57, 4-58, 4-59, 4-60, 4-62, 4-64, 4-67, 4-69, 4-88
Reset
configuration for M68HC16 R-series memory modules 4-67
enable (RSTEN) 4-23
exception processing 4-55
mode selection
timing diagram E-16
operation in SCIM2 4-52
source summary 4-55
states of pins assigned to other MCU modules 4-59
status register (RSR) 4-23, 4-25, 4-56
timing 4-56
resets
effect on FCSM counter 13-5
effect on MCSM 13-10
effect on output flip-flop 13-40
effect on PWMSM counter 13-40
Resistor-divider chain 5-14
Resolution time 5-12
Result word table 5-1, 5-15, 5-51
Resynchronization jump width (RJW) bit field 7-29
Retry operation 4-49
RIE 6-46
RJURR 5-51
RJW 7-11, 7-29
ROM
registers
module configuration register (ROMMCR) 12-3
ROM module configuration register 12-3
ROMBS0–ROMBS3 — ROM bootstrap information words 12-7
ROM base address register (ROMBAH, ROMBAL)
ROMBAH, ROMBAL — ROM base address register 12-5
ROM module configuration register 12-3
ROM module configuration register (ROMMCR) 12-3
ROMBS0–ROMBS3 — ROM bootstrap information words 12-7
RPC 3-23
RSR 4-23, 4-25
RSREG 3-22
RSTEN 4-23
RT 6-54
RTE 4-49
RTR 7-5, 7-6, 7-16
RWD 4-66, 4-67
RWU 6-47, 6-56
RX
Length 7-5
RX14MSKHI 7-30
RX14MSKLO 7-30
RX15MSKHI 7-31
RX15MSKLO 7-31
RXECTR 7-34
RXGMSKHI 7-30
RXGMSKLO 7-30
RXWARN 7-32

–S–

S0 6-7
SAMP 7-28
Sample amplifier bypass (BYP) 5-49
Sampling mode (SAMP) 7-28
SAR 5-14
SASM
block diagram 13-15
capturing the time base bus value 13-17
changing mode 13-17
channel A block diagram 13-16
channel interrupt priority 13-19
clearing flags 13-17, 13-18, 13-19
data register A 13-22
data register B 13-23
effect of reset on output flip-flop 13-15
enabling interrupts 13-21
flag clearing 13-16, 13-21
forcing an output compare 13-18
freeze 13-19
IC mode 13-17
input pin 13-16
interrupts 13-17, 13-18, 13-19
modes of operation 13-16
multiple SASMs 13-19
OC mode 13-17
OCT mode 13-18
OP mode 13-18
output flip-flop 13-16, 13-17, 13-18, 13-19
SDATA — SASM data register A 13-22
SDATB — SASM data register B 13-23
selecting input capture edge sensitivity 13-22
selecting the mode of operation 13-22
selecting the time base bus 13-21
setting the interrupt level 13-21
SICA — SASM status/interrupt/control register A 13-20
SICB — SASM status/interrupt/control register B 13-22
simultaneous interrupts 13-21
spurious interrupts 13-17
status/interrupt control register A 13-20
status/interrupt control register B 13-22
time base bus 13-18
vectors, vector numbers 13-19
SBK 6-47, 6-53

Scan modes
single-scan modes
external trigger 5-20
SCBR 6-45
SCCR0 6-45
SCCR1 6-45
SCDR 6-49
SCI 6-33, 6-41
baud
clock 6-51
rate (SCBR) 6-45
equation 6-45
idle-line detection 6-55
internal loop 6-57
operation 6-50
parity checking 6-52
pins 6-50
receiver
block diagram 6-43
operation 6-54
wakeup 6-56
registers 6-44
SCCR0 6-44
SCCR1 6-44
SCI Baud Rates 6-52
SCI SUBMODULE 6-12
SCSR 6-44
transmitter
block diagram 6-42
operation 6-52
SCI Control Register 0 (SCCR0) 6-44
SCI Control Register 1 (SCCR1) 6-44
SCI Status Register (SCSR) 6-44
SCIM
clock synthesizer control register 4-14
module configuration register 4-2
SCIM2
block diagram 4-2
bus operation 4-38
chip-select 4-73
clock signal (SCIMCLK) 4-21
external bus interface (EBI) 4-31
general-purpose I/O ports 4-87
halt monitor 4-26
interrupts 4-70
operating modes 4-1
periodic interrupt timer 4-28
block diagram (with software watchdog) 4-27
pin reset states 4-58
registers
chip-select
base address
registers (CSBAR) 4-78
option
registers (CSOR) 4-80
pin assignment registers (CSPAR) 4-75
clock synthesizer control register (SYNCR) 4-14
module configuration register (SCIMCR) 4-2, 4-4
periodic interrupt
control register (PICR) 4-30
timer register (PITR) 4-29, 4-31
port A
data register (PORTA) 4-88
port A/B data direction register (DDRAB) 4-88, 4-89
port B
data register (PORTB) 4-88
port C data register (PORTC) 4-78
port E
data direction register (DDRE) 4-89
data register (PORTE) 4-89
pin assignment register (PEPAR) 4-89
port F
data direction register (DDRF) 4-92
data registers (PORTF) 4-91
edge-detect flag register (PORTFE) 4-93
edge-detect interrupt level register (PFLVR) 4-90, 4-93
edge-detect interrupt vector register (PFIVR) 4-90, 4-93
pin assignment register (PFPAR) 4-92
port G
data direction register (DDRG) 4-94
data register (PORTG) 4-94
port H
data direction register (DDRH) 4-94
data register (PORTH) 4-94
software service register (SWSR) 4-28
system
protection control register (SYPCR) 4-24
reset 4-52
software watchdog 4-26
spurious interrupt monitor 4-26
system
clock 4-5
protection 4-23
SCIMCLK 4-21
SCIMCR 4-2, 4-4
SCIMMCR 4-2, 4-14
SCK 6-10, 6-32, 6-37
actual delay before SCK (equation) 6-35
baud rate (equation) 6-34
S-clock 7-9
SCSR 6-47
SDATA 13-22
SDATA — SASM data register A 13-22
SDATB 13-23
SDATB — SASM data register B 13-23
Send break (SBK) 6-47, 6-53
Serial
clock baud rate (SPBR) 6-17
communication interface (SCI) 6-41
formats 6-51
interface 3-24
mode (M) bit 6-51
shifter 6-53
Serial Clock (SCK) 6-10
Service
request breakpoint flag (SRBK) 8-13
SFC 3-7

SGLR 8-18
SHEN 4-3, 4-51
Show cycle
    enable (SHEN) 4-3, 4-5, 4-51
    operation 4-51
    timing diagram E-15
SICA 13-20
SICA — SASM status/interrupt/control register A 13-20
SICB 13-22
SICB — SASM status/interrupt/control register B 13-22
SIGHI
    SIGHI — ROM signature high register 12-6
SIGHI — ROM signature high register 12-6
SIGLO
    SIGLO — ROM signature low register 12-6
SIGLO — ROM signature low register 12-6
Single
    -chip
        mode 4-64, 4-65
SIZ 4-34, 4-37, 4-51, 4-52
Size signals (SIZ) 4-34
    encoding 4-34
Slave Select (SS) 6-41
Slave select signal ($\overline{SS}$) 6-37, 6-38
SLIMP 4-23
SOF 7-9
Soft reset control field (SOFT_RST) 8-19
SOFT_RST 8-19
SOFTRST 7-11
Software
    breakpoints 4-43
    service register (SWSR) 4-26
    watchdog 4-26
        block diagram 4-27
        enable (SWE) 4-26
        prescale (SWP) 4-26
        ratio of SWP and SWT bits 4-27
        timeout period calculation 4-27
        timing field (SWT) 4-26
Software watchdog
    enable (SWE) 4-24
    prescale (SWP) 4-24
    reset (SW) 4-56
    timing field (SWT) 4-24
SPACE (address space select) 4-82
SPBR 6-17
SPCR0 6-16
SPCR1 6-18
SPCR2 6-18
SPCR3 6-19
SPE 6-18, 6-41
SPI 3-25
    finished interrupt enable (SPIFIE) 6-19
    TSBD 6-7
SPI Test Scan Path Select (TSBD) 6-7
SPIF 6-21
SPIFIE 6-19
SPSR 6-19, 6-41
Spurious interrupt exception 4-71
SPWM D-39
SR 3-6
SRAM
    registers
        array base address register (RAMBAH, RAMBAL) 11-5
    supervisor space only 6-7
SRBK 8-13
SRR 7-6
SS 6-41
$\overline{SS}$ 6-37, 6-38
SSE1 5-39
SSE2 5-41
SSP 3-10
Stack pointer (SP) 3-1
Standard
    message format 7-1
        frames 7-5
Standby RAM module w/ TPU emulation (TPURAM). See TPURAM 1-5
Start
    bit (beginning of data frame) 6-50
    -of-frame (SOF) symbol 7-9
State machine 5-25, 6-54
Status register (QASR) 5-42
STEXT 4-21
STF 8-11
STOP 4-30, 5-35, 7-17, 8-11
Stop
    clocks to TCRs (CLKS) 8-12
    enable (STOP) bit
        QADC 5-6
        QSM 6-5
        TOUCAN 7-17
        TPU 8-11
    flag (STF) 8-11
    mode
        external clock (STEXT) 4-21
        SCIM clock (STSCIM) 4-21
    SCI end of data frame bit 6-50
STRB (address strobe/data strobe) bit 4-42, 4-81
STSCIM 4-21
STUFFERR 7-31
Subqueue 5-15
Substitute remote request (SRR) 7-6
Successive approximation register (SAR) 5-14
Supervisor
    /unrestricted data space (SUPV)
        QADC 5-35, 11-4
        SCIM2 4-3, 13-21
        TPU 8-11
    stack pointer (SSP) 3-10
Supervisor mode
    and SRAM 6-7
SUPV 4-3, 4-87, 5-8, 5-35, 11-4, 13-21
SW 4-56
SWE 4-24, 4-26
SWP 4-24, 4-26
SWSR 4-26, 4-28
SWT 4-24, 4-26
synchronization of counters 13-5

synchronization of submodules 13-54
Synchronization to CLKOUT 4-39
Synchronized pulse-width modulation (SPWM) D-39
    parameters D-40, D-41
Synchronous
    resets 4-55
SYNCR 4-14
SYPCR 4-24
SYS 4-56
System
    clock 4-1, 4-5
        output (CLKOUT) 4-39
        sources 4-6
    configuration 4-1, 4-2
    protection block (SCIM2) 4-1
    reset (SYS) 4-56

–T–

T2CFILTER 8-19
T2CG 8-7, 8-11
T2CLK pin filter control (T2CFILTER) 8-19
T2CSL 8-12
Table stepper motor (TSM) D-7
    parameters
        master mode D-8
        slave mode D-9
TBL 3-15
TC 6-48, 6-53
TCIE 6-46, 6-54
TCNMCR 7-23
TCR 8-12
TCR1P 8-11
TCR2 clock/gate control (T2CG) 8-11
TDRE 6-48
TE 6-44, 6-46
Temporary register A (ATEMP) 3-22
Test submodule reset (TST) 4-56
Three-state control (TSC) 4-69
TICR 8-14, 8-20
TIE 6-46, 6-54
Time
    quanta clock 7-9
    stamp 7-5, 7-11
time base bus
    DASM 13-36
    MCSM 13-11
    reading 13-52
    SASM 13-17, 13-18
    selecting 13-11, 13-21, 13-36
Timeout period calculations 4-27
TIMER 7-29
Timer
    count register
        1 prescaler control (TCR1P) 8-11
    synchronize mode (TSYNC) 7-28
toggle 13-18
TOR1 5-43
TOR2 5-43
Totem pole 13-30
totem pole 13-34, 13-36
TOUCAN
    address
        map 7-20
    bit timing configuration 7-8
        operation 7-9
    external pins 7-2
    initialization sequence 7-11
    interrupts 7-19
    message buffer address map 7-23
    operation 7-3
    receive process 7-13
    registers
        control register 0 (CANCTRL0) 7-26
        control register 1 (CTRL1) 7-8
        control register 1(CANCTRL1) 7-27
        control register 2 (CANCTRL2) 7-29
        control register 2 (CTRL2) 7-8
        error and status register (ESTAT) 7-31
        free running timer register (TIMER) 7-29
        interrupt
            configuration register (CANICR) 7-25
            flag register (IFLAG) 7-33
            mask register (IMASK) 7-33
        module configuration register (TCNMCR) 7-23
        receive
            buffer 14 mask registers (RX14MSKHI/LO) 7-30
            buffer 15 mask registers (RX15MSKHI/LO) 7-31
            global mask registers (RXGMSKLO/HI 7-30
        RX/TX error counter registers (RXECTR/TXECTR) 7-34
    special operating modes 7-16
        auto power save mode 7-18
        debug mode 7-16
        low-power stop mode 7-17
    transmit process 7-12
TPU
    address map 8-8
    components 8-2
    FREEZE flag (TPUF) 8-13
    function library 8-4
    host interface 8-2
    interrupts 8-5
    microengine 8-2
    operation 8-3
        coherency 8-4
        emulation support 8-4
        event timing 8-3
        interchannel communication 8-4
        programmable channel service priority 8-4
    parameter RAM 8-2, 8-21
    registers
        channel
            function select registers (CFSR) 8-15
            interrupt
                enable register (CIER) 8-5, 8-14
                status register (CISR) 8-5, 8-18
            priority registers (CPR) 8-17
        decoded channel number register (DCNR) 8-18

development
support control register (DSCR) 8-12
support status register (DSSR) 8-13
host
sequence registers (HSQR) 8-15
service request registers (HSSR) 8-16
link register (LR) 8-18
module configuration register (TPUMCR) 8-10
service grant latch register (SGLR) 8-18
test configuration register (TCR) 8-12
TPU interrupt configuration register (TICR) 8-14, 8-20
scheduler 8-2
time
bases 8-2
timer channels 8-2
TPU Reference Manual 8-3, 8-16
TPU2
module configuration register 2 (TPUMCR2) 8-18
TPUF 8-13
TPUMCR 8-10
TPUMCR2 8-18
Trace
on instruction execution 3-19
Transfer
length options 6-36
Transmission
complete
(TC) flag 6-53
interrupt enable (TCIE) 6-54
Transmit
/receive status (TX/$\overline{RX}$) 7-32
bit error (BITERR) 7-31
complete
bit (TC) 6-48
interrupt enable (TCIE) 6-46
data
register empty (TDRE) flag 6-48
error status flag (TXWARN) 7-31
interrupt enable (TIE) 6-46, 6-54
pin configuration control (TXMODE) 7-26
RAM 6-22
Transmitter Enable (TE) 6-44
Transmitter enable (TE) 6-46, 6-53
Trigger
event 5-46
TSC 4-69
TSM D-7
T_{SR} 5-7
TST 4-56
TSYNC 7-28
TX
Length 7-5
TX/$\overline{RX}$ 7-32
TXECTR 7-34
TXMODE 7-26
TXWARN 7-31

–U–

UART D-12
Unimplemented instruction emulation 3-19
Universal asynchronous receiver/transmitter (UART) D-12
parameters
receiver parameters D-14
transmitter parameters D-13
User stack pointer (USP) 3-10
Using the TPU Function Library and TPU Emulation Mode 8-4
USP 3-10

–V–

V (overflow) flag 3-6
VBR 3-7, 3-16
V_{DD} 4-57
V_{DD} 5-6
V_{DDA} 5-6
$V_{DDA/2}$ 5-14
V_{DDSYN}/MODCLK 4-6, 4-57, 4-60
Vector base register (VBR) 3-7, 3-16
V_{IH} 5-8
V_{IL} 5-8
Virtual memory 3-9
Voltage
reference pins 5-5
V_{RH} 5-5, 5-14, 5-49
V_{RL} 5-5, 5-14, 5-49
V_{SS} 5-6
V_{SSA} 5-6

–W–

WAKE 6-46, 6-56
Wake interrupt (WAKEINT) 7-32
WAKEINT 7-17, 7-32
WAKEMSK 7-17
Wakeup
address mark (WAKE) 6-46, 6-56
Wired-OR
mode
for QSPI pins (WOMQ) 6-17
for SCI pins (WOMS) 6-46, 6-53
wired-or 13-36
WOMQ 6-17
WOMS 6-46, 6-53
WOR - bit in DASMSIC 13-36
Wrap
enable (WREN) 6-19
to (WRTO) 6-19
Wraparound mode 6-15
master 6-37
WREN 6-19
Write
cycle 4-40
flowchart 4-41
Write cycle
timing diagram E-11
WRTO 6-19

–X–

X
(extend) flag 3-6

–Z–

Z (zero) flag 3-6
Zero percent and one hundered percent ‘pulses’ 13-41