# MC68F333

USER'S
MANUAL



INTRODUCTION 1

SIGNAL DESCRIPTIONS 2

SINGLE-CHIP INTEGRATION MODULE 3

CENTRAL PROCESSING UNIT 4

TIME PROCESSOR UNIT 5

QUEUED SERIAL MODULE 6

ANALOG-TO-DIGITAL CONVERTER 7

FLASH EEPROM 8

STANDBY RAM MODULE 9

STANDBY RAM WITH TPU EMULATION 10

ELECTRICAL CHARACTERISTICS A

MECHANICAL DATA AND ORDERING INFORMATION B

DEVELOPMENT SUPPORT C

REGISTER SUMMARY D

INDEX I

1 INTRODUCTION

2 SIGNAL DESCRIPTIONS

3 SINGLE-CHIP INTEGRATION MODULE

4 CENTRAL PROCESSING UNIT

5 TIME PROCESSOR UNIT

6 QUEUED SERIAL MODULE

7 ANALOG-TO-DIGITAL CONVERTER

8 FLASH EEPROM

9 STANDBY RAM MODULE

10 STANDBY RAM WITH TPU EMULATION

A ELECTRICAL CHARACTERISTICS

B MECHANICAL DATA AND ORDERING INFORMATION

C DEVELOPMENT SUPPORT

D REGISTER SUMMARY

I INDEX

# MC68F333
# USER'S MANUAL

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

# PREFACE

This manual describes the capabilities, operation, and functions of the MC68F333 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.

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.

# TABLE OF CONTENTS


| Paragraph | Title | Page |
|---|---|---|

## Section 1
## Introduction

| | | |
|---|---|---|
| 1.1 | MC68F333 Features | 1–1 |
| 1.1.1 | Single-Chip Integration Module | 1–2 |
| 1.1.2 | CPU32 | 1–2 |
| 1.1.3 | Intelligent 16-Bit Time Processor Unit | 1–2 |
| 1.1.4 | Analog-to-Digital Converter | 1–2 |
| 1.1.5 | Queued Serial Module | 1–2 |
| 1.1.6 | Two Flash EEPROM Memory Modules | 1–2 |
| 1.1.7 | Standby RAM | 1–3 |
| 1.1.8 | Standby RAM with TPU Emulation Capability | 1–3 |
| 1.2 | Intermodule Bus | 1–3 |
| 1.3 | System Block Diagram and Pin Assignment Diagrams | 1–3 |
| 1.4 | Memory Map | 1–6 |
| 1.5 | Reset | 1–7 |
| 1.5.1 | SCIM Reset | 1–7 |
| 1.5.2 | MCU Module Pin Function During Reset | 1–8 |

## Section 2
## Signal Descriptions

| | | |
|---|---|---|
| 2.1 | Pin Characteristics | 2–1 |
| 2.2 | Signal Characteristics and Functions | 2–3 |

## Section 3
## Single-Chip Integration Module

| | | |
|---|---|---|
| 3.1 | System Configuration and Protection | 3–2 |
| 3.1.1 | Module Mapping | 3–3 |
| 3.1.2 | Privilege Levels | 3–4 |
| 3.1.3 | Response to FREEZE Assertion | 3–4 |
| 3.1.4 | Interrupt Arbitration | 3–4 |
| 3.1.5 | Single-Chip Operation Support | 3–5 |
| 3.1.6 | Show Internal Cycles | 3–5 |

# TABLE OF CONTENTS
# (Continued)


| Paragraph | Title | Page |
|---|---|---|
| 3.1.7 | Factory Test Mode | 3–6 |
| 3.1.8 | Bus Monitor | 3–6 |
| 3.1.9 | Halt Monitor | 3–6 |
| 3.1.10 | Spurious Interrupt Monitor | 3–7 |
| 3.1.11 | Software Watchdog | 3–7 |
| 3.1.12 | Periodic Interrupt Timer | 3–9 |
| 3.1.13 | Low-Power STOP Operation | 3–11 |
| 3.2 | System Clock | 3–11 |
| 3.2.1 | Clock Sources | 3–12 |
| 3.2.2 | Clock Synthesizer Operation | 3–13 |
| 3.2.3 | External Bus Clock | 3–19 |
| 3.2.4 | Low-Power Operation | 3–19 |
| 3.2.5 | Loss of Clock | 3–20 |
| 3.2.5.1 | Loss of Reference Frequency | 3–21 |
| 3.2.5.2 | Loss of External Clock | 3–22 |
| 3.2.5.3 | Loss-of-Clock Reset | 3–22 |
| 3.3 | External Bus Interface | 3–23 |
| 3.3.1 | Bus Signals | 3–24 |
| 3.3.1.1 | Address Bus | 3–24 |
| 3.3.1.2 | Address Strobe | 3–24 |
| 3.3.1.3 | Data Bus | 3–24 |
| 3.3.1.4 | Data Strobe | 3–25 |
| 3.3.1.5 | Read/Write Signal | 3–25 |
| 3.3.1.6 | Size Signals | 3–25 |
| 3.3.1.7 | Function Codes | 3–25 |
| 3.3.1.8 | Data and Size Acknowledge Signals | 3–26 |
| 3.3.1.9 | Bus Error Signal | 3–26 |
| 3.3.1.10 | Halt Signal | 3–27 |
| 3.3.1.11 | Autovector Signal | 3–27 |
| 3.3.2 | Dynamic Bus Sizing | 3–27 |
| 3.3.3 | Operand Alignment | 3–28 |
| 3.3.4 | Misaligned Operands | 3–29 |
| 3.3.5 | Operand Transfer Cases | 3–29 |
| 3.3.6 | Bus Operation | 3–30 |
| 3.3.6.1 | Synchronization to CLKOUT | 3–30 |
| 3.3.6.2 | Regular Bus Cycles | 3–31 |
| 3.3.6.3 | Read Cycle | 3–32 |
| 3.3.6.4 | Write Cycle | 3–33 |
| 3.3.6.5 | Fast-Termination Cycles | 3–34 |
| 3.3.7 | CPU Space Cycles | 3–35 |

# TABLE OF CONTENTS
## (Continued)


| Paragraph | Title | Page |
|---|---|---|
| 3.3.7.1 | Breakpoint Acknowledge Cycle | 3–36 |
| 3.3.7.1.1 | Software Breakpoints | 3–36 |
| 3.3.7.1.2 | Hardware Breakpoints | 3–37 |
| 3.3.7.2 | LPSTOP Broadcast Cycle | 3–39 |
| 3.3.8 | Bus Exception Control Cycles | 3–39 |
| 3.3.8.1 | Bus Error Exceptions | 3–41 |
| 3.3.8.2 | Double Bus Faults | 3–42 |
| 3.3.8.3 | Retry Operation | 3–42 |
| 3.3.8.4 | Halt Operation | 3–43 |
| 3.3.9 | External Bus Arbitration | 3–44 |
| 3.3.10 | Slave (Factory Test) Mode Arbitration | 3–46 |
| 3.3.11 | Show Cycles | 3–46 |
| 3.4 | Reset | 3–46 |
| 3.4.1 | Reset Exception Processing | 3–47 |
| 3.4.2 | Reset Control Logic | 3–47 |
| 3.4.3 | Operating Configuration out of Reset | 3–48 |
| 3.4.3.1 | Address and Data Bus Pin Functions | 3–49 |
| 3.4.3.2 | Data Bus Mode Selection | 3–49 |
| 3.4.3.3 | Pin Configuration for 16-Bit Data Bus Operation | 3–50 |
| 3.4.3.4 | Pin Configuration for 8-Bit Data Bus Operation | 3–52 |
| 3.4.3.5 | Pin Configuration for Single-Chip Operation | 3–53 |
| 3.4.3.6 | Clock Mode Selection | 3–54 |
| 3.4.3.7 | Breakpoint Mode Selection | 3–55 |
| 3.4.3.8 | Emulation Mode Selection | 3–55 |
| 3.4.4 | MCU Module Pin Function During Reset | 3–55 |
| 3.4.5 | Pin State During Reset | 3–56 |
| 3.4.5.1 | Reset States of SCIM Pins | 3–56 |
| 3.4.5.2 | Reset States of Pins Assigned to Other MCU Modules | 3–58 |
| 3.4.6 | Reset Timing | 3–58 |
| 3.4.7 | Power-On Reset | 3–59 |
| 3.4.8 | Use of Three-State Control Pin | 3–60 |
| 3.4.9 | Reset Processing Summary | 3–61 |
| 3.4.10 | Reset Status Register | 3–61 |
| 3.5 | Interrupts | 3–62 |
| 3.5.1 | Interrupt Exception Processing | 3–62 |
| 3.5.2 | Interrupt Priority and Recognition | 3–62 |
| 3.5.3 | Interrupt Acknowledge and Arbitration | 3–63 |
| 3.5.4 | Interrupt Processing Summary | 3–65 |
| 3.5.5 | Interrupt Acknowledge Bus Cycles | 3–66 |
| 3.6 | Chip Selects | 3–66 |

# TABLE OF CONTENTS
(Continued)


| Paragraph | Title | Page |
|---|---|---|
| 3.6.1 | Chip-Select Registers | 3–68 |
| 3.6.1.1 | Chip-Select Pin Assignment Registers | 3–69 |
| 3.6.1.2 | Chip-Select Base Address Registers | 3–70 |
| 3.6.1.3 | Chip-Select Option Registers | 3–71 |
| 3.6.1.4 | Port C Data Register | 3–73 |
| 3.6.2 | Chip-Select Operation | 3–73 |
| 3.6.3 | Using Chip-Select Signals in Interrupt Acknowledge Cycles | 3–74 |
| 3.6.4 | Chip-Select Reset Operation | 3–75 |
| 3.6.5 | Emulation-Support Chip Select | 3–76 |
| 3.7 | General-Purpose I/O | 3–77 |
| 3.7.1 | Ports A and B | 3–78 |
| 3.7.2 | Port E | 3–78 |
| 3.7.3 | Port F | 3–79 |
| 3.7.3.1 | Port F Data Register | 3–80 |
| 3.7.3.2 | Port F Pin Assignment Register | 3–80 |
| 3.7.3.3 | Port F Data Direction Register | 3–81 |
| 3.7.3.4 | Port F Edge-Detect Flag Register | 3–81 |
| 3.7.3.5 | Port F Interrupt Registers | 3–81 |
| 3.7.4 | Port G | 3–82 |
| 3.7.5 | Port H | 3–82 |
| 3.8 | Factory Testing | 3–82 |


## Section 4
## Central Processing Unit


| | | |
|---|---|---|
| 4.1 | CPU32 Registers | 4–2 |
| 4.1.1 | Data Registers | 4–4 |
| 4.1.2 | Address Registers | 4–5 |
| 4.1.3 | Program Counter | 4–6 |
| 4.1.4 | Control Registers | 4–6 |
| 4.1.4.1 | Status Register | 4–6 |
| 4.1.4.2 | Alternate Function Code Registers | 4–7 |
| 4.1.5 | Vector Base Register | 4–7 |
| 4.2 | Memory Organization | 4–7 |
| 4.3 | Virtual Memory | 4–9 |
| 4.4 | Addressing Modes | 4–9 |
| 4.5 | Processing States | 4–10 |
| 4.6 | Privilege Levels | 4–10 |
| 4.7 | Instructions | 4–10 |
| 4.7.1 | M68000 Family Compatibility | 4–13 |

# TABLE OF CONTENTS
# (Continued)


| Paragraph | Title | Page |
|---|---|---|
| 4.7.2 | New Instructions | 4–13 |
| 4.7.2.1 | Low-Power Stop (LPSTOP) | 4–13 |
| 4.7.2.2 | Table Lookup and Interpolate (TBL) | 4–13 |
| 4.8 | Exception Processing | 4–13 |
| 4.8.1 | Exception Vectors | 4–13 |
| 4.8.2 | Types of Exceptions | 4–16 |
| 4.8.3 | Exception Processing Sequence | 4–16 |
| 4.9 | Development Support | 4–17 |
| 4.9.1 | M68000 Family Development Support | 4–17 |
| 4.9.2 | Background Debugging Mode | 4–17 |
| 4.9.3 | Deterministic Opcode Tracking | 4–19 |
| 4.9.4 | On-Chip Breakpoint Hardware | 4–20 |
| 4.10 | Loop Mode Instruction Execution | 4–20 |


## Section 5
## Time Processor Unit


| | | |
|---|---|---|
| 5.1 | Overview | 5–2 |
| 5.2 | TPU Components | 5–2 |
| 5.2.1 | Time Bases | 5–2 |
| 5.2.2 | Timer Channels | 5–2 |
| 5.2.3 | Scheduler | 5–3 |
| 5.2.4 | Microengine | 5–3 |
| 5.2.5 | Host Interface | 5–3 |
| 5.2.6 | Parameter RAM | 5–3 |
| 5.3 | TPU Operation | 5–4 |
| 5.3.1 | Event Timing | 5–4 |
| 5.3.2 | Channel Orthogonality | 5–4 |
| 5.3.3 | Interchannel Communication | 5–5 |
| 5.3.4 | Programmable Channel Service Priority | 5–5 |
| 5.3.5 | Coherency | 5–5 |
| 5.3.6 | Emulation Support | 5–5 |
| 5.3.7 | TPU Interrupts | 5–6 |
| 5.4 | Time Functions | 5–7 |
| 5.4.1 | Discrete Input/Output (DIO) | 5–7 |
| 5.4.2 | Input Capture/Input Transition Counter (IC/ITC) | 5–7 |
| 5.4.3 | Output Compare (OC) | 5–7 |
| 5.4.4 | Pulse-Width Modulation (PWM) | 5–8 |
| 5.4.5 | Synchronized Pulse-Width Modulation (SPWM) | 5–8 |
| 5.4.6 | Period Measurement with Additional Transition Detect (PMA) | 5–8 |

# TABLE OF CONTENTS
## (Continued)


| Paragraph | Title | Page |
|---|---|---|
| 5.4.7 | Period Measurement with Missing Transition Detect (PMM) | 5–9 |
| 5.4.8 | Position-Synchronized Pulse Generator (PSP) | 5–9 |
| 5.4.9 | Stepper Motor (SM) | 5–9 |
| 5.4.10 | Period/Pulse-Width Accumulator (PPWA) | 5–10 |
| 5.5 | Host Interface Registers | 5–11 |
| 5.5.1 | System Configuration Registers | 5–11 |
| 5.5.1.1 | Prescaler Control for TCR1 | 5–11 |
| 5.5.1.2 | Prescaler Control for TCR2 | 5–12 |
| 5.5.1.3 | Emulation Control | 5–13 |
| 5.5.1.4 | Low-Power Stop Control | 5–13 |
| 5.5.2 | Channel Control Registers | 5–13 |
| 5.5.2.1 | Channel Interrupt Enable and Status Registers | 5–13 |
| 5.5.2.2 | Channel Function Select Registers | 5–14 |
| 5.5.2.3 | Host Sequence Registers | 5–14 |
| 5.5.2.4 | Host Service Registers | 5–14 |
| 5.5.2.5 | Channel Priority Registers | 5–16 |
| 5.5.3 | Development Support and Test Registers | 5–16 |


## Section 6
## Queued Serial Module


| | | |
|---|---|---|
| 6.1 | Overview | 6–1 |
| 6.2 | QSM Registers and Address Map | 6–2 |
| 6.2.1 | QSM Global Registers | 6–2 |
| 6.2.1.1 | Low-Power Stop Operation | 6–3 |
| 6.2.1.2 | Freeze Operation | 6–3 |
| 6.2.1.3 | QSM Interrupts | 6–3 |
| 6.2.2 | QSM Pin Control Registers | 6–4 |
| 6.3 | Queued Serial Peripheral Interface | 6–5 |
| 6.3.1 | QSPI Registers | 6–7 |
| 6.3.1.1 | Control Registers | 6–8 |
| 6.3.1.2 | Status Register | 6–8 |
| 6.3.2 | QSPI RAM | 6–8 |
| 6.3.2.1 | Receive RAM | 6–8 |
| 6.3.2.2 | Transmit RAM | 6–9 |
| 6.3.2.3 | Command RAM | 6–9 |
| 6.3.3 | QSPI Pins | 6–10 |
| 6.3.4 | QSPI Operation | 6–10 |
| 6.3.5 | QSPI Operating Modes | 6–11 |
| 6.3.5.1 | Master Mode | 6–18 |

# TABLE OF CONTENTS
# (Continued)


| Paragraph | Title | Page |
|---|---|---|
| 6.3.5.2 | Master Wraparound Mode | 6–21 |
| 6.3.5.3 | Slave Mode | 6–21 |
| 6.3.5.4 | Slave Wraparound Mode | 6–23 |
| 6.3.6 | Peripheral Chip Selects | 6–23 |
| 6.4 | Serial Communication Interface | 6–23 |
| 6.4.1 | SCI Registers | 6–23 |
| 6.4.1.1 | Control Registers | 6–24 |
| 6.4.1.2 | Status Register | 6–27 |
| 6.4.1.3 | Data Register | 6–27 |
| 6.4.2 | SCI Pins | 6–27 |
| 6.4.3 | SCI Operation | 6–28 |
| 6.4.3.1 | Definition of Terms | 6–28 |
| 6.4.3.2 | Serial Formats | 6–28 |
| 6.4.3.3 | Baud Clock | 6–29 |
| 6.4.3.4 | Parity Checking | 6–29 |
| 6.4.3.5 | Transmitter Operation | 6–30 |
| 6.4.3.6 | Receiver Operation | 6–31 |
| 6.4.3.7 | Idle-Line Detection | 6–32 |
| 6.4.3.8 | Receiver Wakeup | 6–33 |
| 6.4.3.9 | Internal Loop | 6–34 |
| 6.5 | QSM Initialization | 6–34 |


## Section 7
## Analog-to-Digital Converter


| | | |
|---|---|---|
| 7.1 | Overview | 7–1 |
| 7.2 | External Connections | 7–1 |
| 7.2.1 | Analog Input Pins | 7–2 |
| 7.2.2 | Digital Output Pins | 7–3 |
| 7.2.3 | Analog Reference Pins | 7–3 |
| 7.2.4 | Analog Supply Pins | 7–3 |
| 7.3 | Programmer's Model | 7–3 |
| 7.4 | ADC Bus Interface Unit | 7–4 |
| 7.5 | Special Operating Modes | 7–4 |
| 7.5.1 | Low-Power Stop Mode | 7–4 |
| 7.5.2 | Freeze Mode | 7–4 |
| 7.6 | Analog Subsystem | 7–5 |
| 7.6.1 | Multiplexer | 7–5 |
| 7.6.2 | Sample Capacitors and Buffer Amplifier | 7–6 |
| 7.6.3 | RC DAC Array | 7–6 |

# TABLE OF CONTENTS
# (Continued)


| Paragraph | Title | Page |
|---|---|---|
| 7.6.4 | Comparator | 7–7 |
| 7.7 | Digital Control Subsystem | 7–7 |
| 7.7.1 | Control and Status Registers | 7–7 |
| 7.7.2 | Clock and Prescaler Control | 7–7 |
| 7.7.3 | Sample Time | 7–8 |
| 7.7.4 | Resolution | 7–8 |
| 7.7.5 | Conversion Control Logic | 7–9 |
| 7.7.5.1 | Conversion Parameters | 7–9 |
| 7.7.5.2 | Conversion Modes | 7–9 |
| 7.7.6 | Conversion Timing | 7–14 |
| 7.7.7 | Successive Approximation Register | 7–17 |
| 7.7.8 | Result Registers | 7–17 |


## Section 8
## Flash EEPROM


| | | |
|---|---|---|
| 8.1 | Overview | 8–1 |
| 8.2 | Signal Descriptions | 8–2 |
| 8.3 | Flash EEPROM Control Block | 8–2 |
| 8.4 | Flash EEPROM Array | 8–3 |
| 8.5 | Module Configuration | 8–3 |
| 8.5.1 | Low-Power Stop Control | 8–3 |
| 8.5.2 | Response to FREEZE Assertion | 8–3 |
| 8.5.3 | System Boot Configuration | 8–4 |
| 8.5.4 | Write-Lock Protection | 8–4 |
| 8.5.5 | Array Space | 8–4 |
| 8.5.6 | Wait States | 8–4 |
| 8.6 | Base Address Registers | 8–5 |
| 8.7 | Flash EEPROM Control Register | 8–5 |
| 8.7.1 | Program/Erase Verification | 8–5 |
| 8.7.2 | Erasure Control | 8–6 |
| 8.7.3 | Latch Control | 8–6 |
| 8.7.4 | Enabling Programming and Erasure | 8–6 |
| 8.8 | Flash EEPROM Bootstrap Words | 8–6 |
| 8.9 | Flash EEPROM Operation | 8–7 |
| 8.9.1 | Reset Operation | 8–7 |
| 8.9.2 | Bootstrap Operation | 8–7 |
| 8.9.3 | Normal Operation | 8–8 |
| 8.9.4 | Program/Erase Operation | 8–8 |
| 8.9.4.1 | Programming Operation | 8–8 |

# TABLE OF CONTENTS
# (Continued)


| Paragraph | Title | Page |
|---|---|---|
| 8.9.4.2 | Erase Operation | 8–11 |
| 8.9.5 | FLASH Program/Erase Voltage Signal Conditioning | 8–13 |

## Section 9
## Standby RAM Module

| | | |
|---|---|---|
| 9.1 | SRAM Register Block | 9–1 |
| 9.2 | SRAM Array Address Mapping | 9–1 |
| 9.3 | SRAM Array Address Space Type | 9–1 |
| 9.4 | Normal Access | 9–2 |
| 9.5 | Standby Operation | 9–2 |
| 9.6 | Low-Power Stop Operation | 9–3 |
| 9.7 | Reset | 9–3 |

## Section 10
## Standby RAM with TPU Emulation

| | | |
|---|---|---|
| 10.1 | TPURAM Register Block | 10–1 |
| 10.2 | TPURAM Array Address Mapping | 10–2 |
| 10.3 | TPURAM Privilege Level | 10–2 |
| 10.4 | Normal Operation | 10–2 |
| 10.5 | Standby Operation | 10–3 |
| 10.6 | Low-Power Stop Operation | 10–3 |
| 10.7 | Reset | 10–4 |
| 10.8 | TPU Microcode Emulation | 10–4 |

## Appendix A
## Electrical Characteristics

## Appendix B
## Mechanical Data and Ordering Information

## Appendix C
## Development Support

# TABLE OF CONTENTS
# (Continued)

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

## Appendix D
## Register Summary


D.1 Central Processing Unit ..... D–2
D.1.1 CPU32 Register Model ..... D–2
D.1.2 SR — Status Register ..... D–3
D.2 Analog-to-Digital Converter Module ..... D–4
D.2.1 ADCMCR — ADC Module Configuration Register ..... D–5
D.2.2 ADTEST — ADC Test Register ..... D–5
D.2.3 PORTAD — Port AD Data Register ..... D–5
D.2.4 ADCTL0 — ADC Control Register 0 ..... D–6
D.2.5 ADCTL1 — ADC Control Register 1 ..... D–7
D.2.6 ADSTAT — ADC Status Register ..... D–9
D.2.7 RSLT[0:7] — ADC Result Registers ..... D–10
D.2.7.1 RJURR — Unsigned Right-Justified Result Registers ..... D–10
D.2.7.2 LJSRR — Signed Left-Justified Result Registers ..... D–10
D.2.7.3 LJURR — Unsigned Left-Justified Result Registers ..... D–10
D.3 Flash EEPROM Modules ..... D–11
D.3.1 FEE1MCR — Flash EEPROM Configuration Register ..... D–12
FEE2MCR — Flash EEPROM Configuration Register ..... D–12
D.3.2 FEE1TST — Flash EEPROM Test Register ..... D–13
FEE2TST — Flash EEPROM Test Register ..... D–13
D.3.3 FEE1BAH — Flash EEPROM Base Address High Register ..... D–13
FEE2BAH — Flash EEPROM Base Address High Register ..... D–13
D.3.4 FEE1BAL — Flash EEPROM Base Address Low Register ..... D–13
FEE2BAL — Flash EEPROM Base Address Low Register ..... D–13
D.3.5 FEE1CTL — Flash EEPROM Control Register ..... D–14
FEE2CTL — Flash EEPROM Control Register ..... D–14
D.3.6 FEE1BS[3:0] — Flash EEPROM Bootstrap Words ..... D–15
FEE2BS[3:0] — Flash EEPROM Bootstrap Words ..... D–15
D.4 Single-Chip Integration Module ..... D–16
D.4.1 SCIMCR — SCIM Configuration Register ..... D–17
D.4.2 SCIMTR — SCIM Test Register ..... D–19
D.4.3 SYNCR — Clock Synthesizer Control Register ..... D–19
D.4.4 RSR — Reset Status Register ..... D–20
D.4.5 PORTA — Port A Data Register ..... D–21
PORTB — Port B Data Register ..... D–21
D.4.6 PORTG — Port G Data Register ..... D–21
PORTH — Port H Data Register ..... D–21

# TABLE OF CONTENTS
## (Continued)


| Paragraph | Title | Page |
|---|---|---|
| D.4.7 | DDRG — Port G Data Direction Register | D–21 |
| | DDRH — Port H Data Direction Register | D–21 |
| D.4.8 | PORTE0, PORTE1 — Port E Data Register | D–22 |
| D.4.9 | DDRAB — Port A/B Data Direction Register | D–22 |
| | DDRE — Port E Data Direction Register | D–22 |
| D.4.10 | PEPAR — Port E Pin Assignment Register | D–22 |
| D.4.11 | PORTF0, PORTF1 — Port F Data Register | D–23 |
| D.4.12 | DDRF — Port F Data Direction Register | D–23 |
| D.4.13 | PFPAR — Port F Pin Assignment Register | D–24 |
| D.4.14 | PORTFE — Port F Edge-Detect Flag Register | D–24 |
| D.4.15 | PFIVR — Port F Edge-Detect Interrupt Vector Register | D–25 |
| D.4.16 | PFLVR — Port F Edge-Detect Interrupt Level Register | D–25 |
| D.4.17 | SYPCR — System Protection Control Register | D–25 |
| D.4.18 | PICR — Periodic Interrupt Control Register | D–26 |
| D.4.19 | PITR — Periodic Interrupt Timer Register | D–27 |
| D.4.20 | SWSR — Software Service Register | D–27 |
| D.4.21 | SCIM Test Registers | D–27 |
| D.4.22 | PORTC — Port C Data Register | D–28 |
| D.4.23 | CSPAR0 — Chip Select Pin Assignment Register 0 | D–28 |
| D.4.24 | CSPAR1 — Chip Select Pin Assignment Register 1 | D–28 |
| D.4.25 | CSBARBT — Chip Select Base Address Register Boot ROM | D–29 |
| D.4.26 | CSBAR[10:0] — Chip Select Base Address Registers | D–29 |
| D.4.27 | CSORBT — Chip Select Option Register Boot ROM | D–30 |
| D.4.28 | CSOR[10:0] — Chip Select Option Registers | D–30 |
| D.5 | Standby RAM Module with TPU Emulation Capability (TPURAM) | D–32 |
| D.5.1 | TRAMMCR — TPURAM Module Configuration Register | D–32 |
| D.5.2 | TRAMTST — TPURAM Test Register | D–32 |
| D.5.3 | TRAMBAR — TPURAM Base Address and Status Register | D–33 |
| D.6 | Standby RAM Module (SRAM) | D–34 |
| D.6.1 | SRAMMCR — SRAM Module Configuration Register | D–34 |
| D.6.2 | SRAMTST — SRAM Test Register | D–35 |
| D.6.3 | SRAMBAH — SRAM Array Base Address Register High | D–35 |
| D.6.4 | SRAMBAL — SRAM Array Base Address Register Low | D–35 |
| D.7 | Queued Serial Module (QSM) | D–36 |
| D.7.1 | QSMCR — QSM Configuration Register | D–36 |
| D.7.2 | QTEST — QSM Test Register | D–37 |
| D.7.3 | QILR — QSM Interrupt Level Register | D–37 |
| | QIVR — QSM Interrupt Vector Register | D–37 |
| D.7.4 | PORTQS — Port QS Data Register | D–38 |

# TABLE OF CONTENTS
# (Continued)


| Paragraph | Title | Page |
|---|---|---|
| D.7.5 | PQSPAR — PORT QS Pin Assignment Register | D–38 |
| | DDRQS — PORT QS Data Direction Register | D–38 |
| D.7.6 | SPCR0 — QSPI Control Register 0 | D–40 |
| D.7.7 | SPCR1 — QSPI Control Register 1 | D–41 |
| D.7.8 | SPCR2 — QSPI Control Register 2 | D–42 |
| D.7.9 | SPCR3 — QSPI Control Register 3 | D–42 |
| | SPSR — QSPI Status Register | D–42 |
| D.7.10 | RR[0:F] — Receive Data RAM | D–43 |
| D.7.11 | TR[0:F] — Transmit Data RAM | D–43 |
| D.7.12 | CR[0:F] — Command RAM | D–44 |
| D.7.13 | SCCR0 — SCI Control Register 0 | D–45 |
| D.7.14 | SCCR1 — SCI Control Register 1 | D–45 |
| D.7.15 | SCSR — SCI Status Register | D–47 |
| D.7.16 | SCDR — SCI Data Register | D–48 |
| D.8 | Time Processor Unit (TPU) | D–49 |
| D.8.1 | TPUMCR — TPU Module Configuration Register | D–49 |
| D.8.2 | TICR — TPU Interrupt Configuration Register | D–51 |
| D.8.3 | CIER — Channel Interrupt Enable Register | D–51 |
| D.8.4 | CISR — Channel Interrupt Status Register | D–52 |
| D.8.5 | CFSR0 — Channel Function Select Register 0 | D–52 |
| D.8.6 | CFSR1 — Channel Function Select Register 1 | D–52 |
| D.8.7 | CFSR2 — Channel Function Select Register 2 | D–52 |
| D.8.8 | CFSR3 — Channel Function Select Register 3 | D–52 |
| D.8.9 | HSQR0 — Host Sequence Register 0 | D–53 |
| D.8.10 | HSQR1 — Host Sequence Register 1 | D–53 |
| D.8.11 | HSRR0 — Host Service Request Register 0 | D–53 |
| D.8.12 | HSRR1 — Host Service Request Register 1 | D–53 |
| D.8.13 | CPR0 — Channel Priority Register 0 | D–55 |
| D.8.14 | CPR1 — Channel Priority Register 1 | D–55 |
| D.8.15 | Parameter RAM | D–55 |
| D.8.16 | DSCR — Development Support Control Register | D–57 |
| D.8.17 | DSSR — Development Support Status Register | D–58 |
| D.8.18 | LR — Link Register | D–58 |
| D.8.19 | SGLR — Service Grant Latch Register | D–59 |
| D.8.20 | DCNR — Decoded Channel Number Register | D–59 |
| D.8.21 | TCR — Test Configuration Register | D–59 |
| D.9 | Register Bit and Field Mnemonics | D–60 |

**Index**

# LIST OF ILLUSTRATIONS


| Figure | Title | Page |
|---|---|---|
| 1–1 | MC68F333 Block Diagram | 1–4 |
| 1–2 | MC68F333 Pin Assignment | 1–5 |
| 1–3 | MC68F333 Memory Map | 1–6 |
| 2–1 | MC68F333 Signal Pin Assignments | 2–7 |
| 3–1 | Single-Chip Integration Module Block Diagram | 3–1 |
| 3–2 | System Configuration and Protection | 3–3 |
| 3–3 | Periodic Interrupt Timer and Software Watchdog Timer | 3–9 |
| 3–4 | System Clock Block Diagram | 3–12 |
| 3–5 | Loss of Reference Frequency | 3–21 |
| 3–6 | Loss of External Clock Signal | 3–22 |
| 3–7 | MC68F333 Basic System | 3–23 |
| 3–8 | Operand Byte Order | 3–28 |
| 3–9 | Word Read Cycle Flowchart | 3–32 |
| 3–10 | Write Cycle Flowchart | 3–33 |
| 3–11 | Fast-Termination Timing | 3–35 |
| 3–12 | CPU Space Address Encoding | 3–36 |
| 3–13 | Breakpoint Operation Flowchart | 3–38 |
| 3–14 | LPSTOP Interrupt Mask Level | 3–39 |
| 3–15 | Bus Arbitration Flowchart for Single Request | 3–45 |
| 3–16 | Data Bus Mode Select Conditioning | 3–50 |
| 3–17 | Power-On Reset Timing | 3–60 |
| 3–18 | Basic MC68F333 System | 3–67 |
| 3–19 | Chip-Select Circuit Block Diagram | 3–68 |
| 3–20 | CPU Space Encoding for Interrupt Acknowledge | 3–74 |
| 4–1 | CPU32 Block Diagram | 4–2 |
| 4–2 | User Programming Model | 4–3 |
| 4–3 | Supervisor Programming Model Supplement | 4–3 |
| 4–4 | Data Organization in Data Registers | 4–5 |
| 4–5 | Address Organization in Address Registers | 4–6 |
| 4–6 | Memory Operand Addressing | 4–8 |
| 4–7 | Traditional In-Circuit Emulator Diagram | 4–18 |
| 4–8 | Bus State Analyzer Configuration | 4–18 |
| 4–9 | Loop Mode Instruction Execution | 4–20 |

# LIST OF ILLUSTRATIONS
## (Continued)


| Figure | Title | Page |
|---|---|---|
| 5–1 | TPU Block Diagram | 5–1 |
| 5–2 | TCR1 Prescaler Control | 5–11 |
| 5–3 | TCR2 Prescaler Control | 5–12 |
| 6–1 | QSM Block Diagram | 6–1 |
| 6–2 | QSPI Block Diagram | 6–7 |
| 6–3 | QSPI RAM | 6–9 |
| 6–4 | Flowchart of QSPI Initialization Operation | 6–12 |
| 6–5 | Flowchart of QSPI Master Operation | 6–13 |
| 6–6 | Flowchart of QSPI Slave Operation | 6–16 |
| 6–7 | SCI Transmitter Block Diagram | 6–25 |
| 6–8 | SCI Receiver Block Diagram | 6–26 |
| 7–1 | ADC Block Diagram | 7–2 |
| 7–2 | 8-Bit Conversion Timing | 7–15 |
| 7–3 | 10-Bit Conversion Timing | 7–16 |
| 8–1 | FLASH Programming Flow | 8–10 |
| 8–2 | FLASH Erasure Flow | 8–12 |
| 8–3 | VFPE Operating Range | 8–13 |
| 8–4 | VFPE Conditioning Circuit | 8–14 |
| A–1 | CLKOUT Output Timing Diagram | A–16 |
| A–2 | External Clock Input Timing Diagram | A–16 |
| A–3 | ECLK Output Timing Diagram | A–16 |
| A–4 | Read Cycle Timing Diagram | A–18 |
| A–5 | Write Cycle Timing Diagram | A–20 |
| A–6 | Show Cycle Timing Diagram | A–22 |
| A–7 | Reset and Mode Select Timing Diagram | A–24 |
| A–8 | Bus Arbitration Timing Diagram — Active Bus Case | A–26 |
| A–9 | Bus Arbitration Timing Diagram — Idle Bus Case | A–28 |
| A–10 | Fast Termination Read Cycle Timing Diagram | A–30 |
| A–11 | Fast Termination Write Cycle Timing Diagram | A–32 |
| A–12 | ECLK Timing Diagram | A–34 |
| A–13 | Chip Select Timing Diagram | A–36 |
| A–14 | Background Debugging Mode Timing Diagram — Serial Communication | A–38 |
| A–15 | Background Debugging Mode Timing Diagram — Freeze Assertion | A–38 |
| A–16 | QSPI Timing Master, CPHA = 0 | A–40 |
| A–17 | QSPI Timing Master, CPHA = 1 | A–40 |

# LIST OF ILLUSTRATIONS
# (Continued)


| Figure | Title | Page |
|---|---|---|
| A–18 | QSPI Timing Slave, CPHA = 0 | A–42 |
| A–19 | QSPI Timing Slave, CPHA = 1 | A–42 |
| B–1 | 160-Pin Plastic Surface Mount Package Pin Assignments | B–2 |
| B–2 | 160-Pin Package Dimensions | B–3 |
| B–3 | 160-Pin Molded Carrier Ring Assembly | B–4 |
| D–1 | User Programming Model | D–2 |
| D–2 | Supervisor Programming Model Supplement | D–2 |

# LIST OF TABLES


| Table | Title | Page |
|---|---|---|
| 1–1 | Basic Configuration Options | 1–7 |
| 1–2 | Address and Data Bus Configuration Options | 1–8 |
| 2–1 | MC68F333 Pin Characteristics | 2–1 |
| 2–2 | MC68F333 Driver Types | 2–2 |
| 2–3 | MC68F333 Power Connections | 2–3 |
| 2–4 | MC68F333 Signal Characteristics | 2–3 |
| 2–5 | MC68F333 Signal Functions | 2–5 |
| 3–1 | Show Cycle Enable Bits | 3–5 |
| 3–2 | Bus Monitor Period | 3–6 |
| 3–3 | MODCLK Pin and SWP Bit During Reset | 3–7 |
| 3–4 | Software Watchdog Divide Ratio | 3–8 |
| 3–5 | MODCLK Pin and PTP Bit at Reset | 3–9 |
| 3–6 | Periodic Interrupt Priority | 3–10 |
| 3–7 | Clock Control Multipliers | 3–15 |
| 3–8 | System Frequencies with a 32.768-kHz Reference | 3–17 |
| 3–9 | Clock Control | 3–20 |
| 3–10 | Size Signal Encoding | 3–25 |
| 3–11 | Address Space Encoding | 3–26 |
| 3–12 | Effect of $\overline{\text{DSACK}}$ Signals | 3–27 |
| 3–13 | Operand Alignment | 3–29 |
| 3–14 | $\overline{\text{DSACK}}$, $\overline{\text{BERR}}$, and $\overline{\text{HALT}}$ Assertion Results | 3–40 |
| 3–15 | Reset Sources | 3–48 |
| 3–16 | Basic Configuration Options | 3–48 |
| 3–17 | Bus and I/O Port Pin Functions | 3–49 |
| 3–18 | 16-Bit Data Bus Mode Reset Pin Configuration | 3–52 |
| 3–19 | 8-Bit Expanded Mode Reset Configuration | 3–53 |
| 3–20 | Single-Chip Mode Reset Pin Configuration | 3–54 |
| 3–21 | Module Pin Function | 3–56 |
| 3–22 | Pin Reset States | 3–57 |
| 3–23 | Module Pin States | 3–58 |
| 3–24 | Chip-Select Pin Functions | 3–69 |
| 3–25 | Pin Assignment Field Encoding | 3–70 |
| 3–26 | Block Size Encoding | 3–71 |
| 3–27 | Option Register Function Summary | 3–72 |

# LIST OF TABLES
## (Continued)


| Table | Title | Page |
|---|---|---|
| 3–28 | $\overline{\text{CSBOOT}}$ Base and Option Register Reset Values | 3–76 |
| 3–29 | General-Purpose I/O Ports | 3–77 |
| 3–30 | Port E Pin Assignments | 3–79 |
| 3–31 | Port F Pin Assignments | 3–80 |
| 3–32 | PFPAR *Pin* Encodings | 3–80 |
| 4–1 | Instruction Set Summary | 4–12 |
| 4–2 | Exception Vector Assignments | 4–15 |
| 4–3 | Background Mode Command Summary | 4–19 |
| 5–1 | TCR1 Prescaler Control | 5–12 |
| 5–2 | TCR2 Prescaler Control | 5–13 |
| 5–3 | Host Sequence Code/Host Service Request Code | 5–15 |
| 5–4 | Channel Priority Encodings | 5–16 |
| 6–1 | QSM Pin Function | 6–5 |
| 6–2 | QSPI Pin Function | 6–10 |
| 6–3 | BITS Encoding | 6–19 |
| 6–4 | SCI Pin Function | 6–27 |
| 6–5 | Serial Frame Formats | 6–29 |
| 6–6 | Effect of Parity Checking on Data Size | 6–30 |
| 7–1 | FRZ Field Selection | 7–5 |
| 7–2 | Multiplexer Channels | 7–6 |
| 7–3 | Prescaler Output | 7–8 |
| 7–4 | STS Field Selection | 7–8 |
| 7–5 | ADC Conversion Modes | 7–9 |
| 7–6 | Single-Channel Conversions | 7–12 |
| 7–7 | Multiple-Channel Conversions | 7–13 |
| 8–1 | Array Space Encodings | 8–4 |
| 8–2 | Wait States | 8–5 |
| 8–3 | Reset Vector Assignments | 8–7 |
| 9–1 | SRAM Array Address Space Type | 9–2 |
| A–1 | Maximum Ratings | A–2 |
| A–2 | Thermal Characteristics | A–3 |
| A–3 | Clock Control Timing | A–4 |
| A–4 | DC Characteristics | A–5 |
| A–5 | AC Timing | A–7 |
| A–6 | Background Debugging Mode Timing | A–10 |

# LIST OF TABLES
## (Continued)


| Table | Title | Page |
|---|---|---|
| A–7 | ECLK Bus Timing | A–10 |
| A–8 | QSPI Timing | A–11 |
| A–9 | ADC Maximum Ratings | A–12 |
| A–10 | ADC DC Electrical Characteristics (Operating) | A–13 |
| A–11 | ADC AC Characteristics (Operating) | A–14 |
| A–12 | ADC Conversion Characteristics (Operating) | A–14 |
| B–1 | MC68F333 Ordering Information | B–6 |
| D–1 | MC68F333 Module Address Map | D–1 |
| D–2 | ADC Module Address Map | D–4 |
| D–3 | Flash EEPROM Address Map | D–11 |
| D–4 | SCIM Address Map | D–16 |
| D–5 | TPURAM Address Map | D–32 |
| D–6 | SRAM Address Map | D–34 |
| D–7 | QSM Address Map | D–36 |
| D–8 | TPU Address Map | D–49 |
| D–9 | Register Bit and Field Mnemonics | D–60 |

# SECTION 1
# INTRODUCTION

The MC68F333, a 32-bit highly-integrated microcontroller, combines high-performance data manipulation capabilities with powerful peripheral subsystems. Based on the powerful MC68020, the CPU32 instruction processing module provides enhanced system performance and also uses the extensive software base for the Motorola M68000 Family.

The MC68F333 incorporates a 32-bit CPU (CPU32), a single-chip integration module (SCIM), an 8-channel, 10-bit analog-to-digital converter (ADC), a time processor unit (TPU), a queued serial module (QSM), a 512-byte standby RAM (SRAM), a 3.5-Kbyte RAM with TPU emulation capabilities (TPURAM), and two flash EEPROM modules (FLASH16, FLASH48). These modules are connected to the CPU32 by the intermodule bus (IMB).

The maximum system clock for the MC68F333 is 16.78 MHz. A phase-locked loop circuit synthesizes the clock from a frequency reference. Either a crystal with a nominal frequency of 32.768 kHz or an externally generated signal can be used. System hardware and software support changes in the clock rate during operation. Because the MC68F333 is a fully static design, register and memory contents are not affected by clock rate changes.

High-density complementary metal-oxide semiconductor (HCMOS) architecture makes the basic power consumption of the MC68F333 low. Power consumption can be minimized by stopping the system clock. The instruction set includes a low-power stop (LPSTOP) command that implements this capability efficiently.

## 1.1 MC68F333 Features

The following paragraphs highlight capabilities of each of the microcontroller modules. Each module is discussed separately in a subsequent section of this manual.

### 1.1.1 Single-Chip Integration Module

- External Bus Support
- One Emulation-Specific and Nine Programmable Chip-Select Outputs
- System Clock Based on 32.768-kHz Crystal for Low-Power Operation
- System Protection Logic
- Watchdog Timer, Clock Monitor, and Bus Monitor
- Parallel Ports Option on Address and Data Bus in Single-Chip Mode
- Phase-Locked Loop (PLL) Clock System

### 1.1.2 CPU32

- 32-Bit 68300 Family CPU (Upward Object Code Compatible)
- Virtual Memory Implementation
- Loop Mode of Instruction Execution
- Improved Exception Handling for Controller Applications
- Table Lookup and Interpolate Instruction

### 1.1.3 Intelligent 16-Bit Time Processor Unit

- Dedicated Microengine Operating Independently of CPU32
- 16 Independent Programmable Channels and Pins
- Any Channel can Perform any Microcoded Time Function
- Hardware Scheduler with Three Priority Levels
- Each Channel can be Synchronized to One or Both of the Two 16-Bit Free Running Timer Count Registers (TCR1 and TCR2)

### 1.1.4 Analog-to-Digital Converter

- Eight Channels, Eight Result Registers
- Eight Automated Modes
- Three Result Alignment Modes
- One 8-Bit Digital Input Port, One 8-Bit Digital Output Port

### 1.1.5 Queued Serial Module

- Enhanced Serial Communication Interface with Modulus Baud Rate and Parity
- Queued Serial Peripheral Interface with Two 16-bit Data Queues (One Transmit, One Receive), up to 16 Automatic Transfers, Continuous Cycling, 8 to 16 Bits per Transfer
- Dual Function I/O Port

### 1.1.6 Two Flash EEPROM Memory Modules

- Optional Bootstrap Operation

### 1.1.7 Standby RAM

- 512-Byte Static RAM
- External Standby Voltage Supply Input

### 1.1.8 Standby RAM with TPU Emulation Capability

- 3.5-Kbyte Static RAM
- Can be Used as Standby RAM or TPU Microcode Emulation RAM
- External Standby Voltage Supply Input

## 1.2 Intermodule Bus

The intermodule bus (IMB) is a standardized bus developed to facilitate design of modular microcontrollers. The modules in the MC68F333 communicate with one another and with external components via the IMB. The IMB uses 24 address and 16 data lines.

## 1.3 System Block Diagram and Pin Assignment Diagrams

Figure 1–1 is a functional diagram of the MC68F333. Although diagram blocks represent the relative size of the physical modules, there is not a one-to-one correspondence between location and size of blocks in the diagram and location and size of integrated-circuit modules. Figure 1–2 is a pin assignment drawing based on a 160-pin plastic quad flat package. Refer to **APPENDIX B MECHANICAL DATA AND ORDERING INFORMATION** for package dimensions. All pin functions and signal names are shown in these drawings. Refer to **SECTION 2 SIGNAL DESCRIPTIONS** for pin and signal descriptions.

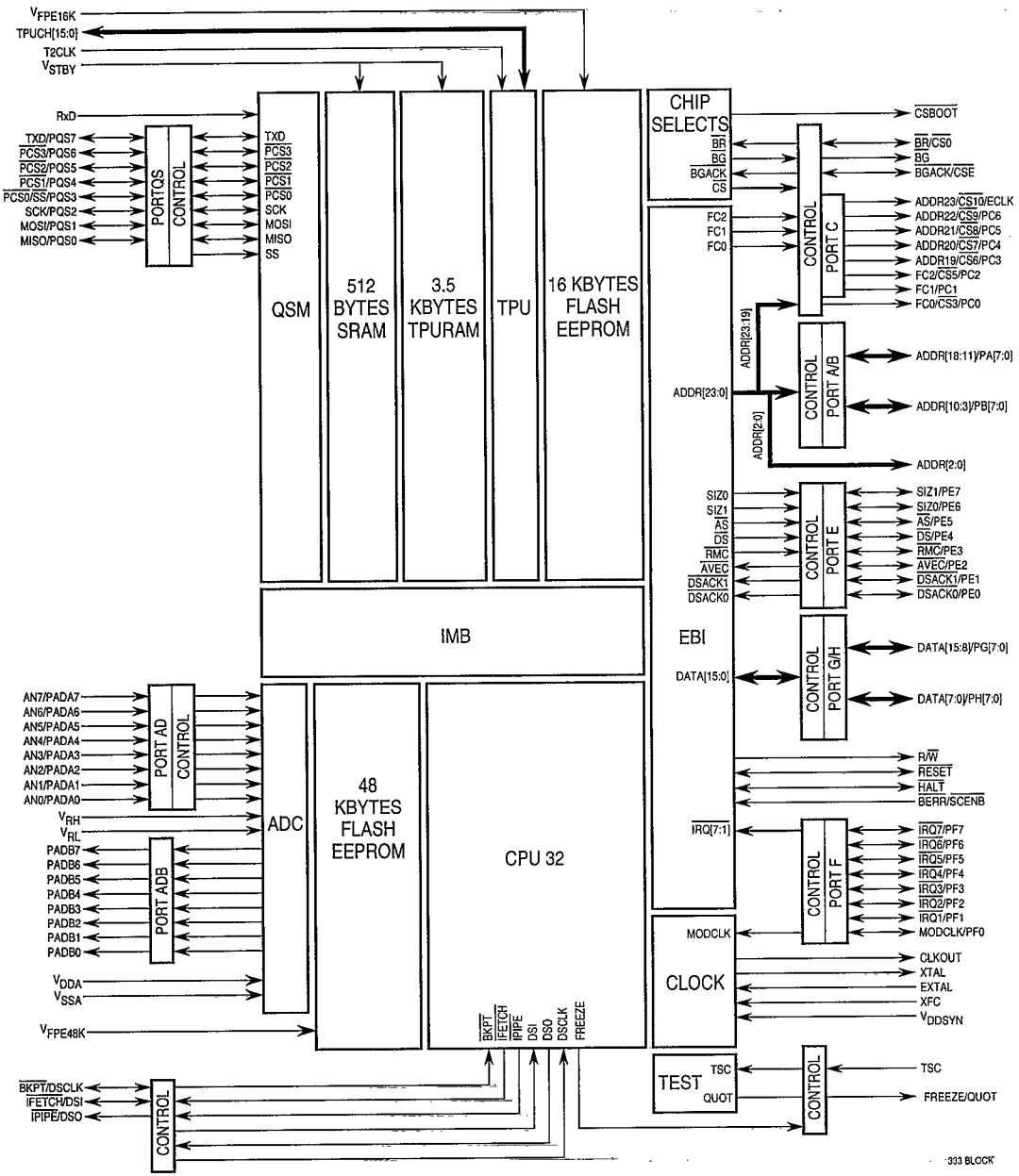



**Figure 1–1. MC68F333 Block Diagram**

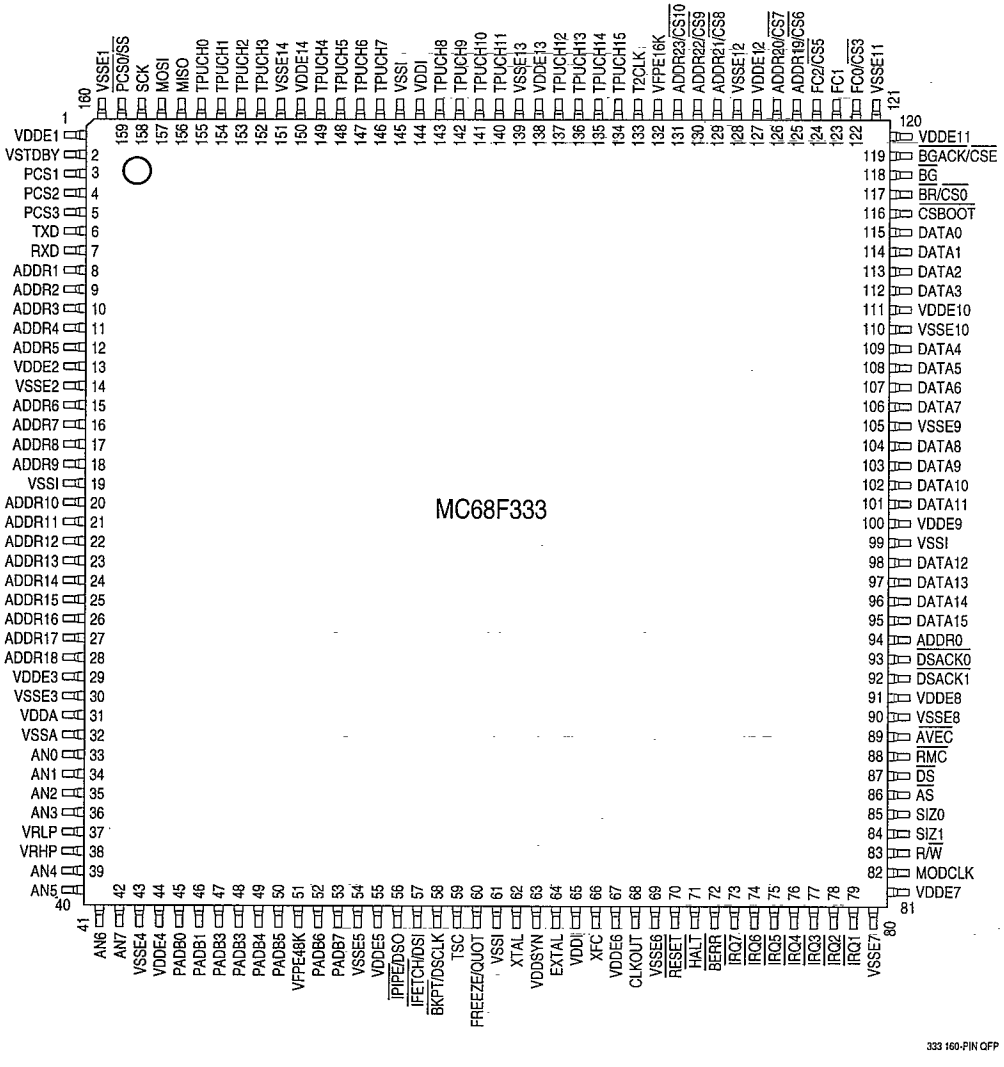
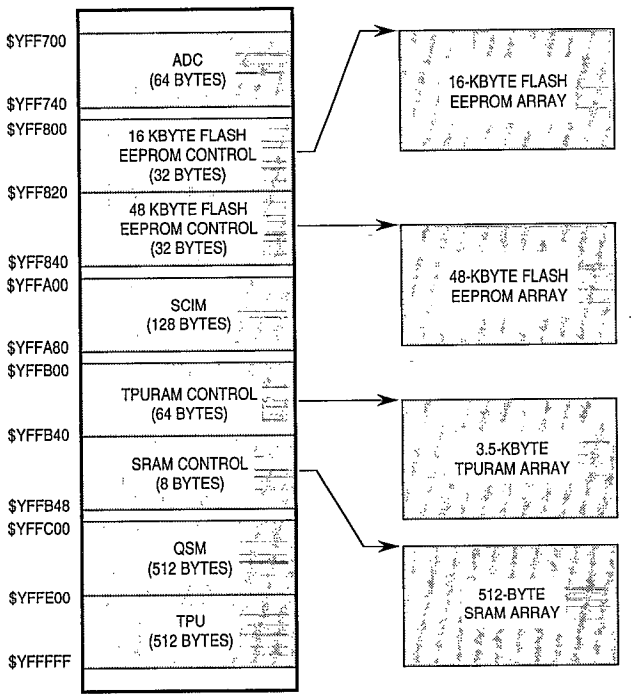



**Figure 1–2. MC68F333 Pin Assignment**

## 1.4 Memory Map

Figure 1–3 is the MC68F333 memory map. The addresses of the TPURAM, SRAM, and the two EEPROM arrays are assigned in the control registers of the corresponding control blocks.

In Figure 1–3, IMB ADDR[23:20] are represented by the letter Y. The value represented by Y determines the base address of MCU module control registers. In the MC68F333, Y is equal to M111, where M is the logic state of the module mapping (MM) bit in the single-chip integration module configuration register (SCIMCR).




Y = M111, where M is the module mapping (MM) bit in the SCIM module configuration register (Y = $7 or Y = $F).

333 MEM MAP

**Figure 1–3. MC68F333 Memory Map**

## 1.5 Reset

The following information is a concise reference only. MC68F333 reset is a complex operation. To understand MCU operation during and after reset, refer to **SECTION 3 SINGLE-CHIP INTEGRATION MODULE**, paragraph **3.4 Reset**.

### 1.5.1 SCIM Reset

The logic states of certain pins during reset determine SCIM operating configuration. During reset, the SCIM reads pin configuration from DATA[11:2] and DATA0, internal module configuration from DATA[15:12], and basic operating information from $\overline{\text{BERR}}$, MODCLK, $\overline{\text{BKPT}}$, and DATA1. These pins are normally pulled high internally during reset, causing the MCU to default to a specific configuration. However, the user can pull the desired pins low during reset to achieve alternate configurations.

Basic operating options include system clock selection, background mode disable/enable, and external bus configuration. The SCIM supports three external bus configurations: fully-expanded operation with a 24-bit address bus and 16-bit data bus with chip selects, single-chip operation with no external address or data bus, and partially-expanded operation with a 24-bit address bus and an 8-bit external data bus.

**Table 1–1. Basic Configuration Options**

| Select Pin | Pin High at Reset | Pin Low at Reset |
|---|---|---|
| MODCLK | Synthesized System Clock | External System Clock |
| $\overline{\text{BKPT}}$ | Background Mode Disabled | Background Mode Enabled |
| $\overline{\text{BERR}}$ | Expanded Mode | Single-Chip Mode |
| DATA1 (if $\overline{\text{BERR}}$ = 1) | 8-Bit Expanded Mode | 16-Bit Expanded Mode |

$\overline{\text{BERR}}$, $\overline{\text{BKPT}}$, and MODCLK do not have internal pull-ups and must be driven to the desired state during reset.

External bus configuration determines which address and data bus lines are used and which general-purpose I/O ports are available. Table 1–2 summarizes address and data bus configuration.

**Table 1–2. Address and Data Bus Configuration Options**

| Mode | Address Bus | Data Bus | I/O Ports |
|---|---|---|---|
| 16-Bit Expanded | ADDR[18:3] | DATA[15:0] | — |
| 8-Bit Expanded | ADDR[18:3] | DATA[15:8] | DATA[7:0] = Port H |
| Single Chip | None | None | ADDR[18:11] = Port A<br>ADDR[10:3] = Port B<br>DATA[15:8] = Port G<br>DATA[7:0] = Port H |

Many pins on the MC68F333, including data and address bus pins, have multiple functions. Reset value for these pins depends on external bus configuration. In expanded mode, the value of DATA[11:0] coming out of reset determines the function of these pins. For details on pin configuration for each external bus configuration, refer to **SECTION 3 SINGLE-CHIP INTEGRATION MODULE**, paragraph **3.4 Reset**.

### 1.5.2 MCU Module Pin Function During Reset

Generally, module pins default to port functions, and input/output ports are set to input state. This is accomplished by disabling pin functions in the appropriate control registers, and by clearing the appropriate port data direction registers.

# SECTION 2
# SIGNAL DESCRIPTIONS

This section describes the characteristics of MC68F333 pins and provides brief descriptions of signal functions. For more detailed discussion of signal function, refer to the section that discusses the module with which the signal is associated. Refer to Figure 2–1 for signal pin assignments.

## 2.1 Pin Characteristics

Table 2–1 describes MC68F333 pin characteristics. All input pins detect CMOS logic levels. All I/O pins can be put in a high-impedance state, but the method of doing this differs depending upon pin function. An entry in the Discrete I/O column indicates that the pin has an alternate I/O function. Port designation is given when it applies. Refer to Table 2–2 for a description of output drivers. Table 2–3 summarizes the MC68F333 power connections.

**Table 2–1. MC68F333 Pin Characteristics**

| Pin Mnemonic | Output Driver | Input Synchronized | Input Hysteresis | Discrete I/O | Port Designation |
|---|---|---|---|---|---|
| ADDR23/$\overline{CS10}$/ECLK | A | Y | N | — | — |
| ADDR[22:19]/$\overline{CS[9:6]}$ | A | Y | N | O | PC[6:3] |
| ADDR[18:3] | A | Y | Y | I/O | A[7:0], B[7:0] |
| ADDR[2:0] | A | Y | N | — | — |
| AN[7:0] | — | Y[1] | Y | I | PADA[7:0] |
| $\overline{AS}$ | B | Y | Y | I/O | PE5 |
| $\overline{AVEC}$ | B | Y | N | I/O | PE2 |
| $\overline{BERR}$ | B | Y[2] | N | — | — |
| $\overline{BG}$ | B | — | — | — | — |
| $\overline{BGACK}$/$\overline{CSE}$ | B | Y | N | — | — |
| $\overline{BKPT}$/DSCLK | — | Y | Y | — | — |
| $\overline{BR}$/$\overline{CS0}$ | B | Y | N | — | — |
| CLKOUT | A | — | — | — | — |
| $\overline{CSBOOT}$ | B | — | — | — | — |
| DATA[15:0] | Aw | Y[1] | Y | I/O | PG[7:0], PH[7:0] |
| $\overline{DS}$ | B | Y | Y | I/O | PE4 |
| $\overline{DSACK1}$ | B | Y | N | I/O | PE1 |

## Table 2–1. MC68F333 Pin Characteristics (Continued)

| Pin Mnemonic | Output Driver | Input Synchronized | Input Hysteresis | Discrete I/O | Port Designation |
|---|---|---|---|---|---|
| $\overline{\text{DSACK0}}$ | B | Y | N | I/O | PE0 |
| DSI/$\overline{\text{IFETCH}}$ | A | Y | Y | — | — |
| DSO/$\overline{\text{IPIPE}}$ | A | — | — | — | — |
| EXTAL | — | — | Special | — | — |
| FC[2:0]/$\overline{\text{CS}}$[5:3] | A | Y | N | O | PC[2:0] |
| FREEZE/QUOT | A | — | — | — | — |
| $\overline{\text{HALT}}$ | Bo | Y[2] | N | — | — |
| $\overline{\text{IRQ}}$[7:1] | B | Y | Y | I/O | PF[7:1] |
| MISO | Bo | Y[1] | Y | I/O | QS0 |
| MODCLK | B | Y[1] | Y | I/O | PF0 |
| MOSI | Bo | Y[1] | Y | I/O | QS1 |
| PADB[7:0] | A | — | — | O | PADB[7:0] |
| PCS0/$\overline{\text{SS}}$ | Bo | Y[1] | Y | I/O | QS3 |
| PCS[3:1] | Bo | Y[1] | Y | I/O | QS[6:4] |
| R/$\overline{\text{W}}$ | A | Y | N | — | — |
| $\overline{\text{RESET}}$ | Bo | Y | Y | — | — |
| $\overline{\text{RMC}}$ | B | Y | N | I/O | PE3 |
| RXD | A | N | Y | — | — |
| SCK | Bo | Y[1] | Y | I/O | QS2 |
| SIZ[1:0] | B | Y | Y | I/O | PE[7:6] |
| TPUCH[15:0] | A | Y | Y | — | — |
| TSC | — | Y | Y | — | — |
| TXD | Bo | Y | Y | I/O | QS7 |
| T2CLK | A | Y | Y | — | — |
| XFC | — | — | — | — | — |
| XTAL | — | — | — | — | — |

1. DATA[15:0] are synchronized during reset only. MODCLK and ADC pins are synchronized only when used as input port pins.
2. $\overline{\text{BERR}}$, HALT only synchronized if late $\overline{\text{BERR}}$ or HALT.

## Table 2–2. MC68F333 Driver Types

| Type | I/O | Description |
|---|---|---|
| A | O | Output-only signals that are always driven. No external pull-up required. |
| Aw | O | Type A output with weak P-channel pull-up during reset. |
| B | O | Three-state output that includes circuitry to pull up asserted 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. Assertion of TSC always places these pins in a high-impedance state; in addition, many bus control outputs are placed in a high-impedance state when the bus is granted to an external master. |
| Bo | O | Type B output that can be operated in an open-drain mode. |

**Table 2–3. MC68F333 Power Connections**

| Pin | Description |
|---|---|
| $V_{STBYRAM}$ | Standby RAM Power |
| $V_{DDSYN}$ | Clock Synthesizer Power |
| $V_{DDA}/V_{SSA}$ | A/D Converter Power |
| $V_{RH}/V_{RL}$ | A/D Reference Voltage |
| $V_{SSE}/V_{DDE}$[1] | External Periphery Power (Source and Drain) |
| $V_{SSI}/V_{DDI}$ | Internal Module Power (Source and Drain) |
| $V_{FPE16K}$, $V_{FPE48K}$ | External Flash EEPROM Programming/Erase Power |

1. Figure 2–1 shows which module pins are powered by specific $V_{SSE}/V_{DDE}$ pairs.

## 2.2 Signal Characteristics and Functions

Table 2–4 summarizes the active state and signal type of each MC68F333 signal and the module with which it is associated. Table 2–5 summarizes signal functions. Notice that more than one signal may be associated with a given pin.

**Table 2–4. MC68F333 Signal Characteristics**

| Signal Name | MCU Module | Signal Type | Active State |
|---|---|---|---|
| ADDR[23:0] | SCIM | Bus | 1/0 |
| AN[7:0] | ADC | Input | — |
| $\overline{AS}$ | SCIM | Output | 0 |
| $\overline{AVEC}$ | SCIM | Input | 0 |
| $\overline{BERR}$ | SCIM | Input | 0 |
| $\overline{BG}$ | SCIM | Output | 0 |
| $\overline{BGACK}$ | SCIM | Input | 0 |
| $\overline{BKPT}$ | CPU32 | Input | 0 |
| $\overline{BR}$ | SCIM | Input | 0 |
| CLKOUT | SCIM | Output | — |
| $\overline{CS}$[10:5], $\overline{CS3}$, $\overline{CS0}$ | SCIM | Output | 0 |
| $\overline{CSBOOT}$ | SCIM | Output | 0 |
| $\overline{CSE}$ | SCIM | Output | 0 |
| DATA[15:0] | SCIM | Bus | 1/0 |
| $\overline{DS}$ | SCIM | Output | 0 |
| $\overline{DSACK1}$, $\overline{DSACK0}$ | SCIM | Input | 0 |
| DSCLK | CPU32 | Input | Serial Clock |
| DSI | CPU32 | Input | Serial Data |
| DSO | CPU32 | Output | Serial Data |

## Table 2–4. MC68F333 Signal Characteristics (Continued)

| Signal Name | MCU Module | Signal Type | Active State |
|---|---|---|---|
| ECLK | SCIM | Output | — |
| EXTAL | SCIM | Input | — |
| FC[2:0] | SCIM | Output | 1/0 |
| FREEZE | SCIM | Output | 1 |
| $\overline{\text{HALT}}$ | SCIM | Input/Output | 0 |
| $\overline{\text{IFETCH}}$ | CPU32 | Output | 0 |
| $\overline{\text{IPIPE}}$ | CPU32 | Output | 0 |
| $\overline{\text{IRQ[7:1]}}$ | SCIM | Input | 0 |
| MISO | QSM | Input/Output | — |
| MODCLK | SCIM | Input | — |
| MOSI | QSM | Input/Output | — |
| PADB[7:0] | ADC | Output | — |
| PCS[3:0] | QSM | Input/Output | — |
| QUOT | SCIM | Output | — |
| R/$\overline{\text{W}}$ | SCIM | Output | 1/0 |
| $\overline{\text{RESET}}$ | SCIM | Input/Output | 0 |
| $\overline{\text{RMC}}$ | SCIM | Output | 0 |
| RXD | QSM | Input | — |
| SCK | QSM | Input/Output | — |
| SIZ0/SIZ1 | SCIM | Output | 1 |
| $\overline{\text{SS}}$ | QSM | Output | 0 |
| T2CLK | TPU | Input | — |
| TPUCH[15:0] | TPU | Input/Output | — |
| TSC | SCIM | Input | 1 |
| TXD | QSM | Output | — |
| XFC | SCIM | Input | — |
| XTAL | SCIM | Output | — |

## Table 2–5. MC68F333 Signal Functions

| Mnemonic | Signal Name | Function |
|---|---|---|
| ADDR[23:0] | Address Bus | 24-bit address bus used by CPU32 |
| AN[7:0] | ADC Analog Input | Inputs to ADC multiplexer |
| $\overline{AS}$ | Address Strobe | Indicates that a valid address is on the address bus |
| $\overline{AVEC}$ | Autovector | Requests an automatic vector during interrupt acknowledge |
| $\overline{BERR}$ | Bus Error | Indicates that a bus error has occurred |
| $\overline{BG}$ | Bus Grant | Indicates that the MCU has relinquished the bus |
| $\overline{BGACK}$ | Bus Grant Acknowledge | Indicates that an external device has assumed bus mastership |
| $\overline{BKPT}$ | Breakpoint | Signals a hardware breakpoint to the CPU |
| $\overline{BR}$ | Bus Request | Indicates that an external device requires bus mastership |
| CLKOUT | System Clockout | System clock output |
| $\overline{CS[10:5]}$, $\overline{CS3}$, $\overline{CS0}$ | Chip Selects | Select external devices at programmed addresses |
| $\overline{CSBOOT}$ | Boot Chip Select | Chip select for external boot startup ROM |
| $\overline{CSE}$ | Emulator Chip Select | Select external emulation device at internally-mapped address. Used to emulate I/O ports. |
| DATA[15:0] | Data Bus | 16-bit data bus |
| $\overline{DS}$ | Data Strobe | During a read cycle, indicates when it is possible for an external device to place data on the data bus. During a write cycle, indicates that valid data is on the data bus. |
| $\overline{DSACK[1:0]}$ | Data and Size Acknowledge | Provide asynchronous data transfers and dynamic bus sizing |
| DSI, DSO, DSCLK | Development Serial In, Out, Clock | Serial I/O and clock for background debug mode |
| ECLK | E-Clock | External M6800 bus clock output |
| EXTAL, XTAL | Crystal Oscillator | Connections for clock synthesizer circuit reference; a crystal or an external oscillator can be used |
| FC[2:0] | Function Codes | Identify processor state and current address space |
| FREEZE | Freeze | Indicates that the CPU has entered background mode |
| $\overline{HALT}$ | Halt | Suspend external bus activity |
| $\overline{IFETCH}$ | Instruction Fetch | Identifies bus cycles in which operand is loaded into pipeline |
| $\overline{IPIPE}$ | Instruction Pipeline | Indicates instruction pipeline activity |
| $\overline{IRQ[7:1]}$ | Interrupt Request Level | Provide prioritized interrupts to the CPU |
| MISO | Master In Slave Out | Serial data input to QSPI or output from QSPI |
| MODCLK | Clock Mode Select | Selects the source and type of system clock |
| MOSI | Master Out Slave In | Serial output from QSPI or input to QSPI |
| PADA[7:0] | ADC Digital Input | ADC input port |
| PADB[7:0] | ADC Digital Output | ADC output port |
| PCS[3:0] | Peripheral Chip-Selects | Provide QSPI chip selects |
| QUOT | Quotient Out | Provides the quotient bit of the polynomial divider |

## Table 2–5. MC68F333 Signal Functions (Continued)

| Mnemonic | Signal Name | Function |
|---|---|---|
| $\overline{\text{RESET}}$ | Reset | System reset |
| $\overline{\text{RMC}}$ | Read-Modify-Write Cycle | Indicates an indivisible read-modify-write instruction |
| R/$\overline{\text{W}}$ | Read/Write | Indicates the direction of data transfer on the bus |
| RXD | Receive Data | Serial data input to SCI |
| SCK | QSPI Serial Clock | Clock output from QSPI in master mode; clock input to QSPI in slave mode |
| SIZ[1:0] | Size | Indicates the number of bytes to be transferred during a bus cycle |
| $\overline{\text{SS}}$ | Slave Select | Selects the QSPI when in slave mode |
| T2CLK | TCR2 Clock | TPU clock input |
| TPUCH[15:0] | TPU I/O Channels | Bidirectional TPU channels |
| TSC | Three-State Control | Places all output drivers in a high-impedance state |
| TXD | Transmit Data | Serial data output from SCI |
| XFC | External Filter Capacitor | Connection for external phase-locked loop filter capacitor |

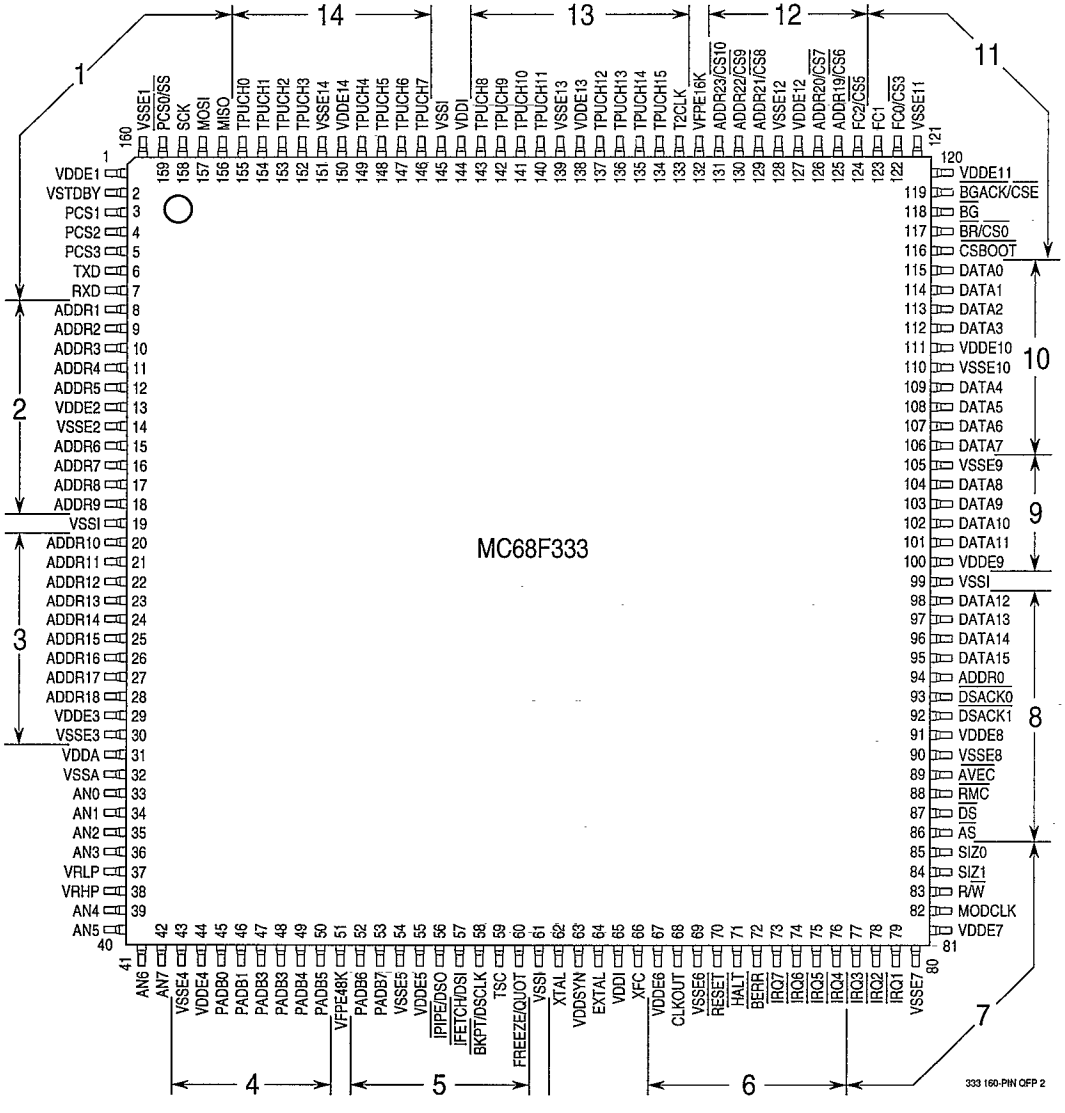



**Figure 2–1. MC68F333 Signal Pin Assignments**

# SECTION 3
# SINGLE-CHIP INTEGRATION MODULE

The single-chip integration module (SCIM) consists of the following functional blocks: system configuration and protection, system clock, external bus interface, chip-select block, and system test block. A functional block diagram of the SCIM is shown in Figure 3–1.

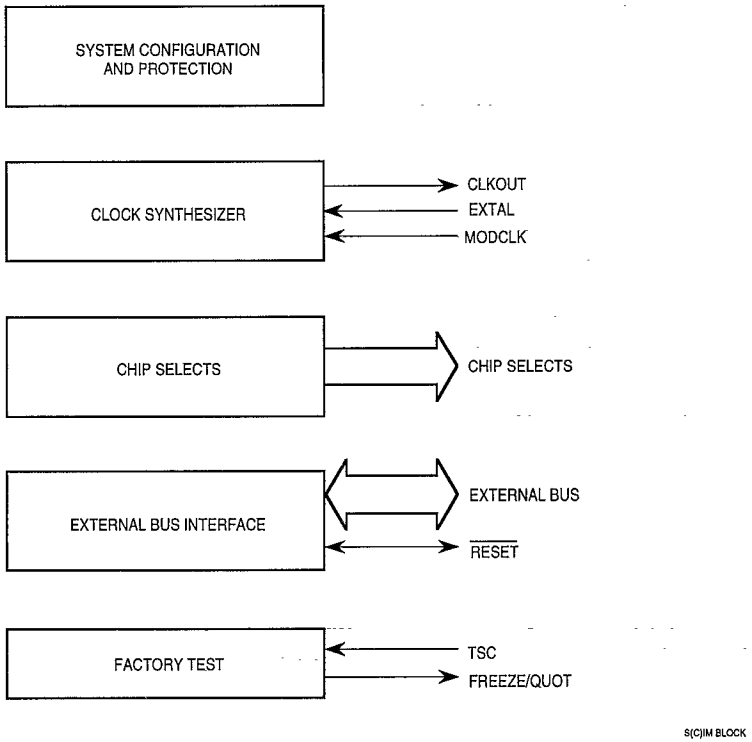



**Figure 3–1. Single-Chip Integration Module Block Diagram**

The system configuration and protection block controls system configuration and provides bus and software watchdog monitors. It also includes a periodic interrupt generator to support the execution of time-critical control routines. **3.1 System Configuration and Protection** describes this block.



The system clock generates clock signals used by the SCIM, other IMB modules, and external devices. **3.2 System Clock** describes this functional block.

The external bus interface handles the transfer of information between IMB modules and external address space. The external bus interface is described in **3.3 External Bus Interface.**

The chip-select block provides nine chip-select signals. Each chip-select signal has an associated base register and option register that contain the programmable characteristics of that chip select. An additional chip-select signal is available for emulation support. **3.6 Chip Selects** describes this block.

The system test block incorporates hardware necessary for testing the MCU. It is used to perform factory tests. Its use in normal applications is not supported.

**3.4 Reset** explains pin and mode configuration and other aspects of reset operation. **3.5 Interrupts** explains interrupt processing on the MC68F333. **3.7 General Purpose I/O** describes the general-purpose I/O ports available on the SCIM.

Refer to **APPENDIX D REGISTER SUMMARY** for SCIM register diagrams and the SCIM memory map.

## 3.1 System Configuration and Protection

The system configuration and protection submodule controls module configuration, preserves reset status, monitors internal activity, and provides periodic interrupt generation. The following registers are involved in system configuration and protection: the SCIM configuration register (SCIMCR), system protection control register (SYPCR), periodic interrupt timer register (PITR), periodic interrupt control register (PICR), and software watchdog service register (SWSR). Refer to **APPENDIX D REGISTER SUMMARY** for register diagrams and bit/field definitions.

Figure 3–2 is a block diagram of the submodule.

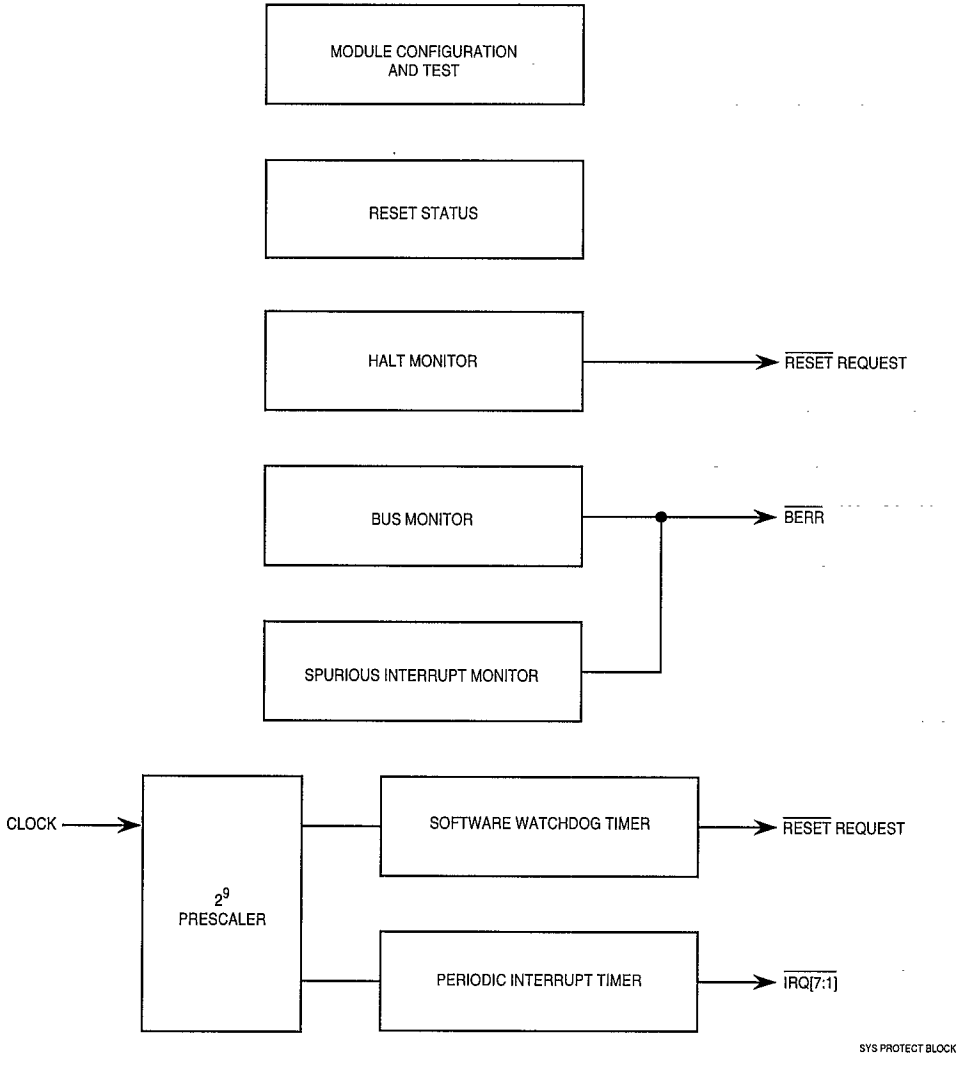



**Figure 3–2. System Configuration and Protection**

### 3.1.1 Module Mapping

Control registers for all the modules in the microcontroller are mapped into a 4-Kbyte block. The state of the module mapping bit (MM) in the SCIMCR determines where the control register block is located in the system memory

map. When MM = 0, register addresses range from $7FF000 to $7FFFFF; when MM = 1, register addresses range from $FFF000 to $FFFFFF.



### 3.1.2 Privilege Levels

To protect system resources, the CPU32 can operate at either the user or supervisor privilege level. Access to most SCIM registers is permitted only when the CPU is operating at the supervisor privilege level. The remaining SCIM registers are programmable to permit supervisor access only or to permit access when the CPU is operating at either the supervisor or user privilege level. If the SUPV bit in the SCIMCR is set, access to SCIM registers is permitted only when the CPU32 is operating at the supervisor level. If SUPV is cleared, then access to certain SCIM registers is permitted when the CPU is operating at either the supervisor or user privilege level. The module memory maps in **APPENDIX D REGISTER SUMMARY** indicate which registers are programmable to allow access from either privilege level.

### 3.1.3 Response to FREEZE Assertion

When the CPU enters background debugging mode, it suspends instruction execution and asserts the internal FREEZE signal. The CPU enters background debugging mode if a breakpoint occurs while background mode is enabled. Refer to the *CPU32 Reference Manual* (CPU32RM/AD) for a discussion of background debugging mode.

Two bits in the SCIMCR determine how the SCIM responds to FREEZE assertion. Setting the freeze bus monitor (FRZBM) bit in the SCIMCR disables the bus monitor when FREEZE is asserted. Setting the freeze software watchdog (FRZSW) bit disables the software watchdog and the periodic interrupt timer when FREEZE is asserted. If these bits are cleared when FREEZE is asserted, the bus monitor, software watchdog, and periodic interrupt timer continue to operate normally.

FREEZE assertion has no affect on the halt monitor or spurious interrupt monitor. They continue to operate normally.

### 3.1.4 Interrupt Arbitration

Each module that can generate interrupt requests, including the SCIM, has an interrupt arbitration (IARB) field in its module configuration register. Each IARB field must have a different value. During an interrupt-acknowledge cycle, IARB fields permit arbitration among simultaneous interrupts of the same priority level.

For arbitration to take place, an IARB field must have a non-zero value. If an interrupt request from a module with an IARB field value of %0000 is recognized, the CPU32 processes a spurious interrupt exception.

Because the SCIM routes external interrupt requests to the CPU32, the SCIM IARB field value is used for arbitration between internal and external interrupts of the same priority. The reset value of IARB for the SCIM is %1111, and the reset IARB value for all other modules is %0000, which prevents SCIM interrupts from being discarded during initialization. Refer to **3.5 Interrupts** for a discussion of interrupt arbitration.



### 3.1.5 Single-Chip Operation Support

The SCIMCR contains three bits that support single-chip operation. Setting the CPU development support disable bit (CPUD) disables the $\overline{\text{IPIPE}}$ and $\overline{\text{IFETCH}}$ instruction tracking pins (by placing them in a high-impedance state) whenever the FREEZE signal is not asserted. When CPUD is cleared to zero, these pins operate normally.

Setting the address bus disable bit (ABD) disables ADDR[2:0] by placing the pins in a high-impedance state. During single-chip operation, the ADDR[23:3] pins are configured for discrete input or output, and ADDR[2:0] should normally be disabled. Setting the R/$\overline{\text{W}}$ disable bit (RWD) disables the R/$\overline{\text{W}}$ pin. This pin is not normally used during single-chip operation.

The reset states of the CPUD, ABD, and RWD bits is one if $\overline{\text{BERR}}$ is held low during reset (configuring the MCU for single-chip operation). If $\overline{\text{BERR}}$ is held high during reset, the reset state of each of these bits is zero.

### 3.1.6 Show Internal Cycles

A show cycle allows internal bus transfers to be monitored externally. The SHEN field in the SCIMCR determines what the external bus interface does during internal transfer operations. Table 3–1 shows whether data is driven externally, and whether external bus arbitration can occur. Refer to **3.3.11 Show Cycles** for more information.

**Table 3–1. Show Cycle Enable Bits**

| SHEN | Action |
|---|---|
| 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 |

### 3.1.7 Factory Test Mode

The internal IMB can serve as a slave to an external master for direct module testing. This test mode is reserved for factory test. Slave mode is enabled by holding DATA11 low during reset. The read-only slave enabled (SLVEN) bit shows the reset state of DATA11.

3

### 3.1.8 Bus Monitor

The internal bus monitor checks data and 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) field in the system protection control register (SYPCR). Table 3–2 shows the periods allowed.

**Table 3–2. Bus Monitor Period**

| BMT | 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 CPU 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.

### 3.1.9 Halt Monitor

The halt monitor responds to an assertion of the $\overline{\text{HALT}}$ signal on the internal bus. Refer to **3.3.8.2 Double Bus Faults** for more information. Halt monitor reset can be inhibited by the halt monitor enable (HME) bit in the SYPCR.

### 3.1.10 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 **3.5 Interrupts** for further information. Refer to **SECTION 4 CENTRAL PROCESSING UNIT** for information about interrupt exception processing.



### 3.1.11 Software Watchdog

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

Perform a software watchdog service sequence as follows:

a. Write $55 to SWSR.
b. Write $AA to SWSR.

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

Software watchdog clock rate is affected by the software watchdog prescale (SWP) and software watchdog timing (SWT) fields in the SYPCR.

SWP determines system clock prescaling for the watchdog timer. One of two options, either no prescaling or prescaling by a factor of 512, can be selected. The value of SWP is determined by the state of the MODCLK pin during reset, as shown in Table 3–3. System software can change SWP value.

**Table 3–3.**
**MODCLK Pin and**
**SWP Bit During Reset**

| MODCLK | SWP |
|---|---|
| 0 (External Clock) | 1 (÷ 512) |
| 1 (Internal Clock) | 0 (÷ 1) |

The SWT field selects the divide ratio used to establish software watchdog timeout period. Timeout period is given by the following equations.

Timeout Period = 1/(EXTAL Frequency/Divide Ratio)

or

Timeout Period = Divide Ratio/EXTAL Frequency

Table 3–4 shows the ratio for each combination of SWP and SWT bits. When SWT[1:0] are modified, a watchdog service sequence must be performed before the new timeout period can take effect.

**Table 3–4. Software Watchdog Divide Ratio**

| SWP | SWT | 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 3–3 is a block diagram of the software watchdog timer and the clock control for the periodic interrupt timer.

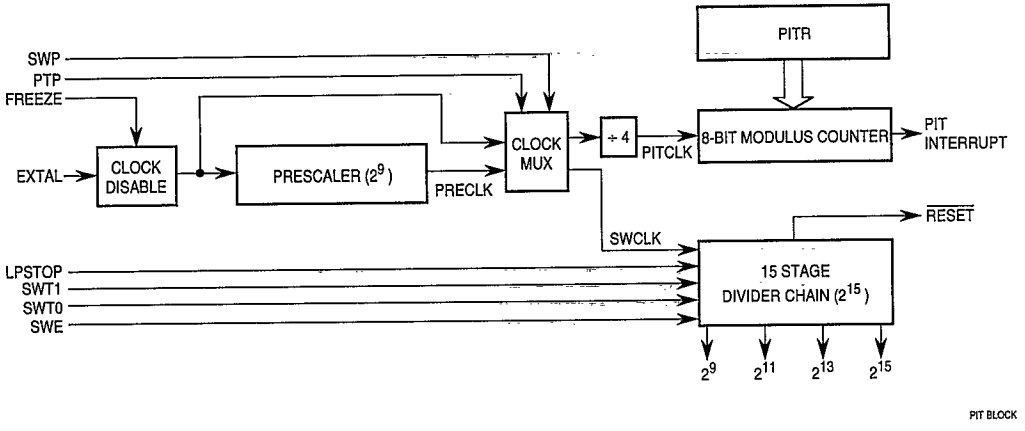



**Figure 3–3. Periodic Interrupt Timer and Software Watchdog Timer**

### 3.1.12 Periodic Interrupt Timer

The periodic interrupt timer 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 timer consists of a prescaler, a modulus counter, and registers that determine interrupt timing, priority and vector assignment. For more information about interrupt exception processing refer to **SECTION 4 CENTRAL PROCESSING UNIT**.

The periodic interrupt modulus counter is clocked by a signal derived from the buffered crystal oscillator (EXTAL) input pin unless an external frequency source is used. The value of the periodic timer prescaler (PTP) bit in the periodic interrupt timer register (PITR) determines system clock prescaling for the watchdog 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 MODCLK pin during reset, as shown in Table 3–5. System software can change the PTP value.

**Table 3–5. MODCLK Pin and PTP Bit at Reset**

| MODCLK | PTP |
|---|---|
| 0 (External Clock) | 1 (÷ 512) |
| 1 (Internal Clock) | 0 (÷ 1) |

Either clock signal (EXTAL or EXTAL ÷ 512) is divided by four before driving the modulus counter (PITCLK). The modulus counter is initialized by writing a value to the periodic timer modulus (PITM) field in the PITR. A zero value turns off the periodic timer. When the modulus counter value reaches zero, an interrupt is generated. The modulus counter is then reloaded with the value in PITM and counting repeats. If a new value is written to the PITR, it is loaded into the modulus counter when the current count is completed.

Use the following expression to calculate timer period.

$$\text{PIT Period} = \frac{(\text{PIT Modulus})(\text{Prescaler Value})(4)}{\text{EXTAL Frequency}}$$

Interrupt priority and vectoring are determined by the values of the periodic interrupt request level (PIRQL) and periodic interrupt vector (PIV) fields in the periodic interrupt control register (PICR).

The content of PIRQL is compared to the CPU32 interrupt priority mask to determine whether the interrupt is recognized. Table 3–6 shows the priority level associated with each PIRQL value. Because of SCIM hardware prioritization, a PIT interrupt is serviced before an external IRQ of the same priority. The periodic timer continues to run when the interrupt is disabled.

**Table 3–6. Periodic Interrupt Priority**

| PIRQL | 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 multiplied by two to form the vector offset, which is added to $0000 to obtain the address of the vector. Reset value of the PIV field is $0F, which generates the uninitialized interrupt vector.

### 3.1.13 Low-Power STOP Operation

When the CPU 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 the SYNCR, and the MC68F333 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, 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 low-power stop ends. The watchdog is not reset by low-power stop. A service sequence must be performed to reset the timer.

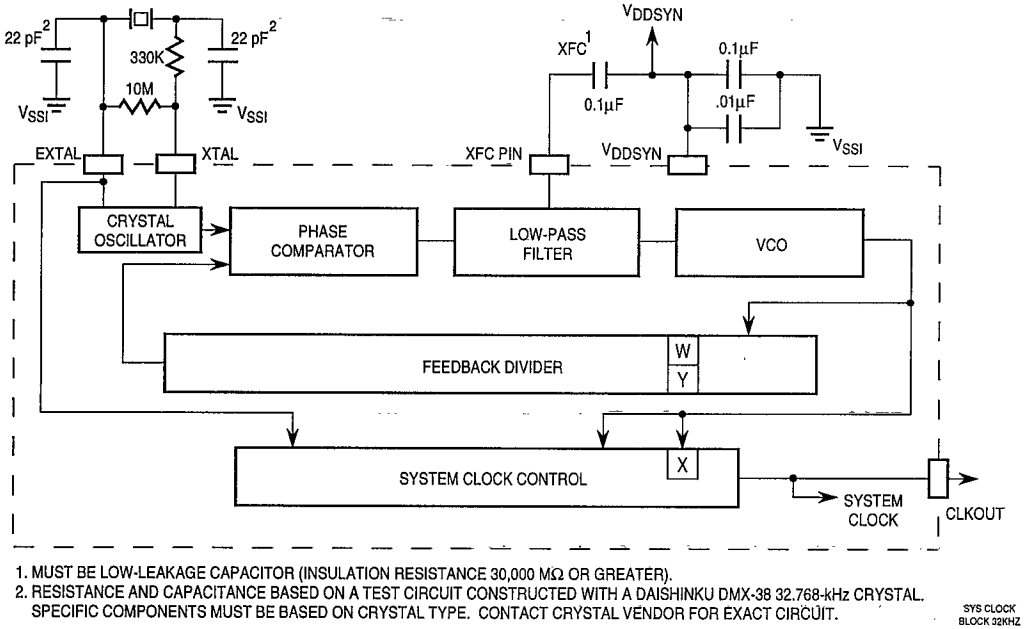
The periodic interrupt timer does not respond to the LPSTOP instruction, but continues to run at the same frequency as EXTAL during LPSTOP. A PIT interrupt can bring the MCU out of the low-power stop condition if it has a higher priority than the interrupt mask value stored in the clock control logic when low-power stop is initiated. To stop the periodic interrupt timer, the PITR must be loaded with a zero value before the LPSTOP instruction is executed.

## 3.2 System Clock

The system clock in the SCIM provides timing signals for the IMB modules and for an external peripheral bus. Because the MC68F333 is a fully static design, register and memory contents are not affected when the clock rate changes. System hardware and software support changes in the clock rate during operation.

An internal phase-locked loop can synthesize the clock from either an internally or externally generated reference frequency or the clock signal can be input from an external source.

Figure 3–4 is a block diagram of the system clock, with a Daishinku DMX-38 32.768-kHz crystal providing the reference signal.




1. MUST BE LOW-LEAKAGE CAPACITOR (INSULATION RESISTANCE 30,000 MΩ OR GREATER).
2. RESISTANCE AND CAPACITANCE BASED ON A TEST CIRCUIT CONSTRUCTED WITH A DAISHINKU DMX-38 32.768-kHz CRYSTAL. SPECIFIC COMPONENTS MUST BE BASED ON CRYSTAL TYPE. CONTACT CRYSTAL VENDOR FOR EXACT CIRCUIT.

SYS CLOCK BLOCK 32KHZ

**Figure 3–4. System Clock Block Diagram**

### 3.2.1 Clock Sources

The state of the clock mode (MODCLK) pin during reset determines clock source. When MODCLK is held high during reset, the clock synthesizer generates a clock signal from either an internal or an external reference frequency. The clock synthesizer control register (SYNCR) determines operating frequency and mode of operation. When MODCLK is held low during reset, the clock synthesizer is disabled and an external system clock signal must be applied. SYNCR control bits have no effect.

A reference crystal must be connected between the EXTAL and XTAL pins to use the internal oscillator. MC68F333 clock synthesizer specifications (Table A–3 in **APPENDIX A ELECTRICAL CHARACTERISTICS**) are based on a typical 32.768-kHz crystal.

If an external system clock signal is applied through the EXTAL pin, the XTAL pin must be left floating. External reference signal frequency must be less than or equal to the maximum specified reference frequency. External system clock signal frequency must be less than or equal to maximum specified system clock frequency.

When an external system clock signal is applied, the duty cycle of the input is critical, especially at operating frequencies close to the maximum. The relationship between clock signal duty cycle and clock signal period is expressed as follows:

$$\text{Minimum external clock period} = \frac{\text{minimum external clock high/low time}}{50\% - \text{percentage variation of external clock input duty cycle}}$$



### 3.2.2 Clock Synthesizer Operation

A voltage controlled oscillator (VCO) generates the system clock signal. A portion of the clock signal is fed back to a divider/counter. The divider controls the frequency of one input to a phase comparator. The other phase comparator input is a reference signal, either from the internal crystal oscillator or from an external source. The comparator generates a control signal proportional to the difference in phase between its two inputs. The signal is low-pass filtered and used to correct VCO output frequency.

The synthesizer locks when VCO frequency is identical to reference frequency. Lock time is affected by the filter time constant and by the amount of difference between the two comparator inputs. Whenever comparator input changes, the synthesizer must relock. Lock status is shown by the SLOCK bit in SYNCR.

The MC68F333 does not come out of reset state until the synthesizer locks. Crystal type, characteristic frequency, and layout of external oscillator circuitry affect lock time.

The low-pass filter requires an external low-leakage capacitor, typically 0.1 µF with an insulation resistance specification of 30,000 MΩ or greater, connected between the XFC and $V_{DDSYN}$ pins.

$V_{DDSYN}$ is used to power the clock circuits. A separate power source increases MCU noise immunity and can be used to run the clock when the MCU is powered down. A quiet power supply must be used as the $V_{DDSYN}$ source. Adequate external bypass capacitors should be placed as close as possible to the $V_{DDSYN}$ pin to assure stable operating frequency.

When the clock synthesizer is used, control register SYNCR determines operating frequency and various modes of operation. SYNCR can be read or written when the CPU is operating at the supervisor privilege level only.

The SYNCR X bit controls a divide by two prescaler that is not in the synthesizer feedback loop. Setting X doubles clock speed without changing VCO speed. There is no VCO relock delay. The SYNCR W bit controls a 3-bit prescaler in

the feedback divider. Setting W increases VCO speed by a factor of four. The SYNCR Y field determines the count modulus for a modulo 64 down counter, causing it to divide by a value of Y + 1. When either the W or Y value change, there is a VCO relock delay (refer to **APPENDIX A ELECTRICAL CHARACTERISTICS**, Table A–3, Clock Control Timing).



Clock frequency is determined by SYNCR bit settings as follows:

$$F_{SYSTEM} = F_{REFERENCE}\,[4(Y + 1)(2^{2W + X})]$$

The VCO frequency is twice the system clock frequency if X = 1 or four times the system clock frequency if X = 0.

For the device to perform correctly, the clock frequency selected by the W, X, and Y bits must be within the limits specified for the MCU. Maximum specified clock frequency with a 32.768-kHz reference is 16.78 MHz.

The reset state of SYNCR ($3F00) produces a modulus-64 count. System frequency is 256 times reference frequency.

Table 3–7 shows multipliers for combinations of SYNCR bits. The range of possible system frequencies can exceed the maximum specified system clock frequency. For instance, with a 32.768-kHz reference and a maximum system frequency of 16.78 MHz (refer to **APPENDIX A ELECTRICAL CHARACTERISTICS**, Table A–3, Clock Control Timing), W and X must not both be set at any count modulus greater than Y = %001111. Table 3–8 shows available clock frequencies for a 16.78-MHz system with a 32.768-kHz reference.

## Table 3–7. Clock Control Multipliers

To obtain clock frequency, find counter modulus in the left column, then multiply reference frequency by value in appropriate prescaler cell. Refer to Table A–1 in Appendix A for the maximum specified system clock frequency ($F_{sys}$).

| Modulus | Prescalers | | | |
|---|---|---|---|---|
| Y | [W:X] = 00 | [W:X] = 01 | [W:X] = 10 | [W:X] = 11 |
| 000000 | 4 | 8 | 16 | 32 |
| 000001 | 8 | 16 | 32 | 64 |
| 000010 | 12 | 24 | 48 | 96 |
| 000011 | 16 | 32 | 64 | 128 |
| 000100 | 20 | 40 | 80 | 160 |
| 000101 | 24 | 48 | 96 | 192 |
| 000110 | 28 | 56 | 112 | 224 |
| 000111 | 32 | 64 | 128 | 256 |
| 001000 | 36 | 72 | 144 | 288 |
| 001001 | 40 | 80 | 160 | 320 |
| 001010 | 44 | 88 | 176 | 352 |
| 001011 | 48 | 96 | 192 | 384 |
| 001100 | 52 | 104 | 208 | 416 |
| 001101 | 56 | 112 | 224 | 448 |
| 001110 | 60 | 120 | 240 | 480 |
| 001111 | 64 | 128 | 256 | 512 |
| 010000 | 68 | 136 | 272 | 544 |
| 010001 | 72 | 144 | 288 | 576 |
| 010010 | 76 | 152 | 304 | 608 |
| 010011 | 80 | 160 | 320 | 640 |
| 010100 | 84 | 168 | 336 | 672 |
| 010101 | 88 | 176 | 352 | 704 |
| 010110 | 92 | 184 | 368 | 736 |
| 010111 | 96 | 192 | 384 | 768 |
| 011000 | 100 | 200 | 400 | 800 |
| 011001 | 104 | 208 | 416 | 832 |
| 011010 | 108 | 216 | 432 | 864 |
| 011011 | 112 | 224 | 448 | 896 |
| 011100 | 116 | 232 | 464 | 928 |
| 011101 | 120 | 240 | 480 | 960 |
| 011110 | 124 | 248 | 496 | 992 |
| 011111 | 128 | 256 | 512 | 1024 |

## Table 3–7. Clock Control Multipliers (Continued)

| Modulus | Prescalers | | | |
|---|---|---|---|---|
| Y | [W:X] = 00 | [W:X] = 01 | [W:X] = 10 | [W:X] = 11 |
| 100000 | 132 | 264 | 528 | 1056 |
| 100001 | 136 | 272 | 544 | 1088 |
| 100010 | 140 | 280 | 560 | 1120 |
| 100011 | 144 | 288 | 576 | 1152 |
| 100100 | 148 | 296 | 592 | 1184 |
| 100101 | 152 | 304 | 608 | 1216 |
| 100110 | 156 | 312 | 624 | 1248 |
| 100111 | 160 | 320 | 640 | 1280 |
| 101000 | 164 | 328 | 656 | 1312 |
| 101001 | 168 | 336 | 672 | 1344 |
| 101010 | 172 | 344 | 688 | 1376 |
| 101011 | 176 | 352 | 704 | 1408 |
| 101100 | 180 | 360 | 720 | 1440 |
| 101101 | 184 | 368 | 736 | 1472 |
| 101110 | 188 | 376 | 752 | 1504 |
| 101111 | 192 | 384 | 768 | 1536 |
| 110000 | 196 | 392 | 784 | 1568 |
| 110001 | 200 | 400 | 800 | 1600 |
| 110010 | 204 | 408 | 816 | 1632 |
| 110011 | 208 | 416 | 832 | 1664 |
| 110100 | 212 | 424 | 848 | 1696 |
| 110101 | 216 | 432 | 864 | 1728 |
| 110110 | 220 | 440 | 880 | 1760 |
| 110111 | 224 | 448 | 896 | 1792 |
| 111000 | 228 | 456 | 912 | 1824 |
| 111001 | 232 | 464 | 928 | 1856 |
| 111010 | 236 | 472 | 944 | 1888 |
| 111011 | 240 | 480 | 960 | 1920 |
| 111100 | 244 | 488 | 976 | 1952 |
| 111101 | 248 | 496 | 992 | 1984 |
| 111110 | 252 | 504 | 1008 | 2016 |
| 111111 | 256 | 512 | 1024 | 2048 |

## Table 3–8. System Frequencies with a 32.768-kHz Reference

To obtain clock frequency, find counter modulus in the left column, then look in appropriate prescaler cell. Shaded cells represent values that exceed maximum system frequency specification ($F_{sys}$ — refer to Table A–1 in **APPENDIX A ELECTRICAL CHARACTERISTICS**)

| Modulus | Prescaler | | | |
|---|---|---|---|---|
| Y | [W:X] = 00 | [W:X] = 01 | [W:X] = 10 | [W:X] = 11 |
| 000000 | 131 | 262 | 524 | 1049 |
| 000001 | 262 | 524 | 1049 | 2097 |
| 000010 | 393 | 786 | 1573 | 3146 |
| 000011 | 524 | 1049 | 2097 | 4194 |
| 000100 | 655 | 1311 | 2621 | 5243 |
| 000101 | 786 | 1573 | 3146 | 6291 |
| 000110 | 918 | 1835 | 3670 | 7340 |
| 000111 | 1049 | 2097 | 4194 | 8389 |
| 001000 | 1180 | 2359 | 4719 | 9437 |
| 001001 | 1311 | 2621 | 5243 | 10486 |
| 001010 | 1442 | 2884 | 5767 | 11534 |
| 001011 | 1573 | 3146 | 6291 | 12583 |
| 001100 | 1704 | 3408 | 6816 | 13631 |
| 001101 | 1835 | 3670 | 7340 | 14680 |
| 001110 | 1966 | 3932 | 7864 | 15729 |
| 001111 | 2097 | 4194 | 8389 | 16777 |
| 010000 | 2228 | 4456 | 8913 | 17826 |
| 010001 | 2359 | 4719 | 9437 | 18874 |
| 010010 | 2490 | 4981 | 9961 | 19923 |
| 010011 | 2621 | 5243 | 10486 | 20972 |
| 010100 | 2753 | 5505 | 11010 | 22020 |
| 010101 | 2884 | 5767 | 11534 | 23069 |
| 010110 | 3015 | 6029 | 12059 | 24117 |
| 010111 | 3146 | 6291 | 12583 | 25166 |
| 011000 | 3277 | 6554 | 13107 | 26214 |
| 011001 | 3408 | 6816 | 13631 | 27263 |
| 011010 | 3539 | 7078 | 14156 | 28312 |
| 011011 | 3670 | 7340 | 14680 | 29360 |
| 011100 | 3801 | 7602 | 15204 | 30409 |
| 011101 | 3932 | 7864 | 15729 | 31457 |
| 011110 | 4063 | 8126 | 16253 | 32506 |
| 011111 | 4194 | 8389 | 16777 | 33554 |

## Table 3–8. System Frequencies with a 32.768-kHz Reference (Continued)

| Y | [W:X] = 00 | [W:X] = 01 | [W:X] = 10 | [W:X] = 11 |
|---|---|---|---|---|
| 100000 | 4325 | 8651 | 17302 | 34603 |
| 100001 | 4456 | 8913 | 17826 | 35652 |
| 100010 | 4588 | 9175 | 18350 | 36700 |
| 100011 | 4719 | 9437 | 18874 | 37749 |
| 100100 | 4850 | 9699 | 19399 | 38797 |
| 100101 | 4981 | 9961 | 19923 | 39846 |
| 100110 | 5112 | 10224 | 20447 | 40894 |
| 100111 | 5243 | 10486 | 20972 | 41943 |
| 101000 | 5374 | 10748 | 21496 | 42992 |
| 101001 | 5505 | 11010 | 22020 | 44040 |
| 101010 | 5636 | 11272 | 22544 | 45089 |
| 101011 | 5767 | 11534 | 23069 | 46137 |
| 101100 | 5898 | 11796 | 23593 | 47186 |
| 101101 | 6029 | 12059 | 24117 | 48234 |
| 101110 | 6160 | 12321 | 24642 | 49283 |
| 101111 | 6291 | 12583 | 25166 | 50332 |
| 110000 | 6423 | 12845 | 25690 | 51380 |
| 110001 | 6554 | 13107 | 26214 | 52428 |
| 110010 | 6685 | 13369 | 26739 | 53477 |
| 110011 | 6816 | 13631 | 27263 | 54526 |
| 110100 | 6947 | 13894 | 27787 | 55575 |
| 110101 | 7078 | 14156 | 28312 | 56623 |
| 110110 | 7209 | 14418 | 28836 | 57672 |
| 110111 | 7340 | 14680 | 29360 | 58720 |
| 111000 | 7471 | 14942 | 2988 | 59769 |
| 111001 | 7602 | 15204 | 30409 | 60817 |
| 111010 | 7733 | 15466 | 30933 | 61866 |
| 111011 | 7864 | 15729 | 31457 | 62915 |
| 111100 | 7995 | 15991 | 31982 | 63963 |
| 111101 | 8126 | 16253 | 32506 | 65011 |
| 111110 | 8258 | 16515 | 33030 | 66060 |
| 111111 | 8389 | 16777 | 33554 | 67109 |

### 3.2.3 External Bus Clock

The state of the external clock division bit (EDIV) in SYNCR determines the clock rate for the external bus clock signal (ECLK) available on pin ADDR23. ECLK is a bus clock for MC6800 devices and peripherals. ECLK frequency can be set to system clock frequency divided by eight or system clock frequency divided by sixteen. The clock is enabled by the CSPA1[4] field in chip select pin assignment register 1 (CSPAR1). The operation of the external bus clock during low-power stop is described in the following paragraph. Refer to **3.6 Chip Selects** for more information about the external bus clock.



### 3.2.4 Low-Power Operation

The CPU32 initiates low-power operation. To reduce power consumption selectively, the CPU can set the STOP bits in each module configuration register. To minimize overall microcontroller power consumption, the CPU can execute the LPSTOP instruction, which causes the SCIM to turn off the system clock.

When STOP bits in the configuration registers of individual modules are set, clock signals inside each module are turned off, but module registers are still accessible.

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 SCIM brings the MCU out of low-power operation when either an interrupt of higher priority than the stored mask or a reset occurs. Refer to **3.3.7.2 LPSTOP Broadcast Cycles** and **SECTION 4 CENTRAL PROCESSING UNIT** for more information.

During a low-power stop, unless the system clock signal is supplied by an external source and that source is removed, the SCIM clock control logic and the SCIM 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 SCIM can also continue to generate the CLKOUT signal while in low-power mode.

The stop mode single-chip integration module clock (STSCIM) and stop mode external clock (STEXT) bits in SYNCR determine clock operation during low-power stop. Table 3–9 is a summary of the effects of STSCIM and STEXT with various clock sources. MODCLK value is the logic level on the MODCLK pin during the last reset before LPSTOP execution. Any clock in the off state is held low.

**Table 3–9. Clock Control**

| Mode | Pins | | SYNCR Bits | | Clock Source | |
|---|---|---|---|---|---|---|
| LPSTOP | MODCLK | EXTAL | STSCIM | STEXT | SCIMCLK | CLKOUT |
| No | 0 | External Clock | X | X | External Clock | External Clock |
| Yes | 0 | External Clock | 0 | 0 | External Clock | Off |
| Yes | 0 | External Clock | 0 | 1 | External Clock | External Clock |
| Yes | 0 | External Clock | 1 | 0 | External Clock | Off |
| Yes | 0 | External Clock | 1 | 1 | External Clock | External Clock |
| No | 1 | Crystal/ Reference | X | X | VCO | VCO |
| Yes | 1 | Crystal/ Reference | 0 | 0 | Crystal/ Reference | Off |
| Yes | 1 | Crystal/ Reference | 0 | 1 | Crystal/ Reference | Crystal/ Reference |
| Yes | 1 | Crystal/ Reference | 1 | 0 | VCO | Off |
| Yes | 1 | Crystal/ Reference | 1 | 1 | VCO | VCO |

### 3.2.5 Loss of Clock

The SCIM can detect the loss of either an external clock signal (MODCLK held low at reset) or a clock signal generated by the PLL (MODCLK held high at reset). Two bits in the SYNCR determine how the SCIM responds to the loss of a clock signal. The loss-of-clock oscillator disable (LOSCD) bit enables (LOSCD = 0) or disables (LOSCD = 1) loss-of-clock detection. The reset enable (RSTEN) bit determines how the SCIM responds when it detects the loss of a clock signal. The LOSCD bit must be cleared for the RSTEN bit to have any effect.

When the RSTEN bit is set and a loss of clock is detected, the SCIM resets the MCU. If RSTEN is cleared when loss of clock is detected, an alternate clock is provided as the system clock until edges are detected on the crystal or external clock input. The alternate clock is the output of an RC oscillator which is also used as the time base for the loss-of-clock detector. All clock switching is done synchronously, so that no short pulses or glitches occur on the system clock.

An MCU using the alternate clock as the system clock is said to be operating in limp mode. The limp mode status bit (SLIMP) in the SYNCR indicates whether the MCU is running in limp mode.

Loss of clock is recognized during low-power operation as well as normal operation, provided the LOSCD bit is cleared. During low-power operation (as during normal operation), when loss of clock is detected, the MCU either operates in limp mode or is reset, depending on the value of the RSTEN bit.

The following paragraphs explain loss-of-clock operation when the PLL is generating the clock signal and when an external clock is provided. In either case, the LOSCD bit must be cleared in order for loss of clock to be detected.



### 3.2.5.1 Loss of Reference Frequency

When the PLL is generating the system clock (MODCLK held high during reset), the RSTEN bit is cleared, and the SCIM detects a loss of reference signal, the alternate clock becomes the system clock. (The LOSCD bit must be cleared for the SCIM to detect a loss of reference signal.) In addition, a nominal voltage is applied to the VCO. This voltage allows the PLL to ramp up quickly if the reference signal is recovered while the SCIM is operating in limp mode. If the reference signal is recovered, the PLL switches back to normal operation.

Figure 3–5 illustrates the loss and recovery of a reference signal to the PLL.

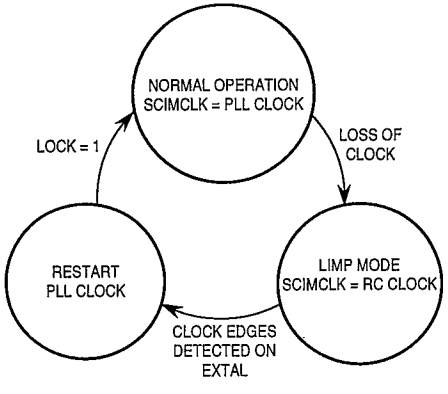



**Figure 3–5. Loss of Reference Frequency**

If the reference signal is lost while the VCO is ramping up, the PLL re-locks at 0 Hz, the loss-of-clock detector is triggered, and the system clock is switched back to the alternate clock.

When the RSTEN bit is set and the SCIM detects a loss of reference signal, the SCIM generates a synchronous reset. Refer to **3.2.5.3 Loss-of-Clock Reset**.

### 3.2.5.2 Loss of External Clock

When the SCIM is configured for an external clock source (MODCLK held low during reset), the RSTEN bit is cleared, and a loss of clock is detected, the alternate clock becomes the system clock until the external clock input re-starts. The LOSCD bit must be cleared for the SCIM to detect the loss of a clock signal. Figure 3–6 illustrates the loss and recovery of an external clock signal.

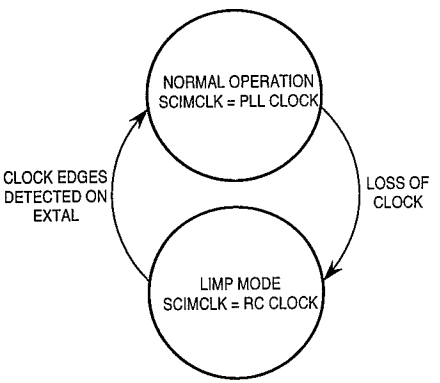



**Figure 3–6. Loss of External Clock Signal**

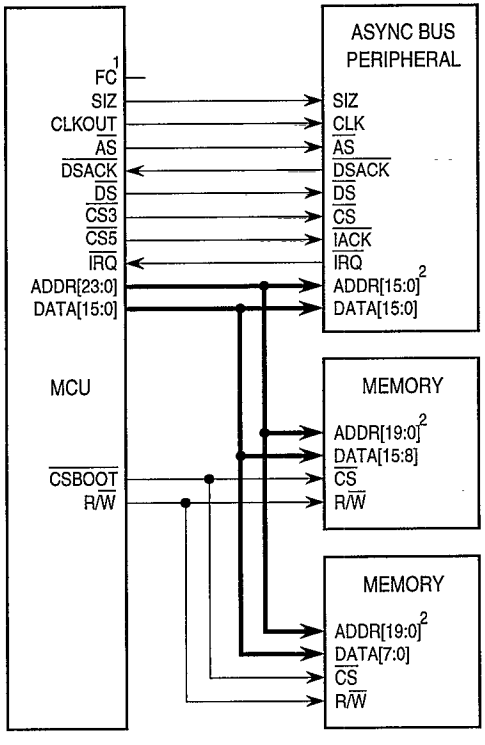
### 3.2.5.3 Loss-of-Clock Reset

When the RSTEN bit is set and the SCIM detects the loss of a clock signal, the SCIM generates an asynchronous reset. For loss of clock to be detected, the LOSCD bit must be cleared. This bit is automatically cleared at reset, ensuring that a clock signal will be present during reset: if the system clock has stopped, the SCIM detects the condition and switches to the alternate clock. This ensures that RESET will be accepted. Refer to **3.4 Reset** for more information on reset procedures.

## 3.3 External Bus Interface

The external bus interface (EBI) transfers information between the internal MCU bus and external devices. Figure 3–7 shows a basic system with external memory and peripherals.






1. CAN BE DECODED TO PROVIDE ADDITIONAL ADDRESS SPACE.
2. VARIES DEPENDING UPON PERIPHERAL MEMORY SIZE.

32 EXAMPLE SYS BLOCK

**Figure 3–7. MC68F333 Basic System**

The external bus, when fully expanded, has 24 address lines and 16 data lines. When partially expanded, the external bus has 24 address lines and 8 data lines. The external bus is not available when the SCIM is operating in single-

chip mode. Where not otherwise noted, this section describes the operation of the external bus interface with a fully expanded bus.

A three-line handshaking interface performs external bus arbitration. The interface supports byte, word, and long-word transfers. The EBI performs dynamic sizing for data accesses.

The maximum number of bits transferred during an access is referred to as port width. Widths of eight and sixteen bits can be accessed by asynchronous bus cycles controlled by the data size (SIZ0 and SIZ1) and the data and size acknowledge ($\overline{\text{DSACK0}}$ and $\overline{\text{DSACK1}}$) signals. Multiple bus cycles may be required for a dynamically sized transfer.

The SCIM chip selects shown in Figure 3–1 provide flexibility and reduce the need for external logic. The other pins depicted in Figure 3–1 are explained in the following paragraphs.

### 3.3.1 Bus 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 it is to take place in, the size of the transfer, and the type of cycle. External devices decode these signals and respond to transfer data and terminate the bus cycle. The EBI operates in an asynchronous mode for any port width.

#### 3.3.1.1 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 $\overline{\text{AS}}$ is asserted.

#### 3.3.1.2 Address Strobe

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

#### 3.3.1.3 Data Bus

Bus signals DATA[15:0] comprise a bidirectional, nonmultiplexed parallel bus that transfers data to or from the MCU. A read or write operation can transfer eight or sixteen bits of data in one bus cycle. During a read cycle, the data is

latched by the MCU on the last falling edge of the clock for that bus cycle. For a write cycle, all 16 bits of the data bus are driven, regardless of the port width or operand size. The MCU places the data on the data bus one-half clock cycle after $\overline{AS}$ is asserted in a write cycle.



### 3.3.1.4 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. The MCU asserts $\overline{DS}$ one full clock cycle after the assertion of $\overline{AS}$ during a write cycle.

### 3.3.1.5 Read/Write Signal

The read/write (R/$\overline{W}$) signal 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 changes when a write cycle is preceded by a read cycle or vice versa. The signal may remain low for two consecutive write cycles.

### 3.3.1.6 Size Signals

Size signals (SIZ[1:0]) indicate the number of bytes remaining to be transferred during an operand cycle. They are valid while the address strobe ($\overline{AS}$) is asserted. Table 3–10 shows SIZ0 and SIZ1 encoding.

**Table 3–10.**
**Size Signal Encoding**

| SIZ1 | SIZ0 | Transfer Size |
|---|---|---|
| 0 | 1 | Byte |
| 1 | 0 | Word |
| 1 | 1 | 3 Byte |
| 0 | 0 | Long Word |

### 3.3.1.7 Function Codes

The CPU generates function code output 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.

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 3–11 shows address space encoding.

**Table 3–11. 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 |

The supervisor bit in the status register determines whether the CPU is operating in supervisor or user mode. Addressing mode and the instruction being executed determine whether a memory access is to program or data space.

### 3.3.1.8 Data and Size Acknowledge Signals

During normal bus transfers, external devices assert the data and size acknowledge signals ($\overline{DSACK1}$ and $\overline{DSACK0}$) 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. $\overline{DSACK1}$ and $\overline{DSACK0}$ can also be supplied internally by chip-select logic. Refer to **3.6 Chip Selects** for more information.

### 3.3.1.9 Bus Error Signal

The bus error ($\overline{BERR}$) signal is asserted in the absence of $\overline{DSACK}$ to indicate a bus error condition. $\overline{BERR}$ can also be asserted with $\overline{DSACK}$ to indicate a bus error condition, provided it meets the appropriate timing requirements. Refer to **3.3.8 Bus Exception Control Cycles** for more information.

The internal bus monitor can generate the $\overline{BERR}$ signal for internal and internal-to-external transfers. An external bus master must provide its own $\overline{BERR}$ generation and drive the $\overline{BERR}$ pin, because the internal $\overline{BERR}$ monitor has no

information about transfers initiated by an external bus master. Refer to **3.3.9 External Bus Arbitration** for more information.

#### 3.3.1.10 Halt Signal

The halt signal ($\overline{\text{HALT}}$) can be asserted by an external device to cause single bus cycle operation. Usually $\overline{\text{HALT}}$ is used for debugging purposes. In addition, asserting $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$ simultaneously indicates a retry termination. Refer to **3.3.8 Bus Exception Control Cycles** for additional information.



#### 3.3.1.11 Autovector Signal

The autovector signal ($\overline{\text{AVEC}}$) can be used to terminate external interrupt acknowledge cycles. Assertion of $\overline{\text{AVEC}}$ causes the CPU32 to generate vector numbers to locate an interrupt handler routine. If it is continuously asserted, autovectors are generated for all external interrupt requests. $\overline{\text{AVEC}}$ is ignored during all other bus cycles. Refer to **3.5 Interrupts** for more information. $\overline{\text{AVEC}}$ for external interrupt requests can also be supplied internally by chip-select logic. Refer to **3.6 Chip Selects** for more information. The autovector function is disabled when there is an external bus master. Refer to **3.3.9 External Bus Arbitration** for more information.

### 3.3.2 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 an operand transfer cycle, an external device signals its port size and indicates completion of the bus cycle to the MCU through the use of the $\overline{\text{DSACK}}$ inputs, as shown in Table 3–12. Chip-select logic can generate data and size acknowledge signals for an external device. Refer to **3.6 Chip Selects** for additional information.

**Table 3–12. Effect of $\overline{\text{DSACK}}$ Signals**

| $\overline{\text{DSACK1}}$ | $\overline{\text{DSACK0}}$ | 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 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 $\overline{\text{DSACK}}$ signals to indicate the port width. For instance, a 16-bit device always returns $\overline{\text{DSACK}}$ 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[15:0], and an 8-bit port must reside on DATA[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 3–8. OP0 – OP3 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.

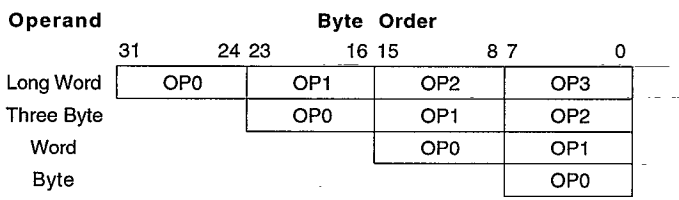



**Figure 3–8. Operand Byte Order**

### 3.3.3 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 size and address outputs. SIZ1 and SIZ0 indicate the remaining number of bytes 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 an operand 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.

The largest amount of data that can be transferred by a single bus cycle is a 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.



### 3.3.4 Misaligned Operands

CPU32 processor architecture uses a basic operand size of 16 bits. An operand is misaligned when it overlaps a word boundary. When ADDR0 = 0, indicating an even address, the address is on a word and byte boundary. When ADDR0 = 1, indicating 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 CPU32 does not support misaligned word transfers.

### 3.3.5 Operand Transfer Cases

Table 3–13 is a summary of 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 3–13. Operand Alignment**

| Transfer Case | SIZ1 | SIZ0 | ADDR0 | $\overline{\text{DSACK1}}$ | $\overline{\text{DSACK0}}$ | DATA [15:8] | DATA [7:0] |
|---|---|---|---|---|---|---|---|
| Byte to 8-Bit Port (Even/Odd) | 0 | 1 | X | 1 | 0 | OP0 | (OP0) |
| Byte to 16-Bit Port (Even) | 0 | 1 | 0 | 0 | X | OP0 | (OP0) |
| Byte to 16-Bit Port (Odd) | 0 | 1 | 1 | 0 | X | (OP0) | OP0 |
| Word to 8-Bit Port (Aligned) | 1 | 0 | 0 | 1 | 0 | OP0 | (OP1) |
| Word to 8-Bit Port (Misaligned)[3] | 1 | 0 | 1 | 1 | 0 | OP0 | (OP0) |
| Word to 16-Bit Port (Aligned) | 1 | 0 | 0 | 0 | X | OP0 | OP1 |
| Word to 16-Bit Port (Misaligned)[3] | 1 | 0 | 1 | 0 | X | (OP0) | OP0 |
| 3 Byte to 8-Bit Port (Aligned)[4] | 1 | 1 | 0 | 1 | 0 | OP0 | (OP1) |
| 3 Byte to 8-Bit Port (Misaligned)[4] | 1 | 1 | 1 | 1 | 0 | OP0 | (OP0) |
| 3 Byte to 16-Bit Port (Misaligned)[4] | 1 | 1 | 1 | 0 | X | (OP0) | OP0 |
| Long Word to 8-Bit Port (Aligned) | 0 | 0 | 0 | 1 | 0 | OP0 | (OP1) |
| Long Word to 16-Bit Port (Aligned) | 0 | 0 | 0 | 0 | X | OP0 | OP1 |

NOTES:
1. X in a column means that the state of the signal has no effect.
2. Operands in parentheses are ignored by the CPU32 during read cycles.
3. The MC68F333 does not support misaligned operand transfers.
4. Three-byte transfer cases occur only as a result of an aligned long word to byte transfer.

### 3.3.6 Bus Operation

Internal microcontroller modules are typically accessed in two system clock cycles, with no wait states. Regular external bus cycles use handshaking between the MCU and external peripherals to manage transfer size and data. These accesses take three system clock cycles, again with no wait states. During regular cycles, wait states can be inserted as needed by bus control logic. Refer to **3.3.6.2 Regular Bus Cycles** for more information. Fast-termination cycles, which are two-cycle external accesses with no wait states, use chip-select logic to generate handshaking signals internally. Chip-select logic can also be used to insert wait states before internal generation of handshaking signals. Refer to **3.3.6.5 Fast-Termination Cycles** and **3.6 Chip Selects** for more information. Bus control signal timing, as well as chip-select signal timing, are specified in **APPENDIX A ELECTRICAL CHARACTERISTICS**. Refer to the *SCIM Reference Manual* (SCIMRM/AD) for more information about each type of bus cycle.

The MCU is responsible for de-skewing signals it issues at both the start and the end of a cycle. In addition, the MCU is responsible for de-skewing acknowledge and data signals from peripheral devices.

#### 3.3.6.1 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, labeled {S0, S1, S2,..., SN} in the appropriate timing diagrams. 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 A 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.

For a read cycle, when assertion of $\overline{\text{DSACK}}$ is recognized on a particular falling edge of the clock, valid data is latched into the MCU on the next falling clock

edge, provided that the data meets the data setup time. In this case, the parameter for asynchronous operation can be ignored.

When a system asserts $\overline{\text{DSACK}}$ for the required window around the falling edge of S2 and obeys the bus protocol by maintaining $\overline{\text{DSACK}}$ and $\overline{\text{BERR}}$ or $\overline{\text{HALT}}$ until and throughout the clock edge that negates $\overline{\text{AS}}$ (with the appropriate asynchronous input hold time), no wait states are inserted. The bus cycle runs at the maximum speed of three clocks per cycle.

To ensure proper operation in a system synchronized to CLKOUT, when $\overline{\text{BERR}}$ (either alone or together with $\overline{\text{HALT}}$) is asserted after $\overline{\text{DSACK}}$, $\overline{\text{BERR}}$ (or $\overline{\text{BERR}}$ and $\overline{\text{HALT}}$) assertion must satisfy the appropriate data-in setup and hold times before the falling edge of the clock cycle after $\overline{\text{DSACK}}$ is recognized.

### 3.3.6.2 Regular Bus Cycles

The following paragraphs contain a discussion of cycles that use external bus control logic. Refer to **3.3.6.5 Fast-Termination Cycles** for information about fast cycles.

To initiate a transfer, the MCU asserts an address 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 **3.3.2 Dynamic Bus Sizing**). 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 that indicates 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 into the MCU, a maximum period between $\overline{\text{DSACK}}$ assertion and $\overline{\text{DS}}$ assertion is specified.

There is no specified maximum for the period between the assertion of $\overline{\text{AS}}$ and $\overline{\text{DSACK}}$. 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 $\overline{\text{DSACK}}$ signal goes low.

**NOTE**

The SCIM bus monitor asserts $\overline{\text{BERR}}$ when response time exceeds a predetermined limit. Bus monitor period is determined by the BMT field in SYPCR. The bus monitor cannot be disabled; maximum monitor period is 64 system clock cycles.

If no peripheral responds to an access, or if an access is invalid, external logic should assert the $\overline{BERR}$ or $\overline{HALT}$ signals to abort the bus cycle. If bus termination signals are not asserted within a specified period, the bus monitor terminates the cycle.

3

### 3.3.6.3 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. Refer to **3.3.2 Dynamic Bus Sizing** and **3.3.4 Misaligned Operands** for more information. Figure 3–9 is a flowchart of a word read cycle.

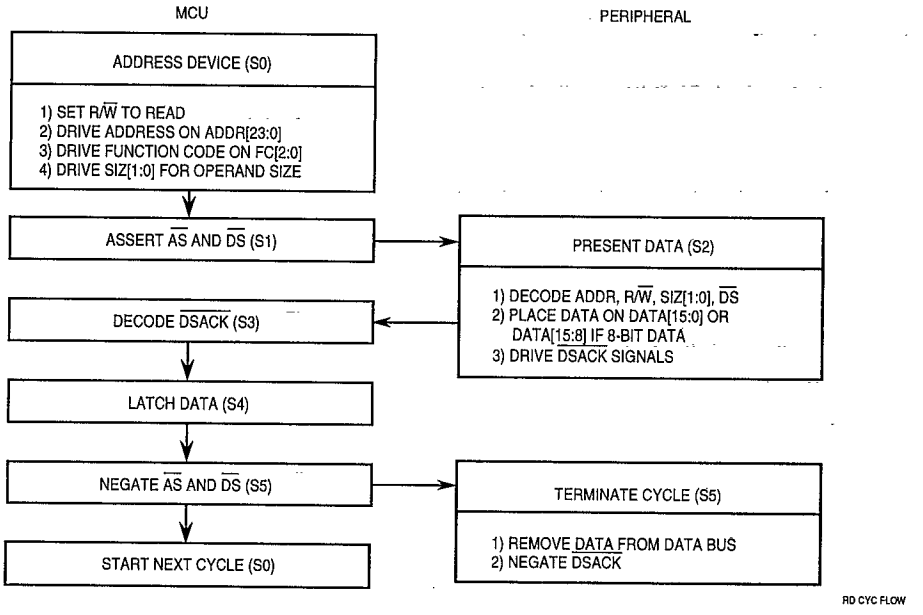



**Figure 3–9. Word Read Cycle Flowchart**

### 3.3.6.4 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. Refer to **3.3.2 Dynamic Bus Sizing** and **3.3.4 Misaligned Operands** for more information. Figure 3–10 is a flowchart of a write-cycle operation for a word transfer.

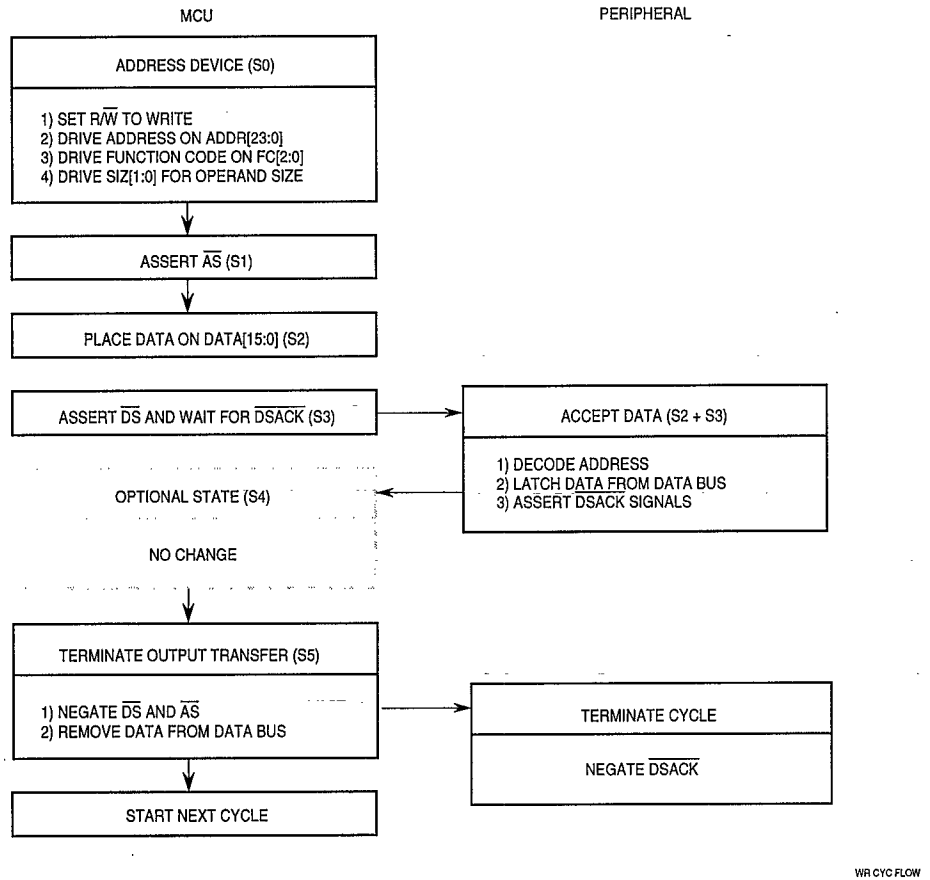



**Figure 3–10. Write Cycle Flowchart**

### 3.3.6.5 Fast-Termination Cycles

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

Fast-termination cycles use internal handshaking signals generated by the chip-select logic. To initiate a transfer, the MCU asserts an address 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 data and size acknowledge signals.

The $\overline{DSACK}$ option fields in the chip-select option registers determine whether internally or externally generated $\overline{DSACK}$ signals are used. Refer to **3.6.1 Chip-Select Registers** for information about fast-termination setup.

To use fast-termination, an external device must be fast enough to have data ready, within the specified setup time, by the falling edge of S4. Figure 3–11 shows the $\overline{DSACK}$ timing for two wait states in read, and a fast-termination read and write.

If multiple chip selects are to be used to select the same device that can support fast termination, and match conditions can occur simultaneously, program the $\overline{DSACK}$ field in each associated chip-select option register for fast termination. Alternately, program one $\overline{DSACK}$ field for fast termination and the remaining $\overline{DSACK}$ fields for external termination.

$\overline{DS}$ is asserted during fast-termination read cycles but not write cycles. 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.

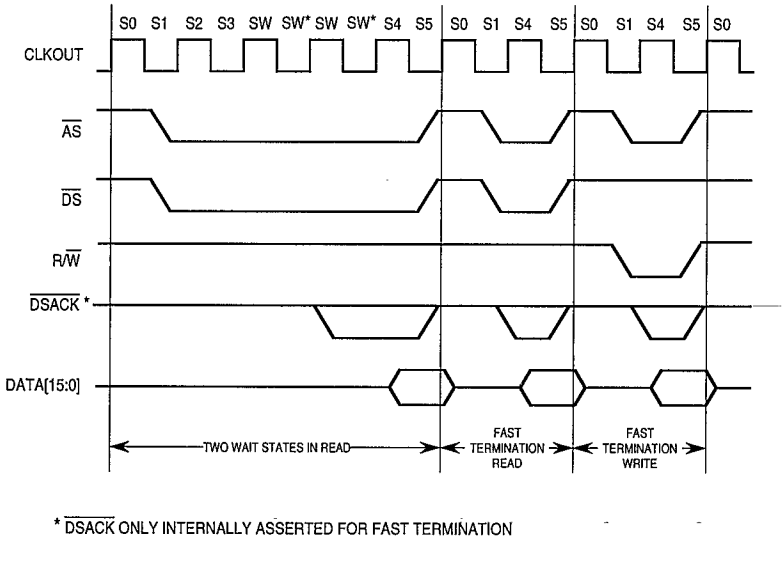



**Figure 3–11. Fast-Termination Timing**

### 3.3.7 CPU Space Cycles

Function code signals FC[2:0] designate which one of eight external address spaces is accessed during a bus cycle. Address space seven is designated 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 **3.3.1.7 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. The MC68F333 uses three encodings, as shown in Figure 3–12. These encodings represent breakpoint acknowledge (Type $0) cycles, low power stop broadcast (Type $3) cycles, and interrupt acknowledge (Type $F) cycles. Refer to **3.5 Interrupts** for information about interrupt acknowledge bus cycles.

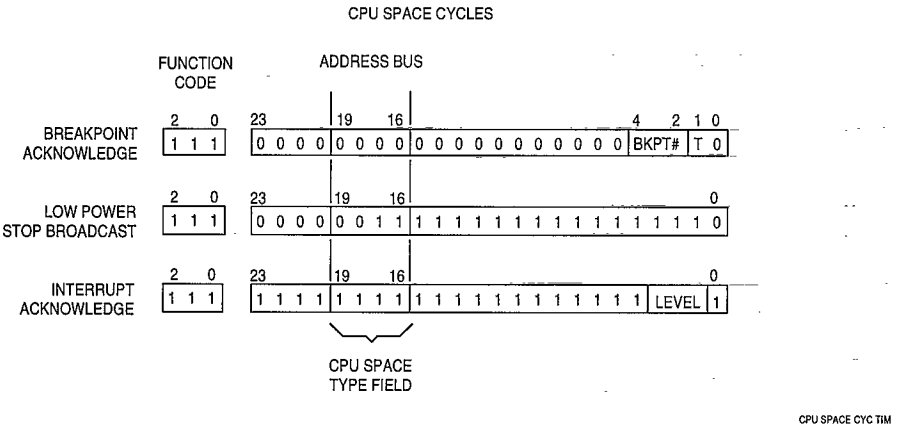



**Figure 3–12. CPU Space Address Encoding**

### 3.3.7.1 Breakpoint Acknowledge Cycle

Breakpoints stop program execution at a predefined point during system development. Breakpoints can be used alone or in conjunction with the background debugging mode. The following paragraphs discuss breakpoint processing when background debugging mode is not enabled. Refer to **SECTION 4 CENTRAL PROCESSING UNIT** for more information on exception processing and the background debugging mode.

On the MC68F333, both hardware and software can initiate breakpoints.

#### 3.3.7.1.1 Software Breakpoints

The CPU32 BKPT instruction allows the user to insert breakpoints through software. The CPU responds to this instruction by initiating a breakpoint-acknowledge read cycle in CPU space. It places the breakpoint acknowledge (%0000) code on ADDR[19:16], the breakpoint number (bits [2:0] of the BKPT opcode) in ADDR[4:2], and %0 (indicating a software breakpoint) on ADDR1.

The external breakpoint circuitry decodes the function code and address lines and responds by either 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 then performs illegal-instruction exception processing: it 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.



#### 3.3.7.1.2 Hardware Breakpoints

Assertion of the $\overline{\text{BKPT}}$ input initiates a hardware breakpoint. The CPU responds by initiating a breakpoint-acknowledge read cycle in CPU space. It places $00001E on the address bus. (The breakpoint acknowledge code of %0000 is placed on ADDR[19:16], the breakpoint number value of %111 is placed on ADDR[4:2], and ADDR1 is set to 1, indicating a hardware breakpoint.)

The external breakpoint circuitry decodes the function code and address lines, places an instruction word on the data bus, and asserts $\overline{\text{BERR}}$. The CPU 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 CPU 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, and 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.

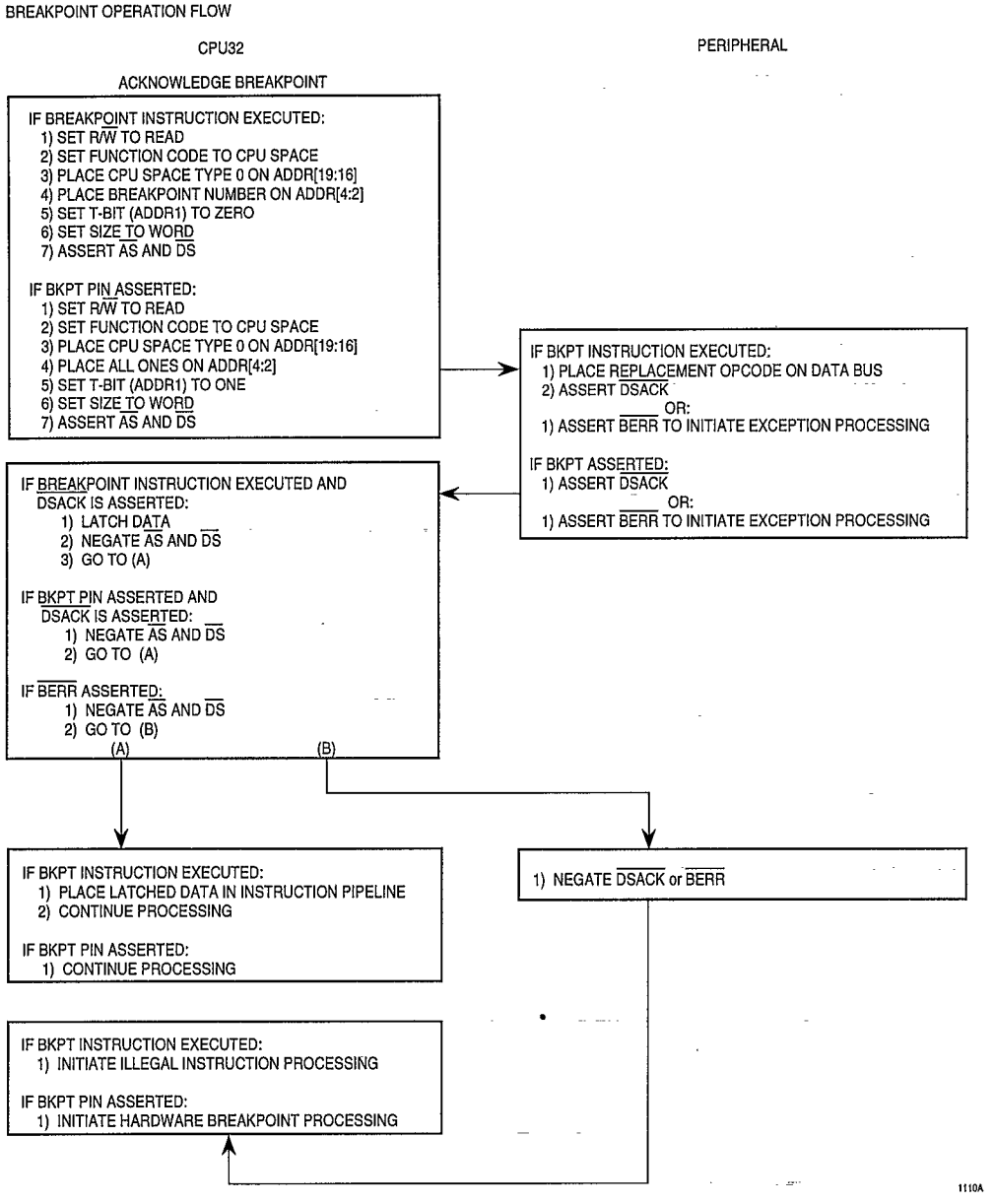



**Figure 3–13. Breakpoint Operation Flowchart**

### 3.3.7.2 LPSTOP Broadcast Cycle

When the CPU executes the LPSTOP instruction, an LPSTOP broadcast cycle is generated. During an LPSTOP broadcast cycle, the CPU performs a CPU space write to address $3FFFE. 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 3–14. 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 SCIM provides an internally generated DSACK response to this cycle. The timing of this bus cycle is the same as for a fast write cycle.

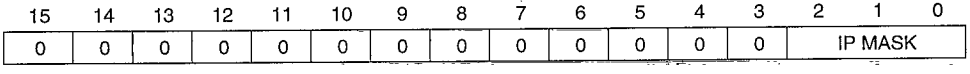



**Figure 3–14. LPSTOP Interrupt Mask Level**

### 3.3.8 Bus Exception Control Cycles

An external device or a chip-select circuit must assert at least one of the DSACK[1:0] signals or the AVEC signal to terminate a bus cycle normally. Bus error processing occurs when bus cycles are not terminated in the expected manner. The internal bus monitor can be used to generate BERR internally, causing a bus error exception to be taken. Bus cycles can also be terminated by assertion of the external BERR or HALT signal, or by assertion of the two signals simultaneously.

Acceptable bus cycle termination sequences are summarized as follows. The case numbers refer to Table 3–14, which indicates the results of each type of bus cycle termination.

Normal Termination

DSACK is asserted; BERR and HALT remain negated (case 1).

Halt Termination

HALT is asserted at the same time, or before DSACK, and BERR remains negated (case 2).

Bus Error Termination

BERR is asserted in lieu of, at the same time as, or before DSACK (case 3), or after DSACK (case 4), and HALT remains negated; BERR is negated at the same time or after 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 $\overline{BERR}$.

3

Table 3–14 shows various combinations of control signal sequences and the resulting bus cycle terminations.

**Table 3–14. $\overline{DSACK}$, $\overline{BERR}$, and $\overline{HALT}$ Assertion Results**

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

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

To properly control termination of a bus cycle for a retry or a bus error condition, $\overline{DSACK}$, $\overline{BERR}$, and $\overline{HALT}$ must be asserted and negated with the rising edge of the MCU clock. 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 A 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.



### 3.3.8.1 Bus Error Exceptions

The CPU treats bus errors as a type of exception. Bus error exception processing begins when the CPU 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.

$\overline{\text{BERR}}$ takes precedence over $\overline{\text{DSACK}}$, provided it meets the timing constraints described in **APPENDIX A ELECTRICAL CHARACTERISTICS**.

**WARNING**

If $\overline{\text{BERR}}$ does not meet these constraints, it may cause unpredictable operation of the MCU. If $\overline{\text{BERR}}$ remains asserted into the next bus cycle, that cycle may operate incorrectly.

$\overline{\text{BERR}}$ assertions do not force immediate exception processing. The signal is synchronized with normal bus cycles and is latched into the CPU at the end of the bus cycle in which it was asserted. Since 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 and acknowledgment depends on 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 will be 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 $FF) is latched into the CPU instruction register, with indeterminate results.

### 3.3.8.2 Double Bus Faults

Exception processing for bus error exceptions follows the standard exception processing sequence. However, a special case of bus error, called double bus fault, can abort exception processing.

3

$\overline{BERR}$ assertion is not detected until an instruction is complete. The $\overline{BERR}$ latch is cleared by the first instruction of the $\overline{BERR}$ exception handler. Double bus faults occur under the following conditions:

- Bus error exception processing begins and a second $\overline{BERR}$ is detected before the first instruction of the first exception handler is executed.
- One or more bus errors occur before the first instruction after a reset exception is executed.
- A bus error occurs while the CPU is loading information from a bus error stack frame during a return from exception (RTE) instruction.

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

Immediately after assertion of a second $\overline{BERR}$ signal, the MCU halts and drives the $\overline{HALT}$ line low. Only a reset can restart a halted MCU. However, bus arbitration can still occur. (Refer to **3.3.9 External Bus Arbitration**.) A bus error or address error that occurs after exception processing has completed (during the execution of the exception handler routine, or later) does not cause a double bus fault. A bus cycle that is retried does not constitute a bus error or cause a double bus fault. The MCU continues to retry the same bus cycle as long as the external hardware requests it.

### 3.3.8.3 Retry Operation

When an external device asserts $\overline{BERR}$ and $\overline{HALT}$ during a bus cycle, the MCU enters the retry sequence.

A delayed retry, similar to the delayed bus error signal, can also occur. The MCU terminates the bus cycle, places the $\overline{AS}$ and $\overline{DS}$ signals in their inactive state, and does not begin another bus cycle until the $\overline{BERR}$ and $\overline{HALT}$ 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 $\overline{BERR}$ signal should be negated before S2 of the read cycle to ensure correct operation of the retried cycle.

If $\overline{BR}$, $\overline{BERR}$, and $\overline{HALT}$ are all asserted on the same cycle, the EBI enters the retry sequence but first relinquishes the bus to an external master. Once the external master returns the bus and negates $\overline{BERR}$ and $\overline{HALT}$, the EBI runs the previous bus cycle. This feature allows an external device to correct the

problem that caused the bus error (e.g., swap in a new page of memory) and then try the bus cycle again.

The MCU retries any read or write cycle of an indivisible read-modify-write operation separately; RMC remains asserted during the entire retry sequence. The MCU will not relinquish the bus while 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 BERR and BR only (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 *CPU32 Reference Manual* (CPU32RM/AD) for details on the special status word.)



### 3.3.8.4 Halt Operation

When HALT is asserted and BERR is not asserted, the MCU halts external bus activity after negation of 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 HALT in accordance with timing requirements provides single-step (bus cycle to bus cycle) operation. The HALT signal affects external bus cycles only, so that a program which 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 HALT is asserted causes the CPU to initiate a retry sequence. Retry cycles must be anticipated while debugging in single-cycle mode. In dynamically sized 8-bit transfers, external bus activity may not stop at the next cycle boundary.

When the MCU completes a bus cycle while the HALT signal is asserted, the data bus goes to high-impedance state and the AS and 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. (Refer to **3.3.9 External Bus Arbitration**.) However, when external bus arbitration occurs while the MCU is halted, address and control signals go to a high-impedance state. If 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.

### 3.3.9 External Bus Arbitration

MCU bus design provides for a single bus master at any one time. When the MCU is not configured for single-chip operation, 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 CPU has halted due to a double bus fault.

The bus controller in the MCU 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 is:

A. An external device asserts the bus request signal ($\overline{\text{BR}}$).

B. The MCU asserts the bus grant signal ($\overline{\text{BG}}$) to indicate that the bus is available.

C. 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.

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

Refer to Figure 3–15, which shows bus arbitration for a single device. The flowchart shows $\overline{BR}$ negated at the same time $\overline{BGACK}$ is asserted.



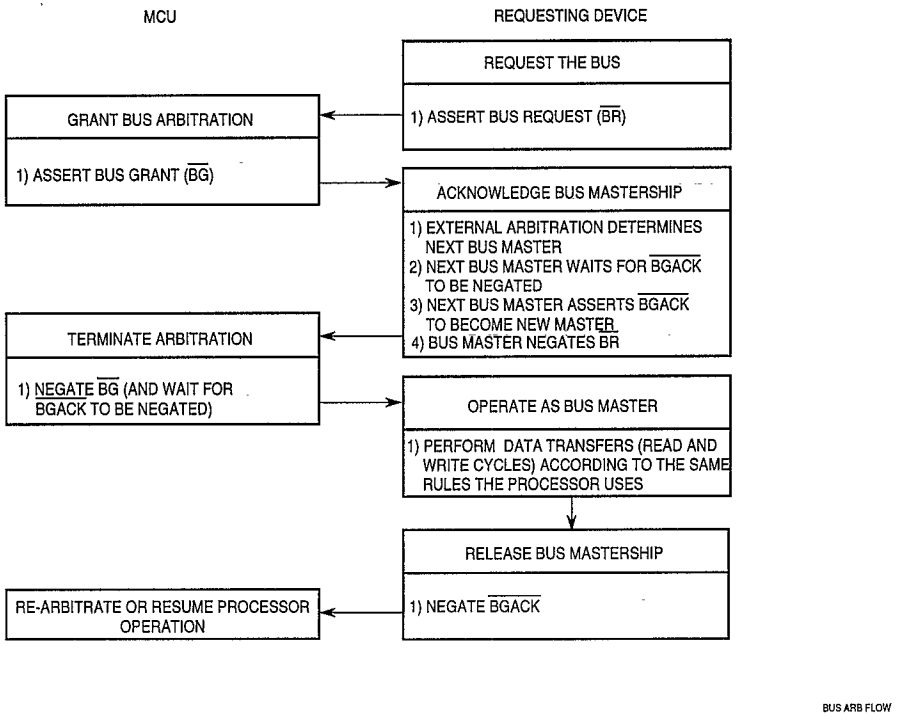



**Figure 3–15. Bus Arbitration Flowchart for Single Request**

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

### 3.3.10 Slave (Factory Test) Mode Arbitration

This mode is used for factory production testing of internal modules. It is not supported as a user operating mode. Slave mode is enabled by holding DATA11 low during reset. In slave mode, when $\overline{BG}$ is asserted, the MCU is slaved to an external master that has full access to all internal registers.

### 3.3.11 Show Cycles

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

Show cycles are controlled by the SHEN field in the SCIMCR (refer to **3.1.6 Show Internal Cycles**). This field is cleared 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 high-impedance state during internal accesses.

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 encoding halts internal bus activity while there is an external master.

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 SCIM causes the value of the byte that is written to be driven out on both bytes of the data bus.

## 3.4 Reset

Reset occurs when an active low logic level on the $\overline{RESET}$ pin is clocked into the SCIM. 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 MC68F333 performs resets with a combination of hardware and software. The single-chip integration module determines whether a reset is valid, asserts control signals, performs basic system configuration and boot ROM selection based on hardware mode-select inputs, then passes control to the CPU32.

### 3.4.1 Reset Exception Processing

The CPU32 processes resets as a type of asynchronous exception. An exception is an event that preempts normal processing. Each exception has an assigned vector that points to an associated handler routine. During exception processing, the CPU fetches the vector assigned to the exception and executes the exception routine to which the vector points.



Exception vectors are stored in a vector table. Out of reset, the table is located beginning at address $000000. The reset vector occupies the first four words of the vector table. The CSBOOT chip-select signal, which responds to memory accesses starting at $000000 out of reset, can be used to select the boot ROM chip with the system initialization routine.

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. **3.4.9 Reset Processing Summary** contains details of exception processing.

### 3.4.2 Reset Control Logic

SCIM reset control logic determines the cause of a reset, synchronizes request signals to CLKOUT, and asserts reset control signals. Reset control logic can drive three different internal signals.

EXTRST (external reset) drives the external reset pin.

CLKRST (clock reset) resets the clock module.

MSTRST (master reset) goes to all other internal circuits.

All resets are gated by CLKOUT. Asynchronous resets, except for system reset (the CPU32 RESET instruction), are assumed to be catastrophic. An asynchronous reset can occur on any clock edge. Synchronous resets are timed (CLKOUT) to occur at the end of bus cycles. The internal bus monitor is automatically enabled for synchronous resets. When a bus cycle does not terminate normally, the bus monitor terminates it. Table 3–15 is a summary of reset sources.

**Table 3–15. Reset Sources**

| Type | Source | Timing | Cause | Lines Asserted by Reset Controller | | |
|---|---|---|---|---|---|---|
| External | External | Synch | External Signal | MSTRST | CLKRST | EXTRST |
| Power Up | EBI | Asynch | $V_{DD}$ | MSTRST | CLKRST | EXTRST |
| Software Watchdog | SW Monitor | Asynch | Time Out | MSTRST | CLKRST | EXTRST |
| $\overline{\text{HALT}}$ Assertion | Halt Monitor | Asynch | Double Bus Fault | MSTRST | CLKRST | EXTRST |
| Loss of Clock | Clock | Synch | Loss of Reference | MSTRST | CLKRST | EXTRST |
| Test | Test | Synch | Test Mode | MSTRST | — | EXTRST |
| System | CPU32 | Asynch | RESET Instruction | — | — | EXTRST |

### 3.4.3 Operating Configuration out of Reset

The logic states of certain pins during reset determine SCIM operating configuration. During reset, the SCIM reads pin configuration from DATA[11:2] and DATA0, internal module configuration from DATA[15:12], and basic operating information from $\overline{\text{BERR}}$, MODCLK, DATA1, and $\overline{\text{BKPT}}$. These pins are normally pulled high internally during reset, causing the MCU to default to a specific configuration. However, the user can drive the desired pins low during reset to achieve alternate configurations.

Basic operating options include system clock selection, background mode disable/enable, and external bus configuration. The SCIM supports three external bus configurations: fully-expanded operation with a 24-bit address bus and 16-bit data bus with chip selects, single-chip operation with no external address and data bus, and partially-expanded operation with a 24-bit address bus and an 8-bit external data bus.

Table 3–16 shows basic configuration options.

**Table 3–16. Basic Configuration Options**

| Select Pin | Default Function (Pin Left High) | Alternate Function (Pin Pulled Low) |
|---|---|---|
| MODCLK | Synthesized System Clock | External System Clock |
| $\overline{\text{BKPT}}$ | Background Mode Disabled | Background Mode Enabled |
| $\overline{\text{BERR}}$ | Expanded Mode | Single-Chip Mode |
| DATA1 (if $\overline{\text{BERR}}$ = 1) | 8-Bit Expanded Mode | 16-Bit Expanded Mode |

$\overline{\text{BERR}}$, $\overline{\text{BKPT}}$, and MODCLK do not have internal pull-ups and must be driven to the desired state during reset.

When $\overline{\text{BERR}}$ is high during reset, the MCU is configured for partially or fully expanded operation. DATA2 is then decoded to select 8- or 16-bit data bus operation, DATA8 is decoded to configure pins for bus control or port E operation, and DATA9 is decoded to configure pins for interrupt requests or port F operation. If DATA2 is held low at reset, selecting 16-bit data bus operation, DATA11, DATA[7:2] and DATA0 are also decoded. The following subsections explain the process in greater detail.



### 3.4.3.1 Address and Data Bus Pin Functions

External bus configuration determines whether certain address and data pins are used for those functions or for general-purpose I/O. ADDR[18:3] serve as pins for ports A and B when the MCU is operating in single-chip mode. DATA[7:0] serve as port H pins in partially expanded and single-chip modes, and DATA[15:8] serve as port G pins during single-chip operation. Table 3–17 summarizes bus and port configuration.

**Table 3–17. Bus and I/O Port Pin Functions**

| | Mode-Select Pins | | Address Bus/Data Bus/Port Distribution | | |
|---|---|---|---|---|---|
| Bus Configuration | $\overline{\text{BERR}}$ | DATA1 | Address Bus Pins [18:3] | Data Bus Pins [15:0] | I/O Ports |
| 16-Bit Expanded | 1 | 0 | ADDR[18:3] | DATA[15:0] | — |
| 8-Bit Expanded | 1 | 1 | ADDR[18:3] | DATA[15:8] | DATA[7:0] = Port H |
| Single Chip | 0 | X | None | None | ADDR[18:11] = Port A<br>ADDR[10:3] = Port B<br>DATA[15:8] = Port G<br>DATA[7:0] = Port H |

ADDR[2:0] are normally placed in a high-impedance state in single-chip mode and function as normal address bus pins in the expanded modes. Refer to **APPENDIX D REGISTER SUMMARY** for a discussion of the ABD bit in the SCIM configuration register.

The ADDR[23:19] pins can also be used as chip selects or discrete output pins, depending on the external bus configuration selected at reset. The following paragraphs contain a summary of pin configuration options for each external bus configuration.

### 3.4.3.2 Data Bus Mode Selection

All data lines have weak internal pull-up drivers. When pins are held high by the internal drivers, the MCU uses a default operating configuration. However, specific lines can be held low externally to achieve an alternate configuration.

**NOTE**

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

3

Use an active device to hold data bus lines low. Data bus configuration logic must release the bus prior to the first bus cycle after reset in order to prevent conflict with external memory devices. The first bus cycle occurs 10 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 3–16 shows a recommended method for conditioning the mode select signals.

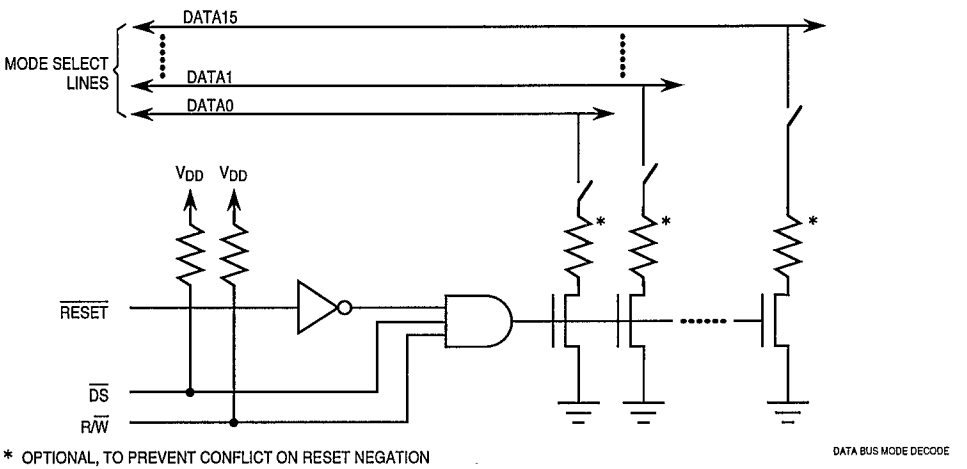



**Figure 3–16. Data Bus Mode Select Conditioning**

### 3.4.3.3 Pin Configuration for 16-Bit Data Bus Operation

16-bit data bus operation is selected when $\overline{\text{BERR}}$ = 1 and DATA1 = 0 during reset. In this configuration, pins ADDR[18:3] and DATA[15:0] are configured as address and data pins, respectively. The alternate functions for these pins as ports A, B, G, and H are unavailable. ADDR[23:20] can be configured as chip selects or address bus pins, as shown in Table 3–18. ADDR[2:0] are configured as address bus pins.

DATA2 determines the functions of $\overline{CS0}$/BR, $\overline{CS3}$/FC0, and $\overline{CS5}$/FC2. DATA[7:3] determine the functions of ADDR[23:19]/$\overline{CS}$[10:6]. A data bus pin pulled low selects the associated chip select and all lower-numbered chip-selects down through $\overline{CS6}$. For example, if DATA5 is pulled low during reset, CS[8:6] are configured as address bus signals ADDR[21:19], and CS[10:9] are configured as chip selects. (On some MCUs, ADDR[23:20] follow the state of ADDR19, and DATA[7:4] have limited use.) Refer to **3.6.4 Chip-Select Reset Operation** for more information.

DATA8 determines the function of the $\overline{DSACK0}$, $\overline{DSACK1}$, $\overline{AVEC}$, $\overline{DS}$, $\overline{AS}$, and SIZ[1:0] pins. If DATA8 is held low during reset, these pins are used for discrete I/O (port E).

DATA9 determines the function of interrupt request pins ($\overline{IRQ[7:0]}$) and the clock mode select pin (MODCLK). When DATA9 is held low during reset, these pins are used for discrete I/O (port F).

DATA11 determines whether the SCIM operates in test mode out of reset. This capability is used for factory testing of the MCU.

DATA0 determines the port size of the boot ROM chip-select signal $\overline{CSBOOT}$. Unlike other chip-select signals, $\overline{CSBOOT}$ is active at the release of reset. When DATA0 is held low, port size is 8 bits; when DATA0 is held high, either by the weak internal pull-up driver or by an external pull-up, port size is 16 bits. Refer to **3.6.4 Chip-Select Reset Operation** for more information.

Table 3–18 is a summary of pin function options for 16-bit data bus operation.

**Table 3–18. 16-Bit Data Bus Mode Reset Pin Configuration**

| Pin(s) Affected | Select Pin | Default Function (Pin Left High) | Alternate Function (Pin Pulled Low) |
|---|---|---|---|
| CSBOOT | DATA0 | CSBOOT 16-Bit | CSBOOT 8-Bit |
| BR/CS0<br>FC0/CS3<br>FC1/PC1<br>FC2/CS5/PC2 | DATA2 | CS0<br>CS3<br>FC1<br>CS5 | BR<br>FC0<br>FC1<br>FC2 |
| ADDR19/CS6/PC3<br>ADDR20/CS7/PC4<br>ADDR21/CS8/PC5<br>ADDR22/CS9/PC6<br>ADDR23/CS10/ECLK | DATA3<br>DATA4<br>DATA5<br>DATA6<br>DATA7 | CS6<br>CS[7:6]<br>CS[8:6]<br>CS[9:6]<br>CS[10:6] | ADDR19<br>ADDR[20:19]<br>ADDR[21:19]<br>ADDR[22:19]<br>ADDR[23:19] |
| DSACK0/PE0<br>DSACK1/PE1<br>AVEC/PE2<br>PE3<br>DS/PE4<br>AS/PE5<br>SIZ0/PE6<br>SIZ1/PE7 | DATA8 | DSACK0<br>DSACK1<br>AVEC<br>PE3<br>DS<br>AS<br>SIZ0<br>SIZ1 | PE0<br>PE1<br>PE2<br>PE3<br>PE4<br>PE5<br>PE6<br>PE7 |
| MODCLK/PF0<br>IRQ[7:1]/PF[7:1] | DATA9 | MODCLK<br>IRQ[7:1] | PF0<br>PF[7:1] |
| BGACK/CSE<br>BG/CSM | DATA10 | BGACK<br>BG | CSE[1]<br>— [2] |
| DATA11 | DATA11 | Slave Mode Disabled[3] | Slave Mode Enabled[3] |

NOTES
1. CSE is enabled when DATA10 = 0 and DATA1 = 0 during reset.
2. Pin tied high when DATA13 = 0, DATA10 = 0, and DATA1 = 0 during reset.
3. Slave mode is used for factory testing only.

### 3.4.3.4 Pin Configuration for 8-Bit Data Bus Operation

The SCIM uses an 8-bit data bus when BERR = 1 and DATA1 = 1 during reset. In this configuration, pins DATA[7:0] are configured as port H, an 8-bit I/O port. Pins DATA[15:8] are configured as data bus pins, and ADDR[18:3] are configured as address bus pins. The alternate functions for these address and data bus pins as ports A, B, and G are unavailable. ADDR[23:19]/CS[10:6] are configured as chip selects. ADDR[2:0] are configured as address bus pins. Emulator mode is always disabled.

DATA8 determines the function of the DSACK0, DSACK1, AVEC, DS, AS, and SIZ[1:0] pins. If DATA8 is held low during reset, these pins are used for discrete I/O (port E).

DATA9 determines the function of interrupt request pins (IRQ[7:0]) and the clock mode select pin (MODCLK). When DATA9 is held low during reset, these pins are used for discrete I/O (port F).

Table 3–19 is a summary of pin function selection for 8-bit data bus operation.

**Table 3–19. 8-Bit Expanded Mode Reset Configuration**

| Pin(s) Affected | Select Pin | Default Function (Pin Left High) | Alternate Function (Pin Pulled Low) |
|---|---|---|---|
| CSBOOT | N/A[1] | CSBOOT 8-Bit | CSBOOT 8-Bit |
| BR/CS0<br>FC0/CS3/PC0<br>FC1/PC1<br>FC2/CS5/PC2 | N/A[1] | CS0<br>CS3<br>FC1<br>CS5 | CS0<br>CS3<br>FC1<br>CS5 |
| ADDR19/CS6/PC3<br>ADDR20/CS7/PC4<br>ADDR21/CS8/PC5<br>ADDR22/CS9/PC6<br>ADDR23/CS10/ECLK | N/A[1] | CS[10:6] | CS[10:6] |
| DSACK0/PE0<br>DSACK1/PE1<br>AVEC/PE2<br>PE3<br>DS/PE4<br>AS/PE5<br>SIZ0/PE6<br>SIZ1/PE7 | DATA8 | DSACK0<br>DSACK1<br>AVEC<br>PE3<br>DS<br>AS<br>SIZ0<br>SIZ1 | PE0<br>PE1<br>PE2<br>PE3<br>PE4<br>PE5<br>PE6<br>PE7 |
| MODCLK/PF0<br>IRQ[7:1]/PF[7:1] | DATA9 | MODCLK<br>IRQ[7:1] | PF0<br>PF[7:1] |
| BGACK/CSE<br>BG | N/A[1] | BGACK<br>BG | BGACK<br>BG |

NOTES
1. These pins have only one reset configuration in 8-bit expanded mode.

### 3.4.3.5 Pin Configuration for Single-Chip Operation

Single-chip operation is selected when BERR = 0 during reset. BERR can be tied low permanently to select this configuration. In single-chip configuration, pins DATA[15:0] are configured as two 8-bit I/O ports, ports G and H. ADDR[18:3] are configured as two 8-bit I/O ports, ports A and B. There is no external data bus path. Expanded mode configuration options are not available: I/O ports A, B, C, E, F, G, and H are always selected. ADDR[2:0] come out of reset in a high-impedance state. After reset, clearing the ABD bit in the SCIM configuration register enables these pins, and leaving the bit set (its single-chip reset state) leaves the pins in a disabled (high-impedance) state. For more information, refer to **APPENDIX D REGISTER SUMMARY** for a discussion of the ABD bit in the SCIM configuration register.

Table 3–20 is a summary of SCIM pin function during single-chip operation.

**Table 3–20. Single-Chip Mode Reset Pin Configuration**

| Pin(s) Affected | Function |
|---|---|
| $\overline{\text{CSBOOT}}$ | $\overline{\text{CSBOOT}}$ 8-Bit |
| ADDR[18:10] | PA[7:0] |
| ADDR[9:3] | PB[7:0] |
| BR/$\overline{\text{CS0}}$ | $\overline{\text{CS0}}$ |
| FC0/$\overline{\text{CS3}}$/PC0<br>FC1/PC1<br>FC2/$\overline{\text{CS5}}$/PC2<br>ADDR19/$\overline{\text{CS6}}$/PC3<br>ADDR20/$\overline{\text{CS7}}$/PC4<br>ADDR21/$\overline{\text{CS8}}$/PC5<br>ADDR22/$\overline{\text{CS9}}$/PC6 | PC[6:0] |
| ADDR23/$\overline{\text{CS10}}$/ECLK | — |
| $\overline{\text{DSACK0}}$/PE0<br>$\overline{\text{DSACK1}}$/PE1<br>$\overline{\text{AVEC}}$/PE2<br>PE3<br>$\overline{\text{DS}}$/PE4<br>$\overline{\text{AS}}$/PE5<br>$\overline{\text{SIZ0}}$/PE6<br>SIZ1/PE7 | PE[7:0] |
| MODCLK/PF0<br>IRQ[7:1]/PF[7:1] | PF0<br>PF[7:1] |
| DATA[15:8] | PG[7:0] |
| DATA[7:0] | PH[7:0] |
| $\overline{\text{BGACK}}$/$\overline{\text{CSE}}$<br>$\overline{\text{BG}}$ | $\overline{\text{BGACK}}$<br>$\overline{\text{BG}}$ |

### 3.4.3.6 Clock Mode Selection

The state of the clock mode (MODCLK) pin during reset determines what clock source the MCU uses. When MODCLK is held high during reset, the clock signal is generated from a reference frequency. When MODCLK is held low during reset, the clock synthesizer is disabled, and an external system clock signal must be applied. Refer to **3.2 System Clock** for more information.

**NOTE**

If the MODCLK pin is also used as a parallel port pin, it should be driven to the desired state with an active device, so that bus loading does not result in incorrect clock mode selection.

### 3.4.3.7 Breakpoint Mode Selection

The MC68F333 uses internal and external breakpoint ($\overline{\text{BKPT}}$) signals. During reset exception processing, at the release of the $\overline{\text{RESET}}$ signal, the CPU32 samples these signals to determine how to handle breakpoints.

If either $\overline{\text{BKPT}}$ signal is at logic level zero when sampled, an internal BDM flag is set, and the CPU32 enters background debugging mode whenever either $\overline{\text{BKPT}}$ input is subsequently asserted.

If both $\overline{\text{BKPT}}$ inputs are at logic level one when sampled, BKPT exception processing begins whenever either $\overline{\text{BKPT}}$ signal is subsequently asserted.

Refer to **SECTION 4 CENTRAL PROCESSING UNIT** for more information on background debugging mode and exceptions. Refer to **3.3.7 CPU Space Cycles** for information concerning breakpoint acknowledge bus cycles.

### 3.4.3.8 Emulation Mode Selection

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 emulator mode, all port A, B, E, G, and H data and data direction registers and the port E pin assignment register are mapped externally. Refer to **3.6.5 Emulation-Support Chip Select** for more information.

## 3.4.4 MCU Module Pin Function During Reset

Usually, module pins default to port functions, and input/output ports are set to input state. This is accomplished by disabling pin functions in the appropriate control registers, and by clearing the appropriate port data direction registers. Refer to individual module sections in this manual for more information. Table 3–21 is a summary of module pin function out of reset. Refer to **APPENDIX D REGISTER SUMMARY** for register function and reset state.

**Table 3–21. Module Pin Function**

| Module | Pin Mnemonic | Function |
|---|---|---|
| ADC | PADA[7:0]/AN[7:0] | Discrete Input |
| | $V_{RH}$ | Reference Voltage |
| | $V_{RL}$ | Reference Voltage |
| CPU | DSI/$\overline{\text{IFETCH}}$ | DSI/$\overline{\text{IFETCH}}$ |
| | DSO/$\overline{\text{IPIPE}}$ | DSO/$\overline{\text{IPIPE}}$ |
| | $\overline{\text{BKPT}}$/DSCLK | $\overline{\text{BKPT}}$/DSCLK |
| QSM | PQS7/TXD | Discrete Input |
| | PQS[6:4]/PCS[3:1] | Discrete Input |
| | PQS3/PCS0/$\overline{\text{SS}}$ | Discrete Input |
| | PQS2/SCK | Discrete Input |
| | PQS1/MOSI | Discrete Input |
| | PQS0/MISO | Discrete Input |
| | RXD | RXD |
| TPU | TP[15:0] | TPU Channel Input/Output |
| | T2CLK | Clock Input to TCR2 |

### 3.4.5 Pin State During Reset

A pin driver can be in an active, inactive, or disabled (high-impedance) state while reset occurs. During power-on reset, pin state is subject to the constraints discussed in **3.4.7 Power-On Reset**.

**NOTE**

Pins that are not used should either be configured as outputs or, if configured as inputs, pulled to the appropriate inactive state. This decreases additional $I_{DD}$ caused by digital inputs floating near mid-supply level.

#### 3.4.5.1 Reset States of SCIM Pins

While the MCU is held in reset, the data bus pins are configured as inputs, and the address bus, function code, and bus control pins ($\overline{\text{AS}}$, $\overline{\text{DS}}$, R/$\overline{\text{W}}$, SIZ[1:0], and $\overline{\text{RMC}}$) are driven high. Any address bus, data bus, or bus control pins configured for I/O (ports A, B, E, F, G, and H) are configured as inputs when the SCIM comes out of reset.

After $\overline{\text{RESET}}$ is released, mode selection occurs, and reset exception processing begins. Pins configured as inputs during reset become active high-impedance loads after $\overline{\text{RESET}}$ is released. Inputs must be driven to the desired

active state; pull-up or pulldown circuitry may be necessary. Pins configured as outputs begin to function after RESET is released.

Table 3–22 is a summary of SCIM pin states during reset. For many SCIM pins, the pin function out of reset is determined by the external bus configuration selected and the state of data bus mode select pins. For these pins, the reset states for both pin functions are provided. The leftmost pin functions in Table 3–22 are the default pin functions for the 16-bit expanded bus configuration.



**Table 3–22. Pin Reset States**

| Mnemonic | State While RESET Asserted | Pin State After RESET Released | | | |
|---|---|---|---|---|---|
| | | Pin Function | Pin State | Pin Function | Pin State |
| CS10/ADDR23 | 1 | CS10 | 1 | ADDR23 | Unknown |
| CS[9:6]/ADDR[22:19]/PC[6:3] | 1 | CS[9:6] | 1 | ADDR[22:19] | Unknown |
| ADDR[18:0] | High-Z Output | ADDR[18:0] | Unknown | ADDR[18:0] | Unknown |
| AS/PE5 | High-Z Output | AS | Output | PE5 | Input |
| AVEC/PE2 | Disabled | AVEC | Input | PE2 | Input |
| BERR | Disabled | BERR | Input | BERR | Input |
| CSM/BG | 1 | CSM | 1 | BG | 1 |
| CSE/BGACK | 1 | CSE | 1 | BGACK | Input |
| CS0/BR | 1 | CS0 | 1 | BR | Input |
| CLKOUT | Output | CLKOUT | Output | CLKOUT | Output |
| CSBOOT | 1 | CSBOOT | 0 | CSBOOT | 0 |
| DATA[15:0] | Mode Select | DATA[15:0] | Input | DATA[15:0] | Input |
| DS/PE4 | Disabled | DS | Output | PE4 | Input |
| DSACK0/PE0 | Disabled | DSACK0 | Input | PE0 | Input |
| DSACK1/PE1 | Disabled | DSACK1 | Input | PE1 | Input |
| CS5/FC2/PC2 | 1 | CS5 | 1 | FC2 | Unknown |
| FC1/PC1 | 1 | FC1 | 1 | FC1 | Unknown |
| CS3/FC0/PC0 | 1 | CS3 | 1 | FC0 | Unknown |
| HALT | Disabled | HALT | Input | HALT | Input |
| IRQ[7:1]/PF[7:1] | Disabled | IRQ[7:1] | Input | PF[7:1] | Input |
| MODCLK/PF0 | Mode Select | MODCLK | Input | PF0 | Input |
| R/W | Disabled | R/W | Output | R/W | Output |
| RESET | Asserted | RESET | Input | RESET | Input |
| RMC | Disabled | RMC | Output | PE3 | Input |
| SIZ[1:0]/PE[7:6] | Disabled | SIZ[1:0] | Unknown | PE[7:6] | Input |
| TSC | Mode Select | TSC | Input | TSC | Input |

### 3.4.5.2 Reset States of Pins Assigned to Other MCU Modules

As a rule, module pins that are assigned to general-purpose I/O ports go to active high-impedance state following reset. However, during power-up reset, module port pins may be in an indeterminate state for a short period. Refer to **3.4.7 Power-On Reset** for more information. Table 3–23 is a summary of module pin states.



**Table 3–23. Module Pin States**

| Module | Pin Mnemonic | State |
|---|---|---|
| ADC | PADA[7:0]/AN[7:0] | Input |
| | $V_{RH}$ | Applied Voltage |
| | $V_{RL}$ | Applied Voltage |
| CPU | DSI/$\overline{\text{IFETCH}}$ | $\overline{\text{IFETCH}}$ Signal |
| | DSO/$\overline{\text{IPIPE}}$ | $\overline{\text{IPIPE}}$ Signal |
| | $\overline{\text{BKPT}}$/DSCLK | $\overline{\text{BKPT}}$ Signal |
| QSM | PQS7/TXD | Input |
| | PQS[6:4]/PCS[3:1] | Input |
| | PQS3/PCS0/$\overline{\text{SS}}$ | Input |
| | PQS2/SCK | Input |
| | PQS1/MOSI | Input |
| | PQS0/MISO | Input |
| | RXD | RXD Signal |
| TPU | TP[15:0] | Input |
| | T2CLK | Input Clock Signal |

### 3.4.6. Reset Timing

The $\overline{\text{RESET}}$ input must be asserted for a specified minimum period for reset to occur. External $\overline{\text{RESET}}$ assertion can be delayed internally for a period equal to the longest bus cycle time (or the bus monitor timeout period) in order to protect write cycles from being aborted by reset. While $\overline{\text{RESET}}$ is asserted, SCIM pins are either in a disabled, high-impedance state or are driven to their inactive states.

When an external device asserts $\overline{\text{RESET}}$ for the proper period, reset control logic clocks the signal into an internal latch. The control logic drives the $\overline{\text{RESET}}$ pin low for an additional 512 CLKOUT cycles after it detects that the $\overline{\text{RESET}}$ signal is no longer being externally driven, to guarantee this length of reset to the entire system.

cycles, the control logic continues to assert $\overline{\text{RESET}}$ until the internal reset signal is negated.

After 512 cycles have elapsed, the reset input pin goes to an inactive, high-impedance state for 10 cycles. At the end of this 10-cycle period, the reset input is tested. When the input is at logic level one, reset exception processing begins. If, however, the reset input is at logic level zero, the reset control logic drives the pin low for another 512 cycles. At the end of this period, the pin again goes to high-impedance state for 10 cycles, then it is tested again. The process repeats until $\overline{\text{RESET}}$ is released.



### 3.4.7 Power-On Reset

When the SCIM clock synthesizer is used to generate system clocks, power-on reset involves special circumstances related to application of system and clock synthesizer power. Regardless of clock source, voltage must be applied to clock synthesizer power input pin $V_{DDSYN}$ for the MCU to operate. The following discussion assumes that $V_{DDSYN}$ is applied before and during reset, which minimizes crystal start-up time. When $V_{DDSYN}$ is applied at power-on, start-up time is affected by specific crystal parameters and by oscillator circuit design. $V_{DD}$ ramp-up time also affects pin state during reset. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for voltage and timing specifications.

During power-on reset, an internal circuit in the SCIM drives the IMB internal (MSTRST) and external (EXTRST) reset lines. The circuit releases MSTRST as $V_{DD}$ ramps up to the minimum specified value, and SCIM pins are initialized as shown in Table 3–22. As $V_{DD}$ reaches specified minimum value, the clock synthesizer VCO begins operation and clock frequency ramps up to specified limp mode frequency. The external $\overline{\text{RESET}}$ line remains asserted until the clock synthesizer PLL locks and 512 CLKOUT cycles elapse.

The SCIM clock synthesizer provides clock signals to the other MCU modules. After the clock is running and MSTRST is asserted for at least four clock cycles, these modules reset. $V_{DD}$ ramp time and VCO frequency ramp time determine how long the four cycles take. Worst case is approximately 15 milliseconds. During this period, module port pins may be in an indeterminate state. While input-only pins can be put in a known state by external pull-up resistors, external logic on input/output or output-only pins during this time must condition the lines. Active drivers require high-impedance buffers or isolation resistors to prevent conflict.

Figure 3–17 is a timing diagram of power-up reset. It shows the relationships between $\overline{\text{RESET}}$, $V_{DD}$, and bus signals.

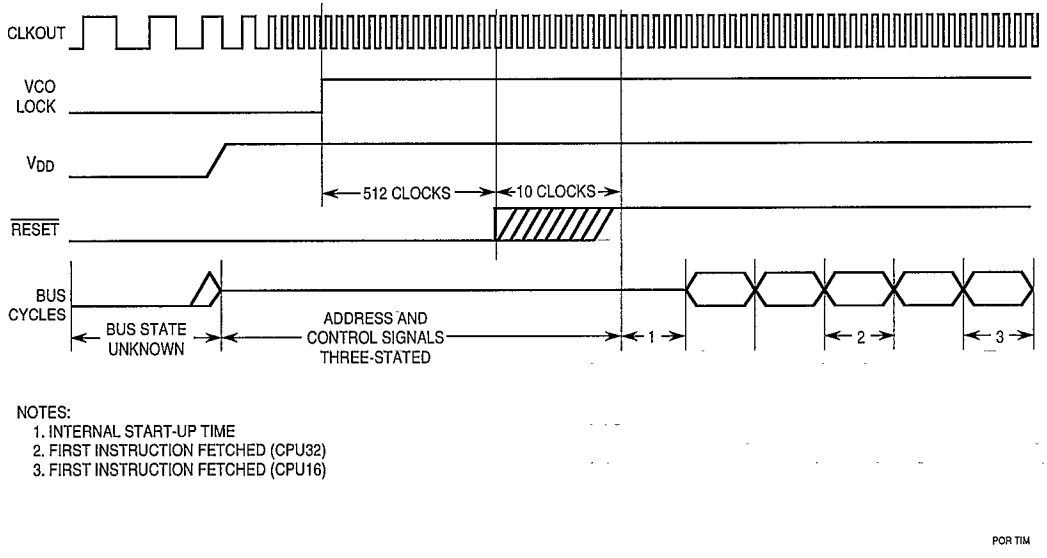



**Figure 3–17. Power-On Reset Timing**

### 3.4.8 Use of Three-State Control Pin

Asserting the three-state control (TSC) input causes the MCU to put all output drivers in an inactive, high-impedance state. The TSC, an active-high input, does not have an internal pull-down and must be tied low when not in use.

TSC must remain asserted for ten clock cycles for drivers to change state. During power-up reset, when the internal clock synthesizer is used (MODCLK held high during reset), synthesizer ramp-up time affects how long the ten cycles take. Worst case is approximately 20 milliseconds from TSC assertion. When an external clock signal is applied (MODCLK held low during reset), pins go to high-impedance state as soon after TSC assertion as 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.

### 3.4.9 Reset Processing Summary

To prevent write cycles in progress from being corrupted, a reset is recognized at the end of a bus cycle, and not at an instruction boundary. Any processing in progress at the time a reset occurs is aborted. After SCIM reset control logic has synchronized an internal or external reset request, it asserts the MSTRST signal.



The following events take place when MSTRST is asserted.

A. Instruction execution is aborted.

B. The status register is initialized.
   1. The T0 and T1 bits are cleared to disable tracing.
   2. The S bit is set to establish supervisor privilege level.
   3. The interrupt priority mask is set to $7, disabling all interrupts below priority 7.

C. The vector base register is initialized to $000000.

The following events take place when MSTRST is negated after assertion.

A. The CPU32 samples the $\overline{\text{BKPT}}$ input.

B. The CPU32 fetches the reset vector:
   1. The first long word of the vector is loaded into the interrupt stack pointer.
   2. The second long word of the vector is loaded into the program counter.

   Vectors can be fetched from internal RAM or from external ROM enabled by the $\overline{\text{CSBOOT}}$ signal.

C. The CPU32 fetches and begins decoding the first instruction to be executed.

### 3.4.10 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 can be set. The reset status register is updated by the reset control logic when the $\overline{\text{RESET}}$ signal is released. Refer to **APPENDIX D REGISTER SUMMARY**.

## 3.5 Interrupts

Interrupt recognition and servicing involve complex interaction between the SCIM, the central processing unit, 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 **3.6 Chip Selects** for more information.

### 3.5.1 Interrupt Exception Processing

The CPU32 processes resets as a type of asynchronous exception. An exception is an event that preempts normal processing. Each exception has an assigned vector that points to an associated handler routine. During exception processing, the CPU fetches the vector assigned to the exception and executes the exception routine to which the vector points.

Exception vectors are stored in a vector table. Out of reset, the table is located beginning at address $000000. This value can be changed by programming the vector base register with the new value. The CPU32 uses vector numbers to calculate displacement into the table. Refer to **SECTION 4 CENTRAL PROCESSING UNIT** for more information concerning exceptions.

### 3.5.2 Interrupt Priority and Recognition

The CPU32 provides for eight levels of interrupt priority (0–7), seven automatic interrupt vectors, and 200 assignable interrupt vectors. All interrupts with priorities less than 7 can be masked by the interrupt priority (IP) field in the status register.

Interrupt recognition is based on the states of interrupt request signals $\overline{\text{IRQ}}$[7:1] and the IP mask value.

Each of the interrupt request signals $\overline{\text{IRQ}}$[7:1] corresponds to an interrupt priority level. $\overline{\text{IRQ1}}$ has the lowest priority, and $\overline{\text{IRQ7}}$ has the highest priority. For periodic timer interrupts, the PIRQ field in the periodic interrupt control register (PICR) determines the priority level. The port F edge-detect interrupt level register (PFLVR) contains the priority level of port F edge-detect interrupts. A priority level of zero in the PICR or PFLVR means that PIT or port F edge-detect interrupts, respectively, are inactive.

Interrupt recognition is based on the interrupt priority level and the interrupt priority mask value. The interrupt priority mask is encoded in the 3-bit interrupt priority (IP) field in the status register. Binary values %000 to %111 provide eight priority masks. Masks prevent an interrupt request of a priority less than or equal to the mask value (except for $\overline{\text{IRQ7}}$) from being recognized and processed. When IP contains %000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

Interrupt request signals can be asserted by external devices or by microcontroller modules. Request lines are connected internally by a wired-NOR. Simultaneous requests of different priorities can be made. Internal assertion of an interrupt request signal does not affect the logic state of the corresponding MCU pin.



External interrupt requests are routed to the CPU32 through the external bus interface and SCIM interrupt control logic. The CPU treats external interrupt requests as if they had come from the SCIM.

External $\overline{\text{IRQ}}$[6:1] are active-low level-sensitive inputs. External $\overline{\text{IRQ7}}$ is an active-low transition-sensitive input that requires both an edge and a voltage level for validity.

$\overline{\text{IRQ}}$[6:1] are maskable. $\overline{\text{IRQ7}}$ is nonmaskable. The $\overline{\text{IRQ7}}$ input is transition-sensitive to prevent redundant servicing and stack overflow. A nonmaskable interrupt is generated each time $\overline{\text{IRQ7}}$ is asserted, and each time the priority mask changes from %111 to a lower number while $\overline{\text{IRQ7}}$ is asserted.

Interrupt request signals 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 periods. Valid requests do not cause immediate exception processing, but are left pending. Pending requests are processed at instruction boundaries or when exception processing of higher-priority exceptions is complete.

The CPU32 does not latch the priority of a pending interrupt request. If an interrupt source of higher priority makes a service request while a lower priority request is pending, the higher priority request is serviced. If an interrupt request of equal or lower priority than the current IP mask value is made, the CPU does not recognize the occurrence of the request in any way.

### 3.5.3 Interrupt Acknowledge and Arbitration

Interrupt acknowledge bus cycles are generated during exception processing. When the CPU32 detects one or more interrupt requests of a priority higher than the interrupt priority mask value, it performs a CPU space read from address $FFFFF : [IP] : 1. (Refer to **3.3.1.7 Function Codes** and **3.3.7 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 also serves two purposes. It is decoded by modules that have requested interrupt service to determine whether the current interrupt acknowledge cycle

pertains to them, and it is latched into the CCR IP field in order 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 interrupt acknowledge 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 make an interrupt service request, including the SCIM, has an IARB field in its configuration register. An IARB field can be assigned a value from %0001 (lowest priority) to %1111 (highest priority). A value of %0000 in an IARB field causes the CPU32 to process a spurious interrupt exception when an interrupt from that module is recognized.

Because the EBI manages external interrupt requests, the SCIM IARB value is used for arbitration between internal and external interrupt requests. The reset value of IARB for the SCIM is %1111, and the reset IARB value for all other modules is %0000. Initialization software must assign different IARB values 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 nonzero value, the CPU interprets multiple vector numbers simultaneously, with unpredictable consequences.

Arbitration must always take place, even when a single source is requesting service. This point is important for two reasons: the EBI does not transfer the CPU interrupt acknowledge cycle to the external bus unless the SCIM wins contention, and failure to contend causes the interrupt acknowledge bus cycle to be terminated early, by a bus error.

When arbitration is complete, the module that wins arbitration must place an interrupt vector number on the data bus and terminate the bus cycle or assert the autovector ($\overline{\text{AVEC}}$) signal. In the case of an external interrupt request, because the interrupt acknowledge cycle is transferred to the external bus, an external device must decode the mask value and respond with a vector number, then generate data and size acknowledge ($\overline{\text{DSACK}}$) cycle termination signals. If the device does not respond in time, the EBI bus monitor asserts the bus error signal ($\overline{\text{BERR}}$), and a spurious interrupt exception is taken. Alternately, an external device can assert $\overline{\text{AVEC}}$ or a chip select can be programmed to

generate $\overline{\text{AVEC}}$ internally. Refer to **3.6.3 Using Chip-Select Signals in Interrupt-Acknowledge Cycles** for additional information.

By hardware convention, when the CPU receives simultaneous interrupt requests of the same level from more than one SCIM source, the periodic interrupt timer is given highest priority, followed by the $\overline{\text{IRQ}}$ pins, and then the port F edge-detect logic.



### 3.5.4 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.

A. The CPU finishes higher priority exception processing or reaches an instruction boundary.

B. Processor state is stacked.

C. The interrupt acknowledge cycle begins:
   1. FC[2:0] are driven to %111 (CPU space) encoding.
   2. The address bus is driven as follows. ADDR[23:20] = %1111; ADDR[19:16] = %1111, which indicates that the cycle is an interrupt acknowledge CPU space cycle; ADDR[15:4] = %111111111111; ADDR[3:1] = the priority of the interrupt request being acknowledged; and ADDR0 = %1.
   3. Request priority is latched into the IP field in the status register from the address bus.

D. Modules or external peripherals that have requested interrupt service, or chip-select circuits programmed to respond to interrupts, decode the priority value in ADDR[3:1]. If request priority is the same as the priority value in the address, interrupt arbitration takes place. When there is no contention, the spurious interrupt monitor asserts $\overline{\text{BERR}}$, and a spurious interrupt exception is processed.

E. After arbitration, the interrupt acknowledge cycle can be completed in one of three ways:
   1. The dominant interrupt source supplies a vector number and $\overline{\text{DSACK}}$ signals appropriate to the access. The CPU32 acquires the vector number.
   2. The $\overline{\text{AVEC}}$ signal is asserted (the signal can be asserted by the interrupt source that wins arbitration or chip-select circuit programmed to assert $\overline{\text{AVEC}}$ internally, or the pin can be tied low), and the CPU32 generates an autovector number corresponding to interrupt priority.

3. The bus monitor asserts $\overline{\text{BERR}}$ and the CPU32 generates the spurious interrupt vector number.

F. The vector number is converted to a vector address.

G. The content of the vector address is loaded into the PC, and the processor transfers control to the exception handler routine.

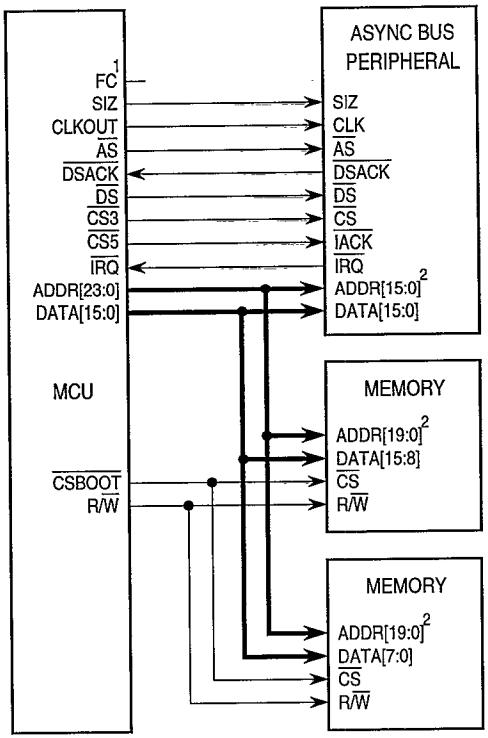


### 3.5.5 Interrupt Acknowledge Bus Cycles

Interrupt acknowledge bus cycles are CPU space cycles that generated during exception processing. Refer to the *SCIM Reference Manual* (SCIMRM/AD) and **APPENDIX A ELECTRICAL CHARACTERISTICS** for additional information about the types of interrupt acknowledge bus cycles that can be executed as part of interrupt exception processing.

## 3.6 Chip Selects

Typical microcontrollers require additional hardware to provide external chip-select signals. The MC68F333 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. An additional chip-select signal ($\overline{\text{CSE}}$) provides support for external emulation of SCIM I/O ports. Figure 3–18 is a diagram of a basic system that uses chip selects.




1. CAN BE DECODED TO PROVIDE ADDITIONAL ADDRESS SPACE.
2. VARIES DEPENDING UPON PERIPHERAL MEMORY SIZE.

32 EXAMPLE SYS BLOCK

**Figure 3–18. Basic MC68F333 System**

Chip-select assertion can be synchronized with bus control signals to provide output enable, read/write strobe, or interrupt acknowledge signals. 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 (if the chip select is programmed to respond during interrupt-acknowledge cycles) to parameters stored in chip-select registers. If all parameters match, the appropriate chip-

select signal is asserted. 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 circuits are configured for operation out of reset. However, all chip-select signals except $\overline{\text{CSBOOT}}$ are disabled and cannot be asserted until the BYTE field in the corresponding option register is programmed to a nonzero value, selecting a transfer size. $\overline{\text{CSBOOT}}$ is automatically asserted out of reset. Alternate functions for chip-select pins are enabled if appropriate data bus pins are held low at the release of the reset signal. Figure 3–19 is a functional diagram of a single chip-select circuit.

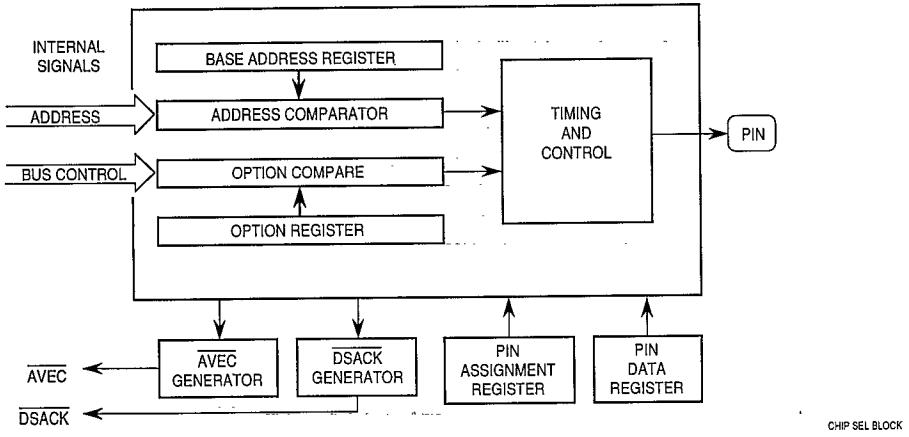



**Figure 3–19. Chip-Select Circuit Block Diagram**

### 3.6.1 Chip-Select Registers

Each chip-select pin can have one or more functions. Chip-select pin assignment registers (CSPAR[1:0]) determine functions of the pins. Pin assignment registers also determine port size (8- or 16-bit) for dynamic bus allocation. The port C data register (PORTC) latches data for chip-select pins that are used for discrete output.

Blocks of addresses are assigned to each chip-select function. Block sizes of two Kbytes to one Mbyte can be selected by writing values to the appropriate

chip select base address register (CSBAR[10:0], CSBARBT). Address blocks for separate chip-select functions can overlap.

Chip select option registers (CSOR[10:5], CSOR3, CSOR0, CSORBT) determine timing of and conditions for assertion of chip-select signals. Eight parameters, including operating mode, access size, synchronization, and wait state insertion can be specified.

Initialization software usually resides in a peripheral memory device controlled by the chip-select circuits. A set of special chip-select functions and registers (CSORBT, CSBARBT) is provided to support bootstrap operation.

Comprehensive address maps and register diagrams are provided in **APPENDIX D REGISTER SUMMARY**.



### 3.6.1.1 Chip-Select Pin Assignment Registers

The pin assignment registers contain eleven 2-bit fields that determine the functions of the chip-select pins. Each pin has two or three possible functions, as shown in Table 3–24.

**Table 3–24. Chip-Select Pin Functions**

| 8- or 16-Bit Chip Select | Alternate Function | Discrete Output or ECLK |
|---|---|---|
| $\overline{\text{CSBOOT}}$ | $\overline{\text{CSBOOT}}$ | — |
| $\overline{\text{CS0}}$ | $\overline{\text{BR}}$ | — |
| —[1] | $\overline{\text{BG}}$ | — |
| $\overline{\text{CSE}}$ | $\overline{\text{BGACK}}$ | — |
| $\overline{\text{CS3}}$ | FC0 | PC0 |
| — | FC1 | PC1 |
| $\overline{\text{CS5}}$ | FC2 | PC2 |
| $\overline{\text{CS6}}$ | ADDR19 | PC3 |
| $\overline{\text{CS7}}$ | ADDR20 | PC4 |
| $\overline{\text{CS8}}$ | ADDR21 | PC5 |
| $\overline{\text{CS9}}$ | ADDR22 | PC6 |
| $\overline{\text{CS10}}$ | ADDR23 | ECLK |

NOTES
1. On the MC68F333, selecting the 8- or 16-bit chip-select function for this pin causes the pin to be held high.

Table 3–25 shows pin assignment field encoding. Pins that have no discrete output function do not use the %00 encoding.

**Table 3–25.**
**Pin Assignment Field Encoding**

| Bit Field | Description |
|---|---|
| 00 | Discrete Output (or ECLK) |
| 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. 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 **3.6.1.3 Chip-Select Option Registers** for more information.

Out of reset chip-select pin function is determined by the logic level on a corresponding data bus pin. (Refer to **3.4.3.1 Address and Data Bus Pin Functions** for more information.) Either 16-bit chip-select function (%11) or alternate function (%01) can be selected during reset. All pins except the boot ROM select pin ($\overline{\text{CSBOOT}}$) are disabled out of reset.

The $\overline{\text{CSBOOT}}$ signal is normally asserted out of reset. The state of the DATA0 line during reset determines what port width $\overline{\text{CSBOOT}}$ uses. If DATA0 is held high (either by the weak internal pull-up driver or by an external pull-up device), 16-bit width is selected. If DATA0 is held low, 8-bit port size is selected.

A pin programmed as a discrete output drives an external signal to the value specified in the pin data register. No discrete output function is available on pins $\overline{\text{CSBOOT}}$, $\overline{\text{BR}}$, $\overline{\text{BG}}$, or $\overline{\text{BGACK}}$. ADDR23 provides 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 $\overline{\text{DSACK}}$ or $\overline{\text{AVEC}}$ internally on an address and control signal match.

### 3.6.1.2 Chip-Select Base Address Registers

Each chip select has an associated base address register. 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 a BLKSZ field. Block addresses for different chip selects can overlap.

The BLKSZ field 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 3–26 shows BLKSZ encoding.



**Table 3–26. Block Size Encoding**

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

The chip-select address compare logic uses only the most significant bits to match an address within a block. The value of the base address must be a multiple of block size. Base address register diagrams show how base register bits correspond to address lines.

After reset, the MCU fetches the initialization routine from the address contained in the reset vector, located beginning at address $000000 of program space. To support bootstrap operation from reset, the base address field in chip-select base address register boot (CSBARBT) has a reset value of all zeros. A memory device containing the reset vector and initialization routine can be automatically enabled by $\overline{\text{CSBOOT}}$ after a reset. The block size field in CSBARBT has a reset value of 512 Kbytes. Refer to **3.6.4 Chip-Select Reset Operation** for more information.

### 3.6.1.3 Chip-Select Option Registers

Option register fields determine timing of and conditions for assertion of chip-select signals. To assert a chip-select signal, and to provide $\overline{\text{DSACK}}$ or autovector support, other constraints set by fields in the option register and in the base address register must also be satisfied. Table 3–27 is a summary of option register functions.

**Table 3–27. Option Register Function Summary**

| MODE | BYTE | R/$\overline{W}$ | STRB | $\overline{DSACK}$ | SPACE | IPL | $\overline{AVEC}$ |
|---|---|---|---|---|---|---|---|
| 0 = Async* | 00 = Disable | 00 = Rsvd | 0 = $\overline{AS}$ | 0000 = 0 wait states | 00 = CPU | 000 = All* | 0 = Off* |
| 1 = Sync. | 01 = Lower | 01 = Read | 1 = $\overline{DS}$ | 0001 = 1 wait state | | 001 = Level 1 | 1 = On |
| | 10 = Upper | 10 = Write | | 0010 = 2 wait states | | 010 = Level 2 | |
| | 11 = Both* | 11 = Both | | 0011 = 3 wait states | | 011 = Level 3 | |
| | | | | 0100 = 4 wait states | | 100 = Level 4 | |
| | | | | 0101 = 5 wait states | | 101 = Level 5 | |
| | | | | 0110 = 6 wait states | | 110 = Level 6 | |
| | | | | 0111 = 7 wait states | | 111 = Level 7 | |
| | | | | 1000 = 8 wait states | 01 = User | 000 = Data/Prog | |
| | | | | 1001 = 9 wait states | 10 = Supv | 001 = Data Sp | |
| | | | | 1010 = 10 wait states | 11 = S/U* | 010 = Prog Sp | |
| | | | | 1011 = 11 wait states | | 011 = Reserved | |
| | | | | 1100 = 12 wait states | | 100 = Reserved | |
| | | | | 1101 = 13 wait states | | 101 = Data Sp | |
| | | | | 1110 = Fast terminate | | 110 = Prog Sp | |
| | | | | 1111 = External | | 111 = Reserved | |

*Use this value when function is not required for chip-select operation.

The **MODE** bit determines whether chip-select assertion is asynchronous or synchronized to the M6800-type bus clock signal (ECLK) available on ADDR23 (refer to **3.2 System Clock** for further information about ECLK).

The **BYTE** field controls bus allocation for chip-select transfers. Port size, set when a chip select is enabled by a pin assignment register, affects signal assertion. When an 8-bit port is assigned, any BYTE field value other than %00 enables the chip select signal. When a 16-bit port is assigned, however, BYTE field value determines when the chip select is enabled. The BYTE fields for all chip selects except $\overline{CSBOOT}$ are cleared during reset. However, both bits in the boot ROM option register (CSORBT) BYTE field are set (%11) when the reset signal is released.

The **R/$\overline{W}$** field causes a chip-select signal to be asserted only for a read, only for a write, or for both read and write. Use this field in conjunction with the STRB bit to generate asynchronous control signals for external devices.

The **STRB** bit controls the timing of a chip-select assertion in asynchronous mode. Selecting address strobe causes a chip-select signal to be asserted synchronized with the address strobe. Selecting data strobe causes a chip-select signal to be asserted synchronized with the data strobe. This bit has no effect in synchronous mode.

The **$\overline{DSACK}$** field specifies the source of data strobe acknowledge signals used in asynchronous mode. It also allows the user to optimize bus speed in a particular application by controlling the number of wait states that are inserted.

The **SPACE** field determines the address space in which a chip select is asserted. An access must have the space type represented by SPACE encoding in order for a chip-select signal to be asserted.

When SPACE = %00, (CPU space) the **IPL** field contains an interrupt priority mask that is compared to the interrupt priority on ADDR[3:1]. If the values are the same (and other option register constraints are satisfied), a chip select signal is asserted. The IPL field only affects the response of chip selects and does not affect interrupt recognition by the CPU. Encoding %000 causes a chip select signal to be asserted regardless of the priority of the interrupt acknowledge cycle, provided all other constraints are met.

When SPACE = %01, %10, or %11, the IPL field specifies whether to assert the chip select during accesses to data space, program space, or both.

The **AVEC** bit selects one of two methods of acquiring an interrupt vector during an external interrupt acknowledge cycle. The internal autovector signal is generated only in response to interrupt requests from the SCIM IRQ pins.

### 3.6.1.4 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. PC[6:0] correspond to CS[9:3]. This is a read/write register. Bit 7 is not used. Writing to this bit has no effect, and it always reads zero.

## 3.6.2 Chip-Select Operation

When the MCU makes an access, enabled chip-select circuits compare the following items:

1. Function codes to SPACE fields, and to the IPL field if the SPACE field encoding is not for CPU space
2. Appropriate ADDR bits to base address fields
3. Read/write status to R/W fields
4. ADDR0 and/or SIZ bits to the BYTE field (16-bit ports only)
5. Priority of the interrupt being acknowledged (ADDR[3:1]) to IPL fields (when the access is an interrupt acknowledge cycle)

When a match occurs, the chip-select signal is asserted. Assertion occurs at the same time as AS or DS assertion in asynchronous mode. Assertion is synchronized with ECLK in synchronous mode. In asynchronous mode, the value of the DSACK field determines whether DSACK signals are generated

internally. If they are, the DSACK field determines the number of wait states inserted before DSACK assertion.

The speed of an external device determines whether internal wait states are needed. Normally, wait states are inserted into the bus cycle during S3 until a peripheral asserts DSACK. If a peripheral does not generate DSACK signals, internal DSACK generation must be selected and a predetermined number of wait states can be programmed into the chip-select option register.

Refer to the *SCIM Reference Manual* (SCIMRM/AD) for additional information.

### 3.6.3 Using Chip-Select Signals in Interrupt-Acknowledge Cycles

Ordinary I/O bus cycles use supervisor space access, but interrupt acknowledge bus cycles use CPU space access. There are no differences in flow for chip selects in each type of space, but base 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 3–20 shows CPU space encoding for an interrupt-acknowledge cycle. FC[2:0] are set to %111, designating CPU space access. ADDR[3:1] indicate interrupt priority, and the space type field (ADDR[19:16]) is set to %1111, the interrupt acknowledge code. The rest of the address lines are set to one.

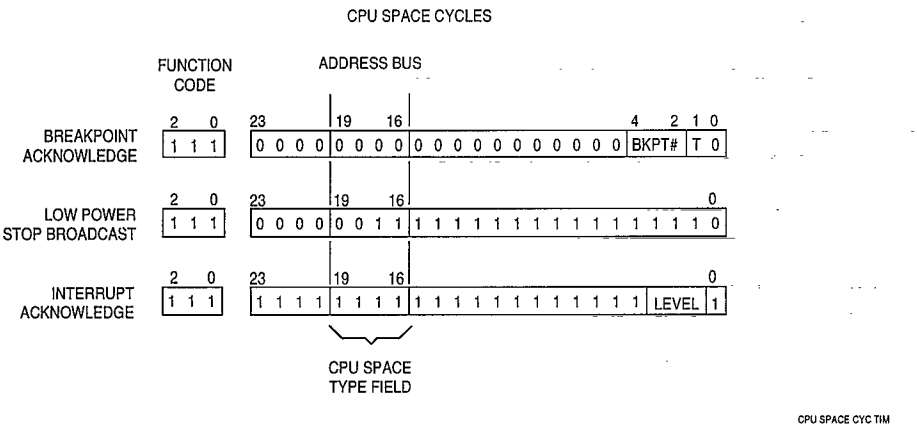



**Figure 3–20. CPU Space Encoding for Interrupt Acknowledge**

Perform the following operations before using a chip select to generate an interrupt acknowledge signal.

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 address lines ADDR[19:16] against the corresponding bits in the base address register. (The CPU places the CPU space type on ADDR[19:16].)
3. Set the R/$\overline{W}$ field to read only. An interrupt acknowledge cycle is performed as a read cycle.
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 acknowledge must be generated. Asserting $\overline{AVEC}$, either by asserting the $\overline{AVEC}$ pin or by generating $\overline{AVEC}$ internally using the chip-select option register, terminates the bus cycle.

### 3.6.4 Chip-Select Reset Operation

The least significant bits of each of the 2-bit pin assignment fields in CSPAR0 and CSPAR1 each have a reset value of one. The reset values of the most significant bits of each field are determined by the states of DATA[7:1] during reset. There are weak internal pull-up drivers for each of the data lines, but these drivers can be overcome by bus loading effects.

The $\overline{CSBOOT}$ assignment field in CSPAR0 is configured differently from the other assignment fields. The MSB, bit 1 of CSPAR0, has a reset value of one. This enables the $\overline{CSBOOT}$ signal to select a boot ROM containing initialization firmware. The LSB value, determined by the logic level of DATA0 during reset, selects boot ROM port size. When DATA0 is held low, port size is eight bits. When DATA0 is held high, port size is 16 bits.

After reset, the MCU fetches the initialization routine from the address contained in the reset vector, located beginning at address $000000 of program space. To support bootstrap operation from reset, the base address field in chip-select base address register boot (CSBARBT) has a reset value of all zeros. A memory device containing the reset vector and initialization routine can be automatically enabled by $\overline{CSBOOT}$ after a reset. The block size field in CSBARBT has a reset value of 1 Mbyte.

The BYTE field in option register CSORBT has a reset value of both bytes. The BYTE fields in all other chip-select option registers have a reset value of disable, as they should not select external devices until an initial program sets up the base and option registers.



Table 3–28 shows the reset values in the base and option registers for $\overline{\text{CSBOOT}}$.

**Table 3–28. $\overline{\text{CSBOOT}}$ Base and Option Register Reset Values**

| Fields | Reset Values |
|---|---|
| ADDR[15:3] | $0000 0000 |
| BLKSZ | 1 Mbyte |
| MODE | Asynchronous Mode |
| BYTE | Both Bytes |
| R/$\overline{\text{W}}$ | Read/Write |
| STRB | $\overline{\text{AS}}$ |
| $\overline{\text{DSACK}}$ | 13 Wait States |
| SPACE | Supervisor/User Space |
| IPL | Any Level |
| $\overline{\text{AVEC}}$ | Interrupt Vector Externally |

### 3.6.5 Emulation-Support Chip Select

The SCIM contains an emulator chip-select signal ($\overline{\text{CSE}}$) and associated logic that supports external emulation of on-chip I/O ports. This emulation support allows system development of a single-chip application in expanded mode.

Chip select pin assignment register 0 (CSPAR0) assigns the $\overline{\text{CSE}}$/$\overline{\text{BGACK}}$ pin for chip-select or alternate function. $\overline{\text{CSE}}$ does not have associated base and option registers.

Emulator 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 emulator 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 register and the port F data, data direction, and pin assignment registers are accessible normally in emulator mode.

An emulator 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 SCIM 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.

On some MCUs, the $\overline{BG}$ and $\overline{CSM}$ (internal module chip select) signals share the same pin. The MC68F333, however, does not support the $\overline{CSM}$ signal. The chip-select pin assignment register 0 (CSPAR0) should be programmed to assign the $\overline{BG}$ pin its bus grant function.



## 3.7 General Purpose I/O

The SCIM contains six general-purpose input/output ports: ports A, B, E, F, G, and H. (Port C, an output-only port, is included in 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 or 8-bit expanded modes only. Ports E, F, G, and H have an associated data direction register to configure each pin as input or output. Ports A and B share a data direction register that configures each port as input or output. Ports E and F have associated pin assignment registers which configure each pin as digital I/O or an alternate function. Port F has an edge-detect flag register which indicates whether a transition has occurred on any of its pins.

Table 3–29 shows the shared functions of the general-purpose I/O ports and the external bus configurations in which the I/O ports are available.

**Table 3–29. General-Purpose I/O Ports**

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

Access to port A, B, E, G, and H data and data direction registers and port E pin assignment register requires three clock cycles, to ensure timing compatibility with external port replacement logic. 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 A-B-G-H port data registers.

If emulator mode is enabled, accesses to ports A, B, E, G, and H data and data direction registers and port E pin assignment register are mapped externally

and cause the emulator chip-select signal ($\overline{CSE}$) to be asserted. The SCIM does not respond to these accesses, allowing external logic, such as a Motorola port replacement unit, to respond. Port F registers, however, remain accessible.



A write to the port A, B, E, F, G, or H data register is stored in the internal data latch, and 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 register.

### 3.7.1 Ports A and B

Ports A and B are available in single-chip mode only. The following registers are involved in the operation of ports A and B:

Port A data register (PORTA)
Port B data register (PORTB)
Port A/B data direction register (DDRAB)

DDRAB controls data direction for both ports. Port A and B registers can be read or written any time the MCU is not in emulator mode.

DDA and DDB are two bits in the DDRAB that control the direction of the pin drivers for ports A and B, respectively, when the pins are configured for I/O. Setting DDA or DDB configures all pins in the corresponding port as outputs. Clearing DDA or DDB to zero configures all pins in the corresponding port as inputs.

### 3.7.2 Port E

Port E can be made available in all external bus configurations. The states of $\overline{BERR}$ and DATA8 during reset control whether the port E pins are used as bus control signals or discrete I/O lines. The following registers are involved in port E operation:

Port E Data Register (PORTE)
Port E Data Direction Register (DDRE)
Port E Pin Assignment Register (PEPAR)

If the MCU is in emulator mode, it does not respond to an access of PORTE, DDRE, or PEPAR. This allows port replacement logic to be supplied externally, giving an emulator access to the bus control signals.

PORTE is a single register that can be accessed in two locations. It can be read or written any time the MCU is not in emulator mode.

The bits in the DDRE control the direction of the pin drivers when the pins are configured for I/O. 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. This register can be read or written any time the MCU is not in emulator mode.



The bits in the PEPAR control the function of each port E pin. Any bit set to one defines the corresponding pin to be a bus control signal, with the function shown in Table 3–30. Any bit cleared to zero defines the corresponding pin to be an I/O pin, controlled by PORTE and DDRE.

**Table 3–30. 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 | $\overline{DSACK0}$ |

$\overline{BERR}$ and DATA8 control the state of the PEPAR following reset. If either $\overline{BERR}$ or DATA8 is low during reset, the PEPAR is set to $00, defining all port E pins to be I/O pins. If $\overline{BERR}$ and DATA8 are both high during reset, the register is set to $FF, and all port E pins are defined to be bus control signals.

### 3.7.3 Port F

Port F consists of eight I/O pins, associated logic, and the following registers:

Port F Data Register (PORTF0, PORTF1)
Port F Data Direction Register (DDRF)
Port F Pin Assignment Register (PFPAR)
Port F Edge-Detect Flag Register (PORTFE)
Port F Edge-Detect Interrupt Vector Register (PFIVR)
Port F Edge-Detect Interrupt Level Register (PFLVR)

Port F pins can be configured as external interrupt request inputs, edge-detect inputs, or discrete inputs or outputs. Port F edge-detect logic can detect rising- or falling-edge transitions. The edge-detection logic can make an interrupt service request when the specified edge is detected. In order to enable the

edge-detect interrupt request, an interrupt priority level must be specified by writing a value to the port F interrupt level register (PFLVR). The edge-detect interrupt has the lowest hardware priority in the SCIM — both PIT and external interrupt requests have higher priority.



### 3.7.3.1 Port F Data Register

The port F data register (PORTF0, PORTF1) is a single register that can be accessed in two locations. It can be read or written any time, including when the MCU is in emulator mode. PORTF returns the logic level of input pins when read and drives the specified logic level onto output pins when written. A write to the port F data register is stored in the internal data latch, and if any port F pin is configured as an output, the value stored for that bit is driven on the pin. A read of PORTF returns the value on a pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the data register.

### 3.7.3.2 Port F Pin Assignment Register

The port F pin assignment register (PFPAR) assigns each pin for rising edge detection, falling edge detection, I/O without edge detection, or for external interrupt requests. The fields in the PFPAR determine the functions of pairs of port F pins as shown in Tables 3–31 and 3–32. $\overline{\text{BERR}}$ and DATA9 determine the reset state of this register. If either $\overline{\text{BERR}}$ or DATA9 is low during reset, this register is set to $00, defining all port F pins to be I/O pins. If $\overline{\text{BERR}}$ and DATA9 are both high during reset, the register is set to $FF, which defines all port F pins except PF0 to be interrupt signals.

**Table 3–31. 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] | $\overline{\text{IRQ1}}$, MODCLK* |

*MODCLK signal is only recognized during reset

**Table 3–32. PFPAR Pin Encodings**

| PFPA Bits | Port F Signal |
|---|---|
| 00 | I/O pin without edge detection |
| 01 | Rising edge detection |
| 10 | Falling edge detection |
| 11 | Interrupt request (or MODCLK) |

When a pin is configured in the PFPAR to detect either a rising or falling edge, and such an edge is detected, the corresponding bit in the PORTFE is set. 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.



Depending on the state of $\overline{\text{BERR}}$ and DATA9 during reset, port F pins come out of reset as either interrupt request inputs or discrete inputs. These registers are unaffected by the CPU32 RESET instruction.

#### 3.7.3.3 Port F Data Direction Register

The port F data direction register (DDRF) assigns each pin configured for discrete I/O as an input or output. Setting any bit in this register configures the corresponding pin as an output. Clearing any bit in this register configures the corresponding pin as an input.

#### 3.7.3.4 Port F Edge-Detect Flag Register

The port F edge-detect flag register (PORTFE) indicates when the proper transition has occurred on a port F pin that is configured for edge detection. Optionally, an interrupt can be generated whenever the proper transition has been detected. When the interrupt is serviced, the interrupt service routine can read PORTFE to determine which pin transition caused the interrupt.

#### 3.7.3.5 Port F Interrupt Registers

To enable interrupt detection for port F edge-detect pins, assign an interrupt priority level greater than zero to the port F interrupt level register (PFLVR), and write the interrupt vector number to the port F interrupt vector register (PFIVR).

Port F edge-detect interrupts have the lowest arbitration priority in the SCIM. An interrupt request of the same level from either the periodic interrupt timer or a port F pin configured as an interrupt request ($\overline{\text{IRQ}}$[7:1]) has priority over an edge-detect interrupt request from the port F control logic.

The PFIVR 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. Refer to the appropriate CPU reference manual for interrupt vector assignments.

PFLVR determines the priority level of the port F edge-detect interrupt. The reset value is $00, indicating that the interrupt is disabled. When several sources of interrupts within the SCIM with the same interrupt request level enter

interrupt arbitration, the port F edge-detect interrupt has the lowest arbitration priority.

### 3.7.4 Port G



Port G is available during single-chip operation only. During single-chip operation, these pins are always configured for general-purpose I/O. Two registers are involved in port G operation: the port G data register (PORTG), and the port G data direction register (DDRG).

PORTG can be read or written any time the MCU is not in emulator mode.

The bits in the DDRG control the direction of the pin drivers when the pins are configured as I/O. 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.

### 3.7.5 Port H

Port H is available during single-chip and partially expanded operation only. The function of these pins is determined by external bus configuration; there is no pin assignment register associated with this port. Two registers are involved in port H operation: the port H data register (PORTH), and the port H data direction register (DDRH).

PORTH can be read or written anytime the MCU is not in emulator mode. Reset has no effect.

The bits in the DDRH control the direction of the pin drivers when the pins are configured for I/O. 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.

## 3.8 Factory Testing

The test submodule supports scan-based testing of the various MCU modules. It is integrated into the SCIM to support production testing. Test submodule registers are intended for Motorola use. Register names and addresses are provided in **APPENDIX D REGISTER SUMMARY** to show the user that these addresses are occupied. The TSTME and QUOT pins are also used for factory testing.

# SECTION 4
# CENTRAL PROCESSING 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. The CPU32 is source code and binary code compatible with the M68000 family.

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. There are eight multifunction data registers and seven general-purpose addressing registers. 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.

As controller applications become more complex and control programs become larger, high-level language will become the system designer's choice in programming languages. High-level languages aid rapid development of complex algorithms, with less error, and are readily portable. The CPU32 instruction set supports high-level languages efficiently.

This section is an overview of the CPU32. For detailed information concerning CPU operation, refer to the *CPU32 Reference Manual* (CPU32RM/AD).

A block diagram of the CPU32 is shown in Figure 4–1. 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.

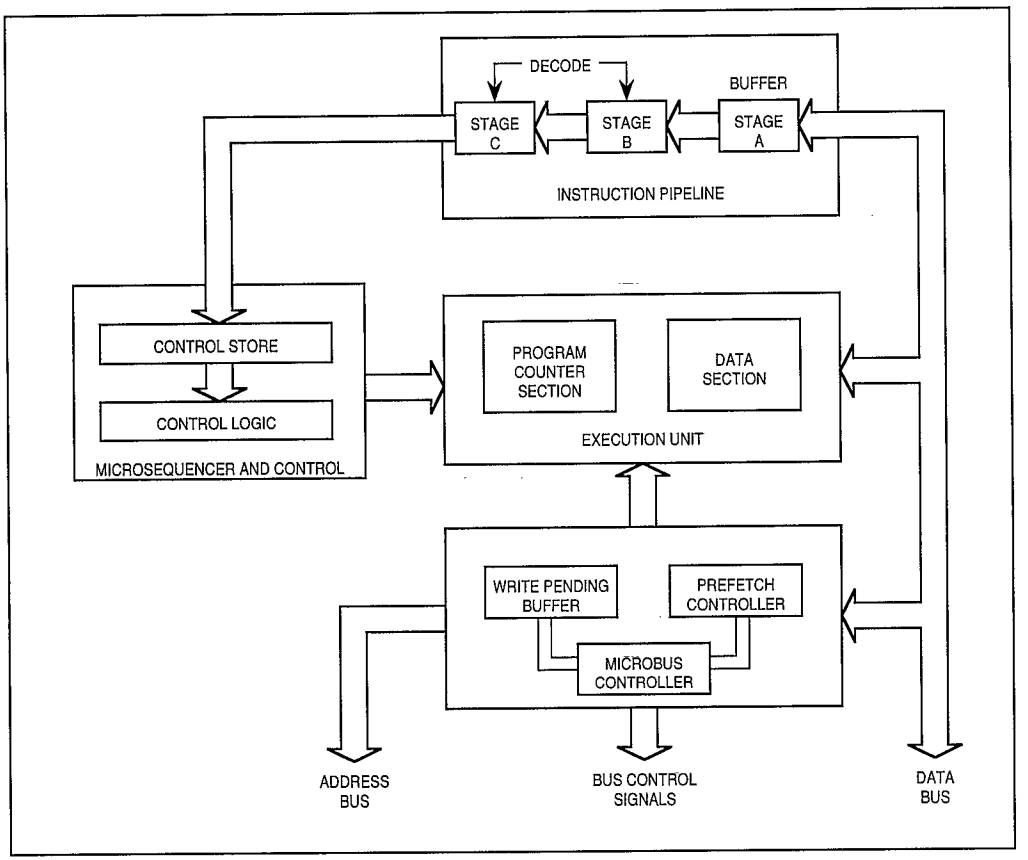



**Figure 4–1. CPU32 Block Diagram**

## 4.1 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 (see Figures 4–2 and 4–3).

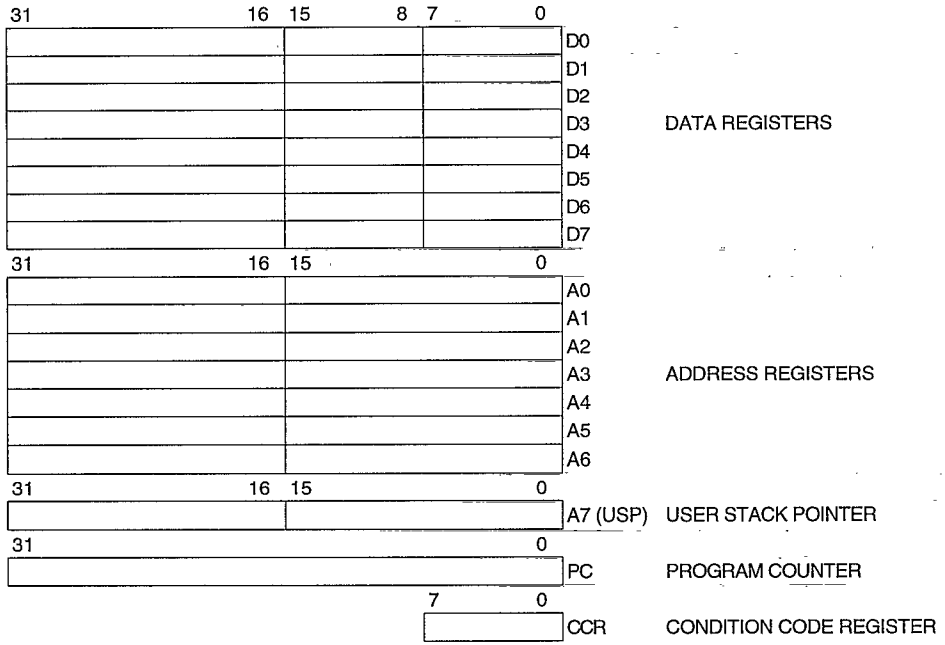



**Figure 4–2. User Programming Model**

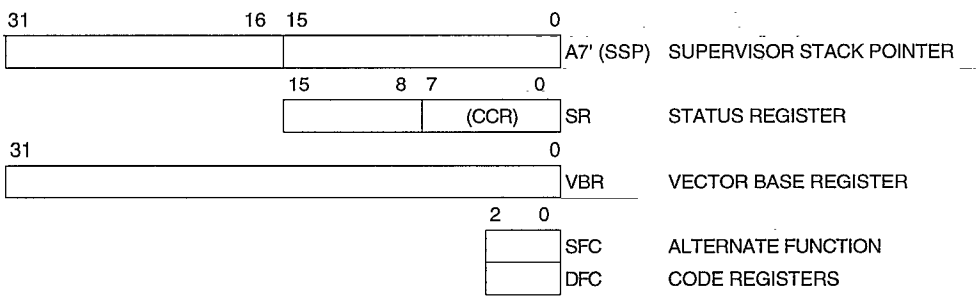



**Figure 4–3. Supervisor Programming Model Supplement**

### 4.1.1 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 4–4 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.

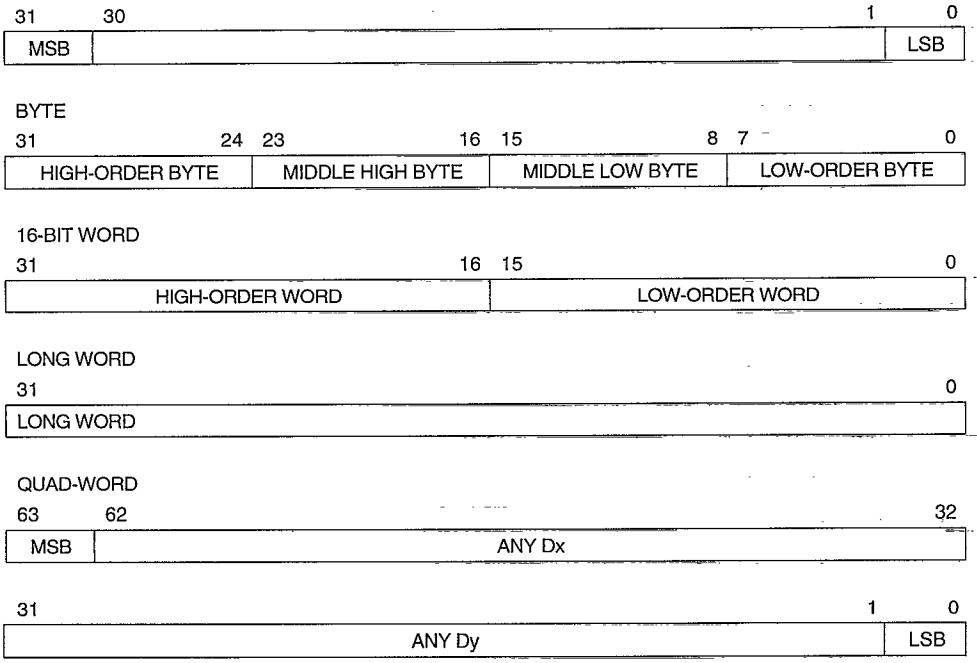



**Figure 4–4. Data Organization in Data Registers**

### 4.1.2 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 operand, the entire register is affected, regardless of the operation size. If the source operand is a word size, it 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 4–5 shows the organization of addresses in address registers.

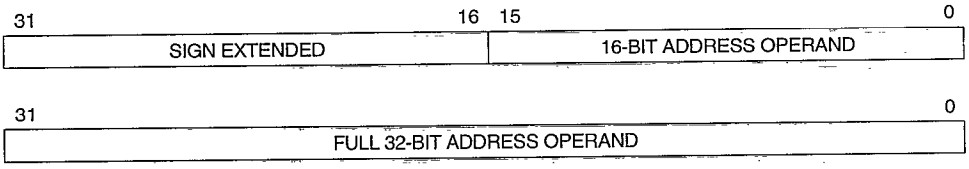



**Figure 4–5. Address Organization in Address Registers**



### 4.1.3 Program Counter

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

### 4.1.4 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), they are accessed only by instructions at the supervisor privilege level.

#### 4.1.4.1 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 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 D REGISTER SUMMARY** for bit/field definitions and a diagram of the status register.

#### 4.1.4.2 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.

### 4.1.5 Vector Base Register

The vector base register (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. Refer to **4.8 Exception Processing** for more information on the VBR and exception processing.

## 4.2 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 instructions to be aligned on word boundaries (refer to Figure 4–6). Data misalignment is not supported.

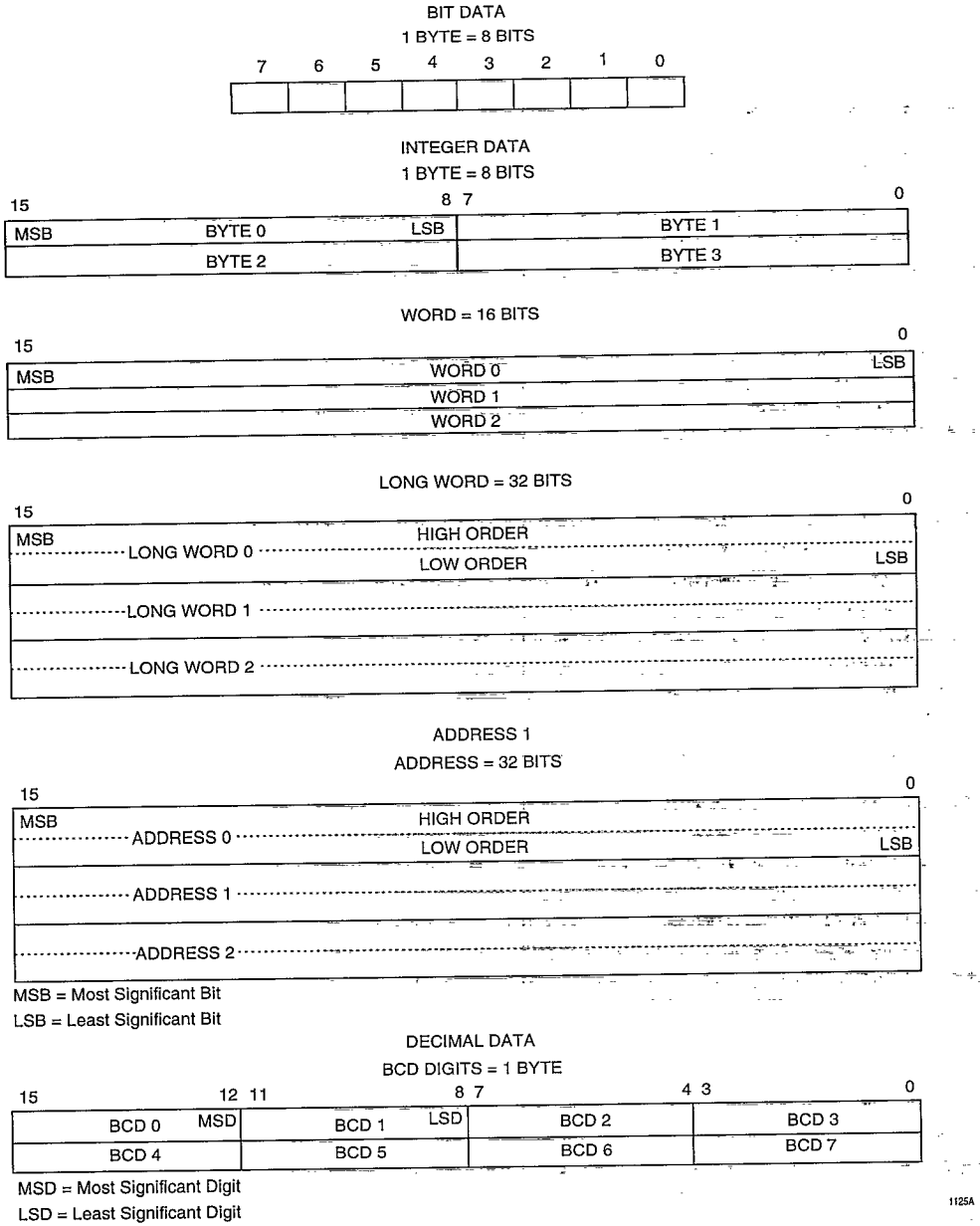



**Figure 4–6. Memory Operand Addressing**

## 4.3 Virtual Memory

The full addressing range of the CPU32 on the MC68F333 is 16 Mbyte 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 Mbyte 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.

## 4.4 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 relative 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.

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

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts.

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.

## 4.6 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, but 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.

## 4.7 Instructions

The CPU32 instruction set is summarized in Table 4–1. 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).

The following MC68020 instructions are not implemented on the CPU32:

| | |
|---|---|
| BFxxx | — 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, cpRESTORE, cpSAVE, cpScc, cpTRAPcc) |
| 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 4–1. Instruction Set Summary

| Mnemonic | Description |
|---|---|
| ABCD | Add Decimal with Extend |
| ADD | Add |
| ADDA | Add Address |
| ADDI | Add Immediate |
| ADDQ | Add Quick |
| ADDX | Add with Extend |
| AND | Logical AND |
| ANDI | Logical AND Immediate |
| ASL, ASR | Arithmetic Shift Left and Right |
| Bcc | Branch Conditionally |
| BCHG | Test Bit and Change |
| BCLR | Test Bit and Clear |
| BGND | Background |
| BKPT | Breakpoint |
| BRA | Branch |
| BSET | Test Bit and Set |
| BSR | Branch to Subroutine |
| BTST | Test Bit |
| CHK, CHK2 | Check Register Against Upper and Lower Bounds |
| CLR | Clear |
| CMP | Compare |
| CMPA | Compare Address |
| CMPI | Compare Immediate |
| CMPM | Compare Memory to Memory |
| CMP2 | Compare Register Against Upper and Lower Bounds |
| DBcc | Test Condition, Decrement and Branch |
| DIVS, DIVSL | Signed Divide |
| DIVU, DIVUL | Unsigned Divide |
| EOR | Logical Exclusive OR |
| EORI | Logical Exclusive OR Immediate |
| EXG | Exchange Registers |
| EXT, EXTB | Sign Extend |
| LEA | Load Effective Address |
| LINK | Link and Allocate |
| LPSTOP | Low Power Stop |
| LSL, LSR | Logical Shift Left and Right |
| ILLEGAL | Take Illegal Instruction Trap |
| JMP | Jump |
| JSR | Jump to Subroutine |
| MOVE | Move |
| MOVE CCR | Move Condition Code Register |
| MOVE SR | Move Status Register |
| MOVE USP | Move User Stack Pointer |
| MOVEA | Move Address |
| MOVEC | Move Control Register |
| MOVEM | Move Multiple Registers |
| MOVEP | Move Peripheral |
| MOVEQ | Move Quick |
| MOVES | Move Alternate Address Space |
| MULS, MULS.L | Signed Multiply |
| MULU, MULU.L | Unsigned Multiply |
| NBCD | Negate Decimal with Extend |
| NEG | Negate |
| NEGX | Negate with Extend |
| NOP | No Operation |
| OR | Logical Inclusive OR |
| ORI | Logical Inclusive OR Immediate |
| PEA | Push Effective Address |
| RESET | Reset External Devices |
| ROL, ROR | Rotate Left and Right |
| ROXL, ROXR | Rotate with Extend Left and Right |
| RTD | Return and Deallocate |
| RTE | Return from Exception |
| RTR | Return and Restore Codes |
| RTS | Return from Subroutine |
| SBCD | Subtract Decimal with Extend |
| Scc | Set Conditionally |
| STOP | Stop |
| SUB | Subtract |
| SUBA | Subtract Address |
| SUBI | Subtract Immediate |
| SUBQ | Subtract Quick |
| SUBX | Subtract with Extend |
| SWAP | Swap Register Words |
| TBLS, TBLSN | Table Lookup and Interpolate (Signed) |
| TBLU, TBLUN | Table Lookup and Interpolate (Unsigned) |
| TAS | Test Operand and Set |
| TRAP | Trap |
| TRAPcc | Trap Conditionally |
| TRAPV | Trap on Overflow |
| TST | Test Operand |
| UNLK | Unlink |

### 4.7.1 M68000 Family Compatibility

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

The CPU32 can be thought of as an intermediate member of the M68000 Family. Object code from an MC68000 or MC68010 may be executed on the CPU32, and many of the instruction and addressing mode extensions of the MC68020 are also supported. Refer to the CPU32 reference manual for a detailed comparison of the CPU32 and MC68020 instruction set.



### 4.7.2 New Instructions

Two new instructions have been added to the MC68000 instruction set for use in controller applications. They are low power stop (LPSTOP) and table lookup and interpolate (TBL).

#### 4.7.2.1 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 a user-specified (or higher) interrupt level or reset occurs.

#### 4.7.2.2 Table Lookup and Interpolate (TBL)

To maximize throughput for real-time applications, reference data is often precalculated and stored in memory for quick access. The storage of each data point could require an inordinate amount of memory. The table instruction requires only a sample of data points stored in the array, reducing memory requirements. Intermediate values are recovered with this instruction via linear interpolation. The results are optionally rounded with the round-to-nearest algorithm.

## 4.8 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.

### 4.8.1 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 4–2 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, are located in supervisor data space. The reset vector is located in supervisor program space. Only the initial reset vector is fixed in the processor memory map. When initialization is complete, there are no fixed assignments. Since the VBR stores the vector table base address, the table can be located anywhere in memory. It can also be dynamically relocated for each task executed by an operating system.

## Table 4–2. 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<br>92 | 040<br>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<br>188 | 080<br>0BC | SD | Trap Instruction Vectors (0–15) |
| 48–58 | 192<br>232 | 0C0<br>0E8 | SD | (Reserved, Coprocessor) |
| 59–63 | 236<br>252 | 0EC<br>0FC | SD | (Unassigned, Reserved) |
| 64–255 | 256<br>1020 | 100<br>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.

### 4.8.2 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.

### 4.8.3 Exception Processing Sequence

For all exceptions other than a reset exception, exception processing occurs in the following sequence. Refer to **3.4 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 $F (the bus cycle is an interrupt acknowledge). For all other exceptions, internal logic provides a vector number.

Next, current processor status is saved. An exception stack frame is created and placed on the supervisor stack. All stack frames contain copies of the 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.

## 4.9 Development Support

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

- M68000 Family Development Support
- Background Debugging Mode
- Deterministic Opcode Tracking
- Hardware Breakpoints



### 4.9.1 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, $4848–$484F, 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.

### 4.9.2 Background Debugging Mode

Microcomputer systems generally provide a debugger, implemented in software, for system analysis at the lowest level. The background debugging 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 (see Figure 4–7), 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 in-circuit emulation. The processor remains in the target system (see Figure 4–8) 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 (e.g., limitations on high-frequency operation, AC and DC parametric mismatches, and restrictions on cable length) are minimized.



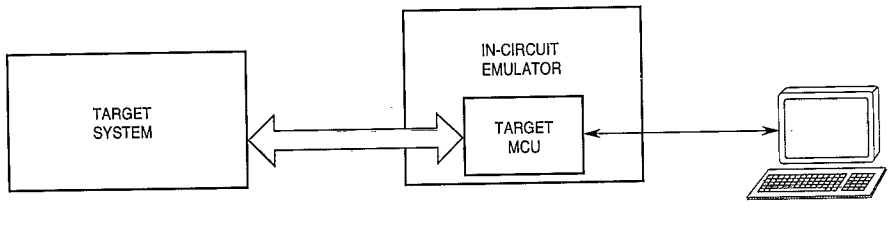



**Figure 4–7. Traditional In-Circuit Emulator Diagram**

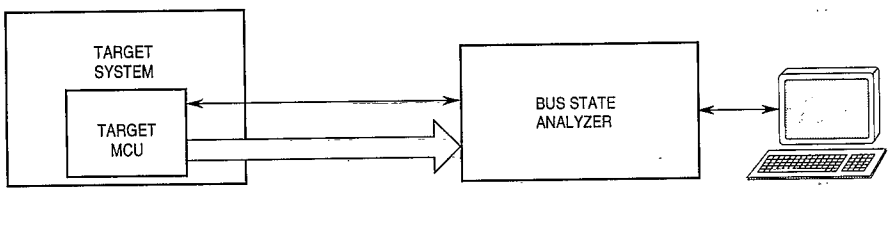



**Figure 4–8. Bus State Analyzer Configuration**

As each command is accumulated in the serial shifter, a microaddress is generated which points to the microcode routine corresponding to that command. If the command can complete without additional serial traffic, it does. However, if addresses or operands are required, the microcode waits as each word is assembled. Result operands are loaded into the output shift register to be shifted out as the next command is read. Table 4–3 summarizes the command set available in the background debugging mode.

**Table 4–3. 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 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. |



### 4.9.3 Deterministic Opcode Tracking

CPU32 function code outputs are augmented by two supplementary signals to monitor the instruction pipeline. The instruction pipe ($\overline{\text{IPIPE}}$) output indicates the start of each new instruction and each mid-instruction pipeline advance. The instruction fetch ($\overline{\text{IFETCH}}$) 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 analyzer to synchronize itself to the instruction stream and monitor its activity.

### 4.9.4 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.

4

## 4.10 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 4–9 shows the required form of an instruction loop for the processor to enter loop mode.

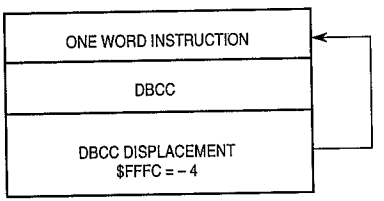



**Figure 4–9. Loop Mode Instruction Sequence**

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.

# SECTION 5
# TIME PROCESSOR UNIT

The time processor unit (TPU) is an intelligent, semi-autonomous microcontroller designed for timing control. Operating simultaneously with the CPU, the TPU schedules tasks, processes ROM instructions, accesses shared data with the CPU, and performs input and output. Figure 5–1 is a simplified block diagram of the TPU.



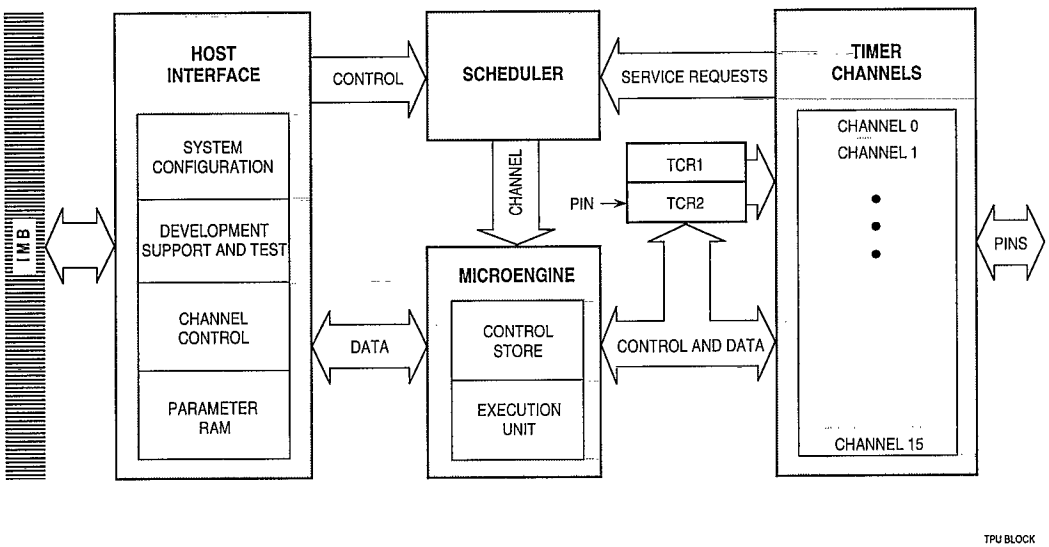



**Figure 5–1. TPU Block Diagram**

## 5.1 Overview

The TPU 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 host CPU interrupt service. The following pre-programmed timing functions are currently available:

- Input capture/input transition counter
- Output compare
- Pulse-width modulation
- Synchronized pulse-width modulation
- Period measurement with additional transition detect
- Period measurement with missing transition detect
- Position-synchronized pulse generator
- Stepper motor
- Period/pulse-width accumulator

5

## 5.2 TPU Components

The TPU module consists of two 16-bit time bases, sixteen independent timer channels, a task scheduler, a microengine, and a host interface. In addition, a dual-port parameter RAM is used to pass parameters between the module and the host CPU.

### 5.2.1 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 host CPU via bit fields in the TPU module configuration register (TPUMCR). Timer count registers TCR1 and TCR2 provide access to current counter values. TCR1 and TCR2 can be read or written by TPU microcode, but are not directly available to the host CPU. The TCR1 clock is derived from the system clock. The TCR2 clock can be derived from the system clock or from an external clock input via the T2CLK pin.

### 5.2.2 Timer Channels

The TPU has 16 independent channels, each connected to an MCU pin. The channels have identical hardware. 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.

### 5.2.3 Scheduler

When a service request is received, the scheduler determines which TPU 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.

### 5.2.4 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 host CPU. Microcode can also be executed from the TPURAM module instead of the control store. The TPURAM module allows emulation and development of custom TPU microcode without the generation of a microcode ROM mask. Refer to **5.3.6 Emulation Support** for more information.



### 5.2.5 Host Interface

Host interface registers allow communication between the host CPU and the TPU, both before and during execution of a time function. The registers are accessible from the IMB through the TPU bus interface unit. Refer to **5.5 Host Interface Registers**, and **APPENDIX D REGISTER SUMMARY** for register bit/field definitions and address mapping.

### 5.2.6 Parameter RAM

Parameter RAM occupies 256 bytes at the top of the system address map. Channel parameters are organized as 128 16-bit words. Although all parameter word locations in RAM can be accessed by all channels, only 100 are normally used: channels 0 to 13 use six parameter words, while channels 14 and 15 each use eight parameter words. The parameter RAM address map in **APPENDIX D REGISTER SUMMARY** shows how parameter words are organized in memory.

The host CPU specifies function parameters by writing the appropriate RAM address. The TPU reads the RAM to determine channel operation. The TPU can also store information to be read by the CPU in 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) for more information.

For pre-programmed functions, one of the parameter words associated with each channel contains three channel control fields. These fields perform the following functions:

PSC — Forces the output level of the pin.
PAC — For input capture, PAC specifies the edge transition to be detected. For output comparison, PAC specifies the logic level to be output when a match occurs.
TBS — Specifies channel direction (input or output) and assigns a time base to the input capture and output compare functions of the channel.



## 5.3 TPU Operation

All TPU 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 TPU can determine precisely when a match or capture event occurs, and respond rapidly. An event register for each channel provides for simultaneity of 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.

### 5.3.1 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. However, before an event can be serviced, any pending previous requests must be serviced. The time needed to respond to and service an event is determined by 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 TPU performance in a given application. Latency can be closely estimated — refer to Motorola *TPU Reference Manual* (TPURM/AD) for more information.

### 5.3.2 Channel Orthogonality

Most timer systems are limited by the fixed number of functions assigned to each pin. All TPU channels contain identical hardware and are functionally

equivalent in operation, 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.

### 5.3.3 Interchannel Communication

The autonomy of the TPU 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.



### 5.3.4 Programmable Channel Service Priority

The TPU 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.

### 5.3.5 Coherency

For data to be coherent, all available portions of it 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.

### 5.3.6 Emulation Support

Although factory-programmed time functions can perform a wide variety of control tasks, they may not be ideal for all applications. The TPU provides emulation capability that allows the user to develop new time functions. Emulation mode is entered by setting the EMU bit in the TPUMCR. In emulation mode, an auxiliary bus connection is made between TPURAM and the TPU module, and access to TPURAM 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, RAM module access timing remains consistent with access timing of the TPU 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 *TPU Reference Manual* (TPURM/AD) for more information about specific functions.

### 5.3.7 TPU 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 nonzero.

The value of the channel interrupt request level (CIRL) field in TICR determines the priority of all TPU interrupt service requests. CIRL values correspond to MCU interrupt request signals $\overline{\text{IRQ}}$[7:1]. $\overline{\text{IRQ7}}$ is the highest-priority request signal; $\overline{\text{IRQ1}}$ has the lowest priority. Assigning a value of %111 to CIRL causes $\overline{\text{IRQ7}}$ 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 %000 disables all interrupts.

The CPU recognizes only interrupt requests of a priority greater than the value contained in the interrupt priority (IP) mask in the condition code register. When the CPU 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 nonzero IARB value in order to implement an arbitration scheme. Arbitration is performed by means of serial assertion of IARB field bit values. IARB is initialized to $0 during reset.

When the TPU wins arbitration, it must respond to the CPU 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 the TPU interrupt configuration register with the 4-bit number of the channel requesting interrupt service. Since the CIBV field has a reset value of %00, 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, or a TPU interrupt service request could cause the CPU to take one of the reserved vectors in the exception vector table.

Refer to **SECTION 3 SINGLE-CHIP INTEGRATION MODULE** for more information about interrupts. For more information about the exception vector table refer to **SECTION 4 CENTRAL PROCESSING UNIT**.

## 5.4 Time Functions

The following paragraphs describe factory-programmed time functions implemented in TPU microcode ROM. A complete description of the functions is beyond the scope of this manual. Refer to the *TPU Reference Manual* (TPURM/AD) for additional information.



### 5.4.1 Discrete Input/Output (DIO)

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.

### 5.4.2 Input Capture/Input Transition Counter (ITC)

Any channel of the TPU can capture the value of a specified TCR upon the occurrence of each transition or specified number of transitions, and then generate an interrupt request to notify the CPU. A channel can perform input captures continually, or a channel can detect a single transition or specified number of transitions, then cease channel activity until reinitialization. After each transition or specified number of transitions, the channel can generate a link to a sequential block of up to eight channels. The user specifies a starting channel of the block and the number of channels within the block. The generation of links depends on the mode of operation. In addition, after each transition or specified number of transitions, one byte of the parameter RAM (at an address specified by channel parameter) can be incremented and used as a flag to notify another channel of a transition.

### 5.4.3 Output Compare (OC)

The output compare 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:

$$\text{OFFSET} = \text{PERIOD} * \text{RATIO}$$

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.



### 5.4.4 Pulse-Width Modulation (PWM)

The TPU can generate a pulse-width modulation 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.

### 5.4.5 Synchronized Pulse-Width Modulation (SPWM)

The TPU generates a PWM waveform in which the CPU can change the period and/or high time at any time. When synchronized to a time function on a second channel, the synchronized PWM low-to-high transitions have a time relationship to transitions on the second channel.

### 5.4.6 Period Measurement with Additional Transition Detect (PMA)

This function and the following function are used primarily in toothed-wheel speed-sensing applications, such as monitoring rotational speed of an engine. The period measurement with additional transition detect function allows for a special-purpose 23-bit period measurement. It can detect the occurrence of an additional transition (caused by an extra tooth on the sensed wheel) indicated by a period measurement that is less than a programmable ratio of the previous period measurement. Once detected, this condition can be counted and compared to a programmable number of additional transitions detected before TCR2 is reset to $FFFF. Alternatively, a byte at an address specified by a channel parameter can be read and used as a flag. A nonzero value of the flag indicates that TCR2 is to be reset to $FFFF once the next additional transition is detected.

### 5.4.7 Period Measurement with Missing Transition Detect (PMM)

Period measurement with missing transition detect allows a special-purpose 23-bit period measurement. It detects the occurrence of a missing transition (caused by a missing tooth on the sensed wheel), indicated by a period measurement that is greater than a programmable ratio of the previous period measurement. Once detected, this condition can be counted and compared to a programmable number of additional transitions detected before TCR2 is reset to $FFFF. In addition, one byte at an address specified by a channel parameter can be read and used as a flag. A nonzero value of the flag indicates that TCR2 is to be reset to $FFFF once the next missing transition is detected.



### 5.4.8 Position-Synchronized Pulse Generator (PSP)

Any channel of the TPU can generate an output transition or pulse, which is a projection in time based on a reference period previously calculated on another channel. Both TCRs are used in this algorithm: TCR1 is internally clocked, and TCR2 is clocked by a position indicator in the user's device. An example of a TCR2 clock source is a sensor that detects special teeth on the flywheel of an automobile using PMA or PMM. The teeth are placed at known degrees of engine rotation; hence, TCR2 is a coarse representation of engine degrees, i.e., each count represents some number of degrees.

Up to 15 position-synchronized pulse generator function channels can operate with a single input reference channel executing a PMA or PMM input function. The input channel measures and stores the time period between the flywheel teeth and resets TCR2 when the engine reaches a reference position. The output channel uses the period calculated by the input channel to project output transitions at specific engine degrees. Because the flywheel teeth might be 30 or more degrees apart, a fractional multiplication operation resolves down to the desired degrees. Two modes of operation allow pulse length to be determined either by angular position or by time.

### 5.4.9 Stepper Motor (SM)

The stepper motor control algorithm provides for linear acceleration and deceleration control of a stepper motor with a programmable number of step rates of up to 14. Any group of channels, up to eight, can be programmed to generate the control logic necessary to drive a stepper motor.

The time period between steps (P) is defined as

$$P(r) = K1 - K2 * r$$

where r is the current step rate (1–14), and K1 and K2 are supplied as parameters.

After providing the desired step position in a 16-bit parameter, the CPU issues a step request. Next, the TPU steps the motor to the desired position through an acceleration/deceleration profile defined by parameters. The parameter indicating the desired position can be changed by the CPU while the TPU is stepping the motor. This algorithm changes the control state every time a new step command is received.

A 16-bit parameter initialized by the CPU for each channel defines the output state of the associated pin. The bit pattern written by the CPU defines the method of stepping, such as full stepping or half stepping. With each transition, the 16-bit parameter rotates one bit. The period of each transition is defined by the programmed step rate.



### 5.4.10 Period/Pulse-Width Accumulator (PPWA)

The period/pulse-width accumulator 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 1 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 1 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 (1 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.

## 5.5 Host Interface Registers

The TPU 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 word accesses. The address space of the TPU memory map occupies 512 bytes. Unused registers within the 512-byte address space return zeros when read.



### 5.5.1 System Configuration Registers

The TPU configuration control registers, TPUMCR and TICR, determine the value of the prescaler, perform emulation control, specify whether the external TCR2 pin functions as a clock source or as gate of the DIV8 clock for TCR2, and determine interrupt request level and interrupt vector number assignment. Refer to **APPENDIX D REGISTER SUMMARY** for more information about TPUMCR and TICR.

#### 5.5.1.1 Prescaler Control for TCR1

Timer control register one (TCR1) is clocked from the output of a prescaler. Two fields in the TPUMCR control TCR1. The prescaler's input is the internal TPU system clock divided by either 4 or 32, depending on the value of the PSCK (prescaler clock) bit. The prescaler divides this input by 1, 2, 4, or 8, depending on the value of TCR1P (timer count register 1 prescaler control). Channels using TCR1 have the capability to resolve down to the TPU system clock divided by 4. Refer to Figure 5–2 and Table 5–1.

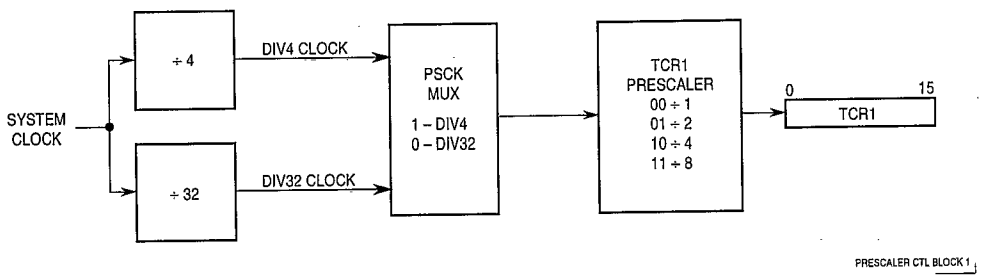



**Figure 5–2. TCR1 Prescaler Control**

**Table 5–1. TCR1 Prescaler Control**

| | | PSCK = 0 | | PSCK = 1 | |
|---|---|---|---|---|---|
| TCR1 Prescaler | Divide By | Number of Clocks | Rate at 16 MHz | Number of Clocks | Rate at 16 MHz |
| 00 | 1 | 32 | 2 ms | 4 | 250 ns |
| 01 | 2 | 64 | 4 ms | 8 | 500 ns |
| 10 | 4 | 128 | 8 ms | 16 | 1 ms |
| 11 | 8 | 256 | 16 ms | 32 | 2 ms |



### 5.5.1.2 Prescaler Control for TCR2

Timer control register two (TCR2), like TCR1, is clocked from the output of a prescaler. The T2CG (TCR2 clock/gate control) bit in TPUMCR determines whether the external TCR2 pin functions as an external clock source for TCR2 or as the gate in the use of TCR2 as a gated pulse accumulator. The function of the T2CG bit is shown in Figure 5–3.

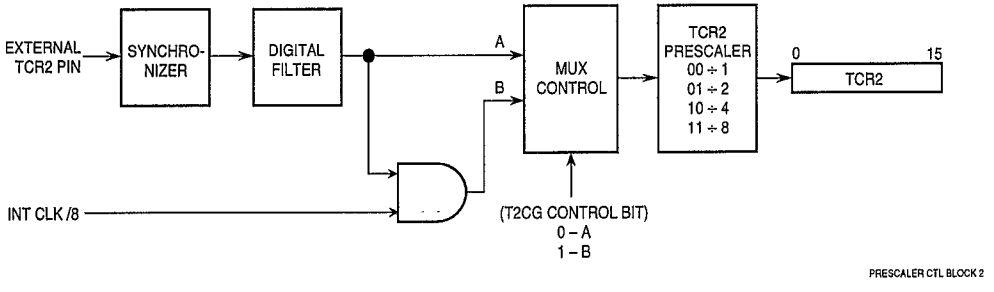



**Figure 5–3. TCR2 Prescaler Control**

When the T2CG bit is set, the external TCR2 pin functions as a gate of the DIV8 clock (the TPU system clock divided by 8). 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 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 TPU system clock divided by 8. Table 5–2 is a summary of prescaler output.

**Table 5–2. 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 |

#### 5.5.1.3 Emulation Control

Asserting the EMU bit in the TPUMCR places the TPU in emulation mode. In emulation mode, the TPU executes microinstructions from TPURAM exclusively. Access to the TPURAM module through the IMB by a host is blocked, and the TPURAM module is dedicated for use by the TPU. After reset, EMU can be written only once.



#### 5.5.1.4 Low-Power Stop Control

If the STOP bit in the TPUMCR is set, the TPU 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 TPU asserts the stop flag (STF) in the TPUMCR to indicate that it has stopped.

### 5.5.2 Channel Control Registers

The channel control and status registers enable the TPU to control channel interrupts, assign time functions to be executed on a specified channel, or select the mode of operation or the type of host service request for the time function specified. Refer to Table 5–3.

#### 5.5.2.1 Channel Interrupt Enable and Status Registers

The channel interrupt enable register (CIER) allows the CPU to enable or disable the ability of individual TPU 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.

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 TPU to make an interrupt service request if the corresponding CIER bit is set and the CIRL field has a nonzero value. To clear a status flag, read CISR, then write a zero to the appropriate bit. CISR is the only TPU register that can be accessed on a byte basis.

#### 5.5.2.2 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 in the TPU ROM are found in Table 5–3.

#### 5.5.2.3 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. Refer to Table 5–3, which is a summary of the host sequence and host service request bits for each time function.



#### 5.5.2.4 Host Service 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 depends on the time function specified. Refer to Table 5–3.

A host service request field cleared to %00 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 nonzero states. It is a good practice to monitor the host service request register and wait until the TPU clears the service request before changing any parameters or issuing a new service request to the channel.

## Table 5–3. Host Sequence Code/Host Service Request Code

| Function Name | Function Code | Host Service Request Code | Host Sequence Code |
|---|---|---|---|
| DIO<br>Discrete<br>Input/Output | $8 | 1= Force Output High | 0 = Trans Mode — Record Pin on Transition |
| | | 2 = Force Output Low | 0 = Trans Mode — Record Pin on Transition |
| | | 3 = Initialization, Input Specified | 0 = Trans Mode — Record Pin on Transition |
| | | 3 = Initialization, Periodic Input | 1 = Match Mode — Record Pin at Match_Rate |
| | | 3 = Update Pin Status Parameter | 2 = Record Pin State on HSR 11 |
| ITC<br>Input Capture/<br>Input Transition<br>Counter | $A | 0 = None<br>1 = Initialization<br>2 = (Not Implemented)<br>3 = (Not Implemented) | 0 = No Link, Single Mode<br>1 = No Link, Continuous Mode<br>2 = Link, Single Mode<br>3 = Link, Continuous Mode |
| OC<br>Output Compare | $E | 0 = None<br>1 = Host-Initiated Pulse Mode<br>2 = (Not Implemented)<br>3 = Continuous Pulse Mode | 0 = Execute All Functions<br>1 = Execute All Functions<br>2 = Only Update TCRn Parameters<br>3 = Only Update TCRn Parameters |
| PWM<br>Pulse-Width<br>Modulation | $9 | 0 = None<br>1 = Immediate Update Request<br>2 = Initialization<br>3 = (Not Implemented) | (None Implemented) |
| SPWM<br>Synchronized<br>Pulse-Width<br>Modulation | $7 | 0 = None<br>1 = (Not Implemented)<br>2 = Initialization<br>3 = Immediate Update Request | 0 = Mode 0<br>1 = Mode 1<br>2 = Mode 2<br>3 = (Not Implemented) |
| PMA/PMM<br>Period<br>Measurement with<br>Additional/Missing<br>Transition Detect | $B | 0 = None<br>1 = Initialization<br>2 = (Not Implemented)<br>3 = (Not Implemented) | 0 = PMA Bank Mode<br>1 = PMA Count Mode<br>2 = PMM Bank Mode<br>3 = PMM Count Mode |
| PSP<br>Position-<br>Synchronized<br>Pulse Generator | $C | 0 = None<br>1 = Immediate Update Request<br>2 = Initialization<br>3 = Force Change | 0 = Pulse Width Set by Angle<br>1 = Pulse Width Set by Time<br>2 = Pulse Width Set by Angle<br>3 = Pulse Width Set by Time |
| SM<br>Stepper Motor | $D | 0 = None<br>1 = None<br>2 = Initialization<br>3 = Step Request | (None Implemented) |
| PPWA<br>Period/Pulse-Width<br>Accumulator | $F | 0 = None<br>1 = (Not Implemented)<br>2 = Initialization<br>3 = (Not Implemented) | 0 = 24-Bit Period<br>1 = 16-Bit Period + Link<br>2 = 24-Bit Pulse Width<br>3 = 16-Bit Pulse Width + Link |

### 5.5.2.5 Channel Priority Registers

The channel priority registers (CPR1, CPR2) assign one of three priority levels to a channel or disable the channel. Table 5–4 indicates the number of time slots guaranteed for each channel priority encoding.

**Table 5–4. Channel Priority Encodings**

| 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 |



### 5.5.3 Development Support and Test Registers

These registers are used for custom microcode development or for factory test. Describing the use of the registers is beyond the scope of this manual. Register descriptions are provided in **APPENDIX D REGISTER SUMMARY**. Refer to the *TPU Reference Manual* (TPURM/AD) for more information.

# SECTION 6
# QUEUED SERIAL MODULE

This section is an overview of the MC68F333 queued serial module (QSM) function. Refer to the *QSM Reference Manual* (QSMRM/AD) for complete information about the QSM.

## 6.1 Overview

The QSM contains two serial interfaces, the queued serial peripheral interface (QSPI) and the serial communication interface (SCI). Figure 6–1 is a block diagram of the QSM.



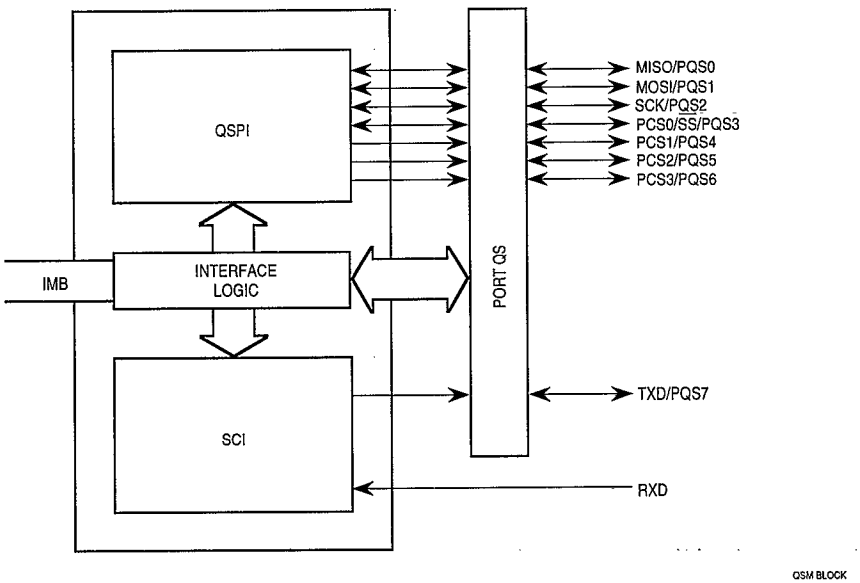



**Figure 6–1. QSM Block Diagram**

The QSPI provides easy peripheral expansion or interprocessor communication through a full-duplex, synchronous, three-line bus. Four programmable peripheral chip selects can select up to 16 peripheral devices. A self-contained RAM queue allows up to sixteen serial transfers of eight to sixteen bits each or transmission of a 256-bit data stream without CPU intervention. A special wraparound mode supports continuous sampling of a serial peripheral, with automatic QSPI RAM updating, for efficient interfacing to A/D converters.

The SCI provides a standard nonreturn to zero (NRZ) mark/space format. It will operate in either full- or half-duplex mode. There are separate transmitter and receiver enable bits and dual data buffers. A modulus-type baud rate generator provides rates from 64 to 524 kbaud (with a 16.78-MHz system clock). Word length of either eight or nine bits can be selected. Optional parity generation and detection provide either even or odd parity check capability. Advanced error detection circuitry catches glitches of up to 1/16 of a bit time in duration. Wakeup functions allow the CPU to run uninterrupted until meaningful data is available.



## 6.2 QSM Registers and Address Map

There are four types of QSM registers: QSM global registers, QSM pin control registers, QSPI registers, and SCI registers. Global registers and pin control registers are discussed in **6.2.1 QSM Global Registers** and **6.2.2 QSM Pin Control Registers**. QSPI and SCI registers are discussed in **6.3 Queued Serial Peripheral Interface** and **6.4 Serial Communication Interface**. Writes to unimplemented register bits have no meaning or effect, and reads from unimplemented bits always return a logic zero value.

The QSM address map includes the QSM registers and the QSPI RAM. The module mapping (MM) bit in the SCIM configuration register (SCIMCR) defines the most significant bit (ADDR23) of the IMB address for each module in the MC68F333.

Refer to **APPENDIX D REGISTER SUMMARY** for a QSM address map and register bit/field definitions. **SECTION 3 SINGLE-CHIP INTEGRATION MODULE** contains more information about how the state of MM affects the system.

### 6.2.1 QSM Global Registers

The QSM configuration register (QSMCR) contains parameters for interfacing to the CPU32 and the intermodule bus. The QSM test register (QTEST) is used during factory test of the QSM. The QSM interrupt level register (QILR) determines the priority of interrupts requested by the QSM and the vector used when an interrupt is acknowledged. The QSM interrupt vector register (QIVR)

contains the interrupt vector for both QSM submodules. QILR and QIVR are 8-bit registers located at the same word address. Refer to **APPENDIX D REGISTER SUMMARY** for register bit and field definitions.

#### 6.2.1.1 Low-Power Stop Operation

When the STOP bit in the QSMCR is set, the system clock input to the QSM is disabled and the module enters a low-power operating state. QSMCR is the only register guaranteed to be readable while STOP is asserted. The QSPI RAM is not readable, but writes to RAM or any register are guaranteed valid while STOP is asserted. STOP can be set by the CPU and by reset.

System software must stop the QSPI and SCI before asserting STOP to prevent data corruption and simplify restart. Disable both SCI receiver and transmitter after transfers in progress are complete. Halt the QSPI by setting the HALT bit in SPCR3 and then setting STOP after the HALTA flag is set. For more information about low-power operation refer to **SECTION 3 SINGLE-CHIP INTEGRATION MODULE.**



#### 6.2.1.2 Freeze Operation

The freeze (FRZ[1:0]) bits in the QSMCR are used to determine what action is taken by the QSM when the IMB FREEZE signal is asserted. FREEZE is asserted when the CPU enters background debugging mode. At the present time, FRZ0 has no effect; setting FRZ1 causes the QSPI to halt on the first transfer boundary following FREEZE assertion. Refer to **SECTION 4 CENTRAL PROCESSING UNIT** for more information about background debugging mode.

#### 6.2.1.3 QSM Interrupts

Both the QSPI and SCI can make interrupt requests on the IMB. Each has a separate interrupt request priority register, but a single vector register is used to generate exception vector numbers.

The values of the ILQSPI and ILSCI fields in the QILR determine the priority of QSPI and SCI interrupt requests. The values in these fields correspond to internal interrupt request signals $\overline{IRQ}$[7:1]. A value of %111 causes $\overline{IRQ7}$ to be asserted when a QSM interrupt request is made; lower field values cause corresponding lower-numbered interrupt request signals to be asserted. Setting field value to %000 disables interrupts. If ILQSPI and ILSCI have the same nonzero 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 interrupt priority (IP) mask in the CPU status register on the address bus. The

QSM compares IP mask value to request priority to determine whether it should contend for arbitration priority. Arbitration priority is determined by the value of the IARB field in the QSMCR. Each module that generates interrupts must have a nonzero IARB value. Arbitration is performed by means of serial assertion of IARB field bit values.

When the QSM wins interrupt arbitration, it responds to the CPU 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, but 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 requests.

At reset, INTV is initialized to $0F, the uninitialized interrupt vector number. To enable interrupt-driven serial communication, a user-defined vector number ($40–$FF) must be written to QIVR, and interrupt handler routines must be located at the addresses pointed to by the corresponding vector. CPU writes to INTV0 have no meaning or effect. Reads of INTV0 return a value of one.

Refer to **SECTION 4 CENTRAL PROCESSING UNIT** and **SECTION 3 SINGLE-CHIP INTEGRATION MODULE** for more information about exceptions and interrupts.

### 6.2.2 QSM Pin Control Registers

The QSM uses nine pins. Eight of the pins can be used for serial communication or for parallel I/O. Clearing a bit in the port QS pin assignment register (PQSPAR) assigns the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI. The PQSPAR does not affect operation of the SCI.

The port QS data direction register (DDRQS) determines whether 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. DDRQS1 determines the direction of the TXD pin only when the SCI transmitter is disabled. When the SCI transmitter is enabled, the TXD pin is an output. PQSPAR and DDRQS are 8-bit registers located at the same word address. Table 6–1 is a summary of QSM pin functions.

The port QS data register (PORTQS) latches I/O data. Writes to PORTQS drive pins defined as outputs. PORTQS reads return data present on the pins. To avoid driving undefined data, first write PORTQS, then configure DDRQS.

**Table 6–1. QSM Pin Function**

| QSM 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 | 0 | Disables Clock Output |
| | | | 1 | Clock Output from QSPI |
| | Slave | | 0 | Clock Input to QSPI |
| | | | 1 | Disables Clock Input |
| PCS0/$\overline{SS}$ | Master | DDQS3 | 0 | Assertion Causes Mode Fault |
| | | | 1 | Chip-Select Output |
| | Slave | | 0 | QSPI Slave Select Input |
| | | | 1 | Disables Select Input |
| PCS[3:1] | Master | DDQS[4:6] | 0 | Disables Chip-Select Output |
| | | | 1 | Chip-Select Output |
| | Slave | | 0 | Inactive |
| | | | 1 | Inactive |
| TXD[2] | Transmit | DDQS7 | X | Serial Data Output from SCI |
| RXD | Receive | None | NA | Serial Data Input to SCI |

1. PQS2 is a digital I/O pin unless the SPI is enabled (SPE in SPCR1 set), in which case it becomes the SPI serial clock SCK.
2. PQS7 is a digital I/O pin unless the SCI transmitter is enabled (TE in SCCR1 set), in which case it becomes SCI serial output TXD and DDRQS has no effect.

## 6.3 Queued Serial Peripheral Interface

The queued serial peripheral interface (QSPI) communicates 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. A variety of transfer rate, clocking, and interrupt-driven communication options are available.

Serial transfer of any number of bits from eight to sixteen can be specified. Programmable transfer length simplifies interfacing to a number of devices that require different data lengths.

An inter-transfer delay of 1 to 500 µs (using a 16.78-MHz system clock) can be specified (default is 1 µs). Programmable delay simplifies the interface to a number of devices that require different delays between transfers.

A dedicated 80-byte RAM is used to store received data, data to be transmitted, and a queue of commands. The CPU can access these locations directly. Serial peripherals can be treated like memory-mapped parallel devices.

The command queue allows the QSPI to perform up to 16 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 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. Multiple-task support can be provided by segmenting the queue.

The QSPI has four peripheral chip-select pins. Chip-select signals simplify interfacing by reducing CPU intervention. If chip-select signals are externally decoded, 16 independent select signals can be generated. Each chip-select pin can drive up to four independent peripherals, depending on loading.



Wraparound operating mode allows continuous execution of queued commands. In wraparound mode, newly received data replaces previously received data in receive RAM. Wraparound can simplify the interface with A/D converters by continuously updating conversion values stored in the RAM.

Continuous transfer mode allows simultaneous transfer of an uninterrupted bit stream. Any number of bits in a range from 8 to 256 can be transferred without CPU intervention. Longer transfers are possible, but minimal CPU intervention is required to prevent loss of data. A 1-µs pause (16.78-MHz system clock) is inserted between each queue entry transfer.

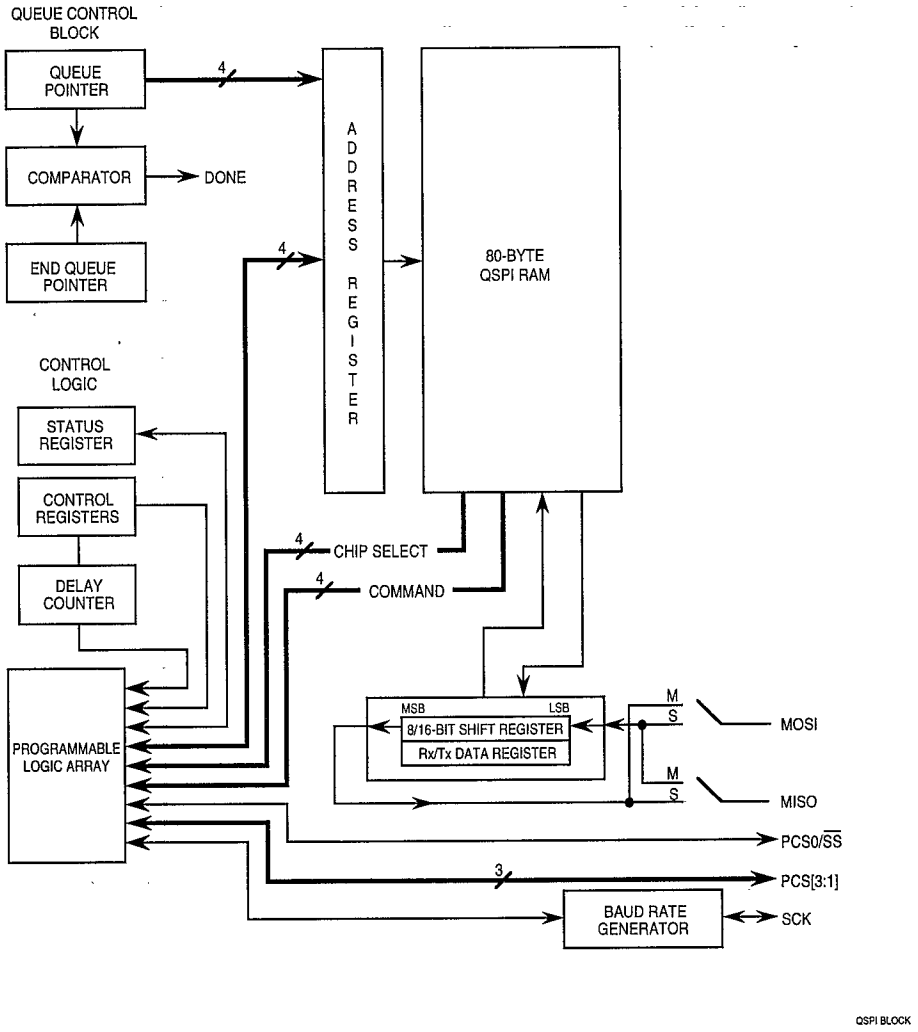



**Figure 6–2. QSPI Block Diagram**

### 6.3.1 QSPI Registers

The programmer's model for the QSPI consists of the QSM global and pin control registers, four QSPI control registers (SPCR[3:0]), a status register (SPCR), and the 80-byte QSPI RAM.

Registers and RAM can be read and written by the CPU. Refer to **APPENDIX D REGISTER SUMMARY** for register bit and field definitions.

#### 6.3.1.1 Control Registers

Control registers contain parameters for configuring the QSPI and enabling various modes of operation. The CPU has read and write access to all control registers, but the QSM has read-only access to all bits except the SPE bit in SPCR1. Control registers must be initialized before the QSPI is enabled to ensure defined operation. SPCR1 must be written last because it contains the QSPI enable bit (SPE).

Writing a new value to any control register except SPCR2 while the QSPI is enabled disrupts operation. SPCR2 is buffered. New SPCR2 values become effective 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 of the buffer.



Writing the same value into any control register except SPCR2 while the QSPI is enabled has no effect on QSPI operation.

#### 6.3.1.2 Status Register

The QSPI status register (SPSR) contains information concerning the current serial transmission. Only the QSPI can set the bits in this register. The CPU reads the SPSR to obtain QSPI status information and writes it to clear status flags.

### 6.3.2 QSPI RAM

The QSPI contains an 80-byte block of dual-access static RAM that can be accessed by both the QSPI and the CPU. The RAM is divided into three segments: receive RAM, transmit RAM, and command RAM. Receive data is information received from a serial device external to the MCU. Transmit data is information stored by the CPU for transmission to an external device. Command control is used to perform transfers. Refer to Figure 6–3, which shows RAM organization.

#### 6.3.2.1 Receive RAM

Data received by the QSPI is stored in this segment. The CPU reads this segment to retrieve data from the QSPI. Data stored in receive RAM is right-justified. Unused bits in a receive queue entry are cleared to zero by the QSPI upon completion of the individual queue entry. The CPU can access the data using byte, word, or long-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.

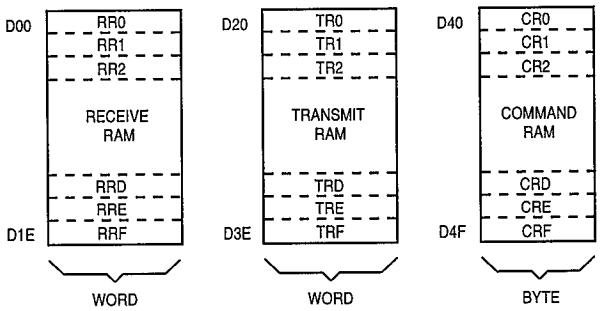



**Figure 6–3. QSPI RAM**

### 6.3.2.2 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.

Information to be transmitted 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.

### 6.3.2.3 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 16 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 16 commands can be in the queue. Queue execution by the QSPI proceeds from the address in NEWQP through the address in ENDQP (both of these fields are in SPCR2).

### 6.3.3 QSPI Pins

The QSPI uses seven pins. These pins can be configured for general-purpose I/O when not needed for QSPI application. When used for QSPI functions, the MOSI, MISO, and $\overline{SS}$ pins should have pull-up resistors.

Table 6–2 shows QSPI input and output pins and their functions.

**Table 6–2. QSPI Pin Function**

| Pin/Signal Name | Mnemonic | Mode | Function |
|---|---|---|---|
| Master In Slave Out | MISO | Master<br>Slave | Serial Data Input to QSPI<br>Serial Data Output from QSPI |
| Master Out Slave In | MOSI | Master<br>Slave | Serial Data Output from QSPI<br>Serial Data Input to QSPI |
| Serial Clock | SCK | Master<br>Slave | Clock Output from QSPI<br>Clock Input to QSPI |
| Peripheral Chip Selects | PCS[3:1] | Master | Select Peripherals |
| Slave Select<br><br>Peripheral Chip Select 0 | $\overline{SS}$<br><br>PCS0 | Master<br>Slave<br>Master | Causes Mode Fault<br>Initiates Serial Transfer<br>Selects Peripherals |

### 6.3.4 QSPI Operation

The QSPI uses a dedicated 80-byte block of static RAM accessible by both the QSPI and the CPU to perform queued operations. The RAM is divided into three segments. There are 16 command control bytes, 16 transmit data words, and 16 receive data words. QSPI RAM is organized so that one byte of command control data, one word of transmit data, and one word of receive data correspond to one queue entry, $0–$F.

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 CPU intervention.

There are four queue pointers. The CPU can access three of them through fields in QSPI registers. The new queue pointer (NEWQP), 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), in the 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 unless wraparound mode is enabled.

At reset, NEWQP is initialized to $0. When the QSPI is enabled, execution begins at queue address $0 unless another value has been written into NEWQP. ENDQP is initialized to $0 at reset, but should be changed to show the last queue entry before the QSPI is enabled. NEWQP and ENDQP can be written at any time. When the NEWQP value 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 $0 causes a single transfer to occur when the QSPI is enabled.



### 6.3.5 QSPI Operating Modes

The QSPI operates in either master or slave mode. Master mode is used when the MCU originates data transfers. Slave mode is used when an external device initiates serial transfers to the MCU through the QSPI. Switching between the modes is controlled by MSTR in SPCR0. Before either mode is entered, appropriate QSM 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 transmit RAM and received by the receive RAM.

In slave mode, operation proceeds in response to $\overline{SS}$ pin activation by an external 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 multimaster 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; Figures 6–5 and 6–6 show QSPI master and slave operation. The CPU must initialize the QSM global and pin registers and the QSPI control registers before enabling the QSPI for either mode of operation (refer to **6.5 QSM Initialization**). 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.



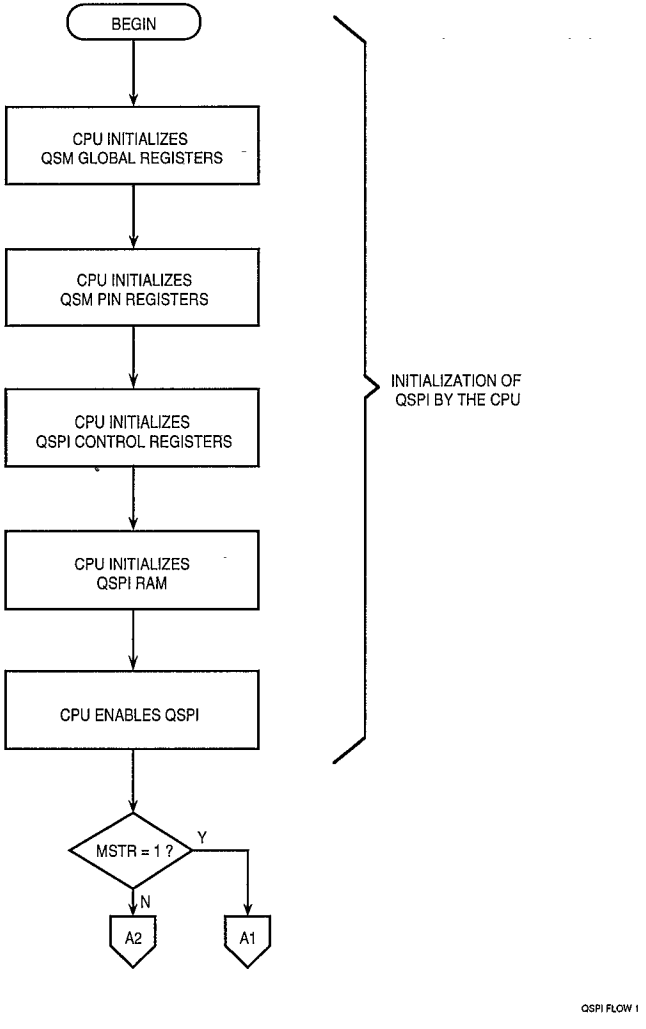



**Figure 6–4. Flowchart of QSPI Initialization Operation**

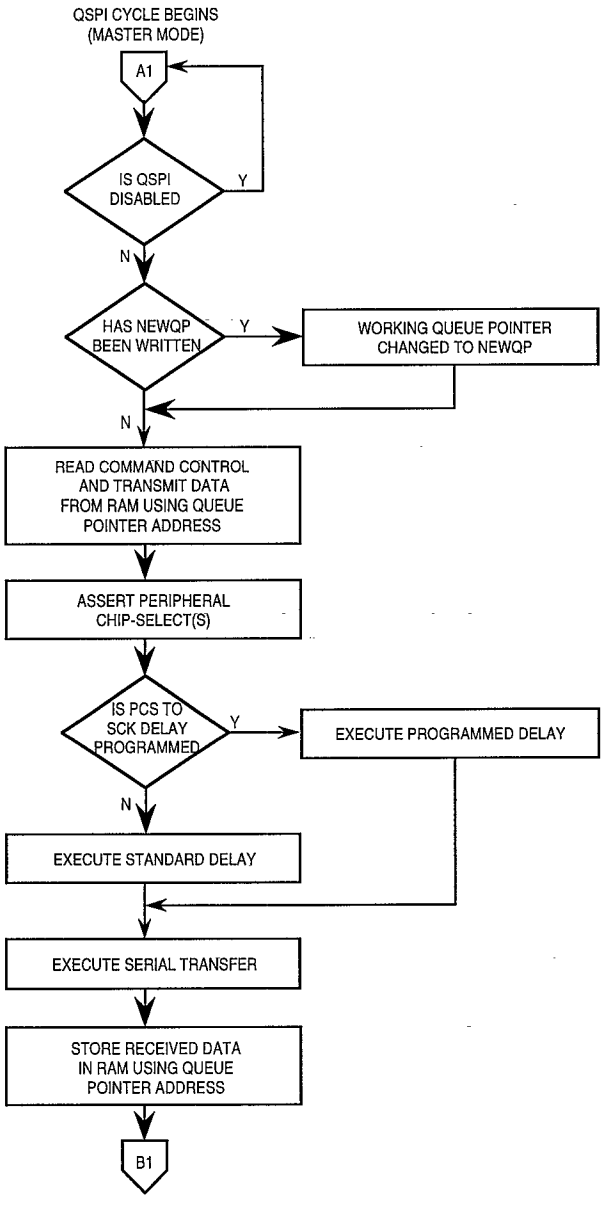



**Figure 6–5. Flowchart of QSPI Master Operation (Part 1)**

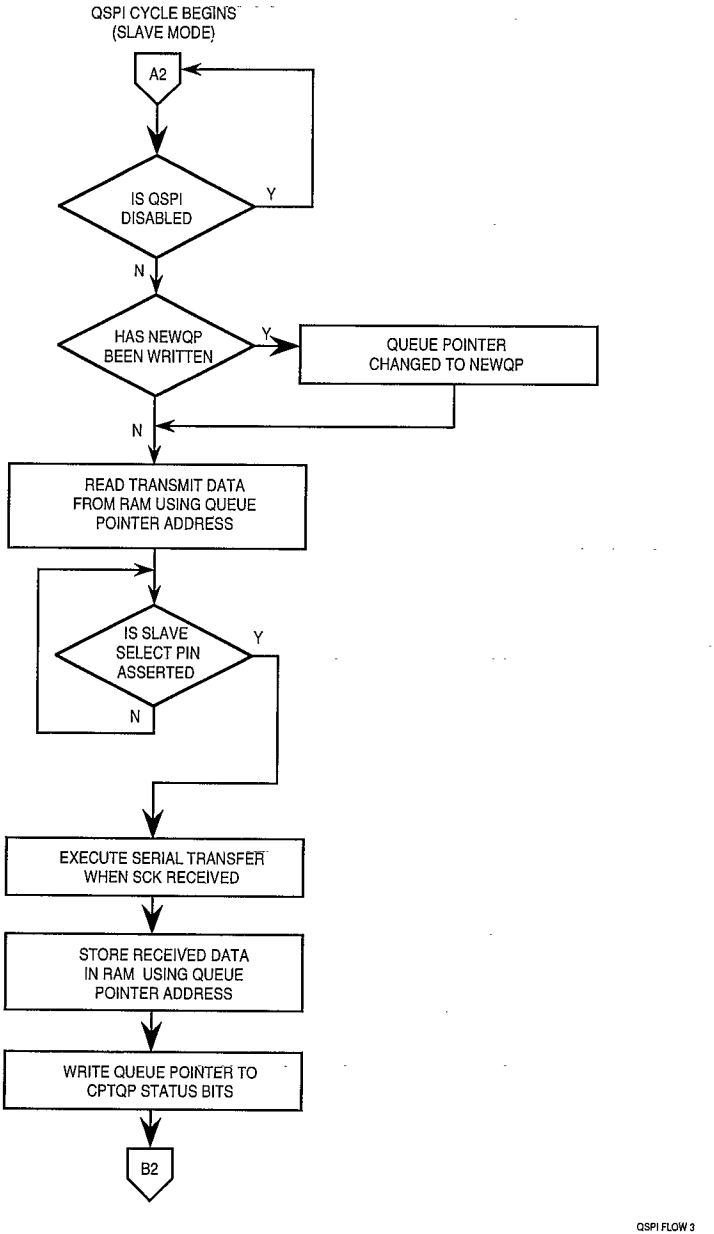



**Figure 6–5. Flowchart of QSPI Master Operation (Part 2)**

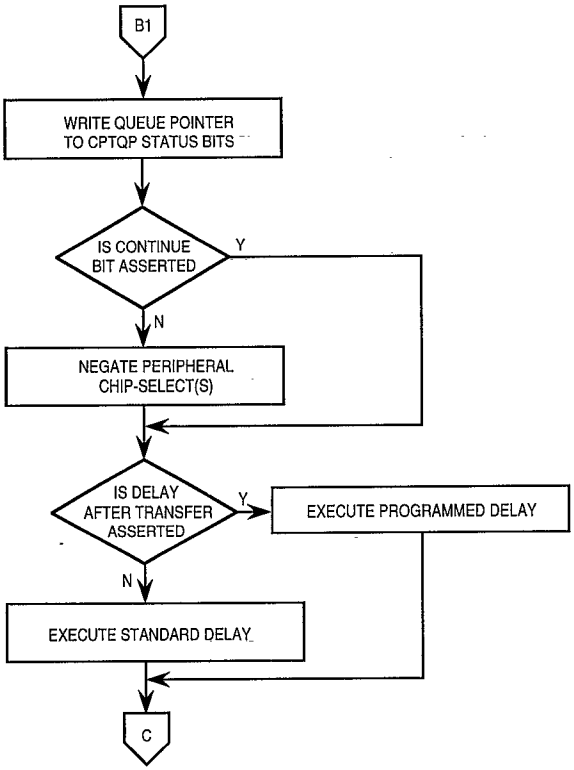



**Figure 6–5. Flowchart of QSPI Master Operation (Part 3)**

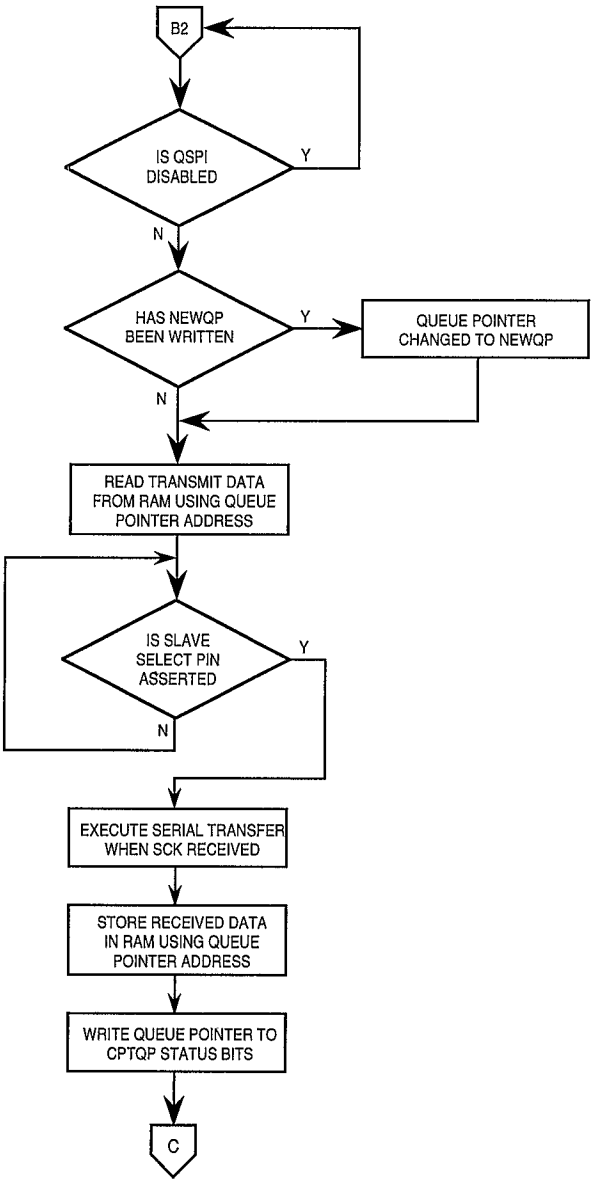



**Figure 6–6. Flowchart of QSPI Slave Operation (Part 1)**

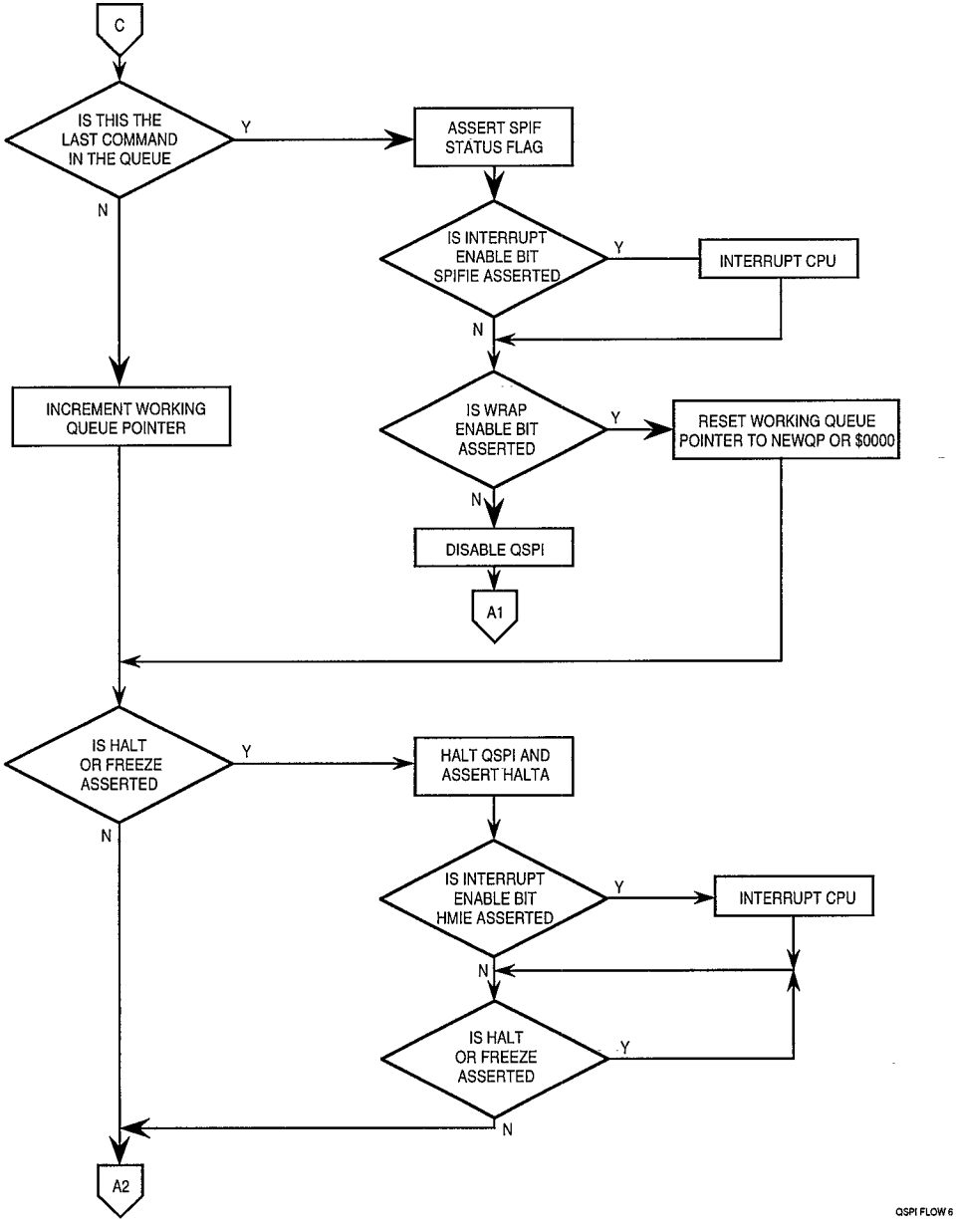



**Figure 6–6. Flowchart of QSPI Slave Operation (Part 2)**

Normally, the SPI bus performs synchronous bidirectional 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 BITSE field 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.



#### 6.3.5.1 Master Mode

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 is initiated, QSM register PQSPAR must be written to assign necessary pins to the QSPI. The pins necessary for master mode operation are MISO and 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.

Before master mode operation is initiated, QSM register DDRQS must be written to direct the data flow on the QSPI pins used. Configure the SCK, MOSI and appropriate chip-select pins PCS[3:0]/$\overline{SS}$ 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.

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.

Baud rate is selected by writing a value from 2 to 255 into the SPBR field in SPCR0. The QSPI uses a modulus counter to derive SCK baud rate from the MCU system clock.

The following expressions apply to SCK baud rate:

SCK Baud Rate = System Clock/(2SPBR)

or

SPBR = System Clock/(2SCK)(Baud Rate Desired)

Giving SPBR a value of zero or one disables the baud rate generator. SCK is disabled and assumes its inactive state value.

The DSCK field in command RAM determines the delay period from chip-select assertion until the leading edge of the serial clock. The DSCKL field in SPCR1 determines the period of delay before the assertion of SCK. The following expression determines the actual delay before SCK:

PCS to SCK Delay = [DSCKL/System Clock Frequency]



where DSCKL equals {1, 2, 3, ..., 127}.

When DSCK equals zero, DSCKL is not used. Instead, the PCS valid-to-SCK transition is one-half the DSCK period.

There are two transfer length options. The user can choose a default value of eight bits, or a programmed value of eight to sixteen bits, inclusive. The programmed value must be written into the BITS field in SPCR0. The BITSE field in command RAM determines whether the default value (BITSE = 0) or the BITS value (BITSE = 1) is used. Table 6–3 shows BITS field encoding.

**Table 6–3. BITS Encoding**

| BITS | Bits per Transfer |
|---|---|
| 0000 | 16 |
| 0001 | Reserved |
| 0010 | Reserved |
| 0011 | Reserved |
| 0100 | Reserved |
| 0101 | Reserved |
| 0110 | Reserved |
| 0111 | Reserved |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | 10 |
| 1011 | 11 |
| 1100 | 12 |
| 1101 | 13 |
| 1110 | 14 |
| 1111 | 15 |

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. There are two transfer delay options. The user can choose to delay a standard period (one µs with a 16.78-MHz system clock) after serial transfer is complete or can specify a delay period. Writing a value to the DTL field in SPCR1 specifies a delay period. The DT bit in command RAM 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:

Delay after Transfer = [(32DTL)/System Clock Frequency]

where DTL equals {1, 2, 3, ..., 255}.

A zero value for DTL causes a delay-after-transfer value of 8192/system clock.



Standard Delay after Transfer = [17/System Clock]

Adequate delay between transfers must be specified for long data streams. The QSPI requires time, approximately one µs with a 16.78-MHz system clock, to load a transmit RAM entry for transfer. Receiving devices need at least one µs of delay between successive transfers. If the system clock is operating at a slower rate, the delay between transfers must be increased proportionately.

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 command RAM is set, PCS pins are continuously driven in 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 DDRQS is driven 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.

### 6.3.5.2 Master Wraparound Mode

Wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address $0 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 SPI service is used, the service routine must clear the SPIF bit to abort the current request. Additional interrupt requests during servicing can be prevented by clearing SPIFIE, but SPIFIE is buffered. Clearing it does not abort a current request.

There are two recommended methods of exiting wraparound mode: clearing the WREN bit or 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.



### 6.3.5.3 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 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, QSM register PQSPAR must be written to assign necessary pins to the QSPI. The pins necessary for slave mode operation are MISO and MOSI, SCK, and PCS0/$\overline{SS}$. 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. Assertion of the active-low slave select signal ($\overline{SS}$) 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/$\overline{SS}$ 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. Unused portions of QSPI RAM can be used by the CPU as general-purpose RAM. Initialize the queue pointers as appropriate.

When SPE is set and MSTR is clear, a low state on the slave select (PCS0/$\overline{SS}$) 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 on which SCK edge to latch incoming data from the MISO pin and to drive outgoing data from the MOSI 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 PCS0/$\overline{SS}$ pin is used only as an input.

The SPBR, DT and DSCK 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 specifies the number of bits to be transferred for all transfers in the queue. When the number of bits designated by BITS 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/$\overline{SS}$ is asserted, unless the CPU writes to NEWQP first.

The QSPI shifts one bit for each pulse of SCK until the slave select input goes high. If $\overline{SS}$ 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 $\overline{SS}$ is asserted. The maximum value that the BITS field can have is 16. If more than 16 bits are transmitted before $\overline{SS}$ 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 $\overline{SS}$ is negated. $\overline{SS}$ does not need to go high between transfers as the QSPI transfers data until reaching the end of the queue, whether $\overline{SS}$ 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.

#### 6.3.5.4 Slave Wraparound Mode

Slave wraparound mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address $0 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.

### 6.3.6 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 bits in the command. 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 ($\overline{SS}$) signal, which initiates slave mode serial transfer. If $\overline{SS}$ is taken low when the QSPI is in master mode, a mode fault occurs.



To set up a chip-select function, 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 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, base state of chip-select signals is zero and chip-select pins are configured for active-high operation.

## 6.4 Serial Communication Interface

The serial communication interface (SCI) communicates with external devices through an asynchronous serial bus. The SCI uses a standard nonreturn to zero (NRZ) transmission format. The SCI is fully compatible with other Motorola SCI systems, such as those in M68HC11 and M68HC05 devices. Figure 6–7 is a block diagram of the SCI transmitter; Figure 6–8 is a block diagram of the SCI receiver.

### 6.4.1 SCI Registers

The SCI programming model includes the QSM global and pin control registers, and four SCI registers. There are two SCI control registers (SCCR0 and SCCR1), one status register (SCSR), and one data register (SCDR). Refer to **APPENDIX D REGISTER SUMMARY** for register bit and field definition.

#### 6.4.1.1 Control Registers

SCCR0 contains the baud rate selection field. Baud rate must be set before the SCI is enabled. The CPU can read and write this register at any time.

SCCR1 contains a number of SCI configuration parameters, including transmitter and receiver enable bits, interrupt enable bits, and operating mode enable bits. The CPU can read and write this register at any time. The SCI can modify the RWU bit under certain circumstances.

Changing the value of SCI control bits during a transfer operation may disrupt operation. Before changing register values, allow the SCI to complete the current transfer, then disable the receiver and transmitter.

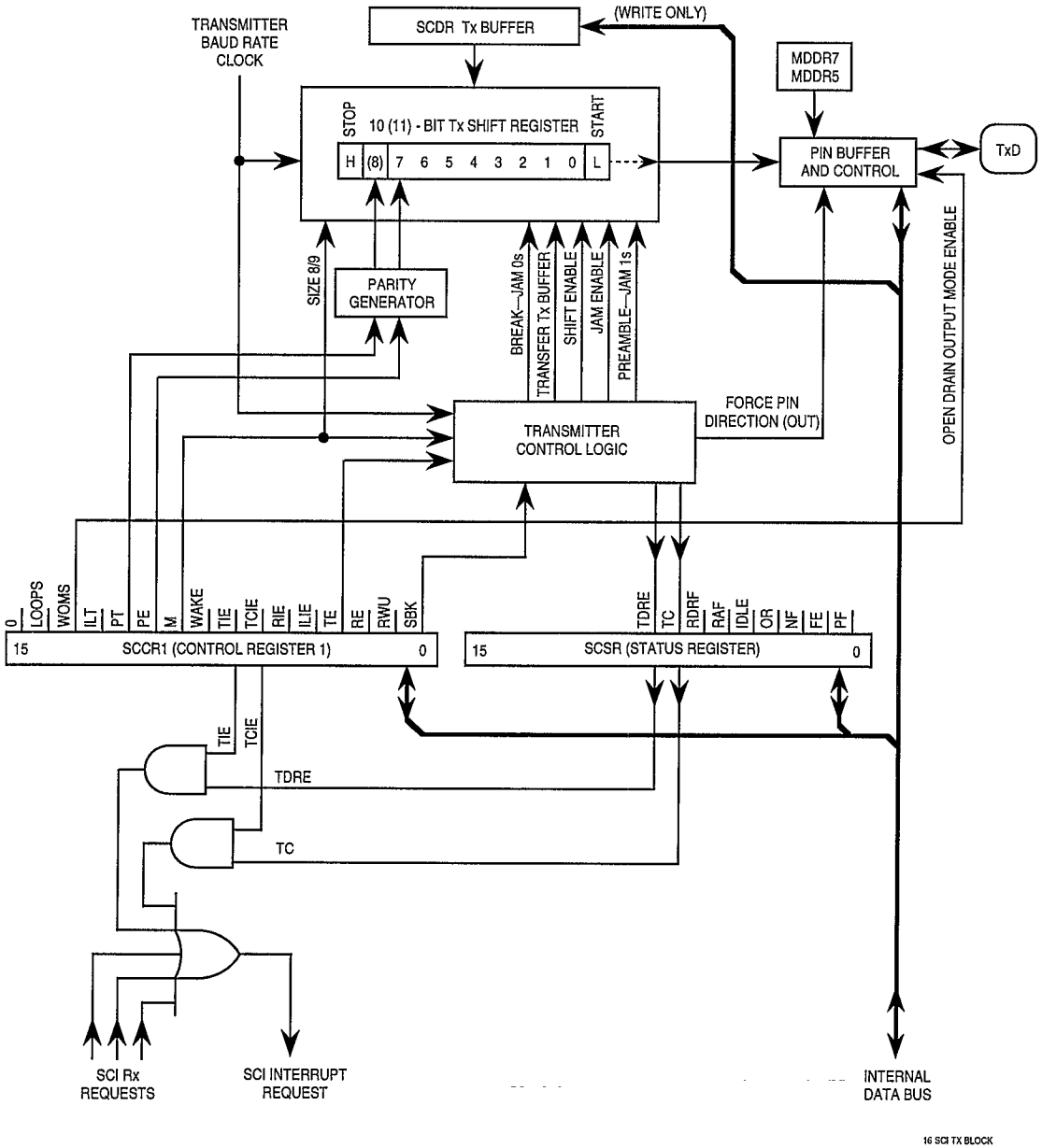



**Figure 6–7. SCI Transmitter Block Diagram**

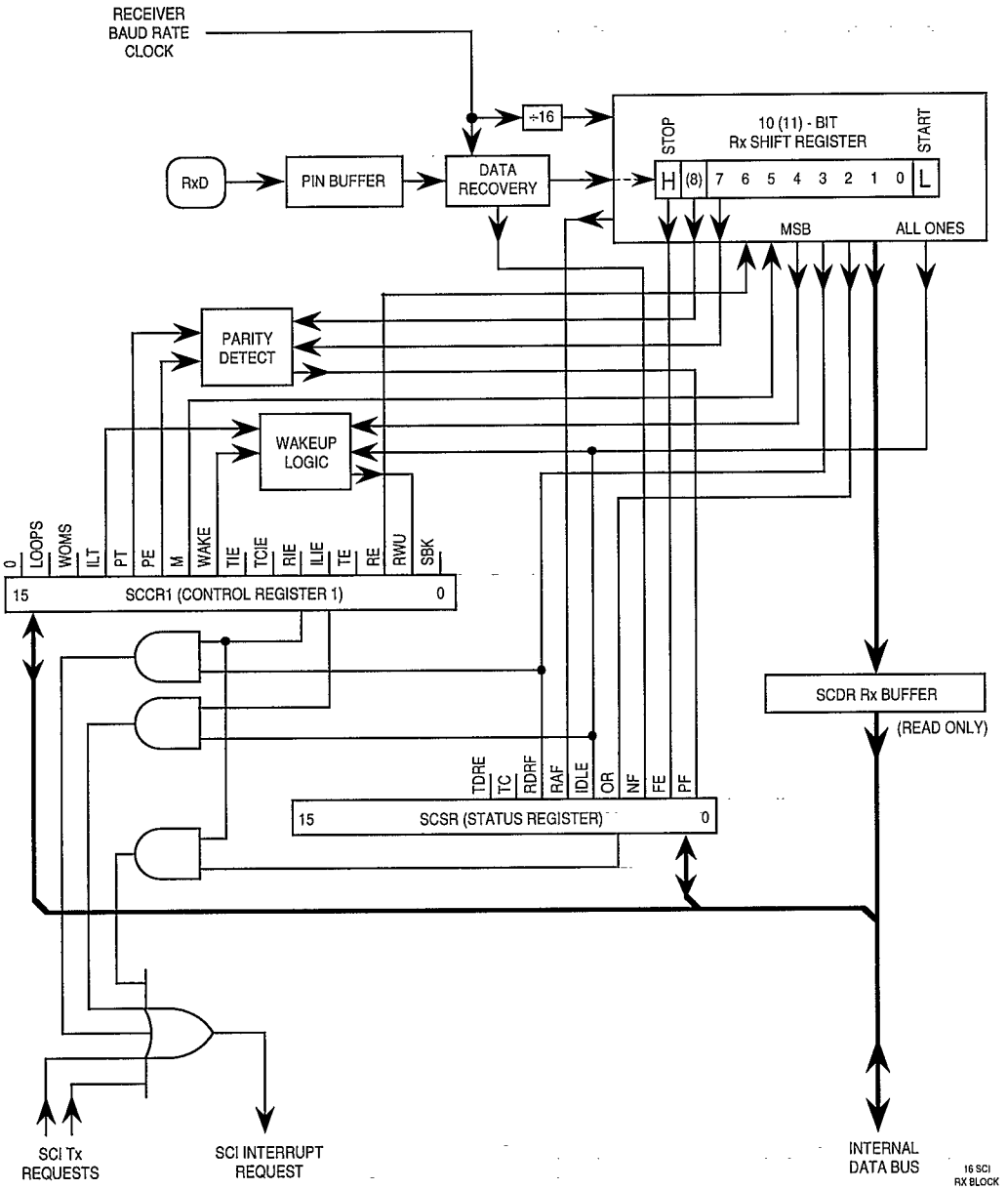



**Figure 6–8. SCI Receiver Block Diagram**

### 6.4.1.2 Status Register

The SCI status register (SCSR) contains flags that show SCI operating conditions. These flags are cleared either by SCI hardware or by a read/write sequence. In general, flags are cleared by reading the SCSR, then reading (receiver status bits) or writing (transmitter status bits) the SCDR. A long-word read can consecutively access both the SCSR and SCDR. This action clears receive status flag bits that were set at the time of the read, but does not clear TDRE or TC flags.

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 written or read the SCDR, the newly set status bit is not cleared. The SCSR must be read again with the bit set, and the SCDR must be written or read before the status bit is cleared.

Reading either byte of the SCSR causes all 16 bits to be accessed, and any status bit already set in either byte is cleared on a subsequent read or write of the SCDR.



### 6.4.1.3 Data Register

The SCDR contains two data registers at the same address. The RDR is a read-only register that contains data received by the SCI serial interface. The data comes into the receive serial shifter and is transferred to the RDR. The TDR is a write-only register that contains data to be transmitted. The data is first written to the TDR, then transferred to the transmit serial shifter, where additional format bits are added before transmission. R[7:0]/T[7:0] contain either the first eight data bits received when the SCDR is read, or the first eight data bits to be transmitted when the SCDR is written. R8/T8 are used when the SCI is configured for 9-bit operation. When it is configured for 8-bit operation, they have no meaning or effect.

## 6.4.2 SCI Pins

Two unidirectional pins, TXD (transmit data) and RXD (receive data), are associated with the SCI. TXD can be used by the SCI or for general-purpose I/O. Function is assigned by the port QS pin assignment register (PQSPAR). The receive data (RXD) pin is dedicated to the SCI. Table 6–4 shows SCI pin function.

**Table 6–4. SCI Pin Function**

| Pin Names | Mnemonics | Mode | Function |
|---|---|---|---|
| Receive Data | RXD | Receiver Disabled<br>Receiver Enabled | Not Used<br>Serial Data Input to SCI |
| Transmit Data | TXD | Transmitter Disabled<br>Transmitter Enabled | General-Purpose I/O<br>Serial Data Output from SCI |

### 6.4.3 SCI Operation

SCI status flags in the SPSR support polled operation, or interrupt-driven operation can be employed by the interrupt enable bits in SCCR1.

#### 6.4.3.1 Definition of Terms

Bit-Time — The time required to transmit or receive one bit of data; 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 (RT) 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.

#### 6.4.3.2 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 serial mode (M) bit in SCI control register one (SCCR1) specifies the number of bits per frame.

The most common ten-bit data frame format for NRZ serial interface consists of one start bit, eight data bits (LSB first), and one stop bit. The most common eleven-bit data frame contains one start bit, eight data bits, a parity or control bit, and one stop bit. Ten-bit and eleven-bit frames are shown in Table 6–5.

**Table 6–5. 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 |

### 6.4.3.3 Baud Clock

The SCI baud clock is programmed by writing a 13-bit value to the baud rate (SCBR) field in SCI control register zero (SCCR0). Baud clock is derived from the MCU system clock by a modulus counter. Writing a value of zero to SCBR disables the baud rate generator. Baud clock rate is calculated as follows:

$$\text{SCI Baud Clock Rate} = \text{System Clock}/(32\text{SCBR})$$

where SCBR 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 clock generator produces a receive time (RT) sampling clock with a frequency 16 times that of the SCI baud clock. The SCI determines the position of bit boundaries from transitions within the received waveform, and adjusts sampling points to the proper positions within the bit period.

### 6.4.3.4 Parity Checking

The parity type (PT) bit in SCCR1 selects either even (PT = 0) or odd (PT = 1) parity. PT affects received and transmitted data. The parity enable (PE) bit in SCCR1 determines whether parity checking is enabled (PE = 1) or disabled (PE = 0). When PE is set, the MSB of the data in a frame 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 parity flag (PF) in the SCI status register (SCSR) 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 6–6 shows possible data and parity formats.

**Table 6–6. 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 |

### 6.4.3.5 Transmitter Operation

The transmitter consists of a serial shifter and a parallel data register (TDR) located in the SCI data register (SCDR). The serial shifter cannot be directly accessed by the CPU. The transmitter is double-buffered, which means that data can be loaded into the TDR while other data is shifted out. The transmitter enable (TE) bit in SCCR1 enables (TE = 1) and disables (TE = 0) the transmitter.

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 wired-OR mode select bit (WOMS) in SCCR1 determines whether TXD is an open-drain (wired-OR) output or a normal CMOS output. An external pull-up resistor on the TXD pin is necessary for wired-OR operation. WOMS controls TXD function whether the pin is used for SCI transmissions (TE = 1) or as a general-purpose I/O pin.

Data to be transmitted is written to TDR, then transferred to the serial shifter. The transmit data register empty (TDRE) flag in SCSR shows the status of TDR. When TDRE = 0, TDR contains data that has not been transferred to the shifter. Writing to TDR again overwrites the data. TDRE is set when the data in TDR is transferred to the shifter. Before new data can be written to TDR, however, the processor must clear TDRE by writing to SCSR. If new data is written to TDR without first clearing TDRE, the data will not be transmitted.

The transmission complete (TC) flag in SCSR 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 SCSR while TC is set, then writing new data to TDR.

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 send break (SBK) bit in SCCR1 is used to insert break frames in a transmission. A nonzero integer number of break frames is 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 are cleared. If SBK is set while a transmission is in progress, that transmission finishes normally before the break begins. To assure 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 the TXD pin reverts to control by 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 (DDRQS), then write a one to PORTQS7. 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 nonlistening receivers in wakeup mode between transmissions, or to signal a retransmission 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 will mark idle. Otherwise, normal transmission of the next sequence will begin.

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 SCCR1. Service routines can load the last byte of data in a sequence into the TDR, then terminate the transmission when a TDRE interrupt occurs.

#### 6.4.3.6 Receiver Operation

The receiver enable (RE) bit in SCCR1 enables (RE = 1) and disables (RE = 0) the transmitter. The receiver contains a receive serial shifter and a parallel receive data register (RDR) located in the SCI data register (SCDR). The serial shifter cannot be directly accessed by the CPU. The receiver is double-buffered, allowing data to be held in RDR 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 (RT) clock is used to control sampling and synchronization. Data is shifted into the receive serial shifter according to the most recent synchronization of the RT 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 *QSM 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 RDR. The receiver data register flag (RDRF) is set when the data is transferred.

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 SCSR are not set until data is transferred from the serial shifter to RDR.

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 the SCSR is set. OR indicates that the CPU needs to service RDR faster. When OR is set, the data in RDR is preserved, but the data in the serial shifter is lost. Because framing, noise, and parity errors are detected while data is in the serial shifter, FE, NF, and PF cannot occur at the same time as OR.

When the CPU reads the SCSR and the SCDR in sequence, it acquires status and data, and also clears the status flags. Reading the SCSR acquires status and arms the clearing mechanism. Reading the SCDR acquires data and clears the SCSR.

When RIE in SCCR1 is set, an interrupt request is generated whenever RDRF is set. Because receiver status flags are set at the same time as RDRF, they do not have separate interrupt enables.

#### 6.4.3.7 Idle-Line Detection

During a typical serial transmission, frames are transmitted isochronously 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 SCCR1.

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 SCCR1 determines

which type of detection is used. When an idle line condition is detected, the IDLE flag in SCSR 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, CPU 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 idle line interrupt enable (ILIE) bit in SCCR1 is set, an interrupt request is generated when the IDLE flag is set. The flag is cleared by reading SCSR and SCDR in sequence. 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.



#### 6.4.3.8 Receiver Wakeup

The receiver wakeup 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 wakeup mode by setting the receiver wakeup (RWU) bit in SCCR1. While RWU is set, receiver status flags and interrupts are disabled. Although the CPU can clear RWU, it is normally cleared by hardware during wakeup.

The WAKE bit in SCCR1 determines which type of wakeup is used. When WAKE = 0, idle-line wakeup is selected. When WAKE = 1, address-mark wakeup is selected. Both types require a software-based device addressing and recognition scheme.

Idle-line wakeup 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 byte is received normally, transferred to register RDR, 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 wakeup 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 wakeup 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 byte is received normally, transferred to register RDR, 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 wakeup 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.

#### 6.4.3.9 Internal Loop

The LOOPS bit in SCCR1 controls a feedback path on the data serial shifter. When LOOPS is set, 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.



### 6.5 QSM Initialization

After reset, the QSM remains in an idle state until initialized. A general sequence guide for initialization follows.

A. Global
   1. Configuration register (QSMCR)
      a. Write an interrupt arbitration priority value into the IARB field.
      b. Clear the FREEZE and/or STOP bits for normal operation.
   2. Interrupt vector and interrupt level registers (QIVR and QILR)
      a. Write QSPI/SCI interrupt vector into QIVR.
      b. Write QSPI (ILSPI) and SCI (ILSCI) interrupt priorities into QILR.
   3. Port data and data direction registers (PORTQS and DDRQS)
      a. Write a data word to PORTQS.
      b. Establish direction of QSM pins used for I/O by writing to DDRQS.
   4. Assign pin functions by writing to the pin assignment register (PQSPAR)

B. Queued Serial Peripheral Interface
   1. Write appropriate values to QSPI command RAM.
   2. QSPI control register zero (SPCR0)
      a. Write a transfer rate value into the BR field.
      b. Determine clock phase (CPHA), and clock polarity (CPOL).
      c. Determine number of bits to be transferred in a serial operation (BIT).
      d. Select master or slave operating mode (MSTR).
      e. Enable or disable wired-OR operation (WOMQ).
   3. QSPI control register one (SPCR1)
      a. Establish a delay following serial transfer by writing to the DTL field.
      b. Establish a delay before serial transfer by writing to the DSCKL field.

4. QSPI control register two (SPCR2)
   a. Write an initial queue pointer value into the NEWQP field.
   b. Write a final queue pointer value into the ENDQP field.
   c. Enable or disable queue wraparound (WREN).
   d. Write wraparound address into the WRTO field.
   e. Enable or disable QSPI flag interrupt (SPIFIE).
5. QSPI control register three (SPCR3)
   a. Enable or disable halt at end of queue (HALT).
   b. Enable or disable halt and mode fault interrupts (HMIE).
   c. Enable or disable loopback (LOOPQ).
6. To enable the QSPI, set the SPE bit in SPCR1.

C. Serial Communication Interface (SCI)
1. SCI control register zero (SCCR0)
   a. Write a transfer rate (baud) value into the BR field.
2. SCI control register one (SCCR1)
   a. Select serial mode (M)
   b. Enable use (PE) and type (PT) of parity check.
   c. Select use (RWU) and type (WAKE) of receiver wakeup.
   d. Enable idle-line detection (ILT) and interrupt (ILIE).
   e. Enable or disable wired-OR operation (WOMS).
   f. Enable or disable break transmission (BK).
3. To receive, set the receiver (RE) and receiver interrupt (RIE) bits in SCCR1.
4. To transmit
   a. Set transmitter (TE) and transmitter interrupt (TIE).
   b. Clear transmitter data register empty (TDRE) and transmit complete (TC) indicators by reading the serial communication interface status register (SCSR).
   c. Write transmit data to the serial communication data register (SCDR).

# SECTION 7
# ANALOG-TO-DIGITAL CONVERTER

This section is an overview of ADC function. Refer to the *ADC Reference Manual* (ADCRM/AD) for a comprehensive discussion of ADC capabilities. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for ADC timing and electrical specifications.

## 7.1 Overview

The analog-to-digital converter module (ADC) is a unipolar, successive-approximation converter with eight modes of operation. It has selectable 8- or 10-bit resolution. Monotonicity is guaranteed in both modes.

A bus interface unit handles communication between the ADC and other microcontroller modules, and supplies IMB timing signals to the ADC. Special operating modes and test functions are controlled by a module configuration register (ADCMCR) and a factory test register (ADCTST).

ADC module conversion functions can be grouped into three basic subsystems: an analog front end, a digital control section, and result storage. Figure 7–1 is a functional block diagram of the ADC module.

In addition to use as multiplexer inputs, the eight analog inputs can be used as a general-purpose digital input port (port ADA), provided signals are within logic level specification. In addition, eight pins serve as a dedicated output port (port ADB). A port data register (PORTAD) is used to access input and output data.

## 7.2 External Connections

The ADC uses 20 pins on the MC68F333 package. Eight pins are analog inputs (which can also be used as digital inputs), eight pins are digital outputs, two pins are analog reference connections, and two pins are analog supply connections.

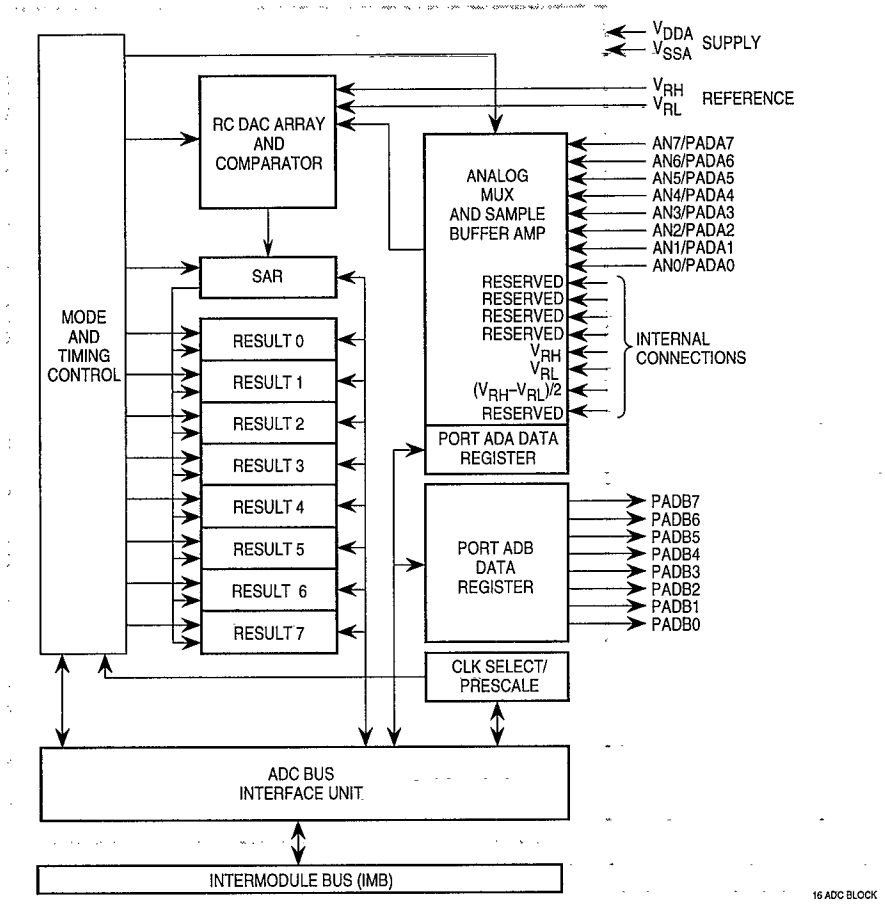



**Figure 7–1. ADC Block Diagram**

### 7.2.1 Analog Input Pins

Each of the eight analog input pins (AN[7:0]) is connected to a multiplexer in the ADC. The multiplexer selects an analog input for conversion to digital data.

Analog input pins can also be read as digital inputs, provided the applied voltage meet $V_{IH}$ and $V_{IL}$ specification. When used as digital inputs, the pins are organized into an 8-bit port (port ADA), and referred to as ADA[7:0]. Digital input data is accessed through a port data register (PORTAD). There is no data direction register because port pins are used only for input.

### 7.2.2 Digital Output Pins

The eight digital output pins (PADB[7:0]) make up port ADB, an output-only port. Data for port ADB is latched in the upper half of the port data register (PORTAD). A read of the upper byte of PORTAD returns the digital value in the port ADB output register.

### 7.2.3 Analog Reference Pins

Separate high ($V_{RH}$) and low ($V_{RL}$) analog reference voltages are connected to the analog reference pins. The pins permit connection of regulated and filtered supplies that allow the ADC to achieve its highest degree of accuracy.

### 7.2.4 Analog Supply Pins

Pins $V_{DDA}$ and $V_{SSA}$ supply power to analog circuitry associated with the RC DAC. Other circuitry in the ADC is powered from the digital power bus (pins $V_{DDI}$ and $V_{SSI}$). Dedicated analog power supplies are necessary to isolate sensitive ADC circuitry from noise on the digital power bus.

## 7.3 Programmer's Model



The ADC module is mapped into 32 words of address space. Five words are control/status registers, one word is digital port data, and 24 words provide access to the results of AD conversion (eight addresses for each type of converted data). Two words are reserved for expansion. See **APPENDIX D REGISTER SUMMARY** for detailed information concerning the ADC address map and register structure.

The ADC module base address is determined by the value of the MM bit in the single-chip integration module configuration register (SCIMCR). The base address is $YFF700, where Y = $7 or $F, depending on the value of MM.

Internally, the ADC has both a differential data bus and a buffered IMB data bus. Registers not directly associated with conversion functions, such as the module configuration register, the module test register, and the port data register, reside on the buffered bus, while conversion registers and result registers reside on the differential bus.

Registers that reside on the buffered bus are updated immediately when written. However, writes to ADC control registers abort any conversion in progress.

## 7.4 ADC Bus Interface Unit

The ADC is designed to act as a slave device on the intermodule bus. The bus interface unit (ABIU) provides IMB bus cycle termination and synchronizes internal ADC signals with IMB signals. The ABIU also manages data bus routing to accommodate the three conversion data formats, and controls the interface to the module differential data bus.

## 7.5 Special Operating Modes

Low-power stop mode and freeze mode are ADC operating modes associated with assertion of IMB signals by other microcontroller modules or by external sources. These modes are controlled by the values of bits in the ADC module configuration register (ADCMCR).

### 7.5.1 Low-Power Stop Mode

When the STOP bit in ADCMCR is set, the IMB clock signal to the ADC is disabled. This places the module in an idle state, and power consumption is minimized. The ABIU does not shut down and ADC registers are still accessible. If a conversion is in progress when STOP is set, it is aborted.



STOP is set during system reset, and must be cleared before the ADC can be used. Because analog circuit bias currents are turned off during low-power stop, the ADC requires recovery time after STOP is cleared.

Execution of the CPU32 LPSTOP command places the entire modular microcontroller in low-power stop mode. Refer to **SECTION 3 SINGLE-CHIP INTEGRATION MODULE** and **SECTION 4 CENTRAL PROCESSING UNIT** for more information regarding low-power stop operation.

### 7.5.2 Freeze Mode

When the CPU32 in the modular microcontroller enters background debugging mode, the FREEZE signal is asserted. The ADC can respond to internal assertion of FREEZE in one of three different ways. It can ignore FREEZE assertion, finish the current conversion and then freeze, or freeze immediately.

The type of response is determined by the value of the FRZ[1:0] field in the module configuration register (refer to Table 7–1).

**Table 7–1.**
**FRZ Field Selection**

| FRZ[1:0] | Response |
|---|---|
| 00 | Ignore FREEZE |
| 01 | Reserved |
| 10 | Finish conversion, then freeze |
| 11 | Freeze immediately |

When the ADC freezes, the ADC clock stops and all sequential activity ceases. Contents of control and status registers remain valid while frozen. When the FREEZE signal is negated, ADC activity resumes.

If the ADC freezes during a conversion, activity resumes with the next step in the conversion sequence. However, capacitors in the analog conversion circuitry discharge while the ADC is frozen. As a result, the conversion will be inaccurate.

Refer to **SECTION 4 CENTRAL PROCESSING UNIT** for more information on background debugging mode.



## 7.6 Analog Subsystem

The analog subsystem consists of a multiplexer, sample capacitors, a buffer amplifier, an RC DAC array, and a high-gain comparator. Comparator output sequences the successive approximation register (SAR). The interface between the comparator and the SAR is the boundary between ADC analog and digital subsystems.

### 7.6.1 Multiplexer

The multiplexer selects one of 16 sources for conversion. Eight sources are internal and eight are external. Multiplexer operation is controlled by channel selection field CD:CA in register ADCTL1 (refer to Table 7–2). The multiplexer contains positive and negative stress protection circuitry. This circuitry prevents voltages on other input channels from affecting the current conversion.

**Table 7–2. Multiplexer Channels**

| CD:CA Value | Input Source |
|---|---|
| %0000 | AN0 |
| %0001 | AN1 |
| %0010 | AN2 |
| %0011 | AN3 |
| %0100 | AN4 |
| %0101 | AN5 |
| %0110 | AN6 |
| %0111 | AN7 |
| %1000 | Reserved |
| %1001 | Reserved |
| %1010 | Reserved |
| %1011 | Reserved |
| %1100 | $V_{RH}$ |
| %1101 | $V_{RL}$ |
| %1110 | $(V_{RH} - V_{RL}) / 2$ |
| %1111 | Test/Reserved |

7

### 7.6.2 Sample Capacitors and Buffer Amplifier

Each of the input channels has its own sample capacitor. All channels share a single buffer amplifier. After a channel is selected, for the first two ADC clock cycles of a sampling period, multiplexer output is connected to the input of the sample buffer amplifier through the sample capacitor. The sample amplifier buffers the input channel from the relatively large capacitance of the RC DAC array.

During the second two clock cycles of a sampling period, the sample capacitor is disconnected from the multiplexer, and the sample buffer amplifier charges the RC DAC array with the value stored in the sample capacitor.

During the third portion of a sampling period, both sample capacitor and buffer amplifier are bypassed, and multiplexer input charges the DAC array directly. The length of this third portion of a sampling period is determined by the value of the STS field in ADCTL0.

### 7.6.3 RC DAC Array

The RC DAC array consists of binary-weighted capacitors and a resistor-divider chain. The array performs two functions: it acts as a sample hold circuit during conversion, and it provides each successive digital-to-analog comparison voltage to the comparator. Conversion begins with MSB comparison and ends with LSB comparison. Array switching is controlled by the digital subsystem.

### 7.6.4 Comparator

The comparator indicates whether each approximation output from the RC DAC array during resolution is higher or lower than the sampled input voltage. Comparator output is fed to the digital control logic, which sets or clears each bit in the successive approximation register in sequence, MSB first.

## 7.7 Digital Control Subsystem

The digital control subsystem includes control and status registers, clock and prescaler control logic, channel and reference select logic, conversion sequence control logic, and the successive approximation register.

The subsystem controls the multiplexer and the output of the RC array during sample and conversion periods, stores the results of comparison in the successive approximation register, then transfers results to the result registers.

### 7.7.1 Control and Status Registers

There are two control registers (ADCTL0, ADCTL1) and one status register (ADSTAT). ADCTL0 controls conversion resolution, sample time, and clock and prescaler values. ADCTL1 controls analog input selection and conversion mode. A write to ADCTL0 aborts the current conversion sequence and halts the ADC. Conversion must be restarted by writing to ADCTL1. A write to ADCTL1 aborts the current conversion sequence and starts a new sequence with parameters altered by the write. ADSTAT shows conversion sequence status, conversion channel status, and conversion completion status.

The following paragraphs are a general discussion of control function. **APPENDIX D REGISTER SUMMARY** shows the ADC address map and discusses register bits and fields.

### 7.7.2 Clock and Prescaler Control

The ADC clock is derived from the system clock by a programmable prescaler. ADC clock period is determined by the value of the PRS field in ADCTL0.

The prescaler has two stages. The first stage is a 5-bit modulus counter. It divides the system clock by any value from 2 to 32 (PRS[4:0] = %00001 to %11111). The second stage is a divide-by-two circuit. Table 7–3 shows prescaler output values.

**Table 7–3. Prescaler Output**

| PRS[4:0] | ADC Clock |
|---|---|
| %00000 | Reserved |
| %00001 | Sys Clk/4 |
| %00010 | Sys Clk/6 |
| ... | ... |
| %11101 | Sys Clk/60 |
| %11110 | Sys Clk/62 |
| %11111 | Sys Clk/64 |

ADC clock speed must be between 0.5 MHz and 2.1 MHz. The reset value of the PRS field is %00011, which divides a nominal 16.78-MHz system clock by eight, yielding maximum ADC clock frequency. There are a minimum of four IMB clock cycles for each ADC clock cycle.

### 7.7.3 Sample Time



The first two parts of all sample periods require four ADC clock cycles. During the third part of a sample period, the selected channel is connected directly to the RC DAC array for a specified number of clock cycles. The value of the STS field in ADCTL0 determines the number of cycles (refer to Table 7–4). The number of clock cycles required for a sample period is the value specified by STS plus four. Sample time is determined by PRS value.

**Table 7–4. STS Field Selection**

| STS[1:0] | Sample Time |
|---|---|
| 00 | 2 A/D Clock Periods |
| 01 | 4 A/D Clock Periods |
| 10 | 8 A/D Clock Periods |
| 11 | 16 A/D Clock Periods |

### 7.7.4 Resolution

ADC resolution can be either eight or ten bits. Resolution is determined by the state of the RES10 bit in ADCTL0. Both 8-bit and 10-bit conversion results are automatically aligned in the result registers.

### 7.7.5 Conversion Control Logic

Analog-to-digital conversions are performed in sequences. Sequences are initiated by any write to ADCTL1. If a conversion sequence is already in progress, a write to either control register will abort it and reset the SCF and CCF flags in the A/D status register. There are eight conversion modes.

Conversion mode is determined by ADCTL1 control bits. Each conversion mode affects the bits in status register ADSTAT differently. Result storage differs from mode to mode.

### 7.7.5.1 Conversion Parameters

The following conversion parameters are controlled by bits in ADCTL1.

Conversion channel — the value of the channel selection field (CD:CA) in ADCTL1 determines which multiplexer inputs are used in a conversion sequence. There are 16 possible inputs. Eight inputs are external pins (AN[7:0]), and eight are internal.

Length of sequence — A conversion sequence consists of either four or eight conversions. The number of conversions in a sequence is determined by the state of the S8CM bit in ADCTL1.

Single or continuous conversion — Conversion can be limited to a single sequence or a sequence can be performed continuously. The state of the SCAN bit in ADCTL1 determines whether single or continuous conversion is performed.

Single or multiple channel conversion — Conversion sequence(s) can be run on a single channel or on a block of four or eight channels. Channel conversion is controlled by the state of the MULT bit in ADCTL1.

### 7.7.5.2 Conversion Modes

Conversion modes are defined by the state of the SCAN, MULT, and S8CM bits in ADCTL1. Table 7–5 shows mode numbering.

**Table 7–5.**
**ADC Conversion Modes**

| SCAN | MULT | S8CM | Mode |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 5 |
| 1 | 1 | 0 | 6 |
| 1 | 1 | 1 | 7 |

Mode 0 — A single four-conversion sequence is performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the conversion sequence is complete.

Mode 1 — A single eight-conversion sequence is performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the conversion sequence is complete.

Mode 2 — A single conversion is performed on each of four sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the last conversion is complete.



Mode 3 — A single conversion is performed on each of eight sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the last conversion is complete.

Mode 4 — Continuous four-conversion sequences are performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). Previous results are overwritten when a sequence repeats. The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first four-conversion sequence is complete.

Mode 5 — Continuous eight-conversion sequences are performed on a single input channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). Previous results are overwritten when a sequence repeats. The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first eight-conversion sequence is complete.

Mode 6 — Continuous conversions are performed on each of four sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT3). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first four-conversion sequence is complete.

Mode 7 — Continuous conversions are performed on each of eight sequential input channels, starting with the channel specified by the value in CD:CA. Each result is stored in a separate result register (RSLT0 to RSLT7). The appropriate CCF bit in ADSTAT is set as each register is filled. The SCF bit in ADSTAT is set when the first eight-conversion sequence is complete.

Table 7–6 summarizes ADC operation when MULT is cleared (single channel modes). Table 7–7 is a summary of ADC operation when MULT is set (multi-channel modes). Number of conversions per channel is determined by SCAN. Channel numbers are given in order of conversion.

## Table 7–6. Single-Channel Conversions

| S8CM | CD | CC | CB | CA | Input | Result Register |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | AN0 | RSLT[0:3] |
| 0 | 0 | 0 | 0 | 1 | AN1 | RSLT[0:3] |
| 0 | 0 | 0 | 1 | 0 | AN2 | RSLT[0:3] |
| 0 | 0 | 0 | 1 | 1 | AN3 | RSLT[0:3] |
| 0 | 0 | 1 | 0 | 0 | AN4 | RSLT[0:3] |
| 0 | 0 | 1 | 0 | 1 | AN5 | RSLT[0:3] |
| 0 | 0 | 1 | 1 | 0 | AN6 | RSLT[0:3] |
| 0 | 0 | 1 | 1 | 1 | AN7 | RSLT[0:3] |
| 0 | 1 | 0 | 0 | 0 | Reserved | RSLT[0:3] |
| 0 | 1 | 0 | 0 | 1 | Reserved | RSLT[0:3] |
| 0 | 1 | 0 | 1 | 0 | Reserved | RSLT[0:3] |
| 0 | 1 | 0 | 1 | 1 | Reserved | RSLT[0:3] |
| 0 | 1 | 1 | 0 | 0 | $V_{RH}$ | RSLT[0:3] |
| 0 | 1 | 1 | 0 | 1 | $V_{RL}$ | RSLT[0:3] |
| 0 | 1 | 1 | 1 | 0 | $(V_{RH} - V_{RL}) / 2$ | RSLT[0:3] |
| 0 | 1 | 1 | 1 | 1 | Test/Reserved | RSLT[0:3] |
| 1 | 0 | 0 | 0 | 0 | AN0 | RSLT[0:7] |
| 1 | 0 | 0 | 0 | 1 | AN1 | RSLT[0:7] |
| 1 | 0 | 0 | 1 | 0 | AN2 | RSLT[0:7] |
| 1 | 0 | 0 | 1 | 1 | AN3 | RSLT[0:7] |
| 1 | 0 | 1 | 0 | 0 | AN4 | RSLT[0:7] |
| 1 | 0 | 1 | 0 | 1 | AN5 | RSLT[0:7] |
| 1 | 0 | 1 | 1 | 0 | AN6 | RSLT[0:7] |
| 1 | 0 | 1 | 1 | 1 | AN7 | RSLT[0:7] |
| 1 | 1 | 0 | 0 | 0 | Reserved | RSLT[0:7] |
| 1 | 1 | 0 | 0 | 1 | Reserved | RSLT[0:7] |
| 1 | 1 | 0 | 1 | 0 | Reserved | RSLT[0:7] |
| 1 | 1 | 0 | 1 | 1 | Reserved | RSLT[0:7] |
| 1 | 1 | 1 | 0 | 0 | $V_{RH}$ | RSLT[0:7] |
| 1 | 1 | 1 | 0 | 1 | $V_{RL}$ | RSLT[0:7] |
| 1 | 1 | 1 | 1 | 0 | $(V_{RH} - V_{RL}) / 2$ | RSLT[0:7] |
| 1 | 1 | 1 | 1 | 1 | Test/Reserved | RSLT[0:7] |

## Table 7–7. Multiple-Channel Conversions

| S8CM | CD | CC | CB | CA | Input | Result Register |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | AN0 | RSLT0 |
| | | | | | AN1 | RSLT1 |
| | | | | | AN2 | RSLT2 |
| | | | | | AN3 | RSLT3 |
| 0 | 0 | 1 | X | X | AN4 | RSLT0 |
| | | | | | AN5 | RSLT1 |
| | | | | | AN6 | RSLT2 |
| | | | | | AN7 | RSLT3 |
| 0 | 1 | 0 | X | X | Reserved | RSLT0 |
| | | | | | Reserved | RSLT1 |
| | | | | | Reserved | RSLT2 |
| | | | | | Reserved | RSLT3 |
| 0 | 1 | 1 | X | X | $V_{RH}$ | RSLT0 |
| | | | | | $V_{RL}$ | RSLT1 |
| | | | | | $(V_{RH} - V_{RL}) / 2$ | RSLT2 |
| | | | | | Test/Reserved | RSLT3 |
| 1 | 0 | X | X | X | AN0 | RSLT0 |
| | | | | | AN1 | RSLT1 |
| | | | | | AN2 | RSLT2 |
| | | | | | AN3 | RSLT3 |
| | | | | | AN4 | RSLT4 |
| | | | | | AN5 | RSLT5 |
| | | | | | AN6 | RSLT6 |
| | | | | | AN7 | RSLT7 |
| 1 | 1 | X | X | X | Reserved | RSLT0 |
| | | | | | Reserved | RSLT1 |
| | | | | | Reserved | RSLT2 |
| | | | | | Reserved | RSLT3 |
| | | | | | $V_{RH}$ | RSLT4 |
| | | | | | $V_{RL}$ | RSLT5 |
| | | | | | $(V_{RH} - V_{RL}) / 2$ | RSLT6 |
| | | | | | Test/Reserved | RSLT7 |

### 7.7.6 Conversion Timing

Total conversion time is made up of initial sample time, transfer time, final sample time, and resolution time. Initial sample time is the time during which a selected input channel is connected to the sample buffer amplifier through a sample capacitor. During transfer time, the sample capacitor is disconnected from the multiplexer, and the RC DAC array is driven by the sample buffer amp. During final sampling time, the sample capacitor and amplifier are bypassed, and the multiplexer input charges the RC DAC array directly. During resolution time, the voltage in the RC DAC array is converted to a digital value, and the value is stored in the SAR.

Initial sample time and transfer time are fixed at two ADC clock cycles each. Final sample time can be 2, 4, 8, or 16 ADC clock cycles, depending on the value of the STS field in ADCTL0. Resolution time is ten cycles for 8-bit conversion and twelve cycles for 10-bit conversion.

Transfer and resolution require a minimum of 16 ADC clocks (8 µs with a 2.1-MHz ADC clock) for 8-bit resolution or 18 ADC clocks (9 µs with a 2.1-MHz ADC clock) for 10-bit resolution. If maximum final sample time (16 ADC clocks) is used, total conversion time is 15 µs for an 8-bit conversion or 16 µs for a 10-bit conversion (with a 2.1-MHz ADC clock).

Figures 7–2 and 7–3 illustrate the timing for 8- and 10-bit conversions, respectively. These diagrams assume a final sampling period of two ADC clocks.

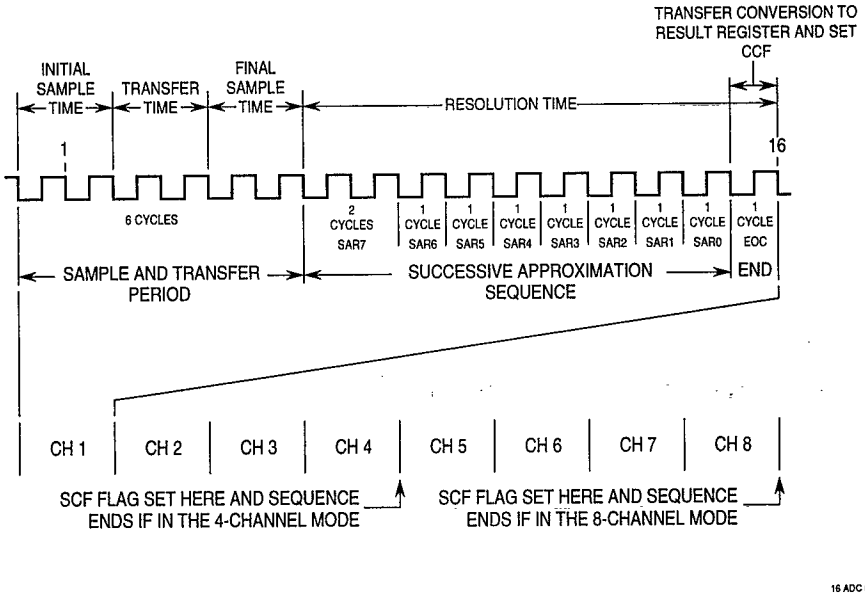



**Figure 7–2. 8-Bit Conversion Timing**

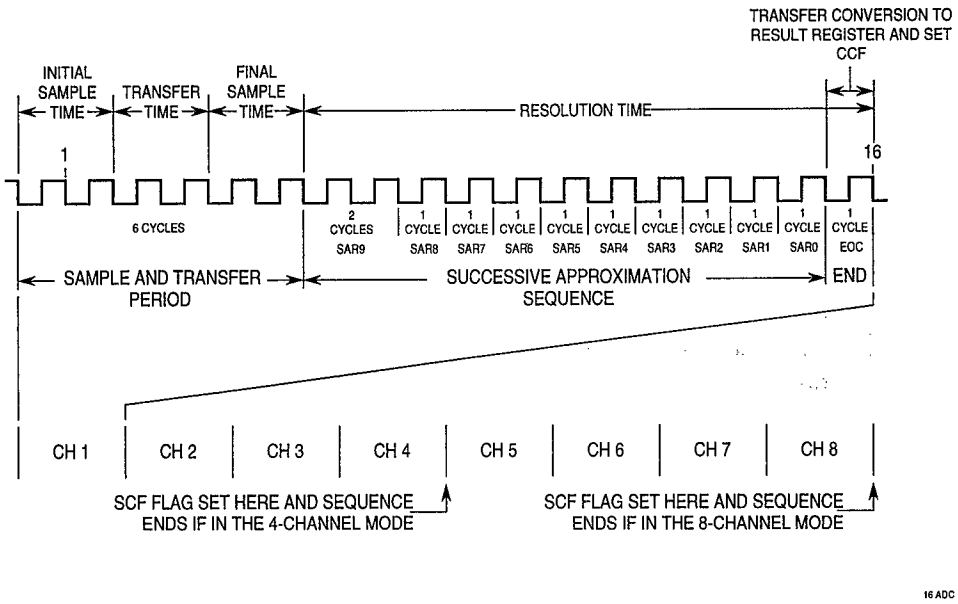



**Figure 7–3. 10-Bit Conversion Timing**

### 7.7.7 Successive Approximation Register

The successive approximation register accumulates the result of each conversion one bit at a time, starting with the most significant bit.

At the start of the resolution period, the MSB of the SAR is set, and all less significant bits are cleared. Depending on the result of the first comparison, the MSB is either left set or cleared. Each successive bit is set or left cleared in descending order until all eight or ten bits have been resolved.

When conversion is complete, the content of the SAR is transferred to the appropriate result register. Refer to **APPENDIX D REGISTER SUMMARY** for register mapping and configuration.

### 7.7.8 Result Registers

Result registers are used to store data after conversion is complete. The registers can be accessed from the IMB under ABIU control. Each register can be read from three different addresses in the ADC memory map. The format of the result data depends on the address from which it is read.



Unsigned Right-Justified Format — Conversion result is unsigned right-justified data. Bits [9:0] are used for 10-bit resolution; bits [7:0] are used for 8-bit conversion (bits [9:8] are zero). Bits [15:10] always return zero when read.

Signed Left-Justified Format — Conversion result is signed left-justified data. Bits [15:6] are used for 10-bit resolution, bits [15:8] are used for 8-bit conversion (bits [7:6] are zero). Although the ADC is unipolar, it is assumed that the zero point is $(V_{RH} - V_{RL}) / 2$ when this format is used. The value read from the register is an offset two's-complement number; for positive input, bit 15 = 0, for negative input, bit 15 = 1. Bits [5:0] always return zero when read.

Unsigned Left-Justified Format — Conversion result is unsigned left-justified data. Bits [15:6] are used for 10-bit resolution, bits [15:8] are used for 8-bit conversion (bits [7:6] are zero). Bits [5:0] always return zero when read.

Refer to **APPENDIX D REGISTER SUMMARY** for register mapping and configuration.

# SECTION 8
# FLASH EEPROM

The MC68F333 contains two flash electrically-erasable programmable read-only memory modules: a 16-Kbyte module and a 48-Kbyte module. These modules serve as nonvolatile, fast-access ROM-emulation memory. The module can be used for program code that must either execute at high speed or is frequently executed, such as operating system kernels and standard subroutines, or it can be used for static data that is read frequently. The module can also be configured to provide bootstrap vectors for system reset.

## 8.1 Overview

Each flash EEPROM module (FLASH) consists of a control-register block that occupies a fixed position in MCU address space and a relocatable EEPROM array. The 16-Kbyte array can be placed on any 16-Kbyte boundary, and the 48-Kbyte array can be placed in the lower 48 Kbytes of any 64-Kbyte block. Each array can be configured to reside in both program and data space or in program space alone.



The EEPROM arrays can be read as either bytes, words, or long-words. The modules respond to back-to-back IMB accesses, providing two-bus-cycle (four system clock) access for aligned long words. Each module can also be programmed to insert up to three wait states per access, to accommodate migration from slower external development memory without re-timing the system.

Both the arrays and individual control bits are programmable and erasable under software control. Program/erase voltage must be supplied through external $V_{FP}$ pins. Programming is by byte or aligned word only. The modules support bulk erasure only. Hardware interlocks protect stored data from corruption if the program/erase voltage to either FLASH array is enabled accidentally.

Holding pin DATA14 low during reset disables the 48-Kbyte FLASH module and places it in stop mode. Holding DATA15 low during reset disables the 16-Kbyte FLASH module.

## 8.2 Signal Descriptions

The $V_{FPE48K}$ and $V_{FPE16K}$ program/erase voltage pins provide program/erase power to the FLASH modules. During operation, a voltage of at least $V_{DD}$ – 0.35 V must be applied to the $V_{FPE}$ pins at all times or damage to the FLASH module can occur. $V_{FPE}$ must not rise to programming level while $V_{DD}$ is below specified minimum value, and must not fall below minimum specified value while $V_{DD}$ is applied. FLASH modules can be damaged by power-on and power off $V_{FPE}$ transients. **8.9.5 FLASH Program/Erase Voltage Signal Conditioning** contains additional information concerning power supply conditioning. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for specifications.

## 8.3 Flash EEPROM Control Block

Each FLASH control block contains five registers: the flash EEPROM module configuration register (FEEMCR), the flash EEPROM test register (FEETST), the flash EEPROM array base address registers (FEEBAH and FEEBAL), and the flash EEPROM control register (FEECTL). Four additional words in the control block can contain bootstrap information when the FLASH is used as bootstrap memory. Control registers are located in supervisor data space.



Each register in the control block has an associated shadow register that is physically located in a spare FLASH row. During reset, fields within the registers are loaded with default reset information from the shadow registers. Shadow registers are programmed or erased in the same manner as a location in the FLASH array, using the address of the corresponding control registers. While a shadow register is being programmed, the data is not written to the corresponding control register — the new data is not copied into the control register until the next reset. The contents of shadow registers are erased whenever the FLASH array is erased.

Configuration information is specified and programmed independently of the FLASH array. After reset, registers in the control block that contain writable bits can be modified. Writes to these registers do not affect the associated shadow register. Certain registers can be written only when the LOCK bit in the FEEMCR is disabled or when the STOP bit in the FEEMCR is set. These restrictions are noted in the individual register descriptions in **APPENDIX D REGISTER SUMMARY**.

References to FLASH control registers in this section apply to registers in either control block, where not otherwise noted. References to FEEMCR, for example, apply to both FEE1MCR (in the 16-Kbyte module) and FEE2MCR (in the 48-Kbyte module).

The module mapping (MM) bit in the SCIM configuration register (SCIMCR) defines the MSB (ADDR23) of the IMB address for each module in the MC68F333. Refer to **APPENDIX D REGISTER SUMMARY** for address maps and register bit/field definitions for the FLASH modules.

## 8.4 Flash EEPROM Array

The base address registers specify the base address of the FLASH array. A default reset base address can be programmed into the base address shadow register.

Avoid using a base address value that causes the array to overlap control registers. If a portion of the array overlaps the FLASH register block, the registers remain accessible, but accesses to that portion of the array are ignored. However, if the array overlaps the control block of another module, those registers may become inaccessible.

## 8.5 Module Configuration

The FLASH module configuration registers (FEE1MCR and FEE2MCR) control configuration of the 16-Kbyte and 48-Kbyte FLASH modules, respectively. This register can be written only when the control block is not write-locked (when LOCK = 0). All active fields and bits take values from the associated shadow register during reset.



### 8.5.1 Low-Power Stop Control

Setting the STOP bit in the FEEMCR places the module in low-power stop mode. STOP can be set by the processor or by reset if the STOP shadow bit is set. The EEPROM array is inaccessible during low-power stop. The array can be re-enabled by clearing STOP. If STOP is set during programming or erasing, program/erase voltage is automatically turned off. However, the enable programming/erase bit (ENPE) in the FEECTL remains set. If STOP is cleared, program/erase voltage is automatically returned unless ENPE is cleared.

### 8.5.2 Response to FREEZE Assertion

The FRZ bit in the FEEMCR determines the response of the FLASH module to assertion of the FREEZE signal by the CPU. When FRZ = 0, the program/erase voltage is disabled while FREEZE is asserted. When FRZ = 1, the ENPE bit in the FEECTL can turn on the program/erase voltage while FREEZE is asserted.

### 8.5.3 System Boot Configuration

On reset, the BOOT bit takes on the default value stored in the shadow register. If BOOT = 0 and STOP = 0, the module responds to program space accesses to IMB addresses $000000 to $000006 following reset, and the contents of FEEBS[3:0] are used as bootstrap vectors. After address $000006 is read, the module responds normally to control block or array addresses only.

### 8.5.4 Write-Lock Protection

When the LOCK bit in FEEMCR is set, writes to locked registers in the control block have no effect. Once set, LOCK cannot be cleared until reset occurs. The default reset state of LOCK is determined by the user. If the default state is zero, LOCK can be set once to protect the registers after initialization. When a default reset state of zero is used, the initialization routine should set LOCK to prevent inadvertent reconfiguration of the FLASH module.

### 8.5.5 Array Space

The ASPC field in the FEEMCR assigns the FLASH array to supervisor or user data space, and to program space only or program and data space. The four possible encodings for ASPC are summarized in Table 8–1.



**Table 8–1. Array Space Encodings**

| ASPC[1:0] | Type of Access |
|---|---|
| 00 | Unrestricted program and data space |
| 01 | Unrestricted program space |
| 10 | Supervisor program and data space |
| 11 | Supervisor program space |

The field can be written only if LOCK = 0 and STOP = 1. During reset, ASPC takes on the default value programmed into the associated shadow register.

### 8.5.6 Wait States

The WAIT field specifies the number of wait states inserted during accesses to the FLASH module. A wait state has the duration of one system clock cycle. WAIT affects both control block and array access, and can be written only if LOCK = 0 and STOP = 1. Table 8–2 shows wait state encodings and corresponding clock cycles per transfer.

**Table 8–2. Wait States**

| WAIT[1:0] | Wait States | Clocks/Transfer |
|---|---|---|
| 00 | 0 | 3 |
| 01 | 1 | 4 |
| 10 | 2 | 5 |
| 11 | –1 | 2 |

The value of the WAIT field is compatible with the lower two bits of the $\overline{\text{DSACK}}$ field in the SCIM chip-select option registers. An encoding of %11 in the WAIT field corresponds to an encoding for fast termination.

## 8.6 Base Address Registers

Each FLASH base address high register (FEE1BAH, FEE2BAH) contains the 16 high-order bits of the corresponding FLASH array base address. During reset, FEEBAH takes on the default value programmed in the associated shadow register.

The FLASH base address low registers (FEE1BAL, FEE2BAL) contain the low-order bits of the corresponding EEPROM arrays as needed. FEE1BAL contains the two low-order bits of the base address for the 16-Kbyte module. All bits of FEE2BAL (in the 48-Kbyte module) are always zero.

After reset, if LOCK = 0 and STOP = 1, software can write to FEEBAH and FEEBAL to relocate the FLASH array.

## 8.7 Flash EEPROM Control Register

Each FLASH control register (FEE1CTL, FEE2CTL) contains the bits needed to control the programming and erasure of the corresponding FLASH array. FEECTL is accessible in supervisor mode only. For additional information on programming this register when programming or erasing the EEPROM array refer to **8.9.4 Program/Erase Operation**.

### 8.7.1 Program/Erase Verification

The verify program/erase bit (VFPE) invokes a special program-verify circuit. During programming sequences (ERAS = 0), VFPE is used in conjunction with the LAT bit to determine when programming of a location is complete. If VFPE and LAT are both set, a bit-wise exclusive-OR of the latched data with the data in the location being programmed occurs when any valid FLASH location is read. If the location is completely programmed, a value of zero is read. Any other value indicates that the location is not fully programmed. When VFPE is cleared, normal reads of valid FLASH locations occur.

### 8.7.2 Erasure Control

The erase control bit (ERAS) in the FEECTL configures the array for either programming or erasure. Setting ERAS causes all locations in the array and all FLASH control bits in the control block to be configured for erasure at the same time.

When the LAT bit is set, ERAS also determines whether a read returns the data in the addressed location (ERAS = 1) or the address itself (ERAS = 0).

ERAS cannot be changed while program/erase voltage is on (ENPE = 1).

### 8.7.3 Latch Control

The latch control bit (LAT) in the FEECTL configures the EEPROM array for normal reads or for programming. When LAT is cleared, the FLASH address and data buses are connected to the IMB address and data buses and the FLASH is configured for normal reads. When LAT is set, the FLASH address and data buses are connected to parallel internal latches and the FLASH array is configured for programming or erasing.

Once LAT is set, the next write to a valid FLASH module address causes the programming circuitry to latch both address and data. Unless control register shadow bits are to be programmed, the write must be to an array address.

8

LAT cannot be changed while program/erase voltage is on (ENPE = 1).

### 8.7.4 Enabling Programming and Erasure

Setting the enable programming/erasure (ENPE) bit in the FEECTL applies program/erase voltage to the FLASH array. ENPE can be set only after LAT has been set and a write to the data and address latches has occurred. ENPE remains cleared if these conditions are not met. While ENPE is set, the LAT, VFPE, and ERAS bits cannot be changed, and attempts to read a flash EEPROM array location are ignored.

## 8.8 Flash EEPROM Bootstrap Words

The FLASH bootstrap words (FEEBS[3:0]) can be used as system bootstrap vectors. When the BOOT bit in FEEMCR = 1 during reset, the FLASH module responds to program space accesses of IMB addresses $000000 to $000006 after reset. When BOOT = 0, the FLASH module responds only to normal array and register accesses. FEEBS[3:0] can be read at any time, but the values in the words can only be changed by programming the appropriate location. Refer to **8.9.2 Bootstrap Operation** for additional information.

## 8.9 Flash EEPROM Operation

The following paragraphs describe FLASH module reset and using the module for system bootstrap, normal operation, and array programming and erasure.

### 8.9.1 Reset Operation

Reset initializes all FLASH control registers. Some bits have fixed default values, and some take on values that are programmed into the associated FLASH shadow registers.

When the state of the STOP shadow bit is zero, the STOP bit in the FEEMCR is cleared during reset, and the module responds to accesses in the range specified by FEEBAH. When the BOOT bit is cleared, the module also responds to bootstrap vector accesses.

When the state of the STOP shadow bit is one, the STOP bit in the FEEMCR is set during reset and the FLASH array is disabled. The module does not respond to array or bootstrap vector accesses until the STOP bit is cleared. This allows an external device to respond to accesses to the FLASH array address space or to bootstrap accesses. The erased state of the shadow bits is one — an erased module comes out of reset in STOP mode.

### 8.9.2 Bootstrap Operation

The CPU32 begins bootstrap operation by fetching initial values for its internal registers from IMB addresses $000000 through $000006 in program space. These are the addresses of the bootstrap vectors in the exception vector table. If the BOOT and STOP bits in the FEEMCR are cleared during reset, the FLASH module is configured to respond to bootstrap vector accesses. Vector assignments are shown in Table 8–3.

**Table 8–3. Reset Vector Assignments**

| EEPROM Bootstrap Word | IMB Vector Address | MC68F333 Reset Vector Content |
|---|---|---|
| FEEBS0 | $000000 | Initial Stack Pointer (High-Order Word) |
| FEEBS1 | $000002 | Initial Stack Pointer (Low-Order Word) |
| FEEBS2 | $000004 | Initial Program Counter (High-Order Word) |
| FEEBS3 | $000006 | Initial Program Counter (Low-Order Word) |

As soon as address $000006 has been read, FLASH operation returns to normal, and the module no longer responds to bootstrap vector accesses.

### 8.9.3 Normal Operation

The FLASH module performs byte or aligned-word accesses in one bus cycle. Long-word reads or writes require an additional bus cycle. The WAIT field in the FEEMCR can be used to insert wait states.

The module checks function codes to verify address-space access type. Array accesses are defined by the state of ASPC in the FEEMCR. When the FLASH module is configured for normal operation, the array responds to read accesses only; write operations are ignored.

Accesses to an address in the 64-Kbyte block defined by the base address registers that does not fall within the array (the upper 16 Kbytes of the block) are driven externally, allowing an external device to fill the entire address space defined by the base address.

### 8.9.4 Program/Erase Operation

An unprogrammed FLASH bit has a logic state of one. A bit must be programmed to change its state from one to zero. Erasing a bit returns it to the state of one. Programming and erasing the FLASH requires a series of control register writes and a write to a set of programming latches. The same procedure is used to program array locations and control registers that contain FLASH bits. Programming is restricted to a single byte or aligned word at a time. The entire FLASH array and the shadow register bits are erased at the same time. For programming and erasure specifications refer to **APPENDIX A ELECTRICAL CHARACTERISTICS**.

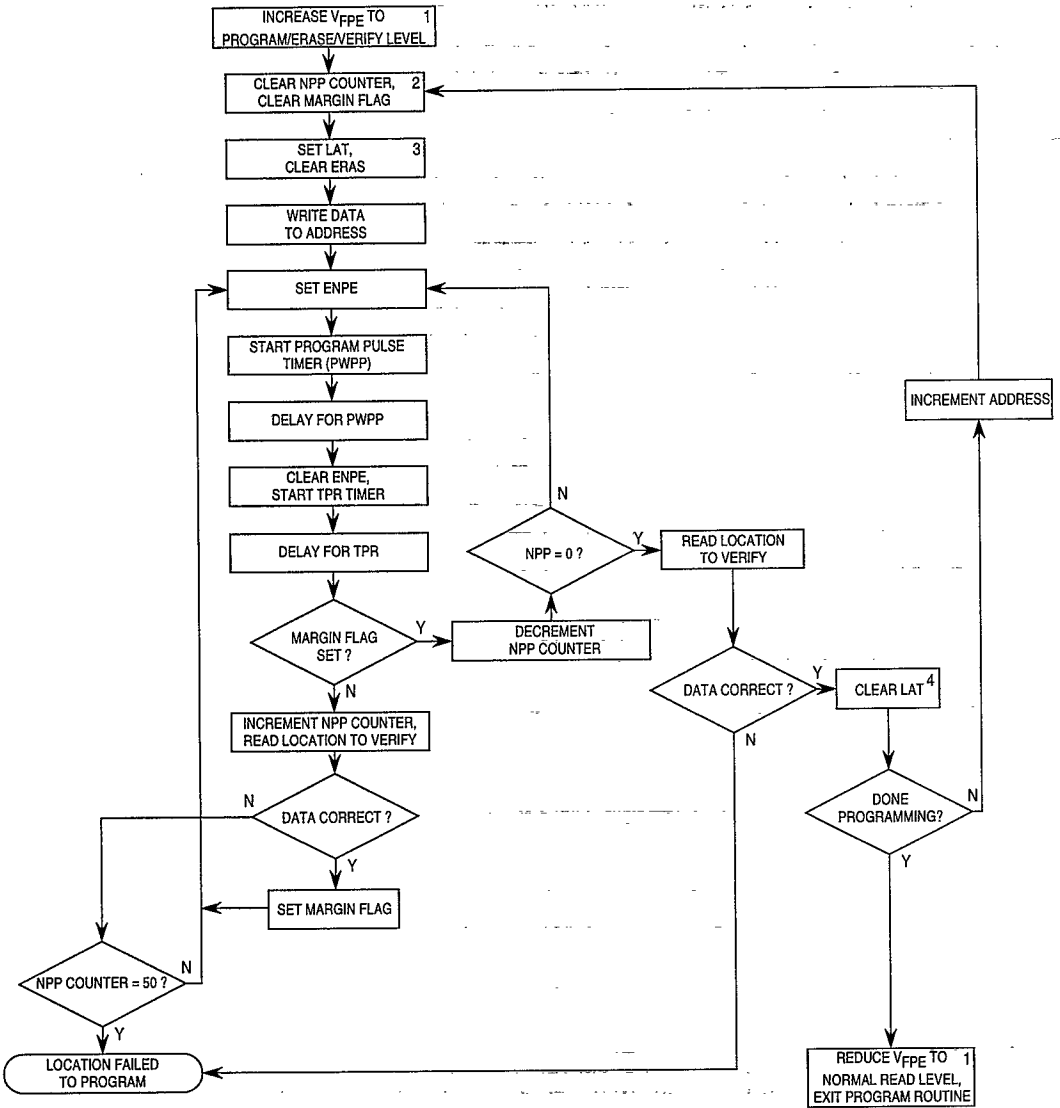
### 8.9.4.1 Programming Operation

Perform the following steps to program the flash EEPROM array. Figure 8–1 is a flowchart of programming operation.

1. Increase voltage applied to the $V_{FPE}$ pin to program/erase/verify level.
2. Clear ERAS bit and set the LAT bit in FEECTL. This enables the programming address and data latches.
3. Write data to the address to be programmed. This latches the address to be programmed and the programming data.
4. Set the ENPE bit in FEECTL. This starts the program pulse.
5. Delay the proper amount of time for one programming pulse to take place. Delay is specified by parameter $pw_{pp}$.
6. Clear the ENPE bit in FEECTL. This stops the program pulse.
7. Delay while high voltage to array is turned off. Delay is specified by parameter $t_{pr}$.
8. Read the address to verify that it has been programmed.

   If the location is not programmed, repeat steps 4 through 7 until the location is programmed, or until the specified maximum number of program pulses has been reached. Maximum number of pulses is specified by parameter $n_{pp}$.

   If the location is programmed, repeat the same number of pulses as required to program the location. This provides 100% program margin.
9. Read the address to verify that it remains programmed.
10. Clear the LAT bit in FEECTL. This disables the programming address and data latches.
11. If more locations are to be programmed, repeat steps 2 through 10.
12. Reduce voltage applied to the $V_{FPE}$ pin to normal read level.




NOTES:
1. SEE APPENDIX A FOR $V_{FPE}$ PIN VOLTAGE SEQUENCING.
2. THE MARGIN FLAG IS A SOFTWARE-DEFINED FLAG THAT INDICATES WHETHER THE PROGRAM SEQUENCE IS GENERATING PROGRAM PULSES OR MARGIN PULSES.
3. TO SIMPLIFY THE PROGRAM OPERATION, THE VFPE BIT IN FEECTL CAN BE SET. SEE APPENDIX D FOR INFORMATION ON FEECTL REGISTER.
4. CLEAR VFPE BIT ALSO IF ROUTINE USES THIS FUNCTION.

FEEPROM PGM FLOW1

**Figure 8–1. FLASH Programming Flow**

### 8.9.4.2 Erase Operation

Perform the following steps to erase the flash EEPROM array. Figure 8–2 is a flowchart of erase operation.

1. Increase voltage applied to the $V_{FPE}$ pin to program/erase/verify level.
2. Set the ERAS bit and the LAT bit in FEECTL. This configures the module for erasure.
3. Perform a write to any valid address in the control block or array. The data written does not matter.
4. Set the ENPE bit in FEECTL. This applies erase voltage to the array.
5. Delay the proper amount of time for one erase pulse. Delay is specified by parameter $t_{epk}$.
6. Clear the ENPE bit in FEECTL. This turns off erase voltage to the array.
7. Delay while high voltage to array is turned off. Delay is specified by parameter $t_{er}$.
8. Clear the LAT and ERAS bits in FEECTL. This allows normal access to the FLASH.
9. Read the entire array and control block to ensure all locations are erased.

   If all locations are not erased, calculate a new value for $t_{epk}$ ($t_{ei}$ * pulse number) and repeat steps 3 through 10 until all locations erase, or the maximum number of pulses has been applied.

   If all locations are erased, calculate the erase margin ($e_m$) and repeat steps 3 through 10 for the single margin pulse.
10. Reduce voltage applied to the $V_{FPE}$ pin to normal read level.

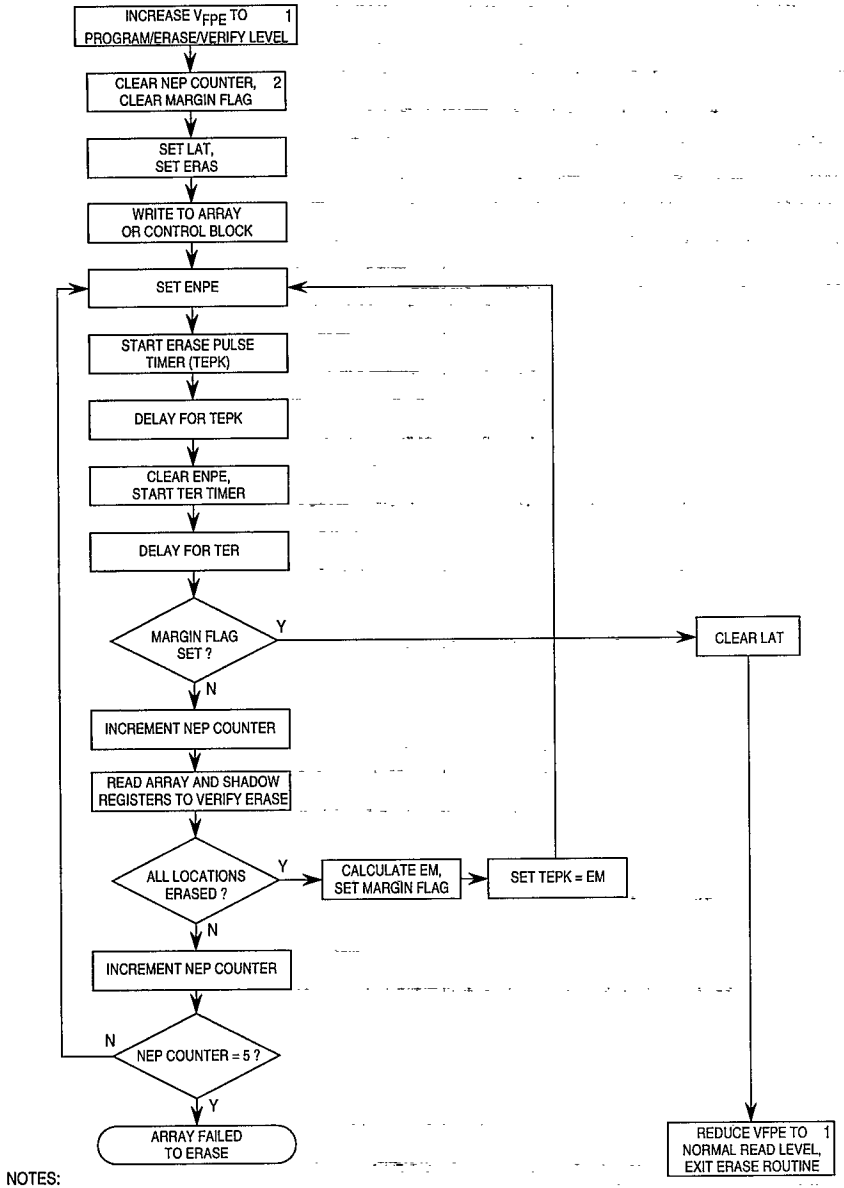



**Figure 8–2. FLASH Erasure Flow**

### 8.9.5 FLASH Program/Erase Voltage Signal Conditioning

A voltage of at least $V_{DD}$ – 0.35 V must be applied at all times to the $V_{FPE}$ pins or damage to the FLASH module can occur. FLASH modules can be damaged by power-on and power off $V_{FPE}$ transients. $V_{FPE}$ must not rise to programming level while $V_{DD}$ is below specified minimum value, and must not fall below minimum specified value while $V_{DD}$ is applied. Figure 8–3 shows the $V_{FPE}$ and $V_{DD}$ operating envelope. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for maximum and operating voltage specifications.

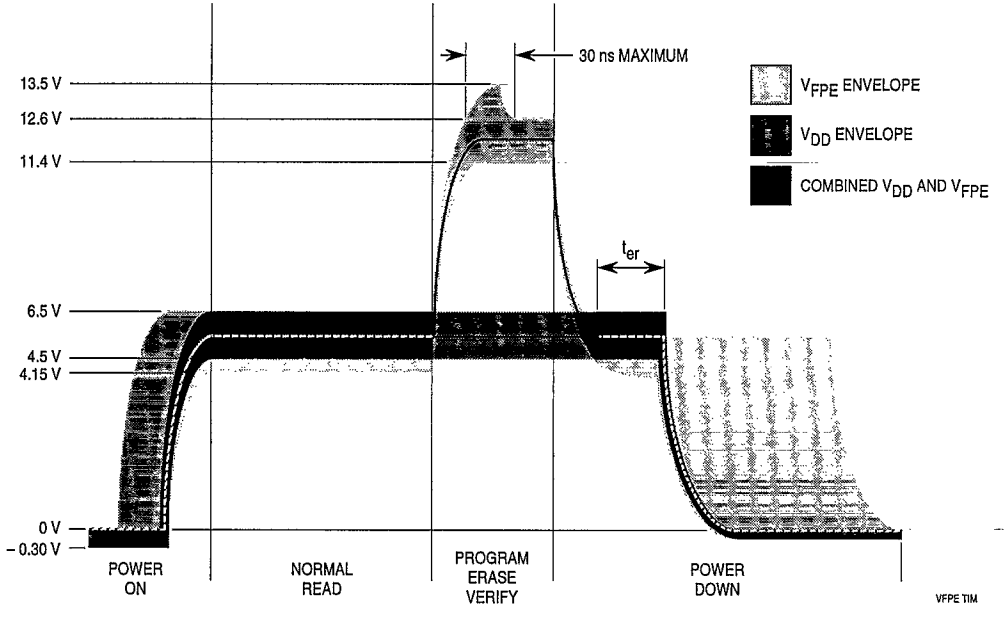



**Figure 8–3. $V_{FPE}$ Operating Range**

Use of an external circuit to condition VFPE is recommended. Figure 8–4 shows a simple circuit that maintains required voltages and filters transients. $V_{FPE}$ is pulled up to $V_{DD}$ via Schottky diode D2. Application of programming voltage via diode D1 reverse-biases D2, protecting $V_{DD}$ from excessive reverse current. D2 also protects the FLASH from damage should programming voltage go to zero. Programming power supply voltage must be adjusted to compensate for the forward-bias drop across D1. The charge time constant of R1 and C1 filters transients, while R2 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 $V_{FPE}$ pin low, to minimize diode voltage drop.

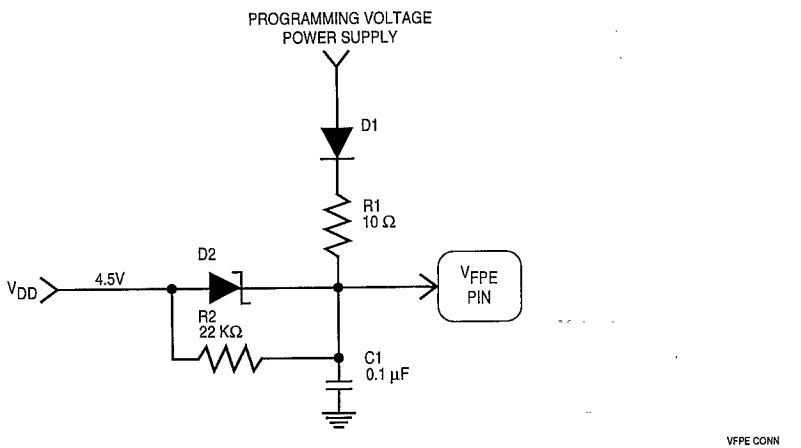



**Figure 8–4. $V_{FPE}$ Conditioning Circuit**

# SECTION 9
# STANDBY RAM MODULE

The standby RAM (SRAM) module consists of a control register block and a 512-byte array of fast (two bus cycle) static RAM. The SRAM is especially useful for system stacks and variable storage. SRAM can be mapped to any 512-byte boundary in the address map, but must not overlap the module control registers. (Overlap makes the registers inaccessible.) Data can be read or written in bytes, words or long words. SRAM is powered by $V_{DD}$ in normal operation. During power-down, SRAM contents can be maintained by power from the $V_{STBY}$ input. Power switching between sources is automatic.

## 9.1 SRAM Register Block

There are four SRAM control registers: the SRAM module configuration register (SRAMMCR), the SRAM test register (SRAMTST), and the SRAM array base address registers (SRAMBAH and SRAMBAL). To protect these registers from accidental modification, they are always mapped to supervisor data space.

The module mapping (MM) bit in the SCIM configuration register (SCIMCR) defines the most significant bit (ADDR23) of the IMB address for each module in the MC68F333. **SECTION 3 SINGLE-CHIP INTEGRATION MODULE** contains more information about how the state of MM affects the system.

There is an eight-byte minimum control register block size for the SRAM module. Unimplemented register addresses are read as zeros, and writes have no effect. Refer to **APPENDIX D REGISTER SUMMARY** for the register block address map and register bit/field definitions.

## 9.2 SRAM Array Address Mapping

Base address registers SRAMBAH and SRAMBAL are used to specify the SRAM array base address in the MC68F333 memory map. RAMBAH and RAMBAL can only be written while the SRAM is in low-power mode (RAMMCR STOP = 1) and the base address lock is disabled (RAMMCR RLCK = 0). RLCK can be written once only to a value of one. This prevents accidental remapping of the array.

## 9.3 SRAM Array Address Space Type

The RASP field in the SRAMMCR determines SRAM array address space type. The SRAM module can respond to both program and data space accesses or to program space accesses only. This allows code to be executed from RAM, and

permits use of program counter relative addressing mode for operand fetches from the array. In addition, RASP specifies whether access to the SRAM module can be made from the supervisor privilege level only or from either the user or supervisor privilege level. If supervisor-only access is specified, an access from the user privilege level is ignored by the SRAM control logic and can be decoded externally.

Table 9–1 shows RASP field encodings.

**Table 9–1. SRAM Array Address Space Type**

| RASP | Space |
|---|---|
| 00 | Unrestricted Program and Data |
| 01 | Unrestricted Program |
| 10 | Supervisor Program and Data |
| 11 | Supervisor Program |

Refer to **SECTION 3 SINGLE-CHIP INTEGRATION MODULE** and **SECTION 4 CENTRAL PROCESSING UNIT** for more information concerning address space types and program and data space access.

## 9.4 Normal Access

The array can be accessed by byte, word, or long word. A byte or aligned word access takes one bus cycle (two system clock cycles). A long word access requires two bus cycles. Refer to **SECTION 4 SINGLE-CHIP INTEGRATION MODULE** for more information concerning access times.



## 9.5 Standby Operation

Standby mode maintains the RAM array when the MCU main power supply is turned off. Low-power mode allows the central processing unit to control MCU power consumption.

Relative voltage levels of the $V_{DD}$ and $V_{STBY}$ pins determine whether the SRAM is in standby mode. SRAM circuitry switches to the standby power source when specified limits are exceeded. If specified standby supply voltage levels are maintained during the transition, there is no loss of memory when switching occurs. The SRAM array cannot be accessed while the module is powered from $V_{STBY}$. If standby operation is not desired, connect the $V_{STBY}$ pin to the $V_{SS}$ pin.

$I_{SB}$ exceeds specified maximum standby current during the time $V_{DD}$ makes the transition from normal operating level to the level specified for standby operation. This occurs within the voltage range $V_{SB} - 0.5\ V \geq V_{DD} \geq V_{SS} + 0.5\ V$. Typically,

$I_{SB}$ peaks when $V_{DD} \approx V_{SB} - 1.5$ V, and averages 1.0 mA over the transition period.

Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for standby switching and power consumption specifications.

To prevent standby supply voltage from going below the specified minimum, a filter capacitor must be attached between the $V_{STBY}$ and $V_{SS}$ pins. To calculate filter capacitance, $V_{DD}$ supply ramp time, available standby voltage, and available standby current must be known. Assuming that the rate of change is constant as $V_{DD}$ changes from 0.0 V to 5.5 V (nominal $V_{SS}$ to nominal $V_{DD}$) and that $V_{SB}$ also drops during this period, capacitance is calculated using the expression:

$$C = \frac{It}{V}$$

Where:

C = Desired capacitance
I = $I_{SB}$ differential (Transient $I_{SB}$ – Available supply current)
t = time of maximum $I_{SB}$ (Typically in the range $V_{SB} - 1.5\ V \pm 0.5\ V$)
V = $V_{SB}$ differential (Available supply voltage – Specified minimum $V_{SB}$)

## 9.6 Low-Power Stop Operation

Setting the STOP bit in the SRAMMCR switches the SRAM module to low-power mode. In low-power mode, the array retains its contents, but cannot be read or written by the CPU. STOP can be written only when the processor is operating at the supervisor privilege level. STOP is set during reset. Stop mode is exited by clearing STOP.

The SRAM module will switch to standby mode while it is in low-power mode, provided the operating constraints discussed above are met.

## 9.7 Reset

Reset places the SRAM in low-power mode, enables program space access, and clears the base address registers and the register lock bit. These actions make it possible to write a new base address into the registers.

When a synchronous reset occurs while a byte or word SRAM access is in progress, the access is completed. If reset occurs during the first word access of a long-word operation, only the first word access is completed. If reset occurs during the second word access of a long-word operation, the entire access is completed. Data being read from or written to the SRAM may be corrupted by asynchronous reset. Refer to **SECTION 3 SINGLE-CHIP INTEGRATION MODULE** for more information concerning resets.

# SECTION 10
# STANDBY RAM WITH TPU EMULATION

The standby RAM module with TPU emulation capability (TPURAM) consists of a control register block and a 3.5-Kbyte array of fast (two bus cycle) static RAM. The TPURAM is especially useful for system stacks and variable storage. The TPURAM responds to both program and data space accesses. This allows code to be executed from RAM and permits use of program counter relative addressing mode for operand fetches from the array. Alternately, the array can be used by the TPU as emulation RAM for new timer algorithms.

The TPURAM can be mapped to the lower 3.5 Kbytes of any 4-Kbyte boundary in the address map, but must not overlap the module control registers. (Overlap makes the registers inaccessible.) Data can be read or written in bytes, words or long words.

The TPURAM is powered by $V_{DD}$ in normal operation. During power-down, TPURAM contents can be maintained by power from the $V_{STBY}$ input. Power switching between sources is automatic.

## 10.1 TPURAM Register Block

There are three TPURAM control registers: the TPURAM module configuration register (TRAMMCR), the TPURAM test register (TRAMTST), and the TPURAM base address and status register (TRAMBAR). To protect these registers from accidental modification, they are always mapped to supervisor data space.



The module mapping (MM) bit in the SCIM configuration register (SCIMCR) defines the most significant bit (ADDR23) of the IMB address for each module in the MC68F333. **SECTION 3 SINGLE-CHIP INTEGRATION MODULE** contains more information about how the state of MM affects the system.

There is an eight-byte minimum control register block size for the TPURAM module. Unimplemented register addresses are read as zeros, and writes have no effect. Refer to **APPENDIX D REGISTER SUMMARY** for the register block address map and register bit/field definitions.

## 10.2 TPURAM Array Address Mapping

Base address and status register TRAMBAR specifies the TPURAM array base address in the MC68F333 memory map. TRAMBAR[15:3] specify the 13 MSB of the base address. The TPU bus interface unit compares these bits to address lines ADDR[23:11]. If the two match, then the low order address lines and the SIZ[1:0] signals are used to access the RAM location in the array. The TPURAM can be mapped to the lower 3.5 Kbytes of any 4-Kbyte boundary in the address map, but must not overlap the module control registers. (Overlap makes the registers inaccessible.) The TPURAM module ignores attempts to access the upper 512 bytes of the 4-Kbyte block, allowing an external device to respond to the address.

The RAM disable (RAMDS) bit, the LSB of the TRAMBAR, indicates whether the TPURAM array is active (RAMDS = 0) or disabled (RAMDS = 1). The array is disabled coming out of reset and remains disabled if the base address field is programmed with an address that overlaps the address of the module control register block. Writing a valid base address to TRAMBAR[15:3] clears RAMDS and enables the array.

The TRAMBAR can be written only once after a master reset. This prevents runaway software from accidentally re-mapping the array. Because the locking mechanism is activated by the first write after a master reset, the base address field should be written in a single word operation. Writing only one half of the register prevents the other half from being written.

## 10.3 TPURAM Privilege Level

The RASP field in the TRAMMCR specifies whether access to the TPURAM module can be made from the supervisor privilege level only or from either the user or supervisor privilege level. If supervisor-only access is specified, an access from the user privilege level is ignored by the TPURAM control logic and can be decoded externally. For more information concerning privilege levels refer to **SECTION 3 SINGLE-CHIP INTEGRATION MODULE** and **SECTION 4 CENTRAL PROCESSING UNIT**.



## 10.4 Normal Operation

In normal operation, TPURAM is accessed via the IMB by a bus master and is powered by $V_{DD}$. The array can be accessed by byte, word, or long word. A byte or aligned word access takes one bus cycle (two system clock cycles). A long word access requires two bus cycles. Refer to **SECTION 3 SINGLE-CHIP INTEGRATION MODULE** for more information concerning access times.

During normal operation, the TPU does not access the array and has no effect on the operation of the TPURAM module.

## 10.5 Standby Operation

Standby mode maintains the RAM array when the MCU main power supply is turned off. Low-power mode allows the central processing unit to control MCU power consumption.

Relative voltage levels of the $V_{DD}$ and $V_{STBY}$ pins determine whether the TPURAM is in standby mode. TPURAM circuitry switches to the standby power source when specified limits are exceeded. If specified standby supply voltage levels are maintained during the transition, there is no loss of memory when switching occurs. The RAM array cannot be accessed while the TPURAM module is powered from $V_{STBY}$. If standby operation is not desired, connect the $V_{STBY}$ pin to the $V_{SS}$ pin.

$I_{SB}$ exceeds specified maximum standby current during the time $V_{DD}$ makes the transition from normal operating level to the level specified for standby operation. This occurs within the voltage range $V_{SB} - 0.5\ V \geq V_{DD} \geq V_{SS} + 0.5\ V$. Typically, $I_{SB}$ peaks when $V_{DD} \approx V_{SB} - 1.5\ V$, and averages 1.0 mA over the transition period.

Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for standby switching and power consumption specifications.

To prevent standby supply voltage from going below the specified minimum, a filter capacitor must be attached between the $V_{STBY}$ and $V_{SS}$ pins. To calculate filter capacitance, $V_{DD}$ supply ramp time, available standby voltage, and available standby current must be known. Assuming that the rate of change is constant as $V_{DD}$ changes from 0.0 V to 5.5 V (nominal $V_{SS}$ to nominal $V_{DD}$) and that $V_{SB}$ also drops during this period, capacitance is calculated using the following expression:

$$C = \frac{It}{V}$$

Where:

C = Desired capacitance
I = $I_{SB}$ differential (Transient $I_{SB}$ – Available supply current)
t = time of maximum $I_{SB}$ (Typically in the range $V_{SB} - 1.5\ V \pm 0.5\ V$)
V = $V_{SB}$ differential (Available supply voltage – Specified minimum $V_{SB}$)

## 10.6 Low-Power Stop Operation

Setting the STOP bit in the TRAMMCR switches the TPURAM module to low-power mode. In low-power mode, the array retains its contents, but cannot be read or written by the CPU. STOP can be written only when the processor is

operating at the supervisor privilege level. STOP is set during reset. Stop mode is exited by clearing STOP.

The TPURAM module will switch to standby mode while it is in low-power mode, provided the operating constraints discussed above are met.

## 10.7 Reset

Reset places the TPURAM in low-power mode, enables supervisor-level access only, clears the base address, and disables the array. These actions make it possible to write a new base address into the base address register.

When a synchronous reset occurs while a byte or word TPURAM access is in progress, the access is completed. If reset occurs during the first word access of a long-word operation, only the first word access is completed. If reset occurs during the second word access of a long-word operation, the entire access is completed. Data being read from or written to the TPURAM may be corrupted by asynchronous reset. For more information concerning resets refer to **SECTION 3 SINGLE-CHIP INTEGRATION MODULE**.

## 10.8 TPU Microcode Emulation

The TPURAM array can emulate the microcode ROM in the TPU module. This provides a means of developing custom TPU code. The TPU selects TPU emulation mode.

The TPU is connected to the TPURAM via a dedicated bus. While the TPURAM array is in TPU emulation mode, the access timing of the TPURAM module matches the timing of the TPU microinstruction ROM to ensure accurate emulation. Normal accesses through the IMB are inhibited and the control registers have no effect, allowing external RAM to emulate the TPURAM at the same addresses. Refer to **SECTION 5 TIME PROCESSOR UNIT** and to the *TPU Reference Manual* (TPURM/AD) for more information.

# APPENDIX A
# ELECTRICAL CHARACTERISTICS

This appendix contains electrical specification tables and reference timing diagrams. Each timing diagram has an associated key table made up of parameters abstracted from the specification tables. Pertinent notes have been included in the key tables.

## Table A–1. Maximum Ratings

| Num | Rating | Symbol | Value | Unit |
|---|---|---|---|---|
| 1 | Supply Voltage[1,2] | $V_{DD}$ | –0.3 to + 6.5 | V |
| 2 | Input Voltage[1,2,3,4] | $V_{in}$ | –0.3 to +6.5 | V |
| 3 | Instantaneous Maximum Current<br>Single pin limit (applies to all pins)[1,4,5,6] | $I_D$ | 25 | mA |
| 4 | Operating Maximum Current<br>Digital input disruptive current [4,5,6]<br>$V_{SS} - 0.3 \leq V_{IN} \leq V_{DD} + 0.3$ | $I_{ID}$ | – 500 to +500 | µA |
| 5 | Flash EEPROM Program/Erase<br>Supply Voltage[7,8] | $V_{FPE}$ | ($V_{DD}$ – 0.35) to +12.6 | V |
| 6 | Operating Temperature Range<br>MC68F333 "C" Suffix | $T_A$ | $T_L$ to $T_H$<br>–40 to +85 | °C |
| 7 | 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 $V_{DD}$.
4. Power supply must maintain regulation within operating $V_{DD}$ range during instantaneous and operating maximum current condition.
5. This parameter is periodically sampled rather than 100% tested.
6. 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.
7. $V_{FPE}$ must not be raised to programming level while $V_{DD}$ is below specified minimum value. $V_{FPE}$ must not be reduced below minimum specified value while $V_{DD}$ is applied.
8. Flash EEPROM modules can be damaged by power-on and power-off $V_{FPE}$ transients. Maximum power-on overshoot tolerance is 13.5 V for periods of less than 30 ns.

A

## Table A–2. Typical Ratings

| Num | Rating | Symbol | Value | Unit |
|---|---|---|---|---|
| 1 | Supply Voltage | $V_{DD}$ | 5.0 | V |
| 2 | Operating Temperature | $T_A$ | 25 | °C |
| 3 | $V_{DD}$ Supply Current<br>RUN<br>LPSTOP, VCO Off<br>LPSTOP, External clock, max $f_{sys}$ | $I_{DD}$ | <br>90<br>125<br>3 | <br>mA<br>µA<br>µA |
| 4 | Clock Synthesizer Operating Voltage | $V_{DDSYN}$ | 5.0 | V |
| 5 | $V_{DDSYN}$ Supply Current<br>VCO on, maximum $f_{sys}$<br>External Clock, maximum $f_{sys}$<br>LPSTOP, VCO off<br>$V_{DD}$ powered down | $I_{DDSYN}$ | <br>1.0<br>4.0<br>250<br>50 | <br>mA<br>mA<br>µA<br>µA |
| 6 | RAM Standby Voltage | $V_{SB}$ | 3.0 | V |
| 7 | RAM Standby Current<br>Normal RAM operation<br>Standby operation | $I_{SB}$ | <br>7.0<br>40 | <br>µA<br>µA |
| 8 | Power Dissipation | $P_D$ | 455 | mW |

A

## Table A–3. Thermal Characteristics

| Num | Characteristic | Symbol | Value | Unit |
|---|---|---|---|---|
| 1 | Thermal Resistance<br>Plastic 160-Pin Surface Mount | $\Theta_{JA}$ | 37 | °C/W |

The average chip-junction temperature ($T_J$) in C can be obtained from:

$$T_J = T_A + (P_D \Theta_{JA}) \quad (1)$$

where

$T_A$ = Ambient Temperature, °C

$\Theta_{JA}$ = 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:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = P_D + (T_A + 273°C) + \Theta_{JA} \times P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring $P_D$ (at 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 $T_A$.

## Table A–4. Clock Control Timing

($V_{DD}$ and $V_{DDSYN}$ = 5.0 Vdc ±10%, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$, 4.194-MHz reference)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | PLL Reference Frequency Range | $f_{ref}$ | 25 | 50 | kHz |
| 2 | System Frequency[1] | $f_{sys}$ | dc | 16.78 | MHz |
| | On-Chip PLL Frequency | | 0.131 | 16.78 | |
| | External Clock Operation | | dc | 16.78 | |
| 3 | PLL Startup Time[2,3,4,5] | $t_{spll}$ | — | 50 | ms |
| 4 | PLL Lock Time[2,4,5,6] | $t_{lpll}$ | — | 20 | ms |
| 5 | Limp Mode Clock Frequency[7] | $f_{limp}$ | | | MHz |
| | SYNCR X Bit = 0 | | — | $f_{sys}$ max /2 | |
| | SYNCR X Bit = 1 | | — | $f_{sys}$ max | |
| 6 | CLKOUT Stability[2,4,5,8] | $C_{stab}$ | | | % |
| | Short term (5 µs interval) | | –0.5 | 0.5 | |
| | Long term (500 µs interval) | | –0.05 | 0.05 | |

NOTES:

1. All internal registers retain data at 0 Hz.
2. This parameter is periodically sampled rather than 100% tested.
3. Parameter measured from the time $V_{DD}$ and $V_{DDSYN}$ are valid to $\overline{\text{RESET}}$ release.
4. Assumes that an external filter capacitor with a value of 0.1 µF and an insulation resistance greater than 30,000 MΩ is attached to the XFC pin. Total external resistance from the XFC pin due to external leakage sources must be greater than 10 MΩ to guarantee this specification.
5. Proper layout procedures must be followed to achieve specifications.
6. Assumes that stable $V_{DDSYN}$ is applied, and that the crystal oscillator is stable. Lock time is measured from the time $V_{DD}$ and $V_{DDSYN}$ are valid to $\overline{\text{RESET}}$ release. This specification also applies to the period required for PLL lock after changing the W and Y frequency control bits in the synthesizer control register (SYNCR) while the PLL is running, and to the period required for the clock to lock after LPSTOP.
7. Determined by the initial control voltage applied to the on-chip VCO. The X bit in SYNCR controls a divide by two scaler on the system clock output.
8. Stability is the average deviation from the 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 $V_{DDSYN}$ and $V_{SS}$ and variation in crystal oscillator frequency increase the $C_{stab}$ percentage for a given interval.

## Table A–5. DC Characteristics

($V_{DD}$ and $V_{DDSYN}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | Input High Voltage | $V_{IH}$ | 0.7 ($V_{DD}$) | $V_{DD}$ + 0.3 | V |
| 2 | Input Low Voltage | $V_{IL}$ | $V_{SS}$ – 0.3 | 0.2 ($V_{DD}$) | V |
| 3 | Input Hysteresis[1,9] | $V_{HYS}$ | 0.5 | — | V |
| 4 | Input Leakage Current[2]<br>$V_{in} = V_{DD}$ or $V_{SS}$ All input-only pins except ADC pins | $I_{in}$ | –2.5 | 2.5 | µA |
| 5 | High Impedance (Off-State) Leakage Current[2]<br>$V_{in} = V_{DD}$ or $V_{SS}$ All input/output and output pins | $I_{OZ}$ | –2.5 | 2.5 | µA |
| 6 | CMOS Output High Voltage[2,3]<br>$I_{OH}$ = –10.0 µA Group 1,2,4 input/output and all output pins | $V_{OH}$ | $V_{DD}$ – 0.2 | — | V |
| 7 | CMOS Output Low Voltage[2]<br>$I_{OL}$ = 10.0 µA Group 1,2,4 input/output and all output pins | $V_{OL}$ | — | 0.2 | V |
| 8 | Output High Voltage [2,3]<br>$I_{OH}$ = –0.8 mA Group 1,2,4 input/output and all output pins | $V_{OH}$ | $V_{DD}$ – 0.8 | — | V |
| 9 | Output Low Voltage[2]<br>$I_{OL}$ = 1.6 mA Group 1 I/O Pins, CLKOUT, FREEZE/QUOT, IPIPE<br>$I_{OL}$ = 5.3 mA Group 2 and Group 4 I/O Pins, CSBOOT, BG/CS<br>$I_{OL}$ = 12 mA Group 3 | $V_{OL}$ | —<br>—<br>— | 0.4<br>0.4<br>0.4 | V |
| 10 | Data Bus Mode Select Pull-up Current[5]<br>$V_{in} = V_{IL}$ DATA[15:0]<br>$V_{in} = V_{IH}$ DATA[15:0] | $I_{MSP}$ | —<br>–15 | –120<br>— | µA |
| 11 | $V_{DD}$ Supply Current[6]<br>RUN[4]<br>RUN, TPU emulation mode<br>RUN, Single-Chip Mode<br>LPSTOP, 32.768-kHz crystal, VCO Off (STSCIM = 0)<br>LPSTOP (External clock input frequency = maximum $f_{sys}$) | $I_{DD}$ | —<br>—<br>—<br>—<br>— | 160<br>170<br>150<br>350<br>5 | mA<br>mA<br>mA<br>µA<br>mA |
| 12 | Clock Synthesizer Operating Voltage | $V_{DDSYN}$ | 4.5 | 5.5 | V |
| 13 | $V_{DDSYN}$ Supply Current[6]<br>32.768-kHz crystal, VCO on, maximum $f_{sys}$<br>External Clock, maximum $f_{sys}$<br>LPSTOP, 32.768-kHz crystal, VCO off (STSCIM = 0)<br>32.768-kHz crystal, $V_{DD}$ powered down | $I_{DDSYN}$ | —<br>—<br>—<br>— | 1<br>5<br>150<br>100 | mA<br>mA<br>µA<br>µA |
| 14 | RAM Standby Voltage[7]<br>Specified $V_{DD}$ applied<br>$V_{DD} = V_{SS}$ | $V_{SB}$ | 0.0<br>3.0 | 5.5<br>5.5 | V |
| 15 | RAM Standby Current[6]<br>Normal RAM operation[10] $V_{DD} > V_{SB} – 0.5$ V<br>Transient condition $V_{SB} – 0.5$ V ≥ $V_{DD}$ ≥ $V_{SS}$ + 0.5 V<br>Standby operation[7] $V_{DD} < V_{SS}$ + 0.5 V | $I_{SB}$ | —<br>—<br>— | TBD<br>5<br>100 | µA<br>mA<br>µA |
| 16 | Power Dissipation[8] | $P_D$ | — | 880 | mW |
| 17 | Input Capacitance[2,9] All input-only pins except ADC pins<br>All input/output pins | $C_{in}$ | —<br>— | 10<br>20 | pF |
| 18 | Load Capacitance[2]<br>Group 1 I/O Pins and CLKOUT, FREEZE/QUOT, IPIPE<br>Group 2 I/O Pins and CSBOOT, BG/CS<br>Group 3 I/O pins<br>Group 4 I/O pins | $C_L$ | —<br>—<br>—<br>— | 90<br>100<br>130<br>200 | pF |

A

## Table A–5. DC Characteristics (Continued)

($V_{DD}$ and $V_{DDSYN}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$)

NOTES:

1. Parameter applies to the following pins:
   Port ADA: PADA[7:0]/AN[7:0]
   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/$\overline{DS}$, PE3
   Port F: PF[7:1]/$\overline{IRQ}$[7:1], PF0/MODCLK
   Port G: PG[7:0]/DATA[15:8]
   Port H: PH[7:0]/DATA[7:0]
   Port QS: PQS7/TXD, PQS[6:4]/PCS[3:1], PQS3/PCS0/$\overline{SS}$, PQS2/SCK, PQS1/MOSI, PQS0/MISO
   Other: $\overline{BKPT}$/DSCLK, PAI, PCLK, $\overline{RESET}$, T2CLK, TPUCH[15:0], TSC
2. Input-Only Pins: $\overline{BKPT}$/DSCLK, EXTAL, PAI, PCLK, TSC
   Output-Only Pins: ADDR[2:0], $\overline{CSBOOT}$, $\overline{BG}$/$\overline{CS1}$, CLKOUT, FREEZE/QUOT, DSO/$\overline{IPIPE}$, PWMA, PWMB
   Input/Output Pins:
   Group 1: Port G: PG[7:0]/DATA[15:8]
   Port H: PH[7:0]/DATA[7:0]
   Other: DSI/$\overline{IFETCH}$, TPUCH[15:0]
   Group 2: Port A: PA[7:0]/ADDR[18:11]
   Port B: PB[7:0]/ADDR[10:3]
   Port C: PC[6:3]/ADDR[22:19]/$\overline{CS}$[9:6], PC2/FC2/$\overline{CS5}$,PC1/FC1, PC0/FC0/$\overline{CS3}$
   Port E: PE[7:6]/SIZ[1:0], PE5/$\overline{AS}$, PE4/$\overline{DS}$, PE3
   Port F: PF[7:1]/$\overline{IRQ}$[7:1], PF0/MODCLK
   Port QS: PQS7/TXD, PQS[6:4]/PCS[3:1], PQS3/PCS0/$\overline{SS}$
   Other: ADDR23/$\overline{CS10}$/ECLK, R/$\overline{W}$, $\overline{BERR}$, $\overline{BR}$/$\overline{CS0}$, $\overline{BGACK}$/$\overline{CS2}$
   Group 3: $\overline{HALT}$, $\overline{RESET}$
   Group 4: Port QS: PQS2/SCK, PQS1/MOSI, PQS0/MISO
3. Does not apply to $\overline{HALT}$, $\overline{RESET}$ (open-drain pins), and port QS in wired-OR mode.
4. Current measured with system clock frequency of 16.78 MHz, all modules active.
5. Use of an active pulldown device is recommended.
6. Total operating current is the sum of the appropriate $I_{DD}$, $I_{DDSYN}$, and $I_{SB}$ values, plus $I_{DDA}$. $I_{DD}$ values include supply currents for device modules powered by $V_{DDE}$ and $V_{DDI}$ pins.
7. RAM modules cannot switch into standby mode as long as $V_{SB}$ does not exceed $V_{DD}$ by more than 0.5 Volt. RAM arrays cannot be accessed while the module is in standby mode.
8. Power dissipation measured with system clock frequency of 16.78 MHz, all modules active. Specification does not apply to TPU emulation mode. Power dissipation can be calculated using the expression:

$$P_D = \text{Maximum } V_{DD}(I_{DD} + I_{DDSYN} + I_{SB}) + \text{Maximum } V_{DDA}(I_{DDA})$$

   $I_{DD}$ includes supply currents for all device modules powered by $V_{DDE}$ and $V_{DDI}$ pins.
9. This parameter is periodically sampled rather than 100% tested.
10. When $V_{SB}$ is more than 0.3 V greater than $V_{DD}$, 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 $V_{STBY}$ pin can contribute to this condition.

A

## Table A–6. AC Timing

($V_{DD}$ and $V_{DDSYN}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| F1 | Frequency of Operation (32.768 kHz crystal)[2] | f | 0.13 | 16.78 | MHz |
| 1 | Clock Period | $t_{cyc}$ | 59.6 | — | ns |
| 1A | ECLK Period | $t_{Ecyc}$ | 476 | — | ns |
| 1B | External Clock Input Period[3] | $t_{Xcyc}$ | 59.6 | — | ns |
| 2, 3 | Clock Pulse Width | $t_{CW}$ | 24 | — | ns |
| 2A, 3A | ECLK Pulse Width | $t_{ECW}$ | 236 | — | ns |
| 2B, 3B | External Clock Input High/Low Time[3] | $t_{XCHL}$ | 29.8 | — | ns |
| 4, 5 | Clock Rise and Fall Time | $t_{Crf}$ | — | 5 | ns |
| 4A, 5A | Rise and Fall Time — All Outputs except CLKOUT | $t_{rf}$ | — | 8 | ns |
| 4B, 5B | External Clock Rise and Fall Time[4] | $t_{XCrf}$ | — | 5 | ns |
| 6 | Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Valid | $t_{CHAV}$ | 0 | 29 | ns |
| 7 | Clock High to ADDR, DATA, FC, SIZE, $\overline{RMC}$ High Impedance | $t_{CHAZx}$ | 0 | 59 | ns |
| 8 | Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Invalid | $t_{CHAZn}$ | 0 | — | ns |
| 9 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted | $t_{CLSA}$ | 2 | 25 | ns |
| 9A | $\overline{AS}$ to $\overline{DS}$ or $\overline{CS}$ Asserted (Read)[5] | $t_{STSA}$ | –15 | 15 | ns |
| 9C | Clock Low to $\overline{IFETCH}$, $\overline{IPIPE}$ Asserted | $t_{CLIA}$ | 2 | 22 | ns |
| 11 | ADDR, FC, SIZE, $\overline{RMC}$ Valid to $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Asserted | $t_{AVSA}$ | 15 | — | ns |
| 12 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated | $t_{CLSN}$ | 2 | 29 | ns |
| 12A | Clock Low to $\overline{IFETCH}$, $\overline{IPIPE}$ Negated | $t_{CLIN}$ | 2 | 22 | ns |
| 13 | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to ADDR, FC, SIZE Invalid (Address Hold) | $t_{SNAI}$ | 15 | — | ns |
| 14 | $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted | $t_{SWA}$ | 100 | — | ns |
| 14A | $\overline{DS}$, $\overline{CS}$ Width Asserted (Write) | $t_{SWAW}$ | 45 | — | ns |
| 14B | $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted (Fast Write Cycle) | $t_{SWDW}$ | 40 | — | ns |
| 15 | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated[6] | $t_{SN}$ | 40 | — | ns |
| 16 | Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ High Impedance | $t_{CHSZ}$ | — | 59 | ns |
| 17 | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to R/$\overline{W}$ High | $t_{SNRN}$ | 15 | — | ns |
| 18 | Clock High to R/$\overline{W}$ High | $t_{CHRH}$ | 0 | 29 | ns |
| 20 | Clock High to R/$\overline{W}$ Low | $t_{CHRL}$ | 0 | 29 | ns |
| 21 | R/$\overline{W}$ High to $\overline{AS}$, $\overline{CS}$ Asserted | $t_{RAAA}$ | 15 | — | ns |
| 22 | R/$\overline{W}$ Low to $\overline{DS}$, $\overline{CS}$ Asserted (Write) | $t_{RASA}$ | 70 | — | ns |
| 23 | Clock High to Data Out Valid | $t_{CHDO}$ | — | 29 | ns |
| 24 | Data Out Valid to Negating Edge of $\overline{AS}$, $\overline{CS}$ (Fast Write Cycle) | $t_{DVASN}$ | 15 | — | ns |

A

## Table A–6. AC Timing (Continued)

($V_{DD}$ and $V_{DDSYN}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 25 | $\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold) | $t_{SNDOI}$ | 15 | — | ns |
| 26 | Data Out Valid to $\overline{DS}$, $\overline{CS}$ Asserted (Write) | $t_{DVSA}$ | 15 | — | ns |
| 27 | Data In Valid to Clock Low (Data Setup) | $t_{DICL}$ | 5 | — | ns |
| 27A | Late $\overline{BERR}$, $\overline{HALT}$ Asserted to Clock Low (Setup Time) | $t_{BELCL}$ | 20 | — | ns |
| 28 | $\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACK}$[1:0], $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated | $t_{SNDN}$ | 0 | 80 | ns |
| 29 | $\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)[7] | $t_{SNDI}$ | 0 | — | ns |
| 29A | $\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance[7,8] | $t_{SHDI}$ | — | 55 | ns |
| 30 | CLKOUT Low to Data In Invalid (Fast Cycle Hold)[7] | $t_{CLDI}$ | 15 | — | ns |
| 30A | CLKOUT Low to Data In High Impedance[7] | $t_{CLDH}$ | — | 90 | ns |
| 31 | $\overline{DSACK}$[1:0] Asserted to Data In Valid[9] | $t_{DADI}$ | — | 50 | ns |
| 33 | Clock Low to $\overline{BG}$ Asserted/Negated | $t_{CLBAN}$ | — | 29 | ns |
| 35 | $\overline{BR}$ Asserted to $\overline{BG}$ Asserted ($\overline{RMC}$ Not Asserted)[10] | $t_{BRAGA}$ | 1 | — | $t_{cyc}$ |
| 37 | $\overline{BGACK}$ Asserted to $\overline{BG}$ Negated | $t_{GAGN}$ | 1 | 2 | $t_{cyc}$ |
| 39 | $\overline{BG}$ Width Negated | $t_{GH}$ | 2 | — | $t_{cyc}$ |
| 39A | $\overline{BG}$ Width Asserted | $t_{GA}$ | 1 | — | $t_{cyc}$ |
| 46 | R/$\overline{W}$ Width Asserted (Write or Read) | $t_{RWA}$ | 150 | — | ns |
| 46A | R/$\overline{W}$ Width Asserted (Fast Write or Read Cycle) | $t_{RWAS}$ | 90 | — | ns |
| 47A | Asynchronous Input Setup Time<br>$\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK}$[1:0], $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$ | $t_{AIST}$ | 5 | — | ns |
| 47B | Asynchronous Input Hold Time | $t_{AIHT}$ | 15 | — | ns |
| 48 | $\overline{DSACK}$[1:0] Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted[11] | $t_{DABA}$ | — | 30 | ns |
| 53 | Data Out Hold from Clock High | $t_{DOCH}$ | 0 | — | ns |
| 54 | Clock High to Data Out High Impedance | $t_{CHDH}$ | — | 28 | ns |
| 55 | R/$\overline{W}$ Asserted to Data Bus Impedance Change | $t_{RADC}$ | 40 | — | 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 | 29 | ns |
| 71 | Data Setup Time to Clock Low (Show) | $t_{SCLDS}$ | 15 | — | ns |
| 72 | Data Hold from Clock Low (Show) | $t_{SCLDH}$ | 10 | — | ns |
| 73 | $\overline{BKPT}$ Input Setup Time | $t_{BKST}$ | 15 | — | ns |
| 74 | $\overline{BKPT}$ Input Hold Time | $t_{BKHT}$ | 10 | — | ns |
| 75 | Mode Select Setup Time | $t_{MSS}$ | 20 | — | $t_{cyc}$ |
| 76 | Mode Select Hold Time | $t_{MSH}$ | 0 | — | ns |
| 77 | $\overline{RESET}$ Assertion Time[12] | $t_{RSTA}$ | 4 | — | $t_{cyc}$ |
| 78 | $\overline{RESET}$ Rise Time[12,13] | $t_{RSTR}$ | — | 10 | $t_{cyc}$ |

A

## Table A–6. AC Timing (Continued)

($V_{DD}$ and $V_{DDSYN}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$)

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
2. Minimum system clock frequency is four times the crystal frequency, subject to specified limits.
3. 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 signal 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).
4. Parameters for an external clock signal applied while the internal PLL is disabled (MODCLK pin held low during reset). Does not pertain to an external VCO reference applied while the PLL is enabled (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.
5. Specification 9A is the worst-case skew between $\overline{AS}$ and $\overline{DS}$ or $\overline{CS}$. The amount of skew depends on the relative loading of these signals. When loads are kept within specified limits, skew will not cause $\overline{AS}$ and $\overline{DS}$ to fall outside the limits shown in specification 9.
6. If multiple chip selects are used, $\overline{CS}$ width negated (specification 15) 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 specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
7. 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.
8. Maximum value is equal to ($t_{cyc}$ / 2) + 25 ns.
9. If the asynchronous setup time (specification 47A) requirements are satisfied, the $\overline{DSACK[1:0]}$ low to data setup time (specification 31) and $\overline{DSACK[1:0]}$ low to $\overline{BERR}$ low setup time (specification 48) can be ignored. The data must only satisfy the data-in to clock low setup time (specification 27) for the following clock cycle. $\overline{BERR}$ must satisfy only the late $\overline{BERR}$ low to clock low setup time (specification 27A) for the following clock cycle.
10. 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 and $\overline{RMC}$ is asserted.
11. In the absence of $\overline{DSACK[1:0]}$, $\overline{BERR}$ is an asynchronous input using the asynchronous setup time (specification 47A).
12. After external $\overline{RESET}$ negation is detected, a short transition period (approximately 2 $t_{cyc}$) elapses, then the SCIM drives $\overline{RESET}$ low for 512 $t_{cyc}$.
13. External logic must pull $\overline{RESET}$ high during this period in order for normal MCU operation to begin.
14. 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.

A

## Table A–7. Background Debugging Mode Timing

($V_{DD}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| B0 | DSI Input Setup Time | $t_{DSISU}$ | 15 | — | ns |
| B1 | DSI Input Hold Time | $t_{DSIH}$ | 10 | — | ns |
| B2 | DSCLK Setup Time | $t_{DSCSU}$ | 15 | — | ns |
| B3 | DSCLK Hold Time | $t_{DSCH}$ | 10 | — | ns |
| B4 | DSO Delay Time | $t_{DSOD}$ | — | 25 | ns |
| B5 | DSCLK Cycle Time | $t_{DSCCYC}$ | 2 | — | $t_{cyc}$ |
| B6 | CLKOUT High to FREEZE Asserted/Negated | $t_{FRZAN}$ | — | 50 | ns |
| B7 | CLKOUT High to $\overline{\text{IFETCH}}$ High Impedance | $t_{IFZ}$ | — | 50 | ns |
| B8 | CLKOUT High to $\overline{\text{IFETCH}}$ Valid | $t_{IF}$ | — | 50 | ns |
| B9 | DSCLK Low Time | $t_{DSCLO}$ | 1 | — | $t_{cyc}$ |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.

## Table A–8. ECLK Bus Timing

($V_{DD}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$)

| 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 $\overline{\text{CS}}$ Valid ($\overline{\text{CS}}$ Delay) | $t_{ECSD}$ | — | 150 | ns |
| E4 | ECLK Low to $\overline{\text{CS}}$ Hold | $t_{ECSH}$ | 15 | — | ns |
| E5 | $\overline{\text{CS}}$ 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 | $\overline{\text{CS}}$ Negated to Data Hold (Read) | $t_{ECDH}$ | 0 | — | ns |
| E10 | $\overline{\text{CS}}$ 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 | $\overline{\text{CS}}$ 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 = $t_{Ecyc} - t_{ECSD} - t_{EDSR}$

A

## Table A–9. QSPI Timing

($V_{DD}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$, 200 pF load on all QSPI pins)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 0 | Operating Frequency<br>Master<br>Slave | $f_{op}$ | <br>DC<br>DC | <br>1/4<br>1/4 | <br>System Clock Frequency<br>System Clock Frequency |
| 1 | Cycle Time<br>Master<br>Slave | $t_{qcyc}$ | <br>4<br>4 | <br>510<br>— | <br>$t_{cyc}$<br>$t_{cyc}$ |
| 2 | Enable Lead Time<br>Master<br>Slave | $t_{lead}$ | <br>2<br>2 | <br>128<br>— | <br>$t_{cyc}$<br>$t_{cyc}$ |
| 3 | Enable Lag Time<br>Master<br>Slave | $t_{lag}$ | <br>—<br>2 | <br>1/2<br>— | <br>SCK<br>$t_{cyc}$ |
| 4 | Clock (SCK) High or Low Time<br>Master<br>Slave[2] | $t_{sw}$ | <br>2 $t_{cyc}$ – 60<br>2 $t_{cyc}$ – n | <br>255 $t_{cyc}$<br>— | <br>ns<br>ns |
| 5 | Sequential Transfer Delay<br>Master<br>Slave (Does Not Require Deselect) | $t_{td}$ | <br>17<br>13 | <br>8192<br>— | <br>$t_{cyc}$<br>$t_{cyc}$ |
| 6 | Data Setup Time (Inputs)<br>Master<br>Slave | $t_{su}$ | <br>30<br>20 | <br>—<br>— | <br>ns<br>ns |
| 7 | Data Hold Time (Inputs)<br>Master<br>Slave | $t_{hi}$ | <br>0<br>20 | <br>—<br>— | <br>ns<br>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)<br>Master<br>Slave | $t_v$ | <br>—<br>— | <br>50<br>50 | <br>ns<br>ns |
| 11 | Data Hold Time (Outputs)<br>Master<br>Slave | $t_{ho}$ | <br>0<br>0 | <br>—<br>— | <br>ns<br>ns |
| 12 | Rise Time<br>Input<br>Output | <br>$t_{ri}$<br>$t_{ro}$ | <br>—<br>— | <br>2<br>30 | <br>µs<br>ns |
| 13 | Fall Time<br>Input<br>Output | <br>$t_{fi}$<br>$t_{fo}$ | <br>—<br>— | <br>2<br>30 | <br>µs<br>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; for low time, n = External SCK fall time.

A

## Table A–10. ADC Maximum Ratings

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | Analog Supply | $V_{DDA}$ | – 0.3 | 6.5 | V |
| 2 | Internal Digital Supply | $V_{DDI}$ | – 0.3 | 6.5 | V |
| 3 | Reference Supply | $V_{RH}$, $V_{RL}$ | – 0.3 | 6.5 | V |
| 4 | $V_{SS}$ Differential Voltage | $V_{SSI} - V_{SSA}$ | – 0.1 | 0.1 | V |
| 5 | $V_{DD}$ Differential Voltage | $V_{DDI} - V_{DDA}$ | – 6.5 | 6.5 | V |
| 6 | $V_{REF}$ Differential Voltage | $V_{RH} - V_{RL}$ | – 6.5 | 6.5 | V |
| 7 | $V_{REF}$ to $V_{DDA}$ Differential Voltage | $V_{RH} - V_{DDA}$ | – 6.5 | 6.5 | V |
| 8 | Disruptive Input Current[1, 2, 3, 4, 5, 6]<br>$V_{NEGCLAMP} \cong -0.3$ V<br>$V_{POSCLAMP} \cong V_{DDA} + 2$ | $I_{NA}$ | – 44 | 44 | µA |
| 9 | Maximum Input Current [3, 4, 6]<br>$V_{NEGCLAMP} \cong -0.3$ V<br>$V_{POSCLAMP} \cong V_{DDA} + 2$ | $I_{MA}$ | – 500 | 500 | µA |

NOTES:

1. Below disruptive current conditions, the channel being stressed will have 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 interfere with conversion 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 than 100% tested.
6. Applies to single pin only.

## Table A–11. ADC DC Electrical Characteristics (Operating)

($V_{SS}$ = 0 Vdc, ADCLK = 2.1 MHz, $T_A = T_L$ to $T_H$)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | Analog Supply[1] | $V_{DDA}$ | 4.5 | 5.5 | V |
| 2 | Internal Digital Supply[1] | $V_{DDI}$ | 4.5 | 5.5 | V |
| 3 | $V_{SS}$ Differential Voltage | $V_{SSI} - V_{SSA}$ | – 1.0 | 1.0 | mV |
| 4 | $V_{DD}$ Differential Voltage | $V_{DDI} - V_{DDA}$ | – 1.0 | 1.0 | V |
| 5 | Reference Voltage Low[2,5] | $V_{RL}$ | $V_{SSA}$ | $V_{DDA}$ / 2 | V |
| 6 | Reference Voltage High[2,5] | $V_{RH}$ | $V_{DDA}$ / 2 | $V_{DDA}$ | V |
| 7 | $V_{REF}$ Differential Voltage[5] | $V_{RH} - V_{RL}$ | 4.5 | 5.5 | V |
| 8 | Input Voltage[2] | $V_{INDC}$ | $V_{SSA}$ | $V_{DDA}$ | V |
| 9 | Input High, Port ADA | $V_{IH}$ | 0.7 ($V_{DDA}$) | $V_{DDA}$ + 0.3 | V |
| 10 | Input Low, Port ADA | $V_{IL}$ | $V_{SSA}$ – 0.3 | 0.2 ($V_{DDA}$) | V |
| 15 | Analog Supply Current<br>Normal Operation[3]<br>Low-Power Stop | $I_{DDA}$ | <br>—<br>— | <br>1.0<br>200 | <br>mA<br>µA |
| 16 | Reference Supply Current | $I_{REF}$ | — | 250 | µA |
| 17 | Input Current, Off Channel[4] | $I_{OFF}$ | — | 150 | nA |
| 18 | Total Input Capacitance, Not Sampling | $C_{INN}$ | — | 10 | pF |
| 19 | Total Input Capacitance, Sampling | $C_{INS}$ | — | 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. Current measured at maximum system clock frequency with ADC active.
4. Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each 10° C decrease from maximum temperature.
5. Accuracy tested and guaranteed at $V_{RH} - V_{RL} \le 5.0\ V \pm 10\%$.

## Table A–12. ADC AC Characteristics (Operating)

($V_{DD}$ and $V_{DDA}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | On-Chip PLL Frequency | $f_{sys}$ | 2.0 | 16.78 | MHz |
| 2 | ADC Clock Frequency | $F_{ADCLK}$ | 0.5 | 2.1 | MHz |
| 3 | 8-Bit Conversion Time (16 ADC Clocks)[1] | $T_{CONV}$ | 7.62 | — | µs |
| 4 | 10-Bit Conversion Time (18 ADC Clocks)[1] | $T_{CONV}$ | 8.58 | — | µs |
| 5 | Stop Recovery Time | $T_{SR}$ | — | 10 | µs |

NOTES:

1. Assumes 2.1-MHz ADC clock and selection of minimum sample time (2 ADC clocks).

## Table A–13. ADC Conversion Characteristics (Operating)

($V_{DD}$ and $V_{DDA}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A = T_L$ to $T_H$, ADCLK = 2.1 MHz)

| Num | Characteristic | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| 1 | 8-Bit Resolution[1] | 1 Count | — | 20 | — | mV |
| 2 | 8-Bit Differential Nonlinearity | DNL | –0.5 | — | 0.5 | Counts |
| 3 | 8-Bit Integral Nonlinearity | INL | –1 | — | 1 | Counts |
| 4 | 8-Bit Absolute Error[2] | AE | –1 | — | 1 | Counts |
| 5 | 10-Bit Resolution[1] | 1 Count | — | 5 | — | mV |
| 6 | 10-Bit Differential Nonlinearity | DNL | –0.5 | — | 0.5 | Counts |
| 7 | 10-Bit Integral Nonlinearity | INL | –2.0 | — | 2.0 | Counts |
| 8 | 10-Bit Absolute Error[3] | AE | –2.5 | — | 2.5 | Counts |
| 9 | Source Impedance at Input[4] | $R_S$ | — | 20 | — | kΩ |

NOTES:

1. At $V_{RH} - V_{RL}$ = 5.12 V, one 10-bit count = 5 mV and one 8-bit count = 20 mV.
2. 8-bit absolute error of 1 count (20 mV) includes 1/2 count (10 mV) inherent quantization error and 1/2 count (10 mV) circuit (differential, integral, and offset) error.
3. 10-bit absolute error of 2.5 counts (12.5 mV) includes 1/2 count (2.5 mV) inherent quantization error and 2 counts (10 mV) circuit (differential, integral, and offset) error.
4. 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 ($V_{errj}$):

   $$V_{errj} = R_S \times I_{OFF}$$

   where $I_{OFF}$ is a function of operating temperature. (See Table A–11, note 4).

   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 A–14. Flash EEPROM Specifications

($V_{DD}$ = 5.0 Vdc ±10%, $V_{SS}$ = 0 Vdc, specified $f_{sys}$, $T_A = T_L$ to $T_H$)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | Program/Erase supply voltage[1]<br>Read operation<br>Program/Erase/Verify operation | $V_{FPE}$ | <br>$V_{DD}$ – 0.35<br>11.4 | <br>5.5<br>12.6 | V |
| 2 | Program/Erase/Verify supply current[2]<br>Read operation<br>Program/Erase/Verify operation<br>Verify (ENPE = 0)<br>Program byte (ENPE = 1)<br>Program word (ENPE = 1)<br>Erase (ENPE = 1) | $I_{FPE}$ | <br>—<br><br>—<br>—<br>—<br>— | <br>15<br><br>50<br>15<br>30<br>4 | <br>µa<br><br>µa<br>ma<br>ma<br>ma |
| 3 | Program recovery time | $t_{pr}$ | — | 10 | µs |
| 4 | Program pulse width | $pw_{pp}$ | 20 | 25 | µs |
| 5 | Number of program pulses[3] | $n_{pp}$ | — | 50 | — |
| 6 | Program margin[4] | $p_m$ | 100 | — | % |
| 7 | Number of erase pulses[3] | $n_{ep}$ | — | 2 | — |
| 8 | Erase pulse time | $t_{epk}$ | — | $t_{ei} \bullet k$ | ms |
| 9 | Amount to increment $t_{ep}$ | $t_{ei}$ | 90 | 110 | ms |
| 10 | Erase margin | $e_m$ | — | $\sum_{k=1}^{n_{ep}} t_{ei} \bullet k$ | ms |
| 11 | Erase recovery time | $t_{er}$ | — | 1 | ms |
| 12 | Low-power stop recovery time[5] | $t_{sb}$ | — | 1 | µs |

NOTES:

1. $V_{FPE}$ must not be raised to programming voltage while $V_{DD}$ is below specified minimum value. $V_{FPE}$ must not be reduced below minimum specified value while $V_{DD}$ is applied.
2. Current parameters apply to each individual EEPROM module.
3. Without margin.
4. At 100% margin, the number of margin pulses required is the same as the number of pulses used to program the byte or word.
5. Parameter measured from end of write cycle that clears STOP bit in FEEMCR.

A

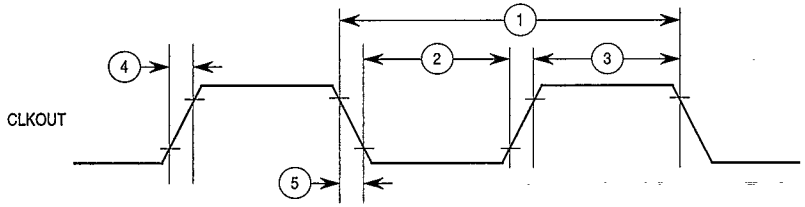
## Table A–15. Flash EEPROM Life

| Num | Parameter | Symbol | Value | Unit |
|---|---|---|---|---|
| 1 | Program-erase endurance[1] | $e_{pe}$ | 100 | cyc |
| 2 | Data retention[2] | $r_d$ | 10 | yr |

NOTES:

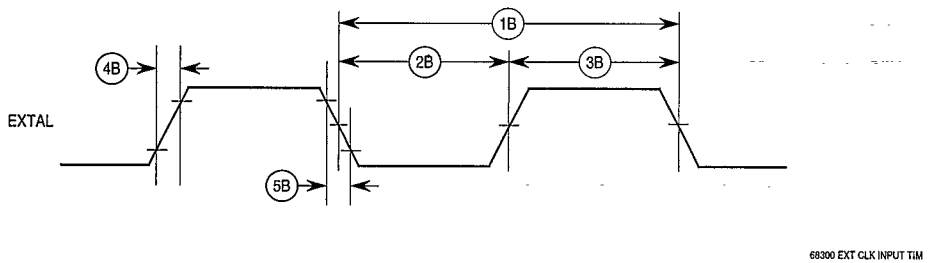
1. Number of program-erase cycles (1 to 0, 0 to 1) per bit.
2. Parameter based on accelerated-life testing with standard test pattern.

## Timing Diagrams




NOTE: Timing shown with respect to 20% and 70% $V_{DD}$.

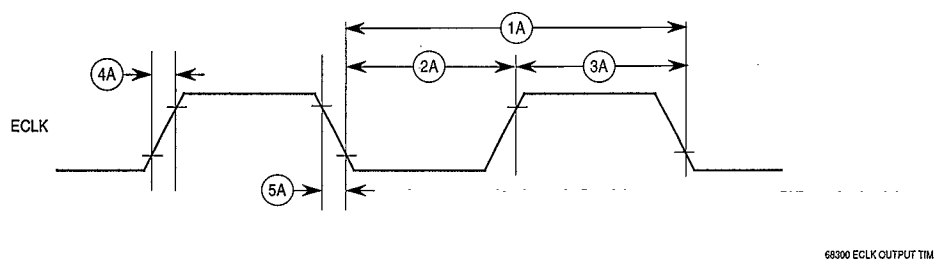
**Figure A–1. CLKOUT Output Timing Diagram**




NOTE: Timing shown with respect to 20% and 70% $V_{DD}$. Pulse width shown with respect to 50% $V_{DD}$.

**Figure A–2. External Clock Input Timing Diagram**

A




NOTE: Timing shown with respect to 20% and 70% $V_{DD}$.

**Figure A–3. ECLK Output Timing Diagram**

**Key to Figures A–1, A–2, A–3**

(Abstracted from Table A–6; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | Clock Period | $t_{cyc}$ | 59.6 | — | ns |
| 1A | ECLK Period | $t_{Ecyc}$ | 476 | — | ns |
| 1B | External Clock Input Period[3] | $t_{Xcyc}$ | 59.6 | — | ns |
| 2, 3 | Clock Pulse Width | $t_{CW}$ | 24 | — | ns |
| 2A, 3A | ECLK Pulse Width | $t_{ECW}$ | 236 | — | ns |
| 2B, 3B | External Clock Input High/Low Time[3] | $t_{XCHL}$ | 29.8 | — | ns |
| 4, 5 | Clock Rise and Fall Time | $t_{Crf}$ | — | 5 | ns |
| 4A, 5A | Rise and Fall Time — All outputs except CLKOUT | $t_{rf}$ | — | 8 | ns |
| 4B, 5B | External Clock Rise and Fall Time[4] | $t_{XCrf}$ | — | 5 | ns |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
3. 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 signal 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).
4. Parameters for an external clock signal applied while the internal PLL is disabled (MODCLK pin held low during reset). Does not pertain to an external VCO reference applied while the PLL is enabled (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.

A

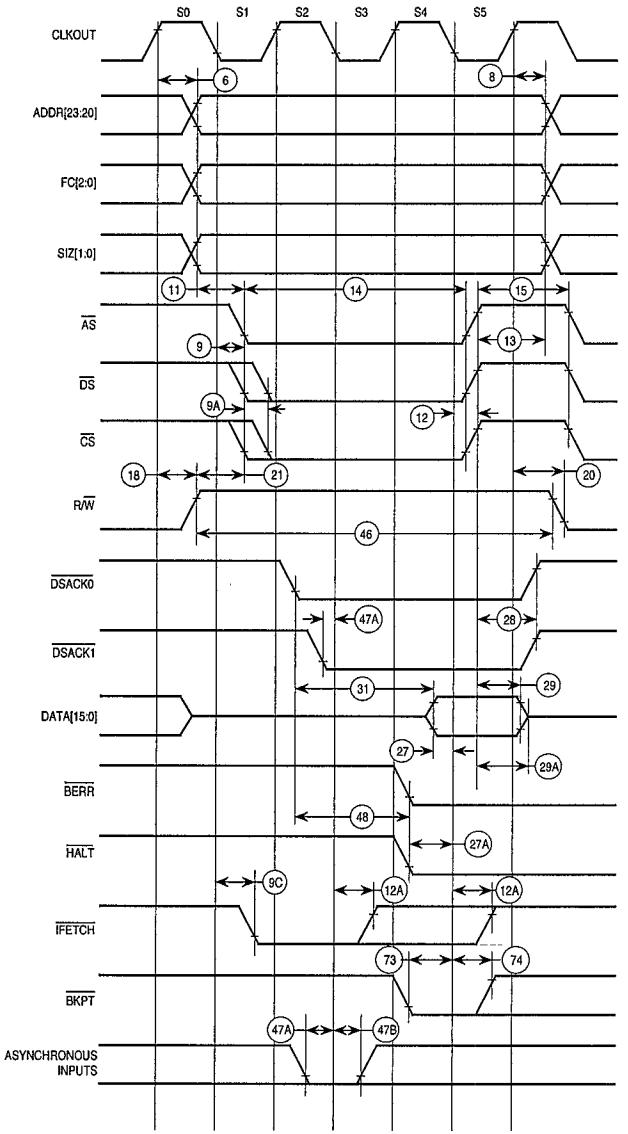



**Figure A–4. Read Cycle Timing Diagram**

## Key to Figure A–4

(Abstracted from Table A–6; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 6 | Clock High to ADDR, FC, SIZE, RMC Valid | $t_{CHAV}$ | 0 | 29 | ns |
| 8 | Clock High to ADDR, FC, SIZE RMC Invalid | $t_{CHAZn}$ | 0 | — | ns |
| 9 | Clock Low to AS, DS, CS Asserted | $t_{CLSA}$ | 2 | 25 | ns |
| 9A | AS to DS or CS Asserted (Read)[5] | $t_{STSA}$ | –15 | 15 | ns |
| 11 | ADDR, FC, SIZE, RMC Valid to AS, CS (and DS Read) Asserted | $t_{AVSA}$ | 15 | — | ns |
| 12 | Clock Low to AS, DS, CS Negated | $t_{CLSN}$ | 2 | 29 | ns |
| 13 | AS, DS, CS Negated to ADDR, FC, SIZE Invalid (Address Hold) | $t_{SNAI}$ | 15 | — | ns |
| 14 | AS, CS (and DS Read) Width Asserted | $t_{SWA}$ | 100 | — | ns |
| 15 | AS, DS, CS Width Negated[6] | $t_{SN}$ | 40 | — | ns |
| 18 | Clock High to R/W High | $t_{CHRH}$ | 0 | 29 | ns |
| 20 | Clock High to R/W Low | $t_{CHRL}$ | 0 | 29 | ns |
| 21 | R/W High to AS, CS Asserted | $t_{RAAA}$ | 15 | — | ns |
| 27 | Data In Valid to Clock Low (Data Setup) | $t_{DICL}$ | 5 | — | ns |
| 27A | Late BERR, HALT Asserted to Clock Low (Setup Time) | $t_{BELCL}$ | 20 | — | ns |
| 28 | AS, DS Negated to DSACK[1:0], BERR, HALT, AVEC Negated | $t_{SNDN}$ | 0 | 80 | ns |
| 29 | DS, CS Negated to Data In Invalid (Data In Hold)[7] | $t_{SNDI}$ | 0 | — | ns |
| 29A | DS, CS Negated to Data In High Impedance[7,8] | $t_{SHDI}$ | — | 55 | ns |
| 31 | DSACK[1:0] Asserted to Data In Valid[9] | $t_{DADI}$ | — | 50 | ns |
| 46 | R/W Width Asserted (Write or Read) | $t_{RWA}$ | 150 | — | ns |
| 47A | Asynchronous Input Setup Time BR, BGACK, DSACK[1:0], BERR, AVEC, HALT | $t_{AIST}$ | 5 | — | ns |
| 47B | Asynchronous Input Hold Time | $t_{AIHT}$ | 15 | — | ns |
| 48 | DSACK[1:0] Asserted to BERR, HALT Asserted[11] | $t_{DABA}$ | — | 30 | ns |
| 73 | BKPT Input Setup Time | $t_{BKST}$ | 15 | — | ns |
| 74 | BKPT Input Hold Time | $t_{BKHT}$ | 10 | — | ns |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
5. Specification 9A is the worst-case skew between AS and DS or CS. The amount of skew depends on the relative loading of these signals. When loads are kept within specified limits, skew will not cause AS and DS to fall outside the limits shown in specification 9.
6. If multiple chip selects are used, CS width negated (specification 15) applies to the time from the negation of a heavily loaded chip select to the assertion of a lightly loaded chip select. The CS width negated specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
7. Hold times are specified with respect to DS or CS on asynchronous reads and with respect to CLKOUT on fast reads. The user is free to use either hold time.
8. Maximum value is equal to ($t_{cyc}$ / 2) + 25 ns.
9. If the asynchronous setup time (specification 47A) requirements are satisfied, the DSACK[1:0] low to data setup time (specification 31) and DSACK[1:0] low to BERR low setup time (specification 48) can be ignored. The data must only satisfy the data-in to clock low setup time (specification 27) for the following clock cycle. BERR must satisfy only the late BERR low to clock low setup time (specification 27A) for the following clock cycle.
11. In the absence of DSACK[1:0], BERR is an asynchronous input — use specification 47A.
14. 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.

A

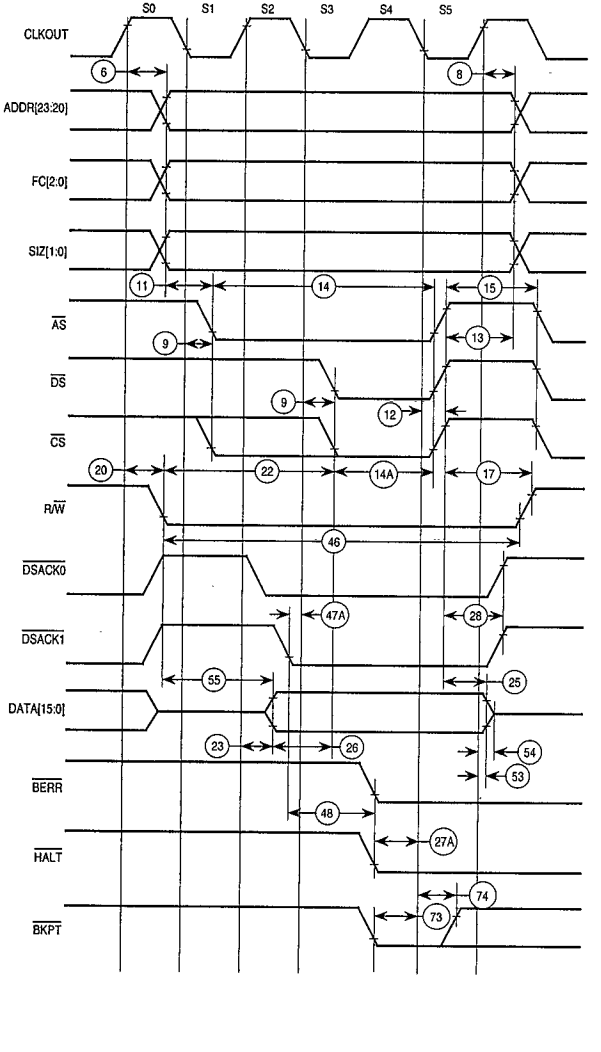



**Figure A–5. Write Cycle Timing Diagram**

**Key to Figure A–5**

(Abstracted from Table A–6; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 6 | Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Valid | $t_{CHAV}$ | 0 | 29 | ns |
| 8 | Clock High to ADDR, FC, SIZE $\overline{RMC}$ Invalid | $t_{CHAZn}$ | 0 | — | ns |
| 9 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted | $t_{CLSA}$ | 2 | 25 | ns |
| 11 | ADDR, FC, SIZE, $\overline{RMC}$ Valid to $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Asserted | $t_{AVSA}$ | 15 | — | ns |
| 12 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated | $t_{CLSN}$ | 2 | 29 | ns |
| 13 | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to ADDR, FC, SIZE Invalid (Address Hold) | $t_{SNAI}$ | 15 | — | ns |
| 14 | $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted | $t_{SWA}$ | 100 | — | ns |
| 14A | $\overline{DS}$, $\overline{CS}$ Width Asserted (Write) | $t_{SWAW}$ | 45 | — | ns |
| 15 | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated[6] | $t_{SN}$ | 40 | — | ns |
| 17 | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to R/$\overline{W}$ High | $t_{SNRN}$ | 15 | — | ns |
| 20 | Clock High to R/$\overline{W}$ Low | $t_{CHRL}$ | 0 | 29 | ns |
| 22 | R/$\overline{W}$ Low to $\overline{DS}$, $\overline{CS}$ Asserted (Write) | $t_{RASA}$ | 70 | — | ns |
| 23 | Clock High to Data Out Valid | $t_{CHDO}$ | — | 29 | ns |
| 25 | $\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold) | $t_{SNDOI}$ | 15 | — | ns |
| 26 | Data Out Valid to $\overline{DS}$, $\overline{CS}$ Asserted (Write) | $t_{DVSA}$ | 15 | — | ns |
| 27A | Late $\overline{BERR}$, $\overline{HALT}$ Asserted to Clock Low (Setup Time) | $t_{BELCL}$ | 20 | — | ns |
| 28 | $\overline{AS}$, $\overline{DS}$ Negated to $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{HALT}$, $\overline{AVEC}$ Negated | $t_{SNDN}$ | 0 | 80 | ns |
| 46 | R/$\overline{W}$ Width Asserted (Write or Read) | $t_{RWA}$ | 150 | — | ns |
| 47A | Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$ | $t_{AIST}$ | 5 | — | ns |
| 48 | $\overline{DSACK[1:0]}$ Asserted to $\overline{BERR}$, $\overline{HALT}$ Asserted[11] | $t_{DABA}$ | — | 30 | ns |
| 53 | Data Out Hold from Clock High | $t_{DOCH}$ | 0 | — | ns |
| 54 | Clock High to Data Out High Impedance | $t_{CHDH}$ | — | 28 | ns |
| 73 | $\overline{BKPT}$ Input Setup Time | $t_{BKST}$ | 15 | — | ns |
| 74 | $\overline{BKPT}$ Input Hold Time | $t_{BKHT}$ | 10 | — | ns |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
6. If multiple chip selects are used, $\overline{CS}$ width negated (specification 15) 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 specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
11. In the absence of $\overline{DSACK[1:0]}$, $\overline{BERR}$ is an asynchronous input — use specification 47A.
14. 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.

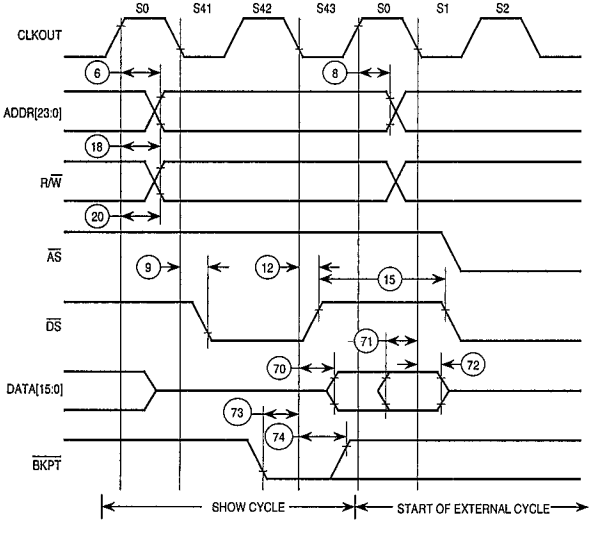



**Figure A–6. Show Cycle Timing Diagram**

A

**Key to Figure A–6**

(Abstracted from Table A–6; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 6 | Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Valid | $t_{CHAV}$ | 0 | 29 | ns |
| 8 | Clock High to ADDR, FC, SIZE $\overline{RMC}$ Invalid | $t_{CHAZn}$ | 0 | — | ns |
| 9 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted | $t_{CLSA}$ | 2 | 25 | ns |
| 12 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated | $t_{CLSN}$ | 2 | 29 | ns |
| 15 | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated[6] | $t_{SN}$ | 40 | — | ns |
| 18 | Clock High to R/$\overline{W}$ High | $t_{CHRH}$ | 0 | 29 | ns |
| 20 | Clock High to R/$\overline{W}$ Low | $t_{CHRL}$ | 0 | 29 | ns |
| 70 | Clock Low to Data Bus Driven (Show) | $t_{SCLDD}$ | 0 | 29 | ns |
| 71 | Data Setup Time to Clock Low (Show) | $t_{SCLDS}$ | 15 | — | ns |
| 72 | Data Hold from Clock Low (Show) | $t_{SCLDH}$ | 10 | — | ns |
| 73 | $\overline{BKPT}$ Input Setup Time | $t_{BKST}$ | 15 | — | ns |
| 74 | $\overline{BKPT}$ Input Hold Time | $t_{BKHT}$ | 10 | — | ns |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
6. If multiple chip selects are used, $\overline{CS}$ width negated (specification 15) 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 specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
14. 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.

A

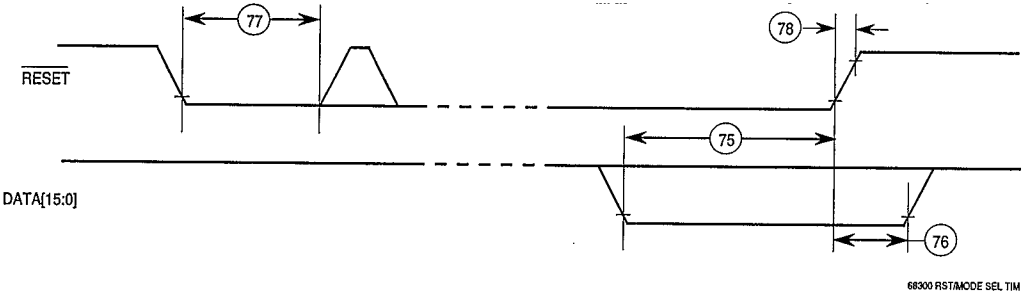



**Figure A–7. Reset and Mode Select Timing Diagram**

A

**Key to Figure A–7**

(Abstracted from Table A–6; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 75 | Mode Select Setup Time | $t_{MSS}$ | 20 | — | $t_{cyc}$ |
| 76 | Mode Select Hold Time | $t_{MSH}$ | 0 | — | ns |
| 77 | $\overline{\text{RESET}}$ Assertion Time[12] | $t_{RSTA}$ | 4 | — | $t_{cyc}$ |
| 78 | $\overline{\text{RESET}}$ Rise Time[12, 13] | $t_{RSTR}$ | — | 10 | $t_{cyc}$ |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
12. After external $\overline{\text{RESET}}$ negation is detected, a short transition period (approximately 2 $t_{cyc}$) elapses, then the SCIM drives $\overline{\text{RESET}}$ low for 512 $t_{cyc}$.
13. External logic must pull $\overline{\text{RESET}}$ high during this period in order for normal MCU operation to begin.

A

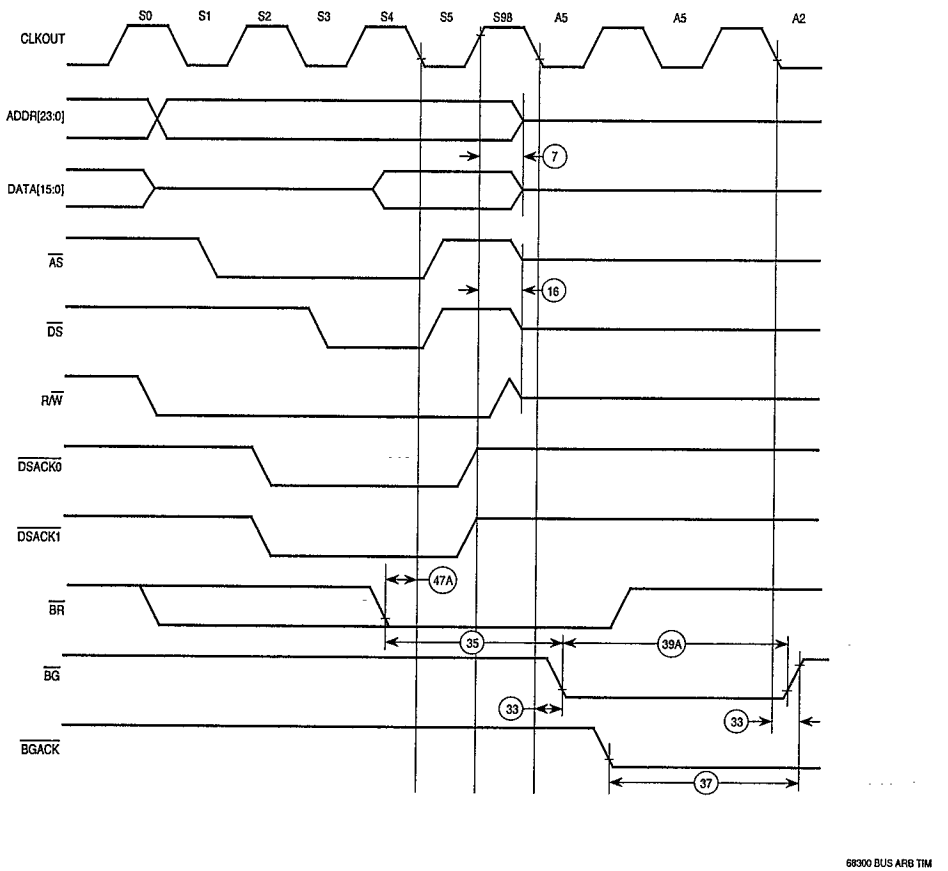



**Figure A–8. Bus Arbitration Timing Diagram — Active Bus Case**

**Key to Figure A–8**
(Abstracted from Table A–6; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 7 | Clock High to ADDR, DATA, FC, SIZE, $\overline{RMC}$ High Impedance | $t_{CHAZx}$ | 0 | 59 | ns |
| 16 | Clock High to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ High Impedance | $t_{CHSZ}$ | — | 59 | ns |
| 33 | Clock Low to $\overline{BG}$ Asserted/Negated | $t_{CLBAN}$ | — | 29 | ns |
| 35 | $\overline{BR}$ Asserted to $\overline{BG}$ Asserted [10] | $t_{BRAGA}$ | 1 | — | $t_{cyc}$ |
| 37 | $\overline{BGACK}$ Asserted to $\overline{BG}$ Negated | $t_{GAGN}$ | 1 | 2 | $t_{cyc}$ |
| 39A | $\overline{BG}$ Width Asserted | $t_{GA}$ | 1 | — | $t_{cyc}$ |
| 47A | Asynchronous Input Setup Time $\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$ | $t_{AIST}$ | 5 | — | ns |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
10. 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 and RMC is asserted.

A

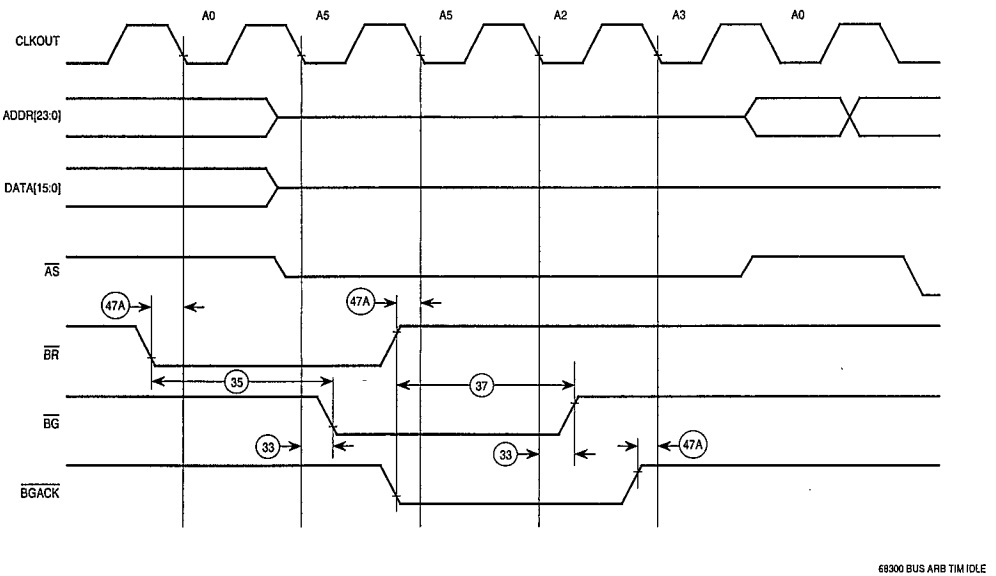



**Figure A–9. Bus Arbitration Timing Diagram — Idle Bus Case**

A

**Key to Figure A–9**

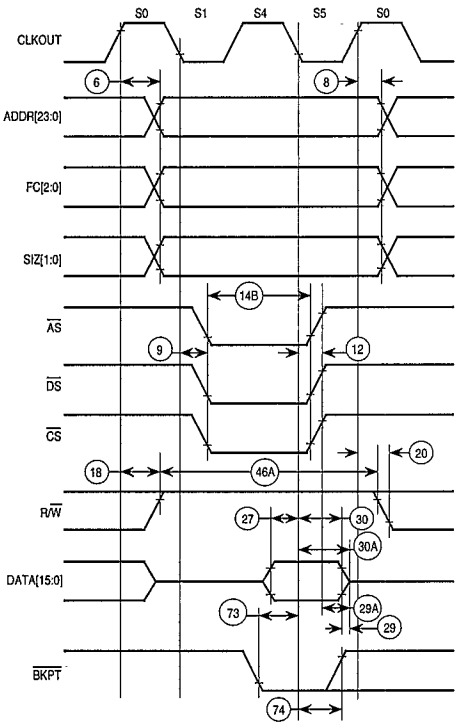
(Abstracted from Table A–6; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 33 | Clock Low to $\overline{BG}$ Asserted/Negated | $t_{CLBAN}$ | — | 29 | ns |
| 35 | $\overline{BR}$ Asserted to $\overline{BG}$ Asserted ($\overline{RMC}$ Not Asserted)[10] | $t_{BRAGA}$ | 1 | — | $t_{cyc}$ |
| 37 | $\overline{BGACK}$ Asserted to $\overline{BG}$ Negated | $t_{GAGN}$ | 1 | 2 | $t_{cyc}$ |
| 47A | Asynchronous Input Setup Time<br>$\overline{BR}$, $\overline{BGACK}$, $\overline{DSACK[1:0]}$, $\overline{BERR}$, $\overline{AVEC}$, $\overline{HALT}$ | $t_{AIST}$ | 5 | — | ns |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
10. 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 and $\overline{RMC}$ is asserted.

A




A

**Figure A–10. Fast Termination Read Cycle Timing Diagram**

**Key to Figure A–10**

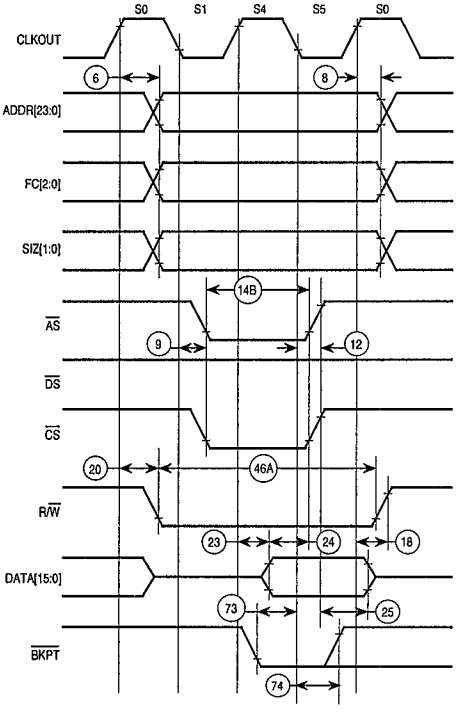
(Abstracted from Table A–6; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 6 | Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Valid | $t_{CHAV}$ | 0 | 29 | ns |
| 8 | Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Invalid | $t_{CHAZn}$ | 0 | — | ns |
| 9 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted | $t_{CLSA}$ | 2 | 25 | ns |
| 12 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated | $t_{CLSN}$ | 2 | 29 | ns |
| 14B | $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted (Fast Write Cycle) | $t_{SWDW}$ | 40 | — | ns |
| 18 | Clock High to R/$\overline{W}$ High | $t_{CHRH}$ | 0 | 29 | ns |
| 20 | Clock High to R/$\overline{W}$ Low | $t_{CHRL}$ | 0 | 29 | ns |
| 27 | Data In Valid to Clock Low (Data Setup) | $t_{DICL}$ | 5 | — | ns |
| 29 | $\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)[7] | $t_{SNDI}$ | 0 | — | ns |
| 29A | $\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance[7,8] | $t_{SHDI}$ | — | 55 | ns |
| 30 | CLKOUT Low to Data In Invalid (Fast Cycle Hold)[7] | $t_{CLDI}$ | 15 | — | ns |
| 30A | CLKOUT Low to Data In High-Impedance[7] | $t_{CLDH}$ | — | 90 | ns |
| 46A | R/$\overline{W}$ Width Asserted (Fast Write or Read Cycle) | $t_{RWAS}$ | 90 | — | ns |
| 73 | $\overline{BKPT}$ Input Setup Time | $t_{BKST}$ | 15 | — | ns |
| 74 | $\overline{BKPT}$ Input Hold Time | $t_{BKHT}$ | 10 | — | ns |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
7. 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.
8. Maximum value is equal to ($t_{cyc}$ / 2) + 25 ns.
14. 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.

A




A

**Figure A–11. Fast Termination Write Cycle Timing Diagram**

## Key to Figure A–11

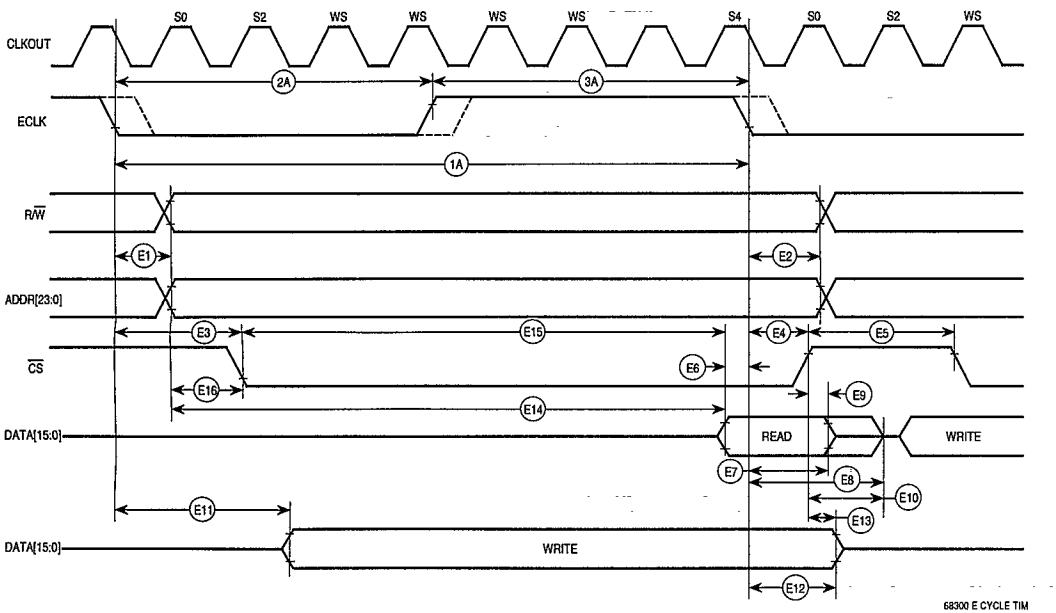
(Abstracted from Table A–6; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 6 | Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Valid | $t_{CHAV}$ | 0 | 29 | ns |
| 8 | Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Invalid | $t_{CHAZn}$ | 0 | — | ns |
| 9 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted | $t_{CLSA}$ | 2 | 25 | ns |
| 12 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated | $t_{CLSN}$ | 2 | 29 | ns |
| 14B | $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted (Fast Write Cycle) | $t_{SWDW}$ | 40 | — | ns |
| 18 | Clock High to R/$\overline{W}$ High | $t_{CHRH}$ | 0 | 29 | ns |
| 20 | Clock High to R/$\overline{W}$ Low | $t_{CHRL}$ | 0 | 29 | ns |
| 23 | Clock High to Data Out Valid | $t_{CHDO}$ | — | 29 | ns |
| 24 | Data Out Valid to Negating Edge of $\overline{AS}$, $\overline{CS}$ (Fast Write Cycle) | $t_{DVASN}$ | 15 | — | ns |
| 25 | $\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold) | $t_{SNDOI}$ | 15 | — | ns |
| 46A | R/$\overline{W}$ Width Asserted (Fast Write or Read Cycle) | $t_{RWAS}$ | 90 | — | ns |
| 73 | $\overline{BKPT}$ Input Setup Time | $t_{BKST}$ | 15 | — | ns |
| 74 | $\overline{BKPT}$ Input Hold Time | $t_{BKHT}$ | 10 | — | ns |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
14. 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.

A




NOTE: Shown with ECLK = system clock/8 — EDIV bit in clock synthesizer control register (SYNCR) = 0.

**Figure A–12. ECLK Timing Diagram**

A

## Key to Figure A–12

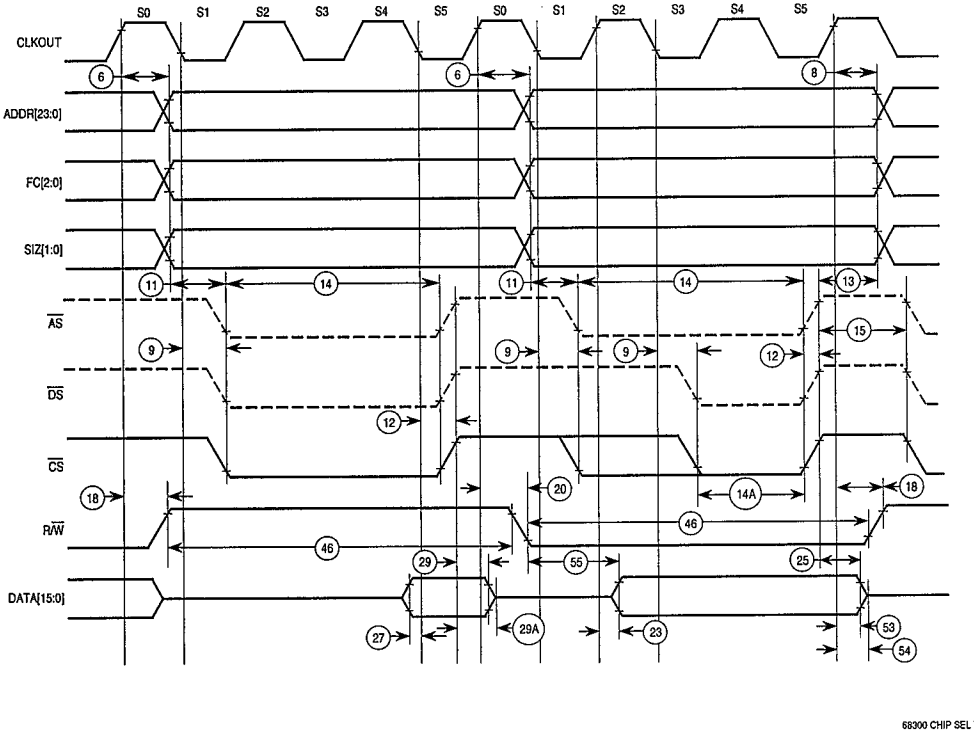
(Abstracted from Tables A–6 and A–8; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1A | ECLK Period | $t_{Ecyc}$ | 476 | — | ns |
| 2A, 3A | ECLK Pulse Width | $t_{ECW}$ | 236 | — | ns |
| 4A, 5A | Rise and Fall Time — All Outputs except CLKOUT | $t_{rf}$ | — | 8 | ns |
| 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 = $t_{Ecyc} - t_{ECSD} - t_{EDSR}$

A




NOTE: $\overline{AS}$ and $\overline{DS}$ timing shown for reference only.

**Figure A–13. Chip Select Timing Diagram**

**Key to Figure A–13**

(Abstracted from Table A–6; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 6 | Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Valid | $t_{CHAV}$ | 0 | 29 | ns |
| 8 | Clock High to ADDR, FC, SIZE, $\overline{RMC}$ Invalid | $t_{CHAZn}$ | 0 | — | ns |
| 9 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Asserted | $t_{CLSA}$ | 2 | 25 | ns |
| 11 | ADDR, FC, SIZE, $\overline{RMC}$ Valid to $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Asserted | $t_{AVSA}$ | 15 | — | ns |
| 12 | Clock Low to $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated | $t_{CLSN}$ | 2 | 29 | ns |
| 13 | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Negated to ADDR, FC, SIZE Invalid (Address Hold) | $t_{SNAI}$ | 15 | — | ns |
| 14 | $\overline{AS}$, $\overline{CS}$ (and $\overline{DS}$ Read) Width Asserted | $t_{SWA}$ | 100 | — | ns |
| 14A | $\overline{DS}$, $\overline{CS}$ Width Asserted (Write) | $t_{SWAW}$ | 45 | — | ns |
| 15 | $\overline{AS}$, $\overline{DS}$, $\overline{CS}$ Width Negated[6] | $t_{SN}$ | 40 | — | ns |
| 18 | Clock High to R/$\overline{W}$ High | $t_{CHRH}$ | 0 | 29 | ns |
| 20 | Clock High to R/$\overline{W}$ Low | $t_{CHRL}$ | 0 | 29 | ns |
| 23 | Clock High to Data Out Valid | $t_{CHDO}$ | — | 29 | ns |
| 25 | $\overline{DS}$, $\overline{CS}$ Negated to Data Out Invalid (Data Out Hold) | $t_{SNDOI}$ | 15 | — | ns |
| 29 | $\overline{DS}$, $\overline{CS}$ Negated to Data In Invalid (Data In Hold)[7] | $t_{SNDI}$ | 0 | — | ns |
| 29A | $\overline{DS}$, $\overline{CS}$ Negated to Data In High Impedance[7,8] | $t_{SHDI}$ | — | 55 | ns |
| 46 | R/$\overline{W}$ Width Asserted (Write or Read) | $t_{RWA}$ | 150 | — | ns |
| 53 | Data Out Hold from Clock High | $t_{DOCH}$ | 0 | — | ns |
| 54 | Clock High to Data Out High Impedance | $t_{CHDH}$ | — | 28 | ns |
| 55 | R/$\overline{W}$ Asserted to Data Bus Impedance Change | $t_{RADC}$ | 40 | — | ns |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.
6. If multiple chip selects are used, $\overline{CS}$ width negated (specification 15) 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 specification between multiple chip selects does not apply to chip selects being used for synchronous ECLK cycles.
7. 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.
8. Maximum value is equal to ($t_{cyc}$ / 2) + 25 ns.

A

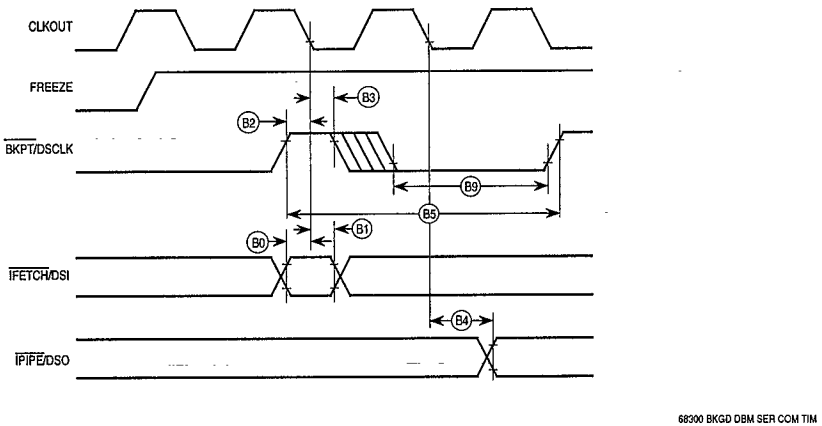



**Figure A–14. Background Debugging Mode Timing Diagram — Serial Communication**

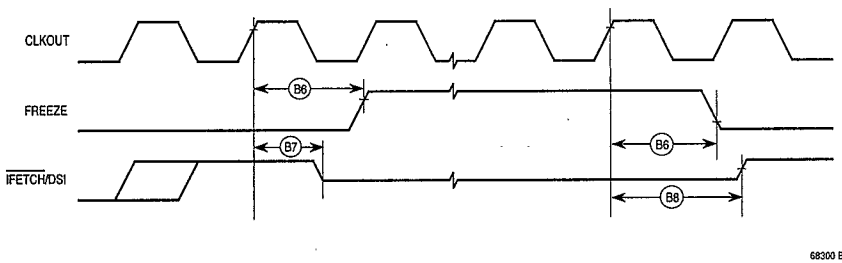



**Figure A–15. Background Debugging Mode Timing Diagram — Freeze Assertion**

**Key to Figures A–14 and A–15**

(Abstracted from Table A–7; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| B0 | DSI Input Setup Time | $t_{DSISU}$ | 15 | — | ns |
| B1 | DSI Input Hold Time | $t_{DSIH}$ | 10 | — | ns |
| B2 | DSCLK Setup Time | $t_{DSCSU}$ | 15 | — | ns |
| B3 | DSCLK Hold Time | $t_{DSCH}$ | 10 | — | ns |
| B4 | DSO Delay Time | $t_{DSOD}$ | — | 25 | ns |
| B5 | DSCLK Cycle Time | $t_{DSCCYC}$ | 2 | — | $t_{cyc}$ |
| B6 | CLKOUT Low to FREEZE Asserted/Negated | $t_{FRZAN}$ | — | 50 | ns |
| B7 | CLKOUT High to $\overline{\text{IFETCH}}$ High Impedance | $t_{IFZ}$ | — | 50 | ns |
| B8 | CLKOUT High to $\overline{\text{IFETCH}}$ Valid | $t_{IF}$ | — | 50 | ns |
| B9 | DSCLK Low Time | $t_{DSCLO}$ | 1 | — | $t_{cyc}$ |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.

A

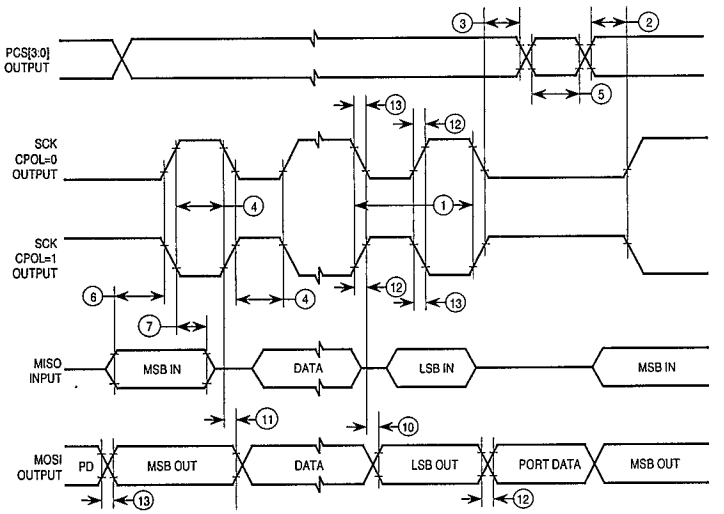



**Figure A–16. SPI Timing Master, CPHA = 0**

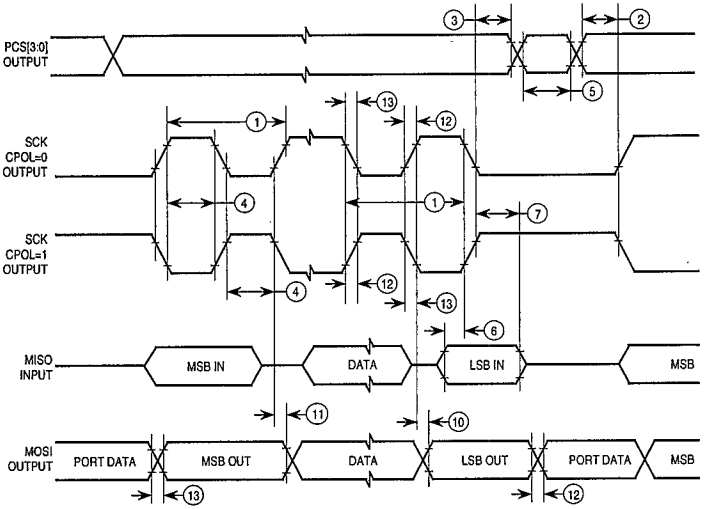



**Figure A–17. SPI Timing Master, CPHA = 1**

**Key to Figures A–16 and A–17**

(Abstracted from Table A–9; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | Master Cycle Time | $t_{qcyc}$ | 4 | 510 | $t_{cyc}$ |
| 4 | Master Clock (SCK) High or Low Time | $t_{sw}$ | 2 $t_{cyc}$ – 60 | 255 $t_{cyc}$ | ns |
| 5 | Master Sequential Transfer Delay | $t_{td}$ | 17 | 8192 | $t_{cyc}$ |
| 6 | Master Data Setup Time (Inputs) | $t_{su}$ | 30 | — | ns |
| 7 | Master Data Hold Time (Inputs) | $t_{hi}$ | 0 | — | ns |
| 10 | Master Data Valid (after SCK Edge) | $t_v$ | — | 50 | ns |
| 11 | Master Data Hold Time (Outputs) | $t_{ho}$ | 0 | — | ns |
| 12 | Rise Time<br>Input<br>Output | <br>$t_{ri}$<br>$t_{ro}$ | <br>—<br>— | <br>2<br>30 | <br>µs<br>ns |
| 13 | Fall Time<br>Input<br>Output | <br>$t_{fi}$<br>$t_{fo}$ | <br>—<br>— | <br>2<br>30 | <br>µs<br>ns |

NOTES:

1. All AC timing is shown with respect to 20% $V_{DD}$ and 70% $V_{DD}$ levels unless otherwise noted.

A

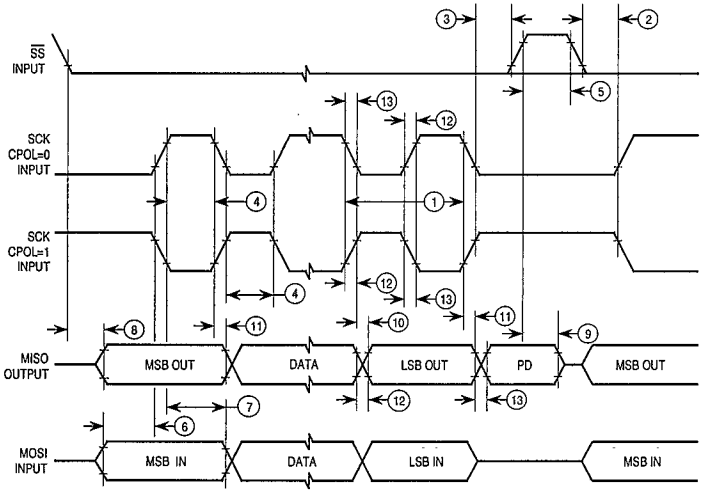



**Figure A–18. SPI Timing Slave, CPHA = 0**

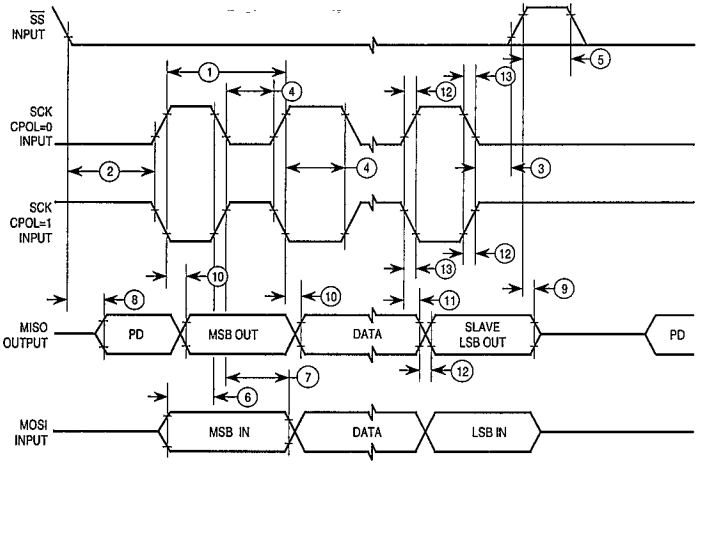



**Figure A–19. SPI Timing Slave, CPHA = 1**

**Key to Figures A–18 and A–19**

(Abstracted from Table A–9; see table for complete notes)

| Num | Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| 1 | Slave Cycle Time | $t_{qcyc}$ | 4 | — | $t_{cyc}$ |
| 2 | Slave Enable Lead Time | $t_{lead}$ | 2 | — | $t_{cyc}$ |
| 3 | Slave Enable Lag Time | $t_{lag}$ | 2 | — | $t_{cyc}$ |
| 4 | Slave Clock (SCK) High or Low Time[2] | $t_{sw}$ | 2 $t_{cyc}$ – n | — | ns |
| 5 | Slave Sequential Transfer Delay (Does Not Require Deselect) | $t_{td}$ | 13 | — | $t_{cyc}$ |
| 6 | Slave Data Setup Time (Inputs) | $t_{su}$ | 20 | — | ns |
| 7 | Slave Data Hold Time (Inputs) | $t_{hi}$ | 20 | — | ns |
| 8 | Slave Access Time | $t_a$ | — | 1 | $t_{cyc}$ |
| 9 | Slave MISO Disable Time | $t_{dis}$ | — | 2 | $t_{cyc}$ |
| 10 | Slave Data Valid (after SCK Edge) | $t_v$ | — | 50 | ns |
| 11 | Slave Data Hold Time (Outputs) | $t_{ho}$ | 0 | — | ns |
| 12 | Rise Time<br>Input<br>Output | <br>$t_{ri}$<br>$t_{ro}$ | <br>—<br>— | <br>2<br>30 | <br>µs<br>ns |
| 13 | Fall Time<br>Input<br>Output | <br>$t_{fi}$<br>$t_{fo}$ | <br>—<br>— | <br>2<br>30 | <br>µs<br>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; for low time, n = External SCK fall time

A

A

# APPENDIX B
# MECHANICAL DATA AND ORDERING INFORMATION

The MC68F333 is available in a 160-pin plastic surface mount package. It can be ordered in a molded carrier ring. This appendix provides package pin assignment drawings, dimensional drawings, and ordering information.

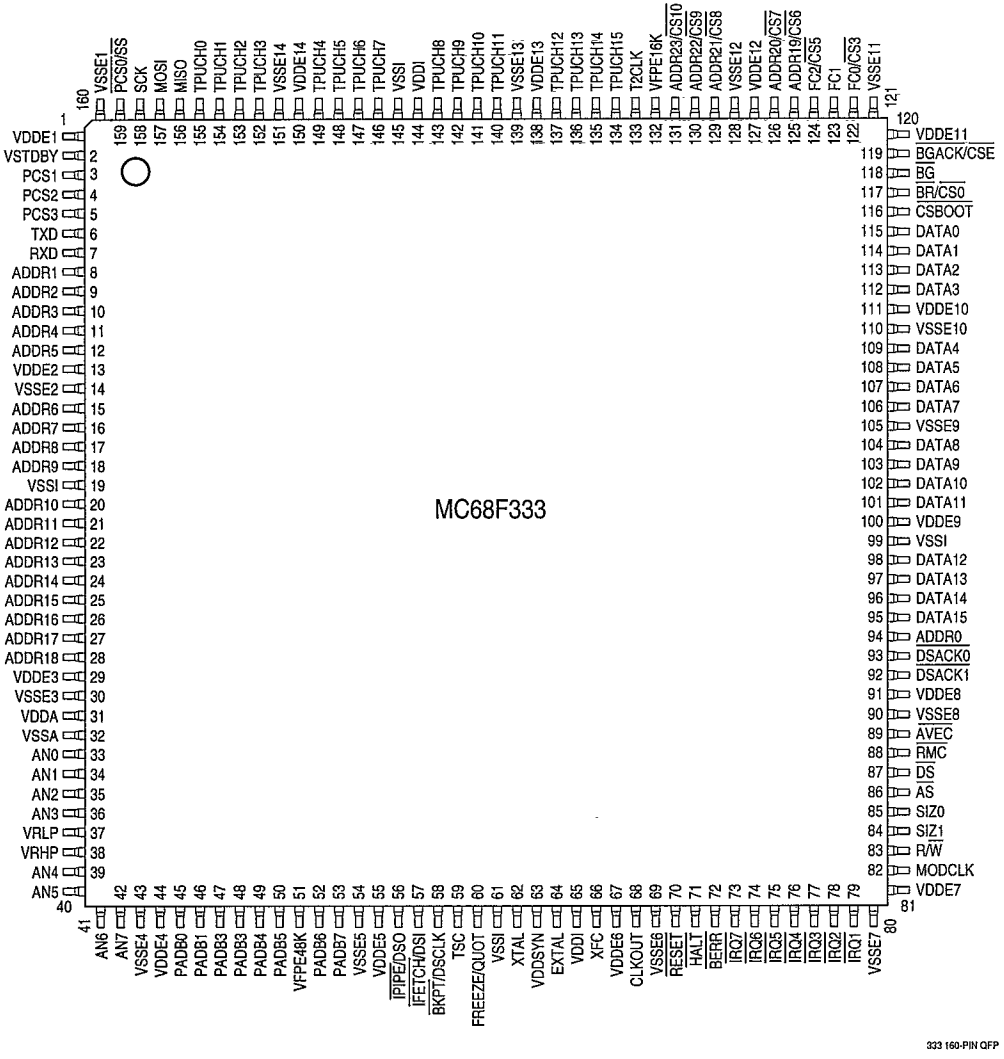
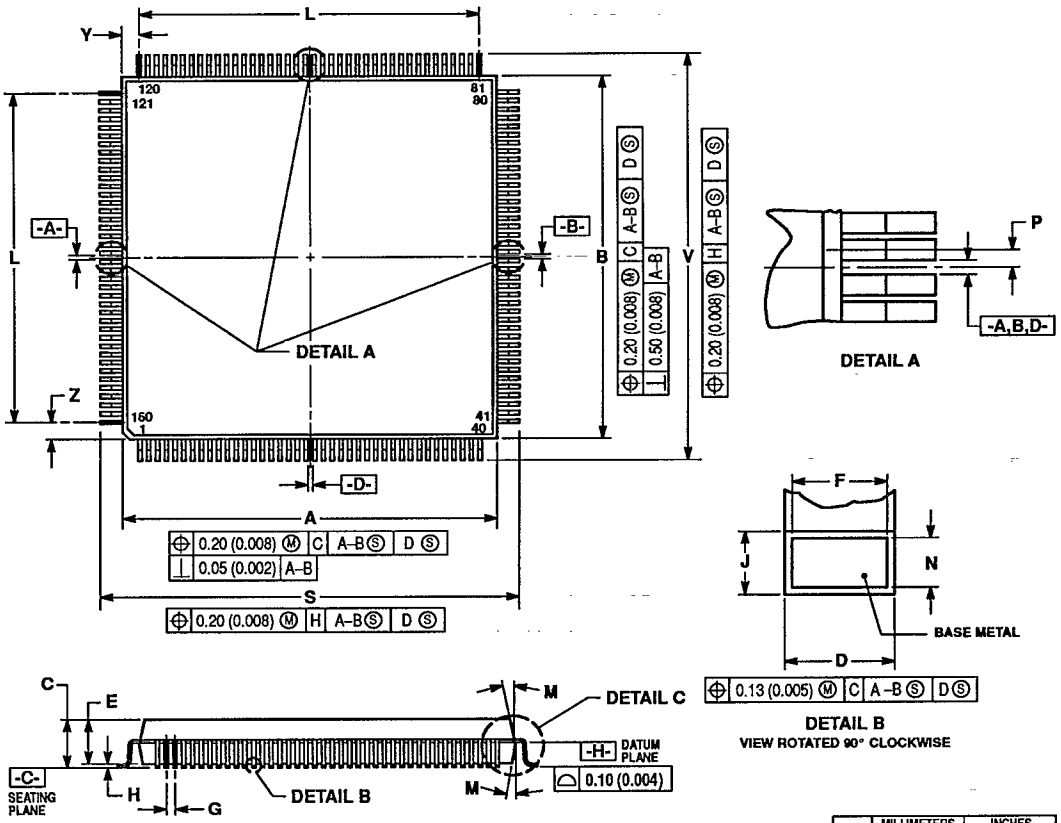



**Figure B–1. 160-Pin Plastic Surface Mount Package Pin Assignments**

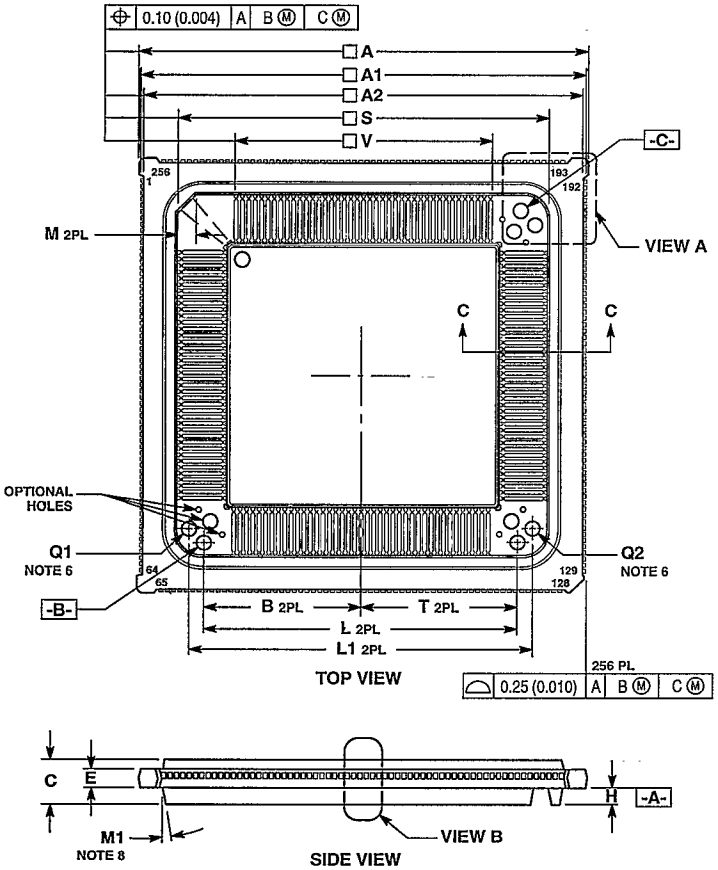



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE -H- IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS -A-, -B- AND -D- TO BE DETERMINED AT DATUM PLANE -H-.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE -C-.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 (0.010) PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE -H-.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 (0.003) TOTAL IN EXCESS OF THE D DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.

| | MILLIMETERS | | INCHES | |
|---|---|---|---|---|
| DIM | MIN | MAX | MIN | MAX |
| A | 27.90 | 28.10 | 1.098 | 1.106 |
| B | 27.90 | 28.10 | 1.098 | 1.106 |
| C | 3.35 | 3.85 | 0.132 | 0.152 |
| D | 0.22 | 0.38 | 0.009 | 0.015 |
| E | 3.20 | 3.50 | 0.126 | 0.138 |
| F | 0.22 | 0.33 | 0.009 | 0.013 |
| G | 0.650 BSC | | 0.0256 BSC | |
| H | 0.25 | 0.35 | 0.010 | 0.012 |
| J | 0.11 | 0.23 | 0.004 | 0.009 |
| K | 0.70 | 0.90 | 0.028 | 0.035 |
| L | 25.35 REF | | 0.998 REF | |
| M | 5° | 16° | 5° | 16° |
| N | 0.11 | 0.19 | 0.004 | 0.007 |
| P | 0.325 BSC | | 0.0130 BSC | |
| Q | 0° | 7° | 0° | 7° |
| R | 0.13 | 0.30 | 0.005 | 0.012 |
| S | 31.00 | 31.40 | 1.220 | 1.236 |
| T | 0.13 | — | 0.005 | — |
| U | 0° | — | 0° | — |
| V | 31.00 | 31.40 | 1.220 | 1.236 |
| W | 0.40 | — | 0.016 | — |
| X | 1.60 REF | | 0.063 REF | |
| Y | 1.33 REF | | 0.052 REF | |
| Z | 1.33 REF | | 0.052 REF | |

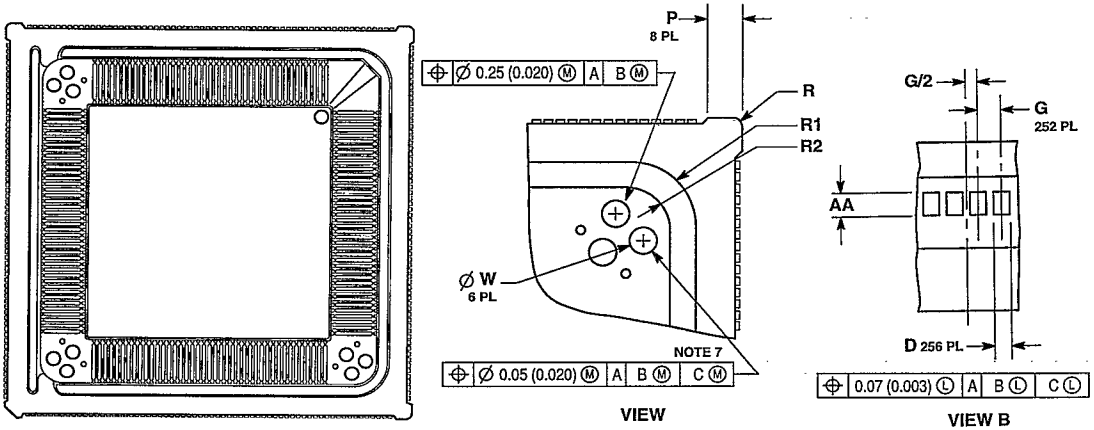
**CASE 864A-01**
**ISSUE B**

**Figure B–2. 160-Pin Package Dimensions**




NOTES:
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. A DIMENSION DOES NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION IS 0.20 (0.008) PER SIDE.
4. A AND S DIMENSIONS INCLUDE MOLD MISMATCH, AND ARE MEASURED AT THE PARTING LINE.
5. UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE SYMMETRICAL ABOUT CENTERLINES.
6. B AND C DATUM HOLES ARE TO BE USED FOR TRIM, FORM, AND EXCISE OF THE MOLDED PACKAGE ONLY. HOLES Q1 AND Q2 ARE TO BE USED FOR ELECTRICAL TESTING ONLY.
7. NON-DATUM HOLES ONLY.
8. APPLIES TO RING AND PACKAGE FEATURES.

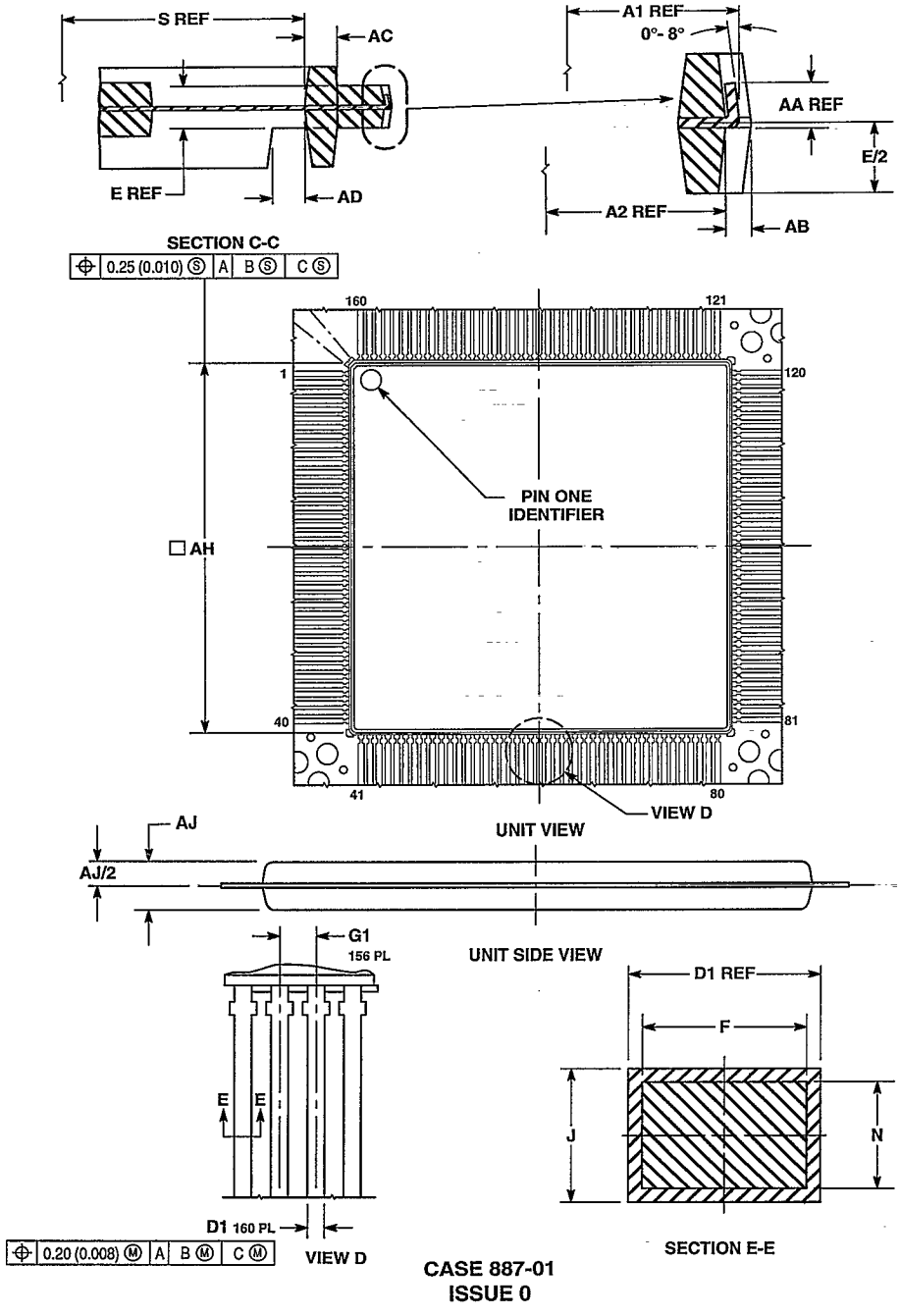
| | MILLIMETERS | | INCHES | |
|---|---|---|---|---|
| DIM | MIN | MAX | MIN | MAX |
| A | 45.87 | 46.13 | 1.806 | 1.816 |
| A1 | 45.70 BSC | | 1.799 BSC | |
| A2 | 45.17 | 45.43 | 1.778 | 1.789 |
| B | 16.10 BSC | | 0.634 BSC | |
| C | 4.70 | 4.90 | 0.185 | 0.193 |
| D | 0.40 | 0.50 | 0.016 | 0.020 |
| D1 | 0.22 | 0.38 | 0.009 | 0.015 |
| E | 1.90 | 2.10 | 0.075 | 0.083 |
| F | 0.22 | 0.33 | 0.009 | 0.013 |
| G | 0.65 BSC | | 0.026 BSC | |
| G1 | 0.65 BSC | | 0.026 BSC | |
| H | 1.70 | 1.90 | 0.067 | 0.075 |
| J | 0.13 | 0.23 | 0.005 | 0.009 |
| L | 32.20 BSC | | 1.268 BSC | |
| L1 | 35.20 BSC | | 1.386 BSC | |
| M | 1.30 | 2.30 | 0.051 | 0.091 |
| M1 | 8° MAX | | 8° MAX | |
| N | 0.13 | 0.17 | 0.005 | 0.007 |
| P | 1.77 | 2.03 | 0.070 | 0.080 |
| R | 0.40 | 0.60 | 0.016 | 0.024 |
| R1 | 3.50 | 4.50 | 0.138 | 0.177 |
| R2 | 2.00 | 3.00 | 0.079 | 0.118 |
| S | 37.87 | 38.13 | 1.491 | 1.501 |
| T | 16.10 BSC | | 0.634 BSC | |
| V | 26.30 | 26.40 | 1.035 | 1.039 |
| W | 1.45 | 1.55 | 0.057 | 0.061 |
| AA | 0.45 | 0.85 | 0.018 | 0.033 |
| AB | 0.30 | 0.60 | 0.012 | 0.024 |
| AC | 1.37 | 1.63 | 0.054 | 0.064 |
| AD | 1.37 | 1.63 | 0.054 | 0.064 |
| AH | 27.90 | 28.10 | 1.098 | 1.106 |
| AJ | 3.20 | 3.40 | 0.126 | 0.134 |




BOTTOM VIEW OF UNIT IN MOLDED CARRIER RING

CASE 887-01
ISSUE 0

**Figure B–3. 160-Pin Molded Carrier Ring Assembly (Part 1)**




CASE 887-01
ISSUE 0

**Figure B–3. 160-Pin Molded Carrier Ring Assembly (Part 2)**

B

## Table B–1. MC68F333 Ordering Information

| Package | Temperature | Frequency | Shipping Method | Order Number |
|---|---|---|---|---|
| 160-Pin Plastic Surface Mount | –40 to +85°C | 16.78 MHz | 24 pc tray | XC68F333CFT16 |
| | | | 2 pc tray | SPAKXCF333CFT16 |
| | | 20 MHz | 24 pc tray | XC68F333CFT20 |
| | | | 2 pc tray | SPAKXCF333CFT20 |
| | | 25 MHz | 24 pc tray | XC68F333CFT25 |
| | | | 2 pc tray | SPAKXCF333CFT25 |
| | –40 to +105°C | 16.78 MHz | 24 pc tray | XC68F333VFT16 |
| | | | 2 pc tray | SPAKXCF333VFT16 |
| | | 20 MHz | 24 pc tray | XC68F333VFT20 |
| | | | 2 pc tray | SPAKXCF333VFT20 |
| | | 25 MHz | 24 pc tray | XC68F333VFT25 |
| | | | 2 pc tray | SPAKXCF333VFT25 |
| | –40 to +125°C | 16.78 MHz | 24 pc tray | XC68F333MFT16 |
| | | | 2 pc tray | SPAKXCF333MFT16 |
| | | 20 MHz | 24 pc tray | XC68F333MFT20 |
| | | | 2 pc tray | SPAKXCF333MFT20 |
| | | 25 MHz | 24 pc tray | XC68F333MFT25 |
| | | | 2 pc tray | SPAKXCF333MFT25 |
| 160-Pin Molded Carrier Ring | –40 to +85°C | 16.78 MHz | 10/tube | XC68F333CFM16 |
| | | 20 MHz | 10/tube | XC68F333CFM20 |
| | | 25 MHz | 10/tube | XC68F333CFM25 |
| | –40 to +105°C | 16.78 MHz | 10/tube | XC68F333VFM16 |
| | | 20 MHz | 10/tube | XC68F333VFM20 |
| | | 25 MHz | 10/tube | XC68F333VFM25 |
| | –40 to +125°C | 16.78 MHz | 10/tube | XC68F333MFM16 |
| | | 20 MHz | 10/tube | XC68F333MFM20 |
| | | 25 MHz | 10/tube | XC68F333MFM25 |

# APPENDIX C
# DEVELOPMENT SUPPORT

This section serves as a brief reference to development support for the MC68F333. In addition to development tools developed by Motorola, a growing number of third-party tools are available. Motorola publication *MCU Tool Box* (MCUTLBX/D) provides an up-to-date list of development tools.

New development support systems for the MC68F333 are currently being produced. Please contact your Motorola representative for more information.

C

# APPENDIX D
# REGISTER SUMMARY

This appendix contains address maps, register diagrams, and bit/field definitions for the MC68F333. More detailed information about register function is provided in the appropriate sections of the manual.

Except for central processing unit resources, information is presented in the intermodule bus address order shown in Table D–1.

**Table D–1. MC68F333 Module Address Map**

| Module | Size | Base Address |
|---|---|---|
| ADC | 64 Bytes | $YFF700 |
| FLASH (16-Kbyte array) | 32 Bytes | $YFF800 |
| FLASH (48-Kbyte array) | 32 Bytes | $YFF820 |
| SCIM | 128 Bytes | $YFFA00 |
| TPURAM | 64 Bytes | $YFFB00 |
| SRAM | 8 Bytes | $YFFB40 |
| QSM | 512 Bytes | $YFFC00 |
| TPU | 512 Bytes | $YFFE00 |

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 the SCIM configuration register (SCIMCR) determines where the control registers block is located in the system memory map. When MM = 0, register addresses range from $7FF000 to $7FFFFF; when MM = 1, register addresses range from $FFF000 to $FFFFFF.

In the module memory maps in this appendix, the "Access" column specifies which registers are accessible at the supervisor privilege level only and which registers can be assigned to either the supervisor or user privilege level.

## D.1 Central Processing Unit

CPU32 registers are not part of the module address map. The following diagram is a functional representation of CPU resources.

### D.1.1 CPU32 Register Model

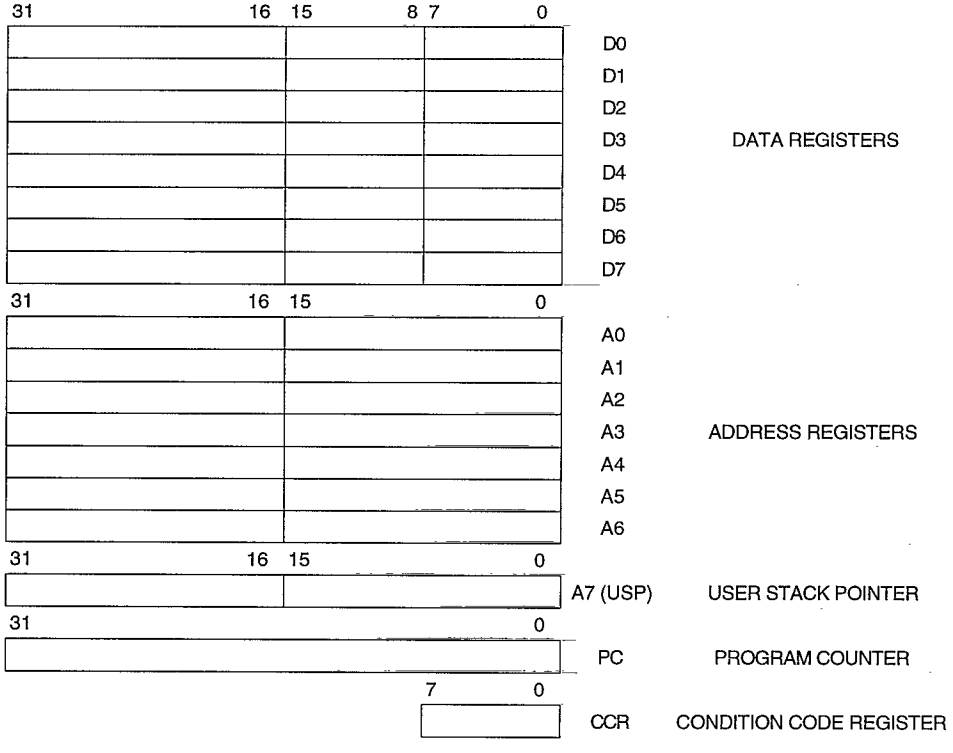



**Figure D–1. User Programming Model**

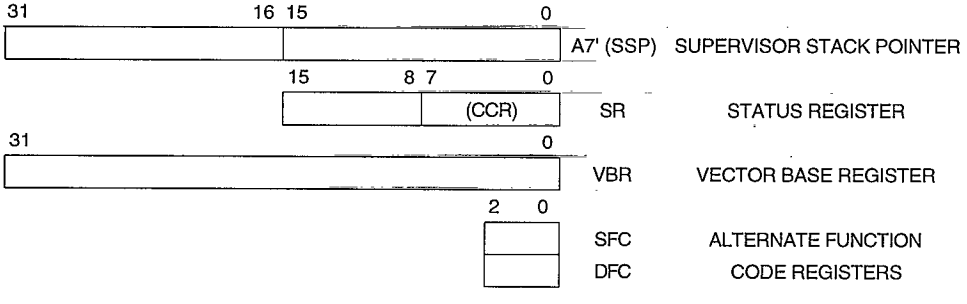



**Figure D–2. Supervisor Programming Model Supplement**

### D.1.2 SR — Status Register

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T1 | T0 | S | 0 | 0 | IP | | | 0 | 0 | 0 | X | N | Z | V | C |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | U | U | U | U | U |

The status register (SR) contains condition codes, an interrupt priority mask, and three control bits. The condition codes are contained in the condition code register (CCR), the lower byte of the SR. (The lower and upper bytes of the status register are also referred to as the user and system bytes, respectively.) At the user privilege level, only the CCR is available. At the supervisor level, software can access the full status register.

T[1:0] — Trace Enable

00 = No tracing
01 = Trace on change of flow
10 = Trace on instruction execution
11 = Undefined; reserved

S — Supervisor/User State

0 = CPU operates at user privilege level
1 = CPU operates at supervisor privilege level

IP[2:0] — Interrupt Priority Mask

The priority value in this field (0 to 7) is used to mask interrupts.

X — Extend Flag

Used in multiple-precision arithmetic operations. In many instructions it is set to the same value as the C bit.

N — Negative Flag

Set when the MSB of a result register is set.

Z — Zero Flag

Set when all bits of a result register are zero.

V — Overflow Flag

Set when two's complement overflow occurs as the result of an operation.

C — Carry Flag

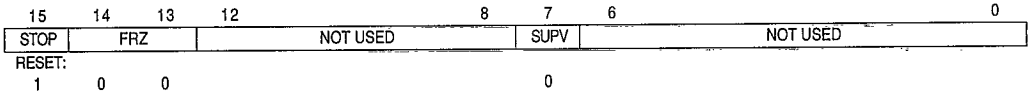
Set when a carry or borrow occurs during an arithmetic operation. Also used during shift and rotate instructions to facilitate multiple word operations.

## D.2 Analog-to-Digital Converter Module

### Table D–2. ADC Module Address Map

| Access | Address | 15 8 | 7 0 |
|---|---|---|---|
| S | $YFF700 | ADC MODULE CONFIGURATION (ADCMCR) | |
| S | $YFF702 | FACTORY TEST (ADTEST) | |
| S | $YFF704 | NOT USED | |
| S/U | $YFF706 | PORT AD DATA (PORTAD) | |
| S/U | $YFF708 | NOT USED | |
| S/U | $YFF70A | ADC CONTROL 0 (ADCTL0) | |
| S/U | $YFF70C | ADC CONTROL 1 (ADCTL1) | |
| S/U | $YFF70E | ADC STATUS (ADSTAT) | |
| S/U | $YFF710 | RIGHT-JUSTIFIED UNSIGNED RESULT 0 (RJURR0) | |
| S/U | $YFF712 | RIGHT-JUSTIFIED UNSIGNED RESULT 1 (RJURR1) | |
| S/U | $YFF714 | RIGHT-JUSTIFIED UNSIGNED RESULT 2 (RJURR2) | |
| S/U | $YFF716 | RIGHT-JUSTIFIED UNSIGNED RESULT 3 (RJURR3) | |
| S/U | $YFF718 | RIGHT-JUSTIFIED UNSIGNED RESULT 4 (RJURR4) | |
| S/U | $YFF71A | RIGHT-JUSTIFIED UNSIGNED RESULT 5 (RJURR5) | |
| S/U | $YFF71C | RIGHT-JUSTIFIED UNSIGNED RESULT 6 (RJURR6) | |
| S/U | $YFF71E | RIGHT-JUSTIFIED UNSIGNED RESULT 7 (RJURR7) | |
| S/U | $YFF720 | LEFT-JUSTIFIED SIGNED RESULT 0 (LJSRR0) | |
| S/U | $YFF722 | LEFT-JUSTIFIED SIGNED RESULT 1 (LJSRR1) | |
| S/U | $YFF724 | LEFT-JUSTIFIED SIGNED RESULT 2 (LJSRR2) | |
| S/U | $YFF726 | LEFT-JUSTIFIED SIGNED RESULT 3 (LJSRR3) | |
| S/U | $YFF728 | LEFT-JUSTIFIED SIGNED RESULT 4 (LJSRR4) | |
| S/U | $YFF72A | LEFT-JUSTIFIED SIGNED RESULT 5 (LJSRR5) | |
| S/U | $YFF72C | LEFT-JUSTIFIED SIGNED RESULT 6 (LJSRR6) | |
| S/U | $YFF72E | LEFT-JUSTIFIED SIGNED RESULT 7 (LJSRR7) | |
| S/U | $YFF730 | LEFT-JUSTIFIED UNSIGNED RESULT 0 (LJURR0) | |
| S/U | $YFF732 | LEFT-JUSTIFIED UNSIGNED RESULT 1 (LJURR1) | |
| S/U | $YFF734 | LEFT-JUSTIFIED UNSIGNED RESULT 2 (LJURR2) | |
| S/U | $YFF736 | LEFT-JUSTIFIED UNSIGNED RESULT 3 (LJURR3) | |
| S/U | $YFF738 | LEFT-JUSTIFIED UNSIGNED RESULT 4 (LJURR4) | |
| S/U | $YFF73A | LEFT-JUSTIFIED UNSIGNED RESULT 5 (LJURR5) | |
| S/U | $YFF73C | LEFT-JUSTIFIED UNSIGNED RESULT 6 (LJURR6) | |
| S/U | $YFF73E | LEFT-JUSTIFIED UNSIGNED RESULT 7 (LJURR7) | |

### D.2.1 ADCMCR — ADC Module Configuration Register $YFF700




The ADCMCR controls ADC operation during low-power stop and freeze modes and determines the privilege level required to access assignable ADC registers.

STOP — STOP Mode

0 = Normal operation

1 = Low-power operation

STOP places the ADC in low-power state. Setting STOP aborts any conversion in progress. STOP is set during reset and must be cleared before ADC operation can begin.

FRZ[1:0] — Freeze

The FRZ field determines ADC response to assertion of the FREEZE signal.

**Freeze Encoding**

| FRZ[1:0] | Response |
|---|---|
| 00 | Ignore FREEZE |
| 01 | Reserved |
| 10 | Finish conversion, then freeze |
| 11 | Freeze immediately |

SUPV — Supervisor/Unrestricted

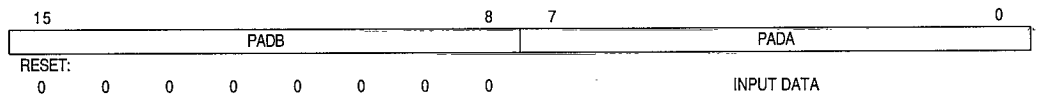
0 = User access permitted to registers controlled by the SUPV bit

1 = Supervisor access only permitted to ADC registers

### D.2.2 ADTEST — ADC Test Register $YFF702

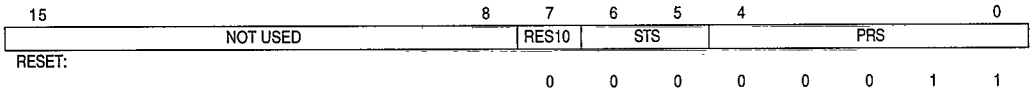
ADTEST is used with the SCIM test register for factory test of the ADC.

### D.2.3 PORTAD — Port AD Data Register $YFF706




Two digital ports are associated with the ADC. Port ADA is an input-only port that uses the eight analog input pins. A read of PORTAD[7:0] returns the logic levels of port ADA pins. If an input is outside specified logic levels, an indeterminate value is read. Use as a digital input does not preclude use as an analog input. Port ADB, an output-only port, uses pins PADB[7:0]. Data for port ADB is latched in the upper half of PORTAD.

### D.2.4 ADCTL0 — ADC Control Register 0 $YFF70A




ADCTL0 is used to select resolution, sample time, and clock/prescaler value. Writing ADCTL0 aborts any conversion in progress. ADC activity halts until ADCTL1 is written.

RES10 — 10-Bit Resolution
0 = 8-bit conversion
1 = 10-bit conversion

STS[1:0] — Sample Time Selection
The STS field selects one of four sample times.

**Sample Time Selection**

| STS[1:0] | Sample Time |
|---|---|
| 00 | 2 A/D Clock Periods |
| 01 | 4 A/D Clock Periods |
| 10 | 8 A/D Clock Periods |
| 11 | 16 A/D Clock Periods |

PRS[4:0] — Prescaler Rate Selection
The ADC clock is generated from the system clock using a modulus counter and a divide-by-two circuit. PRS contains the counter modulus.

**ADC Clock Selection**

| PRS[4:0] | ADCCLK |
|---|---|
| %00000 | Reserved |
| %00001 | Sys Clk/4 |
| %00010 | Sys Clk/6 |
| ... | ... |
| %11101 | Sys Clk/60 |
| %11110 | Sys Clk/62 |
| %11111 | Sys Clk/64 |

D

### D.2.5 ADCTL1 — ADC Control Register 1 $YFF70C

| 15 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| NOT USED | | SCAN | MULT | S8CM | CD | CC | CB | CA |
| RESET: | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

ADCTL1 selects conversion mode and the analog channel or channels. Writing to ADCTL1 aborts any conversion in progress and initiates a new conversion sequence.

SCAN — Scan Mode Selection
- 0 = Single conversion sequence
- 1 = Continuous conversion

MULT — Multichannel Conversion
- 0 = Conversion sequence(s) run on a single channel selected by CD:CA.
- 1 = Sequential conversion of four or eight channels selected by CD:CA.

S8CM — Select Eight-Conversion Sequence Mode
- 0 = Four-conversion sequence
- 1 = Eight-conversion sequence

**ADC Conversion Modes**

| SCAN | MULT | S8CM | MODE |
|---|---|---|---|
| 0 | 0 | 0 | Single 4-conversion single-channel sequence |
| 0 | 0 | 1 | Single 8-conversion single-channel sequence |
| 0 | 1 | 0 | Single 4-conversion multichannel sequence |
| 0 | 1 | 1 | Single 8-conversion multichannel sequence |
| 1 | 0 | 0 | Multiple 4-conversion single-channel sequences |
| 1 | 0 | 1 | Multiple 8-conversion single-channel sequences |
| 1 | 1 | 0 | Multiple 4-conversion multichannel sequences |
| 1 | 1 | 1 | Multiple 8-conversion multichannel sequences |

CD:CA — Channel Selection

Bits in this field select the input channel or channels for analog-to-digital conversion. Conversion mode determines which channel or channels are selected for conversion and which result registers are used to store conversion results. The following tables contain a summary of the effects of ADCTL1 bits and fields.

**Single-Channel Conversions**

| S8CM | CD | CC | CB | CA | Input | Result Register |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | AN0 | RSLT[0:3] |
| 0 | 0 | 0 | 0 | 1 | AN1 | RSLT[0:3] |
| 0 | 0 | 0 | 1 | 0 | AN2 | RSLT[0:3] |
| 0 | 0 | 0 | 1 | 1 | AN3 | RSLT[0:3] |
| 0 | 0 | 1 | 0 | 0 | AN4 | RSLT[0:3] |
| 0 | 0 | 1 | 0 | 1 | AN5 | RSLT[0:3] |
| 0 | 0 | 1 | 1 | 0 | AN6 | RSLT[0:3] |
| 0 | 0 | 1 | 1 | 1 | AN7 | RSLT[0:3] |
| 0 | 1 | 0 | 0 | 0 | Reserved | RSLT[0:3] |
| 0 | 1 | 0 | 0 | 1 | Reserved | RSLT[0:3] |
| 0 | 1 | 0 | 1 | 0 | Reserved | RSLT[0:3] |
| 0 | 1 | 0 | 1 | 1 | Reserved | RSLT[0:3] |
| 0 | 1 | 1 | 0 | 0 | $V_{RH}$ | RSLT[0:3] |
| 0 | 1 | 1 | 0 | 1 | $V_{RL}$ | RSLT[0:3] |
| 0 | 1 | 1 | 1 | 0 | $(V_{RH} - V_{RL}) / 2$ | RSLT[0:3] |
| 0 | 1 | 1 | 1 | 1 | Test/Reserved | RSLT[0:3] |
| 1 | 0 | 0 | 0 | 0 | AN0 | RSLT[0:7] |
| 1 | 0 | 0 | 0 | 1 | AN1 | RSLT[0:7] |
| 1 | 0 | 0 | 1 | 0 | AN2 | RSLT[0:7] |
| 1 | 0 | 0 | 1 | 1 | AN3 | RSLT[0:7] |
| 1 | 0 | 1 | 0 | 0 | AN4 | RSLT[0:7] |
| 1 | 0 | 1 | 0 | 1 | AN5 | RSLT[0:7] |
| 1 | 0 | 1 | 1 | 0 | AN6 | RSLT[0:7] |
| 1 | 0 | 1 | 1 | 1 | AN7 | RSLT[0:7] |
| 1 | 1 | 0 | 0 | 0 | Reserved | RSLT[0:7] |
| 1 | 1 | 0 | 0 | 1 | Reserved | RSLT[0:7] |
| 1 | 1 | 0 | 1 | 0 | Reserved | RSLT[0:7] |
| 1 | 1 | 0 | 1 | 1 | Reserved | RSLT[0:7] |
| 1 | 1 | 1 | 0 | 0 | $V_{RH}$ | RSLT[0:7] |
| 1 | 1 | 1 | 0 | 1 | $V_{RL}$ | RSLT[0:7] |
| 1 | 1 | 1 | 1 | 0 | $(V_{RH} - V_{RL}) / 2$ | RSLT[0:7] |
| 1 | 1 | 1 | 1 | 1 | Test/Reserved | RSLT[0:7] |

**Multichannel Conversions**

| S8CM | CD | CC | CB | CA | Input | Result Register |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | AN[0:3] | RSLT[0:3] |
| 0 | 0 | 1 | X | X | AN[4:7] | RSLT[0:3] |
| 0 | 1 | 0 | X | X | Reserved | RSLT[0:3] |
| 0 | 1 | 1 | X | X | $V_{RH}$ | RSLT0 |
| | | | | | $V_{RL}$ | RSLT1 |
| | | | | | $(V_{RH} - V_{RL}) / 2$ | RSLT2 |
| | | | | | Test/Reserved | RSLT3 |
| 1 | 0 | X | X | X | AN[0:7] | RSLT[0:7] |
| 1 | 1 | X | X | X | Reserved | RSLT0 |
| | | | | | Reserved | RSLT1 |
| | | | | | Reserved | RSLT2 |
| | | | | | Reserved | RSLT3 |
| | | | | | $V_{RH}$ | RSLT4 |
| | | | | | $V_{RL}$ | RSLT5 |
| | | | | | $(V_{RH} - V_{RL}) / 2$ | RSLT6 |
| | | | | | Test/Reserved | RSLT7 |

### D.2.6 ADSTAT — ADC Status Register $YFF70E

| 15 | 14 | 11 | 10 | 8 | 7 | 0 |
|---|---|---|---|---|---|---|
| SCF | NOT USED | | CCTR | | CCF | |
| RESET: | | | | | | |
| 0 | | | 0 0 0 | | 0 0 0 0 0 0 0 0 | |

ADSTAT is a read-only register that contains the sequence complete flag, the conversion counter, and a channel-converted flag for each of the input channels.

SCF — Sequence Complete Flag

0 = Sequence not complete

1 = Sequence complete

When SCAN = 0, SCF is set at the end of a conversion sequence. When SCAN = 1, SCF is set at the end of the first conversion sequence. SCF is cleared when converter activity is halted or restarted by a write to ADCTL0.

CCTR[2:0] — Conversion Counter

This field shows the content of the conversion counter pointer during a conversion sequence. The value is the number of the next result register to be written.

CCF[7:0] — Conversion Complete Flags

Each bit in this field corresponds to an A/D result register. A bit is set when conversion of the corresponding input is complete. It remains set until the result register is read. It is cleared when the register is read.

### D.2.7 RSLT[0:] — ADC Result Registers $YFF710–$YFF73E

Result registers contain conversion results. The data format depends on the address from which the conversion result is read. The notation "10" in a diagram indicates that a bit is used only for 10-bit resolution and is cleared during 8-bit conversion. The notation "8/10" indicates a bit is used for both 8-bit and 10-bit resolution. Unused bits return zeros when read.

#### D.2.7.1 RJURR — Unsigned Right-Justified Result Registers $YFF710–$YFF71F

| 15 | | | | | | | | | | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | | | | | | | | | | 10 | 10 | 8/10 | 8/10 | 8/10 | 8/10 | 8/10 | 8/10 | 8/10 | 8/10 |

#### D.2.7.2 LJSRR — Signed Left-Justified Result Registers $YFF720–$YFF72F

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8/10 | 8/10 | 8/10 | 8/10 | 8/10 | 8/10 | 8/10 | 8/10 | 10 | 10 | NOT USED | |

This data format assumes that the zero reference point is $(V_{RH} - V_{RL}) / 2$. Bit 15 thus indicates the sign of the result. When bit 15 = 1, the result is positive; when bit 15 = 0, the result is negative. Bits [5:0] return zeros when read.

#### D.2.7.3 LJURR — Unsigned Left-Justified Result Registers $YFF730–$YFF73F

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8/10 | 8/10 | 8/10 | 8/10 | 8/10 | 8/10 | 8/10 | 8/10 | 10 | 10 | NOT USED | |

## D.3 Flash EEPROM Modules

The address map of the flash EEPROM modules follows. All control block registers reside in supervisor space.

**Table D–3. Flash EEPROM Address Map**

| Access | Address | Register | Module |
|---|---|---|---|
| S | $YFF800 | FLASH EEPROM MODULE CONFIGURATION (FEE1MCR) | FLASH16 |
| S | $YFF802 | FLASH EEPROM TEST REGISTER (FEE1TST) | |
| S | $YFF804 | FLASH EEPROM BASE ADDRESS HIGH (FEE1BAH) | |
| S | $YFF806 | FLASH EEPROM BASE ADDRESS LOW (FEE1BAL) | |
| S | $YFF808 | FLASH EEPROM CONTROL REGISTER (FEE1CTL) | |
| S | $YFF80A | NOT USED | |
| S | $YFF80C | NOT USED | |
| S | $YFF80E | NOT USED | |
| S | $YFF810 | FLASH EEPROM BOOTSTRAP WORD 0 (FEE1BS0) | |
| S | $YFF812 | FLASH EEPROM BOOTSTRAP WORD 1 (FEE1BS1) | |
| S | $YFF814 | FLASH EEPROM BOOTSTRAP WORD 2 (FEE1BS2) | |
| S | $YFF816 | FLASH EEPROM BOOTSTRAP WORD 3 (FEE1BS3) | |
| S | $YFF818 | NOT USED | |
| S | $YFF81A | NOT USED | |
| S | $YFF81C | NOT USED | |
| S | $YFF81E | NOT USED | |
| S | $YFF820 | FLASH EEPROM MODULE CONFIGURATION (FEE2MCR) | FLASH48 |
| S | $YFF822 | FLASH EEPROM TEST REGISTER (FEE2TST) | |
| S | $YFF824 | FLASH EEPROM BASE ADDRESS HIGH (FEE2BAH) | |
| S | $YFF826 | FLASH EEPROM BASE ADDRESS LOW (FEE2BAL) | |
| S | $YFF828 | FLASH EEPROM CONTROL REGISTER (FEE2CTL) | |
| S | $YFF82A | NOT USED | |
| S | $YFF82C | NOT USED | |
| S | $YFF82E | NOT USED | |
| S | $YFF830 | FLASH EEPROM BOOTSTRAP WORD 0 (FEE2BS0) | |
| S | $YFF832 | FLASH EEPROM BOOTSTRAP WORD 1 (FEE2BS1) | |
| S | $YFF834 | FLASH EEPROM BOOTSTRAP WORD 2 (FEE2BS2) | |
| S | $YFF836 | FLASH EEPROM BOOTSTRAP WORD 3 (FEE2BS3) | |
| S | $YFF838 | NOT USED | |
| S | $YFF83A | NOT USED | |
| S | $YFF83C | NOT USED | |
| S | $YFF83E | NOT USED | |

In the following register diagrams, the reset value SB means the bit assumes the value of its associated shadow bit.

### D.3.1 FEE1MCR — Flash EEPROM Module Configuration Register $YFF800
### FEE2MCR — Flash EEPROM Module Configuration Register $YFF820

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STOP | FRZ | 0 | BOOT | LOCK | 0 | ASPC | | WAIT | | 0 | 0 | 0 | 0 | 0 | 0 |
| RESET: | | | | | | | | | | | | | | | |
| SB | 0 | 0 | SB | SB | 0 | SB | | SB | | 0 | 0 | 0 | 0 | 0 | 0 |

These registers can be written only when the control block is not write-locked (i.e., when LOCK = 0). All fields and bits are set with the values in the shadow register after reset.

STOP — Stop Mode Control

0 = Normal operation

1 = Low-power stop operation

STOP can be set with an external STOP configuration signal or by reset if the STOP shadow bit is set. The array can be reenabled by clearing STOP after reset.

FRZ — Freeze Mode Control

0 = Disable program/erase voltage while FREEZE is asserted

1 = Allow ENPE bit to turn on the program/erase voltage while FREEZE signal is asserted

BOOT — Boot Control

0 = Flash EEPROM module responds to the bootstrap addresses after reset

1 = Flash EEPROM module does not respond to the bootstrap addresses after reset

LOCK — Lock Registers

0 = Write-locking disabled

1 = Write-locked registers protected

If the default reset state of the LOCK is zero, it can be set once after master reset to allow protection of the registers after initialization. Once the LOCK bit is set by software, it cannot be cleared again until after a master reset.

ASPC — Flash EEPROM Array Space

00 = Unrestricted program and data space

01 = Unrestricted program space

10 = Supervisor program and data space

11 = Supervisor program space

ASPC assigns the flash EEPROM array to supervisor or user space, and to program or data space. The field can be written only if LOCK = 0 and STOP = 1.

WAIT — Wait States

The WAIT field specifies the number of wait states inserted during accesses to the flash EEPROM module. A wait state has the duration of one system clock cycle. This field affects both control block and array access.

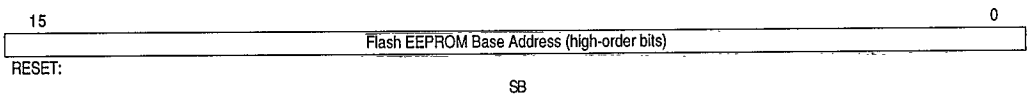
| WAIT[1:0] | Wait States | Clocks/Transfer |
|---|---|---|
| 00 | 0 | 3 |
| 01 | 1 | 4 |
| 10 | 2 | 5 |
| 11 | –1 | 2 |

The value of the WAIT field is compatible with the lower two bits of the DSACK field in the SCIM chip select option registers.

### D.3.2 FEE1TST — Flash EEPROM Test Register $YFF802
### FEE2TST — Flash EEPROM Test Register $YFF822

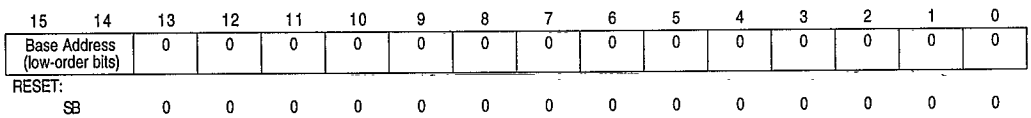
These registers are used for factory test purposes only.

### D.3.3 FEE1BAH — Flash EEPROM Base Address High Register $YFF804
### FEE2BAH — Flash EEPROM Base Address High Register $YFF824




FEEBAH contains the 16 high-order bits of the flash EEPROM array base address. After reset, if LOCK = 0 and STOP = 1, software can write to FEEBAH and FEEBAL to relocate the flash EEPROM array.

### D.3.4 FEE1BAL — Flash EEPROM Base Address Low Register $YFF806
### FEE2BAL — Flash EEPROM Base Address Low Register $YFF826




FEE1BAL contains the two low-order bits of the 16-Kbyte flash EEPROM array base address. After reset, if LOCK = 0 and STOP = 1, software can write to FEEBAH and FEEBAL to relocate the flash EEPROM array. All bits of FEE2BAL (in the 48-Kbyte module) are always zero.

D

### D.3.5 FEE1CTL — Flash EEPROM Control Register $YFF808
### FEE2CTL — Flash EEPROM Control Register $YFF828

| 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 | VFPE | ERAS | LAT | ENPE |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

FEECTL contains the bits needed to control the programming and erasure of the flash EEPROM. This register is accessible in supervisor mode only.

VFPE — Verify Program/Erase

0 = Normal read cycles

1 = Invoke program verify circuit

This bit invokes a special program verify circuit. During programming sequences (ERAS = 0), VFPE is used in conjunction with the LAT bit to determine when programming of a location is complete.

ERAS — Erase Control

0 = Flash EEPROM configured for programming

1 = Flash EEPROM configured for erasure

Asserting ERAS causes all locations in the array and all flash EEPROM control bits in the control block to be configured for erasure at the same time. When the LAT bit is set, ERAS also determines whether a read returns the value of the addressed location (ERAS = 1) or the location being programmed (ERAS = 0). The value of ERAS cannot be changed if the program/erase voltage is turned on (ENPE = 1).

LAT — Latch Control

0 = Programming latches disabled

1 = Programming latches enabled

When LAT = 0, the flash EEPROM address and data buses are connected to the IMB address and data buses and the flash EEPROM is configured for normal reads. When LAT is asserted, the flash EEPROM address and data buses are connected to parallel internal latches and the flash EEPROM array is configured for programming or erasure. The value of LAT cannot be changed if the program/erase voltage is turned on (ENPE = 1).

ENPE — Enable Programming/Erase

0 = Disable program/erase voltage

1 = Apply program/erase voltage to flash EEPROM

### D.3.6 **FEE1BS[3:0]** — Flash EEPROM Bootstrap Words **$YFF810–$YFF816**
**FEE2BS[3:0]** — Flash EEPROM Bootstrap Words **$YFF830–$YFF836**

These words may contain bootstrap information for the processor. If BOOT = 0 in the FEEMCR, these words are mapped to the following addresses:

| Word | Address |
|---|---|
| FEEBS0 | $00000000 |
| FEEBS1 | $00000002 |
| FEEBS2 | $00000004 |
| FEEBS3 | $00000006 |

When BOOT = 0, these words respond only to supervisor program space accesses. When BOOT = 1, they respond only to supervisor data space accesses. FEEBS[3:0] can be read as registers at any time, but they respond to addresses $00 to $06 only when BOOT = 0.

## Table D–4. SCIM Address Map

| Access | Address | 15 8 | 7 0 |
|---|---|---|---|
| S | $YFFA00 | SCIM CONFIGURATION (SCIMCR) | |
| S | $YFFA02 | SCIM FACTORY TEST (SCIMTR) | |
| S | $YFFA04 | CLOCK SYNTHESIZER CONTROL (SYNCR) | |
| S | $YFFA06 | NOT USED | RESET STATUS REGISTER (RSR) |
| S | $YFFA08 | MODULE TEST E (SCIMTRE) | |
| S/U | $YFFA0A | PORT A DATA REGISTER (PORTA) | PORT B DATA REGISTER (PORTB) |
| S/U | $YFFA0C | PORT G DATA REGISTER (PORTG) | PORT H DATA REGISTER (PORTH) |
| S/U | $YFFA0E | PORT G DATA DIRECTION (DDRG) | PORT H DATA DIRECTION (DDRH) |
| S/U | $YFFA10 | NOT USED | PORT E DATA (PORTE0) |
| S/U | $YFFA12 | NOT USED | PORT E DATA (PORTE1) |
| S/U | $YFFA14 | PORT A/B DATA DIRECTION (DDRAB) | PORT E DATA DIRECTION (DDRE) |
| S | $YFFA16 | NOT USED | PORT E PIN ASSIGNMENT (PEPAR) |
| S/U | $YFFA18 | NOT USED | PORT F DATA (PORTF0) |
| S/U | $YFFA1A | NOT USED | PORT F DATA (PORTF1) |
| S/U | $YFFA1C | NOT USED | PORT F DATA DIRECTION (DDRF) |
| S | $YFFA1E | NOT USED | PORT F PIN ASSIGNMENT (PFPAR) |
| S | $YFFA20 | NOT USED | SYSTEM PROTECTION CONTROL (SYPCR) |
| S | $YFFA22 | PERIODIC INTERRUPT CONTROL (PICR) | |
| S | $YFFA24 | PERIODIC INTERRUPT TIMING (PITR) | |
| S | $YFFA26 | NOT USED | SOFTWARE SERVICE (SWSR) |
| S | $YFFA28 | NOT USED | PORTFE |
| S | $YFFA2A | NOT USED | PORT F EDGE-DETECT INTERRUPT VECTOR (PFIVR) |
| S | $YFFA2C | NOT USED | PORT F EDGE-DETECT INTERRUPT LEVEL (PFLVR) |
| S/U | $YFFA2E | NOT USED | NOT USED |
| S | $YFFA30 | TEST SUBMODULE MASTER SHIFT A (TSTMSRA) | |
| S | $YFFA32 | TEST SUBMODULE MASTER SHIFT B (TSTMSRB) | |
| S | $YFFA34 | TEST SUBMODULE SHIFT COUNT (TSTSC) | |
| S | $YFFA36 | TEST SUBMODULE REPETITION COUNTER (TSTRC) | |
| S | $YFFA38 | TEST SUBMODULE CONTROL (CREG) | |
| S/U | $YFFA3A | TEST SUBMODULE DISTRIBUTED REGISTER (DREG) | |
| S/U | $YFFA3C | NOT USED | NOT USED |
| S/U | $YFFA3E | NOT USED | NOT USED |
| S/U | $YFFA40 | NOT USED | PORT C DATA (PORTC) |
| S/U | $YFFA42 | NOT USED | NOT USED |

**Table D–4. SCIM Address Map (Continued)**

| Access | Address | 15 8 | 7 0 |
|---|---|---|---|
| S | $YFFA44 | CHIP-SELECT PIN ASSIGNMENT (CSPAR0) | |
| S | $YFFA46 | CHIP-SELECT PIN ASSIGNMENT (CSPAR1) | |
| S | $YFFA48 | CHIP-SELECT BASE BOOT (CSBARBT) | |
| S | $YFFA4A | CHIP-SELECT OPTION BOOT (CSORBT) | |
| S | $YFFA4C | CHIP-SELECT BASE 0 (CSBAR0) | |
| S | $YFFA4E | CHIP-SELECT OPTION 0 (CSOR0) | |
| S | $YFFA50 | NOT USED | |
| S | $YFFA52 | NOT USED | |
| S | $YFFA54 | NOT USED | |
| S | $YFFA56 | NOT USED | |
| S | $YFFA58 | CHIP-SELECT BASE 3 (CSBAR3) | |
| S | $YFFA5A | CHIP-SELECT OPTION 3 (CSOR3) | |
| S | $YFFA5C | NOT USED | |
| S | $YFFA5E | NOT USED | |
| S | $YFFA60 | CHIP-SELECT BASE 5 (CSBAR5) | |
| S | $YFFA62 | CHIP-SELECT OPTION 5 (CSOR5) | |
| S | $YFFA64 | CHIP-SELECT BASE 6 (CSBAR6) | |
| S | $YFFA66 | CHIP-SELECT OPTION 6 (CSOR6) | |
| S | $YFFA68 | CHIP-SELECT BASE 7 (CSBAR7) | |
| S | $YFFA6A | CHIP-SELECT OPTION 7 (CSOR7) | |
| S | $YFFA6C | CHIP-SELECT BASE 8 (CSBAR8) | |
| S | $YFFA6E | CHIP-SELECT OPTION 8 (CSOR8) | |
| S | $YFFA70 | CHIP-SELECT BASE 9 (CSBAR9) | |
| S | $YFFA72 | CHIP-SELECT OPTION 9 (CSOR9) | |
| S | $YFFA74 | CHIP-SELECT BASE 10 (CSBAR10) | |
| S | $YFFA76 | CHIP-SELECT OPTION 10 (CSOR10) | |
| | $YFFA78 | NOT USED | NOT USED |
| | $YFFA7A | NOT USED | NOT USED |
| | $YFFA7C | NOT USED | NOT USED |
| | $YFFA7E | NOT USED | NOT USED |

## D.4.1 SCIMCR — SCIM Configuration Register $YFFA00

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EXOFF | FRZSW | FRZBM | CPUD | SLVEN | 0 | SHEN | | SUPV | MM | ABD | RWD | IARB | |
| RESET: | | | | | | | | | | | | | |
| 0 | 0 | 0 | $\overline{\text{BERR}}$ | DATA11 | 0 | 0 | 0 | 1 | 1 | $\overline{\text{BERR}}$ | $\overline{\text{BERR}}$ | 1 | 1 | 1 | 1 |

The SCIMCR controls system configuration. The SCIMCR can be read or written at any time, except for the module mapping (MM) bit, which can be written once and then must remain set.

D

EXOFF — External Clock Off

0 = The CLKOUT pin is driven from an internal clock source.

1 = The CLKOUT pin is placed in a high-impedance state.

FRZSW — Freeze Software Enable

0 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters continue to run.

1 = When FREEZE is asserted, the software watchdog and periodic interrupt timer counters are disabled, preventing interrupts during software debug.

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.

CPUD — CPU Development Support Disable

0 = Instruction pipeline signals available on pins $\overline{\text{IPIPE}}$ and $\overline{\text{IFETCH}}$

1 = Unless a breakpoint occurs, pins $\overline{\text{IPIPE}}$ and $\overline{\text{IFETCH}}$ placed in high-impedance state

CPUD is reset to zero when the MCU is in an expanded mode, and to one in single-chip mode.

SLVEN — Factory Test Mode Enabled

0 = IMB is not available to an external master.

1 = An external bus master has direct access to the IMB.

SHEN[1:0] — Show Cycle Enable

This field determines what the EBI does with the external bus during internal transfer operations.

SUPV — Supervisor/User Data Space

0 = Registers with access controlled by this bit are unrestricted.

1 = Registers with access controlled by this bit are restricted to supervisor access only.

MM — Module Mapping

0 = Internal modules are addressed from $7FF000–$7FFFFF.

1 = Internal modules are addressed from $FFF000–$FFFFFF.

This bit can be written only once. Subsequent attempts to change this bit are ignored.

ABD — Address Bus Disable

0 = Pins ADDR[2:0] operate normally.

1 = Pins ADDR[2:0] are disabled.

ABD is reset to zero when the MCU is in an expanded mode, and to one in single-chip mode. ABD can be written only once after reset.

RWD — Read/Write Disable

0 = R/$\overline{W}$ signal operates normally

1 = R/$\overline{W}$ signal placed in high-impedance state

RWD is reset to zero when the MCU is in an expanded mode, and to one in single-chip mode. RWD can be written only once after reset.

IARB[3:0] — Interrupt Arbitration Field

The value in this field is used to arbitrate between simultaneous interrupt service requests of the same priority. Each module that can generate interrupts has an IARB field. To implement an arbitration scheme, each IARB field must be set to a different non-zero value. If an interrupt request from a module that has an IARB field value of $0 is recognized, the CPU32 processes a spurious interrupt exception. The reset value of all IARB fields other than that of the SCIM is $0 (no priority), to preclude interrupt processing during reset.

### D.4.2 SCIMTR — SCIM Test Register $YFFA02

SCIMTR is used for factory test only.

### D.4.3 SYNCR — Clock Synthesizer Control Register $YFFA04

| 15 | 14 | 13 | | | | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| W | X | Y | | | | | | EDIV | 0 | LOSCD | SLIMP | SLOCK | RSTEN | STSCIM | STEXT |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | U | U | 0 | 0 | 0 |

The SYNCR determines system clock operating frequency and mode of operation.

W — Frequency Control (VCO)

0 = Base VCO frequency

1 = VCO frequency multiplied by four

X — Frequency Control Bit (Prescale)

0 = Base system clock frequency

1 = System clock frequency multiplied by two

Y[5:0] — Frequency Control (Counter)

The Y field is the initial value for the modulus 64 down counter in the synthesizer feedback loop. Values range from 0 to 63.

EDIV — ECLK Divide Rate

0 = ECLK is system clock divided by 8

1 = ECLK is system clock divided by 16

LOSCD — Loss-of-Clock Oscillator Disable

0 = Enable loss-of-clock oscillator

1 = Disable loss-of-clock oscillator

SLIMP — Limp Mode
0 = Clock operating normally
1 = Loss of clock

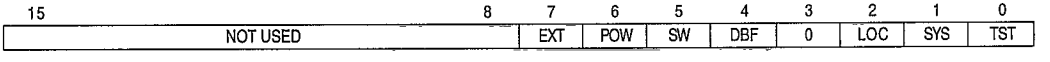
SLOCK — Synthesizer Lock
0 = VCO is enabled, but has not locked.
1 = VCO has locked on the desired frequency or system clock is external.

RSTEN — Reset Enable
0 = Loss of clock causes the MCU to operate in limp mode.
1 = Loss of clock causes system reset.

STSCIM — Stop Mode SCIM Clock
0 = When LPSTOP is executed, the SCIM clock is driven from the crystal oscillator and the VCO is turned off to conserve power.
1 = When LPSTOP is executed, the SCIM clock is driven from the VCO.

STEXT — Stop Mode External Clock
0 = CLKOUT held low during low-power stop to conserve power.
1 = CLKOUT driven from SCIM clock during low-power stop, as determined by the state of the STSCIM bit.

### D.4.4 RSR — Reset Status Register $YFFA06




EXT — External Reset
Reset was caused by an external signal.

POW — Power-On Reset
Reset was caused by the power-on reset circuit.

SW — Software Watchdog Reset
Reset was caused by the software watchdog circuit.

DBF — Double Bus Fault Reset
Reset was caused by a double bus fault.

LOC — Loss-of-Clock Reset
Reset was caused by loss of clock signal.

SYS — System Reset
Reset was caused by the CPU32 RESET instruction.

TST — Test Submodule Reset
Reset was caused by the test submodule. This bit is set during system test only.

### D.4.5 PORTA — Port A Data Register $YFFA0A

### PORTB — Port B Data Register $YFFA0B

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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. One data direction register controls data direction for ports A and B. PORTA and PORTB can be read or written any time the MCU is not in emulator mode.

### D.4.6 PORTG — Port G Data Register $YFFA0C

### PORTH — Port H Data Register $YFFA0D

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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, accessed through the upper byte of the register, is available during single-chip operation only. During single-chip operation, port G pins are always configured for general-purpose I/O.

Port H, accessed through the lower byte of the register, is available during single-chip and partially expanded operation only. The function of port H pins is determined by external bus configuration; there is no pin assignment register associated with this port.

### D.4.7 DDRG — Port G Data Direction Register $YFFA0E

### DDRH — Port H Data Direction Register $YFFA0F

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 the DDRG and DDRH control the direction of the pin drivers when the pins are configured for I/O. 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.

### D.4.8 PORTE0, PORTE1 — Port E Data Register $YFFA10, $YFFA12

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | | | | | | | PE7 | PE6 | PE5 | PE4 | PE3 | PE2 | PE1 | PE0 |
| RESET: | | | | | | | | U | U | U | U | U | U | U | U |

PORTE is a single register that can be accessed in two locations. It can be read or written any time the MCU is not in emulator mode.

### D.4.9 DDRAB — Port A/B Data Direction Register $YFFA14

### DDRE — Port E Data Direction Register $YFFA15

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

DDRAB, in the upper half of the register, contains two bits, DDA and DDB, that control the direction of the pin drivers for ports A and B, respectively, when the pins are configured for I/O. Setting DDA or DDB configures all pins in the corresponding port as outputs. Clearing DDA or DDB to zero configures all pins in the corresponding port as inputs.

The bits in DDRE, in the lower half of the register, control the direction of the pin drivers when the pins are configured for I/O. 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. This register can be read or written any time the MCU is not in emulator mode.

### D.4.10 PEPAR — Port E Pin Assignment Register $YFFA16

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | | | | | | | PEPA3 | | PEPA2 | | PEPA1 | | PEPA0 | |
| RESET (Expanded-bus configuration): | | | | | | | | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 | DATA8 |
| RESET (Single-chip configuration): | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The bits in the PEPAR control the function of each port E pin. Any bit set to one defines the corresponding pin to be a bus control signal, with the function shown in the following table. Any bit cleared to zero defines the corresponding pin to be an I/O pin, controlled by PORTE and DDRE.

**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 | $\overline{DSACK0}$ |

### D.4.11 PORTF0, PORTF1 — Port F Data Register $YFFA18, $YFFA1A

| 15 … 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| NOT USED | PF7 | PF6 | PF5 | PF4 | PF3 | PF2 | PF1 | PF0 |
| RESET: | U | U | U | U | U | U | U | U |

A write to the port F data register is stored in the internal data latch, and if any port F pin is configured as an output, the value stored for that bit is driven onto the pin. A read of PORTF returns the value on a pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the data register.

Port F is a single register that can be accessed in two locations. It can be read or written at any time, including when the MCU is in emulator mode.

### D.4.12 DDRF — Port F Data Direction Register $YFFA1C

| 15 … 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| NOT USED | DDF7 | DDF6 | DDF5 | DDF4 | DDF3 | DDF2 | DDF1 | DDF0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

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

### D.4.13 PFPAR — Port F Pin Assignment Register $YFFA1E

| 15 … 8 | 7 6 | 5 4 | 3 2 | 1 0 |
|---|---|---|---|---|
| NOT USED | PFPA3 | PFPA2 | PFPA1 | PFPA0 |
| RESET (Expanded-bus configuration): | DATA9 DATA9 | DATA9 DATA9 | DATA9 DATA9 | DATA9 DATA9 |
| RESET (Single-chip configuration): | 0 0 | 0 0 | 0 0 | 0 0 |

The fields in the PFPAR determine the functions of pairs of port F pins as shown in the following tables. $\overline{\text{BERR}}$ and DATA9 determine the reset state of this register. If either $\overline{\text{BERR}}$ or DATA9 is low during reset, this register is set to $00, defining all port F pins to be I/O pins. If $\overline{\text{BERR}}$ and DATA9 are both high during reset, the register is set to $FF, which defines all port F pins except PF0 to be interrupt signals.

**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{IRQ}}$1, MODCLK* |

*MODCLK signal is only recognized during reset

**PFPAR Pin Encodings**

| PFPA Bits | Port F Signal |
|---|---|
| 00 | I/O pin without edge detection |
| 01 | Rising edge detection |
| 10 | Falling edge detection |
| 11 | Interrupt request (or MODCLK) |

### D.4.14 PORTFE — Port F Edge-Detect Flag Register $YFFA28

| 15 … 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| NOT USED | EF7 | EF6 | EF5 | EF4 | EF3 | EF2 | EF1 | EF0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

When a pin is configured in the PFPAR to detect either a rising or falling edge, and such an edge is detected, the corresponding bit in the PORTFE is set. 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.

### D.4.15 PFIVR — Port F Edge-Detect Interrupt Vector Register $YFFA2A

| 15 … 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| NOT USED | PFIV7 | PFIV6 | PFIV5 | PFIV4 | PFIV3 | PFIV2 | PFIV1 | PFIV0 |
| RESET: | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

The PFIVR determines which vector in the exception vector table is used for interrupts generated by the port F edge-detect logic. Program PFIV[7:0] to the value pointing to the appropriate interrupt vector. Refer to the *CPU32 Reference Manual* (CPU32RM/AD) for interrupt vector assignments.

### D.4.16 PFLVR — Port F Edge-Detect Interrupt Level Register $YFFA2C

| 15 … 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| NOT USED | 0 | 0 | 0 | 0 | 0 | PFLV2 | PFLV1 | PFLV0 |
| RESET: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

PFLVR determines the priority level of the port F edge-detect interrupt. The reset value is $00, indicating that the interrupt is disabled. When several sources of interrupts within the SCIM with the same interrupt request level enter interrupt arbitration, the port F edge-detect interrupt has the lowest arbitration priority.

### D.4.17 SYPCR — System Protection Control Register $YFFA20

| 15 … 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| NOT USED | SWE | SWP | SWT | | HME | BME | BMT | |
| RESET: | 1 | $\overline{\text{MODCLK}}$ | 0 | 0 | 0 | 0 | 0 | 0 |

SWE — Software Watchdog Enable
- 0 = Software watchdog disabled
- 1 = Software watchdog enabled

SWP — Software Watchdog Prescale
- 0 = Software watchdog clock not prescaled
- 1 = Software watchdog clock prescaled by 512

SWT[1:0] — Software Watchdog Timing
This field selects software watchdog timeout period.

**Software Watchdog Ratio**

| SWP | SWT | 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 | $2^{24}$ |

HME — Halt Monitor Enable
0 = Disable halt monitor function
1 = Enable halt monitor function

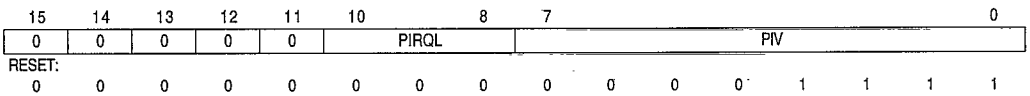
BME — Bus Monitor External Enable
0 = Disable bus monitor function for an internal-to-external bus cycle.
1 = Enable bus monitor function for an internal-to-external bus cycle.

BMT[1:0] — Bus Monitor Timing
This field selects bus monitor timeout period.

**Bus Monitor Period**

| BMT | Bus Monitor Timeout Period |
|---|---|
| 00 | 64 System Clocks |
| 01 | 32 System Clocks |
| 10 | 16 System Clocks |
| 11 | 8 System Clocks |

### D.4.18 PICR — Periodic Interrupt Control Register $YFFA22

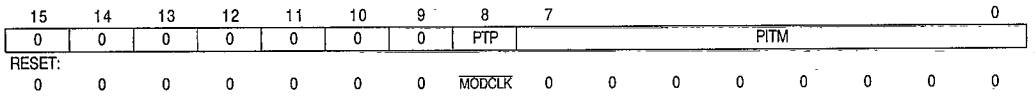



PIRQL[2:0] — Periodic Interrupt Request Level
This field determines the priority of periodic interrupt requests.

PIV[7:0] — Periodic Interrupt Vector
The bits of this field contain the periodic interrupt vector number supplied by the SCIM when the CPU acknowledges an interrupt request.

D

### D.4.19 PITR — Periodic Interrupt Timer Register $YFFA24

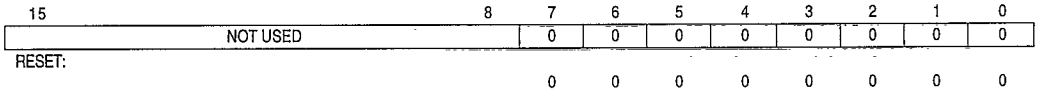



PTP — Periodic Timer Prescaler Control
1 = Periodic timer clock prescaled by a value of 512
0 = Periodic timer clock not prescaled

PITM[7:0] — Periodic Interrupt Timing Modulus
This is the 8-bit timing modulus used to determine periodic interrupt rate. Use the following expression to calculate timer period.

PIT Period = [(PIT Modulus)(Prescaler value)(4)]/EXTAL Frequency

### D.4.20 SWSR — Software Service Register $YFFA26




When enabled, the software watchdog requires that a service sequence be written to the SWSR on a periodic basis. Perform a software watchdog service sequence as follows:

a. Write $55 to the SWSR.
b. Write $AA to the SWSR.

### D.4.21 SCIM Test Registers

SCIM test registers are for factory testing only.

**TSTMSRA** — Test Submodule Master Shift Register A **$YFFA30**

**TSTMSRB** — Test Submodule Master Shift Register B **$YFFA32**

**TSTSC** — Test Submodule Shift Count **$YFFA34**

**TSTRC** — Test Submodule Repetition Count **$YFFA36**

**CREG** — Test Submodule Control Register **$YFFA38**

**DREG** — Test Submodule Distributed Register **$YFFA3A**

### D.4.22 PORTC — Port C Data Register $YFFA40

| 15 … 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| NOT USED | 0 | PC6 | PC5 | PC4 | PC3 | PC2 | PC1 | PC0 |
| RESET: | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

The port C data register latches data for pins programmed as discrete outputs. When a pin is assigned as a discrete output, the value in this register appears at the output.

### D.4.23 CSPAR0 — Chip Select Pin Assignment Register 0 $YFFA44

| 15 | 14 | 13 12 | 11 10 | 9 8 | 7 6 | 5 4 | 3 2 | 1 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | CSPA0[6] | CSPA05 | CSPA04 | CSPA03 | CSPA02 | CSPA01 | $\overline{\text{CSBOOT}}$ |
| RESET: | | | | | | | | |
| 0 | 0 | DATA2 1 | DATA2 1 | DATA2 1 | DATA1 1 | DATA1 1 | DATA1 1 | 1 DATA0 |

CSPAR0 contains six 2-bit fields that determine the functions of corresponding chip-select pins. CSPAR0[15:14] are not used. These bits always read zero; writes have no effect. CSPAR0 bit 1 always reads one; writes to bit 1 of CSPAR0 have no effect.

**CSPAR0 Pin Assignments**

| CSPAR0 Field | Chip Select Signal | Alternate Signal | Discrete Output |
|---|---|---|---|
| CSPA0[6] | $\overline{\text{CS5}}$ | FC2 | PC2 |
| CSPA0[5] | — | FC1 | PC1 |
| CSPA0[4] | $\overline{\text{CS3}}$ | FC0 | PC0 |
| CSPA0[3] | $\overline{\text{CSE}}$ | $\overline{\text{BGACK}}$ | — |
| CSPA0[2] | — | $\overline{\text{BG}}$ | — |
| CSPA0[1] | $\overline{\text{CS0}}$ | $\overline{\text{BR}}$ | — |
| $\overline{\text{CSBOOT}}$ | $\overline{\text{CSBOOT}}$ | — | — |

### D.4.24 CSPAR1 — Chip Select Pin Assignment Register 1 $YFFA46

| 15 | 14 | 13 | 12 | 11 | 10 | 9 8 | 7 6 | 5 4 | 3 2 | 1 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | CSPA1[4] | CSPA1[3] | CSPA1[2] | CSPA1[1] | CSPA1[0] |
| RESET: | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | DATA7 1 | DATA6 1 | DATA5 1 | DATA4 1 | DATA3 1 |

CSPAR1 contains five 2-bit fields that determine the functions of corresponding chip-select pins. CSPAR1[15:10] are not used. These bits always read zero; writes have no effect.

### CSPAR1 Pin Assignments

| CSPAR0 Field | Chip Select Signal | Alternate Signal | Discrete Output |
|---|---|---|---|
| CSPA1[4] | $\overline{CS10}$ | ADDR23 | ECLK |
| CSPA1[3] | $\overline{CS9}$ | ADDR22 | PC6 |
| CSPA1[2] | $\overline{CS8}$ | ADDR21 | PC5 |
| CSPA1[1] | $\overline{CS7}$ | ADDR20 | PC4 |
| CSPA1[0] | $\overline{CS6}$ | ADDR19 | PC3 |

### Pin Assignment Encodings

| Bit Field | Description |
|---|---|
| 00 | Discrete Output |
| 01 | Alternate Function |
| 10 | Chip Select (8-Bit Port) |
| 11 | Chip Select (16-Bit Port) |

## D.4.25 CSBARBT — Chip Select Base Address Register Boot ROM $YFFA48

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | | 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 | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

## D.4.26 CSBAR[10:0] — Chip Select Base Address Registers $YFFA4C–$YFFA74

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | | 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 | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

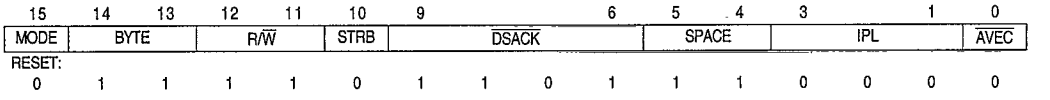
ADDR[15:3] — Base Address Field

This field sets the starting address of a particular address space.

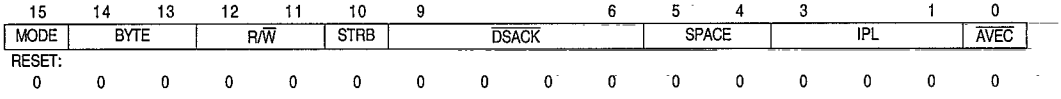
BLKSZ — Block Size Field

This field determines the size of the block above the base address that is enabled by the chip select.

### D.4.27 CSORBT — Chip Select Option Register Boot ROM $YFFA4A




### D.4.28 CSOR[10:0] — Chip Select Option Registers $YFFA4E–$YFFA76




MODE — Timing Mode

The MODE bit determines whether chip-select operation emulates asynchronous bus operation or is synchronized to the M6800-type bus clock signal (ECLK) available on ADDR23.

0 = Emulate asynchronous bus operation
1 = Synchronize chip-select assertion to ECLK

BYTE — Upper/Lower Byte Option

This field enables or disables the chip-select circuit and, for 16-bit ports, determines which combinations of size and ADDR0 pins will cause the chip select to be asserted.

00 = Disable
01 = Lower byte
10 = Upper byte
11 = Both Bytes

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 read and write.

00 = Reserved
01 = Read only
10 = Write only
11 = Read/Write

STRB — Address Strobe/Data Strobe

The STRB bit controls the timing of a chip-select assertion in asynchronous mode. This bit has no effect in synchronous mode.

0 = Synchronize chip select assertion with address strobe
1 = Synchronize chip select assertion with data strobe

$\overline{DSACK}$ — Data and Size Acknowledge

This field specifies the source of $\overline{DSACK}$ when chip-select cycles emulate asynchronous bus cycles, and controls wait state insertion.

SPACE — Address Space Select

The SPACE field determines the address space in which a chip select is asserted. An access must have the space type represented by SPACE encoding in order for a chip-select signal to be asserted.

00 = CPU space
01 = User space
10 = Supervisor space
11 = Supervisor or user space

IPL — Interrupt Priority Level

This field selects the interrupt level when a chip select is used for interrupt acknowledge.

000 = Any Level
001 = Level 1
010 = Level 2
011 = Level 3
100 = Level 4
101 = Level 5
110 = Level 6
111 = Level 7

$\overline{\text{AVEC}}$ — Autovector Enable

This field specifies whether to generate an internal $\overline{\text{AVEC}}$ signal during an interrupt acknowledge cycle initiated by assertion of an $\overline{\text{IRQ}}$ pin when match conditions are met.

0 = Disable $\overline{\text{AVEC}}$ generation
1 = Enable $\overline{\text{AVEC}}$ generation

## D.5 Standby RAM Module with TPU Emulation Capability (TPURAM)

**Table D–5. TPURAM Address Map**

| Access | Address | 15 8 | 7 0 |
|---|---|---|---|
| S | $YFFB00 | TPURAM MODULE CONFIGURATION REGISTER (TRAMMCR) | |
| S | $YFFB02 | TPURAM TEST REGISTER (TRAMTST) | |
| S | $YFFB04 | TPURAM BASE ADDRESS AND STATUS REGISTER (TRAMBAR) | |
| S | $YFFB06 | NOT USED | |

TPURAM responds to both program and data space accesses. The RASP bit in the TRAMMCR determines whether the processor must be operating at the supervisor privilege level to access the array. TPURAM control registers are accessible at the supervisor privilege level only.

### D.5.1 TRAMMCR — TPURAM Module Configuration Register $YFFB00

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 — 0 |
|---|---|---|---|---|---|---|---|---|
| STOP | 0 | 0 | 0 | 0 | 0 | 0 | RASP | NOT USED |
| RESET: | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |

STOP — Stop Control

0 = TPURAM array operates normally.

1 = TPURAM array enters low-power stop mode.

This bit controls whether the RAM array is in stop mode or normal operation. Reset state is zero, for normal operation. In stop mode, the array retains its contents, but cannot be read or written by the CPU.

RASP[1:0] — TPURAM Array Space Field

0 = TPURAM array is accessible from the supervisor or user privilege level.

1 = TPURAM array is accessible from the supervisor privilege level only.

### D.5.2 TRAMTST — TPURAM Test Register $YFFB02

TRAMTST is used for factory test of the TPURAM module.

### D.5.3 TRAMBAR — TPURAM Base Address and Status Register $YFFB04

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 | 0 | 0 | 0 | RAMDS |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

ADDR[23:12] — TPURAM Array Base Address

These bits specify address lines ADDR[23:12] of the base address of the TPURAM array when enabled.

RAMDS — TPURAM Array Disable

0 = TPURAM array is enabled

1 = TPURAM array is disabled

The TPURAM array is disabled by internal logic after a master reset. Writing a valid base address to the TPURAM array base address field (bits [15:3]) automatically clears RAMDS, enabling the TPURAM array.

## D.6 Standby RAM Module (SRAM)

**Table D–6. SRAM Address Map**

| Access | Address | 15 8 | 7 0 |
|---|---|---|---|
| S | $YFFB40 | SRAM MODULE CONFIGURATION REGISTER (SRAMMCR) | |
| S | $YFFB42 | SRAM TEST REGISTER (SRAMTST) | |
| S | $YFFB44 | SRAM ARRAY BASE ADDRESS REGISTER HIGH (SRAMBAH) | |
| S | $YFFB46 | SRAM ARRAY BASE ADDRESS REGISTER LOW (SRAMBAL) | |

SRAM control registers are accessible at the supervisor privilege level only.

### D.6.1 SRAMMCR — Standby RAM Module Configuration Register $YFFB40

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| STOP | 0 | 0 | 0 | RLCK | 0 | RASP | | NOT USED | |
| RESET: | | | | | | | | | |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | | |

The SRAMMCR specifies normal or low-power stop operation for the SRAM array. The register also determines in which space the array resides and controls access to the base array registers. Reads of unimplemented bits return zeros. Writes do not affect unimplemented bits.

STOP — Stop Control

0 = SRAM array operates normally.

1 = SRAM array enters low-power stop mode.

This bit determines whether the SRAM array is in low-power stop mode. Reset state is one, leaving the array configured for low-power stop operation. In stop mode, the array retains its contents, but cannot be read or written by the CPU. This bit can be read or written at any time.

RLCK — RAM Base Address Lock

0 = SRAM base address registers can be written from IMB

1 = SRAM base address registers are locked

RLCK defaults to zero on reset; it can be written once to one.

RASP[1:0] — RAM Array Space

RASP locates the SRAM array in supervisor or user space and in program or data space.

D

**RASP Encoding**

| RASP | Space |
|---|---|
| 00 | Unrestricted Program and Data |
| 01 | Unrestricted Program |
| 10 | Supervisor Program and Data |
| 11 | Supervisor Program |

### D.6.2 SRAMTST — Standby RAM Test Register $YFFB42

SRAMTST is used for factory testing only. Reads of this register return zeros, and writes have no effect.

### D.6.3 SRAMBAH — Standby RAM Array Base Address Register High $YFFB44

| 15 | | | | | | | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | | | | | | | ADDR 23 | ADDR 22 | ADDR 21 | ADDR 20 | ADDR 19 | ADDR 18 | ADDR 17 | ADDR 16 |
| RESET: | | | | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### D.6.4 SRAMBAL — Standby RAM Array Base Address Register Low $YFFB46

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR 15 | ADDR 14 | ADDR 13 | ADDR 12 | ADDR 11 | ADDR 10 | ADDR 9 | ADDR 8 | ADDR 7 | ADDR 6 | ADDR 5 | ADDR 4 | ADDR 3 | ADDR 2 | ADDR 1 | ADDR 0 |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SRAMBAH and SRAMBAL are used to specify an SRAM array base address. SRAMBAH and SRAMBAL can only be written while the SRAM is in low-power mode (SRAMMCR STOP = 1) and the base address lock (SRAMMCR RLCK = 0) is disabled. This prevents accidental remapping of the array.

## D.7 Queued Serial Module (QSM)

**Table D–7. QSM Address Map**

| Access | Address | 15 8 | 7 0 |
|---|---|---|---|
| S | $YFFC00 | QSM MODULE CONFIGURATION (QSMCR) | |
| S | $YFFC02 | QSM TEST (QTEST) | |
| S | $YFFC04 | QSM INTERRUPT LEVEL (QILR) | QSM INTERRUPT VECTOR (QIVR) |
| S/U | $YFFC06 | NOT USED | |
| S/U | $YFFC08 | SCI CONTROL 0 (SCCR0) | |
| S/U | $YFFC0A | SCI CONTROL 1 (SCCR1) | |
| S/U | $YFFC0C | SCI STATUS (SCSR) | |
| S/U | $YFFC0E | SCI DATA (SCDR) | |
| S/U | $YFFC10 | NOT USED | |
| S/U | $YFFC12 | NOT USED | |
| S/U | $YFFC14 | NOT USED | PQS DATA (PORTQS) |
| S/U | $YFFC16 | PQS PIN ASSIGNMENT (PQSPAR) | PQS DATA DIRECTION (DDRQS) |
| S/U | $YFFC18 | SPI CONTROL 0 (SPCR0) | |
| S/U | $YFFC1A | SPI CONTROL 1 (SPCR1) | |
| S/U | $YFFC1C | SPI CONTROL 2 (SPCR2) | |
| S/U | $YFFC1E | SPI CONTROL 3 (SPCR3) | SPI STATUS (SPSR) |
| S/U | $YFFC20–$YFFCFF | NOT USED | |
| S/U | $YFFD00–$YFFD1F | RECEIVE RAM (RR[0:F]) | |
| S/U | $YFFD20–$YFFD3F | TRANSMIT RAM (TR[0:F]) | |
| S/U | $YFFD40–$YFFD4F | COMMAND RAM (CR[0:F]) | |

### D.7.1 QSMCR — QSM Configuration Register $YFFC00

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STOP | FRZ1 | FRZ0 | 0 | 0 | 0 | 0 | 0 | SUPV | 0 | 0 | 0 | IARB | |
| RESET: | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

QSMCR bits enable stop and freeze modes, and determine the arbitration priority of QSM interrupt requests.

STOP — Stop Enable

0 = Normal QSM clock operation

1 = QSM clock operation stopped

When STOP is set, the QSM enters low-power stop mode. System clock input to the module is disabled. While STOP is asserted, only QSMCR reads are guaranteed to be valid, but writes to QSPI RAM or any register are guaranteed valid. STOP is set during reset. The SCI receiver and transmitter must be disabled before STOP is set. To stop the QSPI, set the HALT bit in SPCR3, wait until the HALTA flag is set, then set STOP.

FRZ[1:0] — Freeze Control

0 = Ignore the FREEZE signal on the IMB

1 = Halt the QSPI (on a transfer boundary)

FRZ1 determines what action is taken by the QSPI when the FREEZE signal of the IMB is asserted. FREEZE is asserted whenever the CPU enters background mode. FRZ0 is reserved for future use.

SUPV — Supervisor/Unrestricted

0 = Assignable QSM registers permit unrestricted access.

1 = Assignable QSM registers permit supervisor access only.

IARB — Interrupt Arbitration

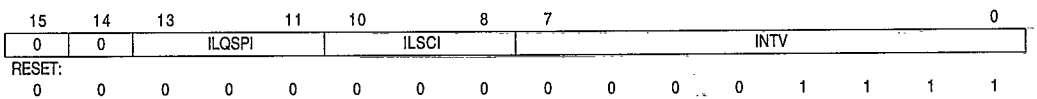
Each module that generates interrupts must have an IARB value. IARB values are used to arbitrate between interrupt requests of the same priority.

### D.7.2 QTEST — QSM Test Register $YFFC02

QTEST is used for factory testing only.

### D.7.3 QILR — QSM Interrupt Level Register $YFFC04

### QIVR — QSM Interrupt Vector Register $YFFC05




The values of the ILQSPI and ILSCI fields in QILR determine the priority of QSPI and SCI interrupt requests. QIVR determines the value of the interrupt vector number the QSM supplies when it responds to an interrupt acknowledge cycle. At reset, QIVR is initialized to vector number $0F, the uninitialized interrupt vector number. To use interrupt-driven serial communication, a user-defined vector number must be written to QIVR.

ILQSPI — Interrupt Level for QSPI

When an interrupt request is made, ILQSPI value determines which of the interrupt request signals is asserted; when a request is acknowledged, the QSM compares this value to a mask value supplied by the CPU32 to determine whether to respond. ILQSPI must have a value in the range $0 (lowest priority) to $7 (highest priority).

ILSCI — Interrupt Level for SCI

When an interrupt request is made, ILSCI value determines which of the interrupt request signals is asserted. When a request is acknowledged, the QSM 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 $0 (lowest priority) to $7 (highest priority).

If ILQSPI and ILSCI have the same nonzero value, and both submodules simultaneously request interrupt service, the QSPI has priority.

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 QSM. 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.

### D.7.4 PORTQS — Port QS Data Register $YFFC14

| 15 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | PQS7 | PQS6 | PQS5 | PQS4 | PQS3 | PQS2 | PQS1 | PQS0 |
| | | RESET: | | | | | | | |
| | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

PORTQS latches I/O data. Writes drive pins defined as outputs. Reads return data present on the pins. To avoid driving undefined data, first write a byte to PORTQS, then configure DDRQS.

### D.7.5 PQSPAR — PORT QS Pin Assignment Register $YFFC16

### DDRQS — PORT QS Data Direction Register $YFFC17

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | PQSPA6 | PQSPA5 | PQSPA4 | PQSPA3 | 0 | PQSPA1 | PQSPA0 | DDQS7 | DDQS6 | DDQS5 | DDQS4 | DDQS3 | DDQS2 | DDQS1 | DDQS0 |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Clearing a bit in the PQSPAR assigns the corresponding pin to general-purpose I/O; setting a bit assigns the pin to the QSPI. The PQSPAR does not affect operation of the SCI.

**PQSPAR Pin Assignments**

| PQSPAR Field | PQSPAR Bit | Pin Function |
|---|---|---|
| PQSPA0 | 0 | PQS0 |
| | 1 | MISO |
| PQSPA1 | 0 | PQS1 |
| | 1 | MOSI |
| — | 0 | PQS2[1] |
| PQSPA3 | 1 | SCK |
| | 0 | PQS3 |
| | 1 | PCS0/$\overline{SS}$ |
| PQSPA4 | 0 | PQS4 |
| | 1 | PCS1 |
| PQSPA5 | 0 | PQS5 |
| | 1 | PCS2 |
| PQSPA6 | 0 | PQS6 |
| | 1 | PCS3 |
| — | 0 | PQS7[2] |
| | 1 | TXD |

NOTES:

1. PQS2 is a digital I/O pin unless the SPI is enabled (SPE in SPCR1 set), in which case it becomes SPI serial clock SCK.
2. PQS7 is a digital I/O pin unless the SCI transmitter is enabled (TE in SCCR1 = 1), in which case it becomes SCI serial output TXD.

DDRQS determines whether 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.

**Effect of DDRQS on PORTQS Pins**

| Pin | DDRQS Bit | Pin Function |
|---|---|---|
| PQS0 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS1 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS2 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS3 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS4 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS5 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS6 | 0 | Digital Input |
| | 1 | Digital Output |
| PQS7 | 0 | Digital Input |
| | 1 | Digital Output |

D

**Effect of DDRQS on QSM Pin Function**

| QSM 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 | 0 | Disables Clock Output |
| | | | 1 | Clock Output from QSPI |
| | Slave | | 0 | Clock Input to QSPI |
| | | | 1 | Disables Clock Input |
| PCS0/$\overline{SS}$ | Master | DDQS3 | 0 | Assertion Causes Mode Fault |
| | | | 1 | Chip-Select Output |
| | Slave | | 0 | QSPI Slave Select Input |
| | | | 1 | Disables Select Input |
| PCS[3:1] | Master | DDQS[4:6] | 0 | Disables Chip-Select Output |
| | | | 1 | Chip-Select Output |
| | Slave | | 0 | Inactive |
| | | | 1 | Inactive |
| TXD[2] | Transmit | DDQS7 | X | Serial Data Output from SCI |
| RXD | Receive | None | NA | Serial Data Input to SCI |

NOTES:

1. PQS2 is a digital I/O pin unless the SPI is enabled (SPE in SPCR1 set), in which case it becomes SPI serial clock SCK.
2. PQS7 is a digital I/O pin unless the SCI transmitter is enabled (TE in SCCR1 =1), in which case it becomes SCI serial output TXD.

DDRQS determines the direction of the TXD pin only when the SCI transmitter is disabled. When the SCI transmitter is enabled, the TXD pin is an output.

### D.7.6 SPCR0 — QSPI Control Register 0 $YFFC18

| 15 | 14 | 13 | | | | 10 | 9 | 8 | 7 | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSTR | WOMQ | BITS | | | | | CPOL | CPHA | SP | | | | | | | |
| RESET: | | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

SPCR0 contains parameters for configuring the QSPI and enabling various modes of operation. The CPU has read/write access to SPCR0, but the QSM 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.

MSTR — Master/Slave Mode Select

0 = QSPI is a slave device.

1 = QSPI is system master.

WOMQ — Wired-OR Mode for QSPI Pins

0 = Outputs have normal MOS drivers.

1 = Pins designated for output by DDRQS have open-drain drivers.

BITS — Bits Per Transfer

The BITS field determines the number of serial data bits transferred.

CPOL — Clock Polarity

0 = The inactive state value of SCK is logic level zero.

1 = The inactive state value of SCK is logic level one.

CPHA — Clock Phase

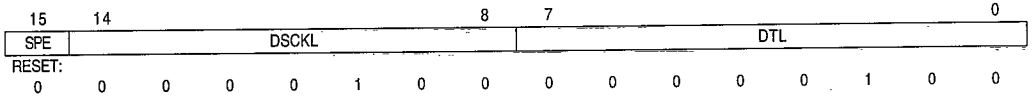
0 = Data captured on the leading edge of SCK and changed on the following edge of SCK.

1 = Data is changed on the leading edge of SCK and captured on the following edge of SCK.

SPBR — Serial Clock Baud Rate

QSPI baud rate is selected by writing a value from 2 to 255 into SPBR. Giving BR a value of zero or one disables SCK (disable state determined by CPOL). At reset, BAUD is initialized to a 2.1-MHz SCK frequency.

### D.7.7 SPCR1 — QSPI Control Register 1 $YFFC1A




SPCR1 enables the QSPI and specified transfer delays. The CPU has read/write access to SPCR1, but the QSM has read access only to all bits except enable bit SPE. SPCR1 must be written last during initialization because it contains SPE. Writing a new value to SPCR1 while the QSPI is enabled disrupts operation.

SPE — QSPI Enable

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.

DSCKL — Delay before SCK

When the DSCK bit in command RAM is set, this field determines the length of delay from PCS valid to SCK transition. PCS can be any of the four peripheral chip-select pins.

DTL — Length of Delay after Transfer

When the DT bit in command RAM is set, this field determines the length of delay after serial transfer.

### D.7.8 SPCR2 — QSPI Control Register 2 $YFFC1C

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPIFIE | WREN | WRTO | 0 | ENDQP | | | | 0 | 0 | 0 | 0 | NEWQP | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SPCR2 contains QSPI queue pointers, wraparound mode control bits, and an interrupt enable bit. The CPU has read/write access to SPCR2, but the QSM has read access only. SPCR2 is 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. SPCR2 reads return the value of the register, not the buffer.

SPIFIE — SPI Finished Interrupt Enable
- 0 = QSPI interrupts disabled
- 1 = QSPI interrupts enabled

WREN — Wrap Enable
- 0 = Wraparound mode disabled
- 1 = Wraparound mode enabled

WRTO — Wrap To
- 0 = Wrap to pointer address $0
- 1 = Wrap to address in NEWQP

ENDQP — Ending Queue Pointer
This field contains the last QSPI queue address.

NEWQP — New Queue Pointer Value
This field contains the first QSPI queue address.

### D.7.9 SPCR3 — QSPI Control Register 3 $YFFC1E
### SPSR — QSPI Status Register $YFFC1F

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | LOOPQ | HMIE | HALT | SPIF | MODF | HALTA | 0 | CPTQP | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SPCR3 contains the loop mode enable bit, halt and mode fault interrupt enables, and the halt control bit. The CPU has read/write access to SPCR3, but the QSM 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. The SPSR contains information concerning the current serial transmission. Only the QSPI can set bits in the SPSR. The CPU reads the SPSR to obtain QSPI status information and writes it to clear status flags.

LOOPQ — QSPI Loop Mode
0 = Feedback path disabled
1 = Feedback path enabled

HMIE — HALTA and MODF Interrupt Enable
0 = HALTA and MODF interrupts disabled
1 = HALTA and MODF interrupts enabled

HALT — Halt
0 = Halt not enabled
1 = Halt enabled

SPIF — QSPI Finished Flag
0 = QSPI not finished
1 = QSPI finished

MODF — Mode Fault Flag
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).

HALTA — Halt Acknowledge Flag
0 = QSPI not halted
1 = QSPI halted

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 ($0) or a pointer to the last command completed in the previous queue.

### D.7.10 RR[0:F] — Receive Data RAM $YFFD00–$YFFD0E

Data received by the QSPI is stored in this segment. The CPU reads this segment to retrieve data from the QSPI. Data stored in receive RAM is right-justified. 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, word, or long-word addressing.

### D.7.11 TR[0:F] — Transmit Data RAM $YFFD20–$YFFD3E

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.

Information to be transmitted must be written to transmit data RAM in a right-justified format. The QSPI cannot modify information in the transmit data RAM. The QSPI copies the information to its data serializer for transmission. Information remains in transmit RAM until overwritten.

## D.7.12 CR[0:F] — Command RAM $YFFD40–$YFFD4F

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CONT | BITSE | DT | DSCK | PCS3 | PCS2 | PCS1 | PCS0* |
| — | — | — | — | — | — | — | — |
| CONT | BITSE | DT | DSCK | PCS3 | PCS2 | PCS1 | PCS0* |

COMMAND CONTROL | PERIPHERAL CHIP SELECT

*The PCS0 bit represents the dual-function PCS0/$\overline{SS}$.

Command RAM is used by the QSPI when 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 16 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 16 commands can be in the queue. Queue execution proceeds from the address in NEWQP through the address in ENDQP (both of these fields are in SPCR2).

CONT — Continue

0 = Chip select output values stored in PORTQS between transfers.
1 = Peripheral chip selects remain asserted after transfer is complete.

BITSE — Bits per Transfer Enable

0 = Eight bits
1 = Number of bits set in BITS field of SPCR0

DT — Delay after Transfer

The QSPI provides a variable delay at the end of serial transfer to facilitate interfacing with peripherals that have a latency requirement. The delay between transfers is determined by the SPCR1 DTL field.

DSCK — PCS to SCK Delay

0 = PCS valid to SCK transition is one-half SCK.
1 = SPCR1 DSCKL field specifies delay from PCS valid to SCK.

PCS[3:0] — Peripheral Chip Select

Peripheral chip-select bits are used 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.

D

### D.7.13 SCCR0 — SCI Control Register 0 $YFFC08

| 15 | 14 | 13 | 12 | | | | | | | | | | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | SCBR | | | | | | | | | | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

SCCR0 contains the SCI baud rate selection field. Baud rate must be set before the SCI is enabled. The CPU can read and write SCCR0 at any time. Changing the value of SCCR0 bits during a transfer operation can disrupt operation.

SCBR — SCI Baud Rate

SCI baud rate is programmed by writing a 13-bit value to this field. Writing a value of zero to SCBR disables the baud rate generator. Baud clock rate is calculated as follows:

SCI Baud Clock Rate = System Clock/(32SCBR)

### D.7.14 SCCR1 — SCI Control Register 1 $YFFC0A

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 |

SCCR1 contains SCI configuration parameters, including transmitter and receiver enable bits, interrupt enable bits, and operating mode enable bits. The CPU can read and write SCCR0 at any time. The SCI can modify the RWU bit under certain circumstances. Changing the value of SCCR1 bits during a transfer operation can disrupt operation.

LOOPS — Loop Mode

0 = Normal SCI operation, no looping, feedback path disabled
1 = Test SCI operation, looping, feedback path enabled

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.

ILT — Idle-Line Detect Type

0 = Short idle-line detect (start count on first one)
1 = Long idle-line detect (start count on first one after stop bit(s))

PT — Parity Type

0 = Even parity
1 = Odd parity

PE — Parity Enable
0 = SCI parity disabled
1 = SCI parity enabled

M — Mode Select
0 = 10-bit SCI frame
1 = 11-bit SCI frame

WAKE — Wakeup by Address Mark
0 = SCI receiver awakened by idle-line detection
1 = SCI receiver awakened by address mark (last bit set)

TIE — Transmit Interrupt Enable
0 = SCI TDRE interrupts inhibited
1 = SCI TDRE interrupts enabled

TCIE — Transmit Complete Interrupt Enable
0 = SCI TC interrupts inhibited
1 = SCI TC interrupts enabled

RIE — Receiver Interrupt Enable
0 = SCI RDRF and OR interrupts inhibited
1 = SCI RDRF and OR interrupts enabled

ILIE — Idle-Line Interrupt Enable
0 = SCI IDLE interrupts inhibited
1 = SCI IDLE interrupts enabled

TE — Transmitter Enable
0 = SCI transmitter disabled (TXD pin can be used as I/O)
1 = SCI transmitter enabled (TXD pin dedicated to SCI transmitter)

RE — Receiver Enable
0 = SCI receiver disabled (status bits inhibited)
1 = SCI receiver enabled

RWU — Receiver Wakeup
0 = Normal receiver operation (received data recognized)
1 = Wakeup mode enabled (received data ignored until awakened)

SBK — Send Break
0 = Normal operation
1 = Break frame(s) transmitted after completion of current frame

## D.7.15 SCSR — SCI Status Register

**$YFFC0C**

| 15 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| NOT USED | | TDRE | TC | RDRF | RAF | IDLE | OR | NF | FE | PF |
| RESET: | | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The SCSR contains flags that show SCI operating conditions. These flags are cleared either by SCI hardware or by a CPU read/write sequence. The sequence consists of reading the SCSR, then reading or writing the SCDR.

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 written or read the SCDR, the newly set status bit is not cleared. The SCSR must be read again with the bit set and the SCDR must be written or read before the status bit is cleared.

A long-word read can consecutively access both the SCSR and the SCDR. This action clears receive status flag bits that were set at the time of the read, but does not clear TDRE or TC flags. Reading either byte of the SCSR causes all 16 bits to be accessed, and any status bit already set in either byte is cleared on a subsequent read or write of the SCDR.

TDRE — Transmit Data Register Empty
- 0 = Register TDR still contains data to be sent to the transmit serial shifter.
- 1 = A new character can now be written to the TDR.

TC — Transmit Complete
- 0 = SCI transmitter is busy.
- 1 = SCI transmitter is idle.

RDRF — Receive Data Register Full
- 0 = The RDR is empty or contains previously read data.
- 1 = The RDR contains new data.

RAF — Receiver Active Flag
- 0 = SCI receiver is idle.
- 1 = SCI receiver is busy.

IDLE — Idle-Line Detected
- 0 = SCI receiver did not detect an idle-line condition.
- 1 = SCI receiver detected an idle-line condition.

OR — Overrun Error
- 0 = RDRF is cleared before new data arrives.
- 1 = RDRF is not cleared before new data arrives.

NF — Noise Error Flag
0 = No noise detected on the received data
1 = Noise occurred on the received data

FE — Framing Error
1 = Framing error or break occurred on the received data
0 = No framing error on the received data

PF — Parity Error
1 = Parity error occurred on the received data
0 = No parity error on the received data

### D.7.16 SCDR — SCI Data Register $YFFC0E

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 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 |

The SCDR consists of two data registers located at the same address. The RDR is a read-only register that contains data received by the SCI serial interface. Data comes into the receive serial shifter and is transferred to the RDR. The TDR is a write-only register that contains data to be transmitted. Data is first written to the TDR, then transferred to the transmit serial shifter, where additional format bits are added before transmission. R[7:0]/T[7:0] contain either the first eight data bits received when the SCDR is read, or the first eight data bits to be transmitted when the SCDR is written. R8/T8 are used when the SCI is configured for 9-bit operation. When the SCI is configured for 8-bit operation, R8/T8 have no meaning or effect.

**Table D–8. TPU Address Map**

| Access | Address | 15 8 | 7 0 |
|---|---|---|---|
| S | $YFFE00 | TPU MODULE CONFIGURATION REGISTER (TMCR) | |
| S | $YFFE02 | TEST CONFIGURATION REGISTER (TCR) | |
| S | $YFFE04 | DEVELOPMENT SUPPORT CONTROL REGISTER (DSCR) | |
| S | $YFFE06 | DEVELOPMENT SUPPORT STATUS REGISTER (DSSR) | |
| S | $YFFE08 | TPU INTERRUPT CONFIGURATION REGISTER (TICR) | |
| S | $YFFE0A | CHANNEL INTERRUPT ENABLE REGISTER (CIER) | |
| S | $YFFE0C | CHANNEL FUNCTION SELECTION REGISTER 0 (CFSR0) | |
| S | $YFFE0E | CHANNEL FUNCTION SELECTION REGISTER 1 (CFSR1) | |
| S | $YFFE10 | CHANNEL FUNCTION SELECTION REGISTER 2 (CFSR2) | |
| S | $YFFE12 | CHANNEL FUNCTION SELECTION REGISTER 3 (CFSR3) | |
| S/U | $YFFE14 | HOST SEQUENCE REGISTER 0 (HSQR0) | |
| S/U | $YFFE16 | HOST SEQUENCE REGISTER 1 (HSQR1) | |
| S/U | $YFFE18 | HOST SERVICE REQUEST REGISTER 0 (HSRR0) | |
| S/U | $YFFE1A | HOST SERVICE REQUEST REGISTER 1 (HSRR1) | |
| S | $YFFE1C | CHANNEL PRIORITY REGISTER 0 (CPR0) | |
| S | $YFFE1E | CHANNEL PRIORITY REGISTER 1 (CPR1) | |
| S | $YFFE20 | CHANNEL INTERRUPT STATUS REGISTER (CISR) | |
| S | $YFFE22 | LINK REGISTER (LR) | |
| S | $YFFE24 | SERVICE GRANT LATCH REGISTER (SGLR) | |
| S | $YFFE26 | DECODED CHANNEL NUMBER REGISTER (DCNR) | |

Y = M111, where M is the state of the MM (module mapping) bit in the SCIMCR (Y = $7 or $F)

### D.8.1 TPUMCR — TPU Module Configuration Register $YFFE00

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STOP | TCR1P | | TCR2P | | EMU | T2CG | STF | SUPV | PSCK | 0 | 0 | IARB | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

STOP — Stop Bit

0 = TPU operating normally

1 = Internal clocks shut down

TCR1P — Timer Count Register 1 Prescaler Control

TCR1 is clocked from the output of a prescaler. The prescaler's input is the internal TPU system clock divided by either 4 or 32, depending on the value of the PSCK bit. The prescaler divides this input by 1, 2, 4, or 8. Channels using TCR1 have the capability to resolve down to the TPU system clock divided by 4.

| | | PSCK = 0 | | PSCK = 1 | |
|---|---|---|---|---|---|
| TCR1 Prescaler | Divide By | Number of Clocks | Rate at 16 MHz | Number of Clocks | Rate at 16 MHz |
| 00 | 1 | 32 | 2 ms | 4 | 250 ns |
| 01 | 2 | 64 | 4 ms | 8 | 500 ns |
| 10 | 4 | 128 | 8 ms | 16 | 1 ms |
| 11 | 8 | 256 | 16 ms | 32 | 2 ms |

TCR2P — Timer Count Register 2 Prescaler Control

TCR2 is clocked from the output of a prescaler. If T2CG = 0, the input to the TCR2 prescaler is the external TCR2 clock source. If T2CG = 1, the input is the TPU system clock divided by eight. The TCR2 field specifies the value of the prescaler: 1, 2, 4, or 8. Channels using TCR2 have the capability to resolve down to the TPU system clock divided by 8. The following table is a summary of prescaler output.

| 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 |

EMU — Emulation Control

In emulation mode, the TPU executes microinstructions from TPURAM exclusively. Access to the TPURAM module through the IMB by a host is blocked, and the TPURAM module is dedicated for use by the TPU. After reset, this bit can be written only once.

0 = TPU and TPURAM not in emulation mode
1 = TPU and TPURAM in emulation mode

T2CG — TCR2 Clock/Gate Control

When the T2CG bit is set, the external TCR2 pin functions as a gate of the DIV8 clock (the TPU system clock divided by 8). 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.

0 = TCR2 pin used as clock source for TCR2
1 = TCR2 pin used as gate of DIV8 clock for TCR2

STF — Stop Flag

0 = TPU operating
1 = TPU stopped (STOP bit has been asserted)

SUPV — Supervisor Data Space

0 = Assignable registers are unrestricted (FC2 is ignored)

1 = Assignable registers are restricted (FC2 is decoded)

PSCK — Prescaler Clock

0 = System clock/32 is input to TCR1 prescaler

1 = System clock/4 is input to TCR1 prescaler

IARB — Interrupt Arbitration Number

This field contains the arbitration number of the TPU that is used to arbitrate for the intermodule bus when two or more modules or peripherals have an interrupt on the same priority level.

### D.8.2 TICR — TPU Interrupt Configuration Register $YFFE08

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | CIRL | | | CIBV | | | | 0 | 0 | 0 | 0 |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

CIRL — Channel Interrupt Request Level

This 3-bit encoded field specifies the interrupt request level for all channels. Level seven for this field indicates a nonmaskable interrupt; level zero indicates that all channel interrupts are disabled.

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.

### D.8.3 CIER — Channel Interrupt Enable Register $YFFE0A

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 |

CH[15:0] — Channel Interrupt Enable/Disable

0 = Channel interrupts disabled

1 = Channel interrupts enabled

### D.8.4 CISR — Channel Interrupt Status Register $YFFE20

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 |

CH[15:0] — Channel Interrupt Status Bit
0 = Channel interrupt not asserted
1 = Channel interrupt asserted

### D.8.5 CFSR0 — Channel Function Select Register 0 $YFFE0C

| 15 | | | 12 | 11 | | | 8 | 7 | | | 4 | 3 | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHANNEL15 | | | | CHANNEL14 | | | | CHANNEL13 | | | | CHANNEL12 | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### D.8.6 CFSR1 — Channel Function Select Register 1 $YFFE0E

| 15 | | | 12 | 11 | | | 8 | 7 | | | 4 | 3 | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHANNEL11 | | | | CHANNEL10 | | | | CHANNEL9 | | | | CHANNEL8 | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### D.8.7 CFSR2 — Channel Function Select Register 2 $YFFE10

| 15 | | | 12 | 11 | | | 8 | 7 | | | 4 | 3 | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHANNEL7 | | | | CHANNEL6 | | | | CHANNEL5 | | | | CHANNEL4 | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### D.8.8 CFSR3 — Channel Function Select Register 3 $YFFE12

| 15 | | | 12 | 11 | | | 8 | 7 | | | 4 | 3 | | | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHANNEL3 | | | | CHANNEL2 | | | | CHANNEL1 | | | | CHANNEL0 | | | |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

CHANNEL[15:0] — Encoded Time Function for each Channel
Encoded 4-bit fields in the channel function select registers specify one of 16 time functions to be executed on the corresponding channel. Encodings for predefined functions are found in the table **Host Service Request and Sequence Codes**.

### D.8.9 HSQR0 — Host Sequence Register 0 $YFFE14

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 |

### D.8.10 HSQR1 — Host Sequence Register 1 $YFFE16

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 |

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. Refer to the table, **Host Service Request and Sequence Codes**, which is a summary of the host sequence and host service request bits for each time function.

### D.8.11 HSRR0 — Host Service Request Register 0 $YFFE18

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 |

### D.8.12 HSRR1 — Host Service Request Register 1 $YFFE1A

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 |

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. The table, **Host Service Request and Sequence Codes**, is a summary of the host sequence and host service request bits for each time function.

## Host Service Request and Sequence Codes

| Function Name | Function Code | Host Service Request Code | Host Sequence Code* |
|---|---|---|---|
| DIO<br>Discrete Input/Output | $8 | 1= Force Output High<br>2 = Force Output Low<br>3 = Initialization, Input Specified<br>3 = Initialization, Periodic Input<br>3 = Update Pin Status Parameter | 0 = Trans Mode — Record Pin on Transition<br>0 = Trans Mode — Record Pin on Transition<br>0 = Trans Mode — Record Pin on Transition<br>1 = Match Mode — Record Pin at Match_Rate<br>2 = Record Pin State on HSR 11 |
| ITC<br>Input Capture/<br>Input Transition Counter | $A | 0 = None<br>1 = Initialization<br>2 = (Not Implemented)<br>3 = (Not Implemented) | 0 = No Link, Single Mode<br>1 = No Link, Continuous Mode<br>2 = Link, Single Mode<br>3 = Link, Continuous Mode |
| OC<br>Output Compare | $E | 0 = None<br>1 = Host-Initiated Pulse Mode<br>2 = (Not Implemented)<br>3 = Continuous Pulse Mode | 0 = Execute All Functions<br>1 = Execute All Functions<br>2 = Only Update TCRn Parameters<br>3 = Only Update TCRn Parameters |
| PWM<br>Pulse-Width Modulation | $9 | 0 = None<br>1 = Immediate Update Request<br>2 = Initialization<br>3 = (Not Implemented) | (None Implemented) |
| SPWM<br>Synchronized Pulse-Width Modulation | $7 | 0 = None<br>1 = (Not Implemented)<br>2 = Initialization<br>3 = Immediate Update Request | 0 = Mode 0<br>1 = Mode 1<br>2 = Mode 2<br>3 = (Not Implemented) |
| PMA/PMM<br>Period Measurement with Additional/Missing Transition Detect | $B | 0 = None<br>1 = Initialization<br>2 = (Not Implemented)<br>3 = (Not Implemented) | 0 = PMA Bank Mode<br>1 = PMA Count Mode<br>2 = PMM Bank Mode<br>3 = PMM Count Mode |
| PSP<br>Position-Synchronized Pulse Generator | $C | 0 = None<br>1 = Immediate Update Request<br>2 = Initialization<br>3 = Force Change | 0 = Pulse Width Set by Angle<br>1 = Pulse Width Set by Time<br>2 = Pulse Width Set by Angle<br>3 = Pulse Width Set by Time |
| SM<br>Stepper Motor | $D | 0 = None<br>1 = None<br>2 = Initialization<br>3 = Step Request | (None Implemented) |
| PPWA<br>Period/Pulse-Width Accumulator | $F | 0 = None<br>1 = (Not Implemented)<br>2 = Initialization<br>3 = (Not Implemented) | 0 = 24-Bit Period<br>1 = 16-Bit Period + Link<br>2 = 24-Bit Pulse Width<br>3 = 16-Bit Pulse Width + Link |

*Host sequence code interpretation is determined by the function; some HSQ codes apply to all HSR codes, some, such as Init, apply to only one.

A host service request field cleared to %00 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 nonzero states. The CPU should monitor the host service request register until the TPU clears the service request to %00 before the CPU changes any parameters or issues a new service request to the channel.

### D.8.13 CPR0 — Channel Priority Register 0 $YFFE1C

| 15 14 | 13 12 | 11 10 | 9 8 | 7 6 | 5 4 | 3 2 | 1 0 |
|---|---|---|---|---|---|---|---|
| CH 15 | CH 14 | CH13 | 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 |

### D.8.14 CPR1 — Channel Priority Register 1 $YFFE1E

| 15 14 | 13 12 | 11 10 | 9 8 | 7 6 | 5 4 | 3 2 | 1 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 |

CH[15:0] — Encoded One of Three Channel Priority Levels

| 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 |

### D.8.15 Parameter RAM

The channel parameter registers are organized as one hundred 16-bit words of RAM. Channels 0 to 13 have six parameters. Channels 14 and 15 each have eight parameters. The parameter registers constitute a shared work space for communication between the bus master and the TPU. The CPU specifies attributes of the time function being executed by writing the associated parameter registers. The TPU reads the registers at the appropriate time to determine channel operation. The TPU also stores information in the parameter registers to be read by the CPU. The TPU cannot access address space outside the TPU module. Consequently, the CPU must provide all such access.

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) for additional information.

## Parameter RAM Address Map

| Channel Number | Base Address | Parameter | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | $YFFF## | 00 | 02 | 04 | 06 | 08 | 0A | — | — |
| 1 | $YFFF## | 10 | 12 | 14 | 16 | 18 | 1A | — | — |
| 2 | $YFFF## | 20 | 22 | 24 | 26 | 28 | 2A | — | — |
| 3 | $YFFF## | 30 | 32 | 34 | 36 | 38 | 3A | — | — |
| 4 | $YFFF## | 40 | 42 | 44 | 46 | 48 | 4A | — | — |
| 5 | $YFFF## | 50 | 52 | 54 | 56 | 58 | 5A | — | — |
| 6 | $YFFF## | 60 | 62 | 64 | 66 | 68 | 6A | — | — |
| 7 | $YFFF## | 70 | 72 | 74 | 76 | 78 | 7A | — | — |
| 8 | $YFFF## | 80 | 82 | 84 | 86 | 88 | 8A | — | — |
| 9 | $YFFF## | 90 | 92 | 94 | 96 | 98 | 9A | — | — |
| 10 | $YFFF## | A0 | A2 | A4 | A6 | A8 | AA | — | — |
| 11 | $YFFF## | B0 | B2 | B4 | B6 | B8 | BA | — | — |
| 12 | $YFFF## | C0 | C2 | C4 | C6 | C8 | CA | — | — |
| 13 | $YFFF## | D0 | D2 | D4 | D6 | D8 | DA | — | — |
| 14 | $YFFF## | E0 | E2 | E4 | E6 | E8 | EA | EC | EE |
| 15 | $YFFF## | F0 | F2 | F4 | F6 | F8 | FA | FC | FE |

— = Not Implemented

Y = M111, where M is the module mapping (MM) bit in the SCIMCR (Y = $7 or $F).

One of the parameter registers associated with each channel contains channel control fields. There are three such fields: pin state control (PSC), pin action control (PAC), and time base/directionality control (TBS). These fields perform the following functions:

PSC — Forces the output level of the pin.

PAC — On input channels, PAC specifies the transition edge to be detected. On output channels, PAC specifies the logic level to be output to the pin because of a match: a greater-than-or-equal-to comparison.

TBS — Specifies both the channel direction (input or output) and the time base (TCR1 or TCR2) associated with the input capture and output compare function of each channel.

### D.8.16 DSCR — Development Support Control Register $YFFE04

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HOT4 | 0 | 0 | 0 | 0 | BLC | CLKS | FRZ1 | FRZ0 | CCL | BP | BC | BH | BL | BM | BT |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

HOT4 — Hang on T4

0 = Exit wait on T4 state caused by assertion of HOT4

1 = Enter wait on T4 state

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.

CLKS — Stop Clocks (to TCRs)

0 = Do not stop TCRs.

1 = Stop TCRs during the halted state.

FRZ[1:0] — IMB FREEZE Response

The FRZ bits specify the TPU microengine response to the FREEZE signal.

| FRZ[1:0] | TPU Response |
|---|---|
| 00 | Ignore Freeze |
| 01 | Reserved |
| 10 | Freeze at End of Current Microcycle |
| 11 | Freeze at Next Time-Slot Boundary |

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.

BP, BC, BH, BL, BM, and B — Breakpoint Enable Bits

DSCR[5:0] are TPU breakpoint enables. Setting a bit enables a breakpoint condition.

BP — Break if µPC equals µPC breakpoint register.

BC — Break if CHAN register equals channel breakpoint register at beginning of state or when CHAN is changed through microcode.

BH — Break if host service latch is asserted at beginning of state.

BL — Break if link service latch is asserted at beginning of state.

BM — Break if MRL is asserted at beginning of state.

BT — Break if TDL is asserted at beginning of state.

### D.8.17 DSSR — Development Support Status Register $YFFE06

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BKPT | PCBK | CHBK | SRBK | TPUF | 0 | 0 | 0 |
| RESET: | | | | | | | | | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

BKPT — Breakpoint Asserted Flag

If an internal breakpoint caused the TPU to enter the halted state, the TPU asserts the BKPT signal on the IMB and the BKPT flag. The TPU continues to assert BKPT until it recognizes a breakpoint acknowledge cycle from a host, or until the FREEZE signal on the IMB is asserted.

PCBK — μPC Breakpoint Flag

PCBK is asserted if a breakpoint occurs because of a μPC register match with the μPC breakpoint register. PCBK is negated when the BKPT flag is negated.

CHBK — Channel Register Breakpoint Flag

CHBK is asserted if a breakpoint occurs because of a CHAN register match with the channel register breakpoint register. CHBK is negated when the BKPT flag is negated.

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 negated.

TPUF — TPU FREEZE Flag

TPUF is asserted whenever the TPU is in a halted state as a result of FREEZE being asserted. This flag is automatically negated when the TPU exits the halted state because of FREEZE being negated.

### D.8.18 LR — Link Register $YFFE22

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 |

CH[15:0] — Test Mode Link Service Request Enable Bit

0 = Link bit not asserted

1 = Link bit asserted

D

### D.8.19 SGLR — Service Grant Latch Register

**$YFFE24**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 |

CH[15:0] — Service Granted Bits

### D.8.20 DCNR — Decoded Channel Number Register

**$YFFE26**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 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 |

CH[15:0] — Service Status Bits

### D.8.21 TCR — Test Configuration Register

**$YFFE02**

The TCR is used for factory test of the MCU.

## D.9 Register Bit and Field Mnemonics

### Table D–9. Register Bit and Field Mnemonics

| Mnemonic | Name | Register Location |
|---|---|---|
| ADDR[23:11] | Base Address | CSBAR[0:10], CSBARBT, TRAMBAR |
| ADDR[15:0] | SRAM Array Base Address | SRAMBAL |
| ADDR[31:16] | SRAM Array Base Address | SRAMBAH |
| ASPC | Flash EEPROM Array Space | FEE1MCR, FEE2MCR |
| $\overline{\text{AVEC}}$ | Autovector Enable | CSOR[0:10], CSORBT |
| BP, BC, BH, BL, BM, BT | Breakpoint Enable Bits | DSCR |
| BITS | Bits Per Transfer | SPCR0 |
| BITSE | Bits Per Transfer Enable | CR[0:F] |
| BKPT | Breakpoint Asserted Flag | DSSR |
| BLC | Branch Latch Control | DSCR |
| BLKSZ | Block Size | CSBAR[0:10], CSBARBT |
| BME | Bus Monitor External Enable | SYPCR |
| BMT[1:0] | Bus Monitor Timing | SYPCR |
| BOOT | Boot Control | FEE1MCR, FEE2MCR |
| BYTE | Upper/Lower Byte Option | CSOR[0:10], CSORBT |
| C | Carry Flag | SR |
| CCF[7:0] | Conversion Complete Flags | ADSTAT |
| CCL | Channel Conditions Latch | DSCR |
| CCTR[2:0] | Conversion Counter | ADSTAT |
| CD:CA | Channel Selection | ADCTL1 |
| CFSR[15:0] | Channel Time Function | CFSR[0:3] |
| CH[15:0] | Channel Function Select | CFSR[0:3] |
| CH[15:0] | Channel Interrupt Enable | CIER |
| CH[15:0] | Channel Interrupt Status | CISR |
| CH[15:0] | Channel Priority | CPR[0:1] |
| CH[15:0] | Decoded Channel Number | DCNR |
| CH[15:0] | Host Sequence | HSQR[0:1] |
| CH[15:0] | Host Service Request | HSRR[0:1] |
| CH[15:0] | Link | LR |
| CH[15:0] | Service Grant Latch | SGLR |
| CHBK | Channel Register Breakpoint Flag | DSSR |
| CIBV | Channel Interrupt Base Vector | TICR |
| CIRL | Channel Interrupt Request Level | TICR |
| CLKS | Stop Clocks | DSCR |
| CONT | Continue | CR[0:F] |
| CPHA | Clock Phase | SPCR0 |

## Table D–9. Register Bit and Field Mnemonics (Continued)

| Mnemonic | Name | Register Location |
|---|---|---|
| CPOL | Clock Polarity | SPCR0 |
| CPTQP | Completed Queue Pointer | SPSR |
| CSPA0[6:1] | Chip-Select [6:1] | CSPAR0 |
| CSPA1[4:0] | Chip-Select [4:0] | CSPAR1 |
| $\overline{\text{CSBOOT}}$ | Boot ROM Chip Select | CSPAR0 |
| DDQS[7:0] | Data Direction | DDRQS |
| $\overline{\text{DSACK}}$ | Data Strobe Acknowledge | CSOR[0:10], CSORBT |
| DSCK | PCS to SCK Delay | CR[0:F] |
| DSCKL | Delay Before SCK | SPCR1 |
| DT | Delay After Transfer | CR[0:F] |
| DTL | Length of Delay After Transfer | SPCR1 |
| EDIV | ECLK Divide Rate | SYNCR |
| EMU | Emulation Control | TPUMCR |
| ENPE | Enable Programming/Erase | FEE1CTL, FEE2CTL |
| ENDQP | Ending Queue Pointer | SPCR2 |
| ERAS | Erase Control | FEE1CTL, FEE2CTL |
| EV | Extension Bit Overflow Flag | CCR |
| EXOFF | External Clock Off | SCIMCR |
| EXT | External Reset | RSR |
| FE | Framing Error | SCSR |
| FRZ | Freeze Control | DSCR, ADCMCR, QSMCR, FEE1MCR, FEE2MCR |
| FRZBM | Freeze Bus Monitor Enable | SCIMCR |
| FRZSW | Freeze Software Enable | SCIMCR |
| HALT | Halt | SPCR3 |
| HALTA | Halt Acknowledge Flag | SPSR |
| HLT | Halt Monitor Reset | RSR |
| HME | Halt Monitor Enable | SYPCR |
| HMIE | HALTA and MODF Interrupt Enable | SPCR3 |
| HOT4 | Hang on T4 | DSCR |
| IARB | Interrupt Arbitration | QSMCR, SCIMCR, TPUMCR |
| IDLE | Idle-Line Detected | SCSR |
| ILIE | Idle-Line Interrupt Enable | SCCR1 |
| ILQSPI | Interrupt Level for QSPI | QILR |
| ILSCI | Interrupt Level for SCI | QILR |
| ILT | Idle-Line Detect Type | SCCR1 |
| INTV[7:0] | Interrupt Vector Number | QIVR |
| IPL | Interrupt Priority Level | CSOR[0:10], CSORBT |
| IP[2:0] | Interrupt Priority Field | SR |
| LAT | Latch Control | FEE1CTL, FEE2CTL |
| LOC | Loss of Clock Reset | RSR |
| LOCK | Lock Registers | FEE1MCR, FEE2MCR |

## Table D–9. Register Bit and Field Mnemonics (Continued)

| Mnemonic | Name | Register Location |
|---|---|---|
| LOOPQ | QSPI Loop Mode | SPCR3 |
| LOOPS | Loop Mode | SCCR1 |
| LOSCD | Loss of Clock Oscillator Disable | SYNCR |
| M | Mode Select | SCCR1 |
| MISO | Master In Slave Out | DDRQS |
| MISO | Master In Slave Out | PQSPAR |
| MM | Module Mapping | SCIMCR |
| MODE | Asynchronous/Synchronous Mode | CSOR[0:10], CSORBT |
| MODF | Mode Fault Flag | SPSR |
| MOSI | Master Out Slave In | DDRQS |
| MOSI | Master Out Slave In | PQSPAR |
| MSTR | Master/Slave Mode Select | SPCR0 |
| MULT | Multichannel Conversion | ADCTL1 |
| MV | Accumulator M Overflow Flag | CCR |
| N | Negative Flag | CCR |
| NEWQP | New Queue Pointer Value | SPCR2 |
| NF | Noise Error | SCSR |
| OR | Overrun Error | SCSR |
| PC[6:0] | Port C Pin Data | PORTC |
| PCBK | µPC Breakpoint Flag | DSSR |
| PE | Parity Enable | SCCR1 |
| PEPA[7:0] | Port E Pin Assignment | PEPAR |
| PF | Parity Error | SCSR |
| PFPA[7:0] | Port F Pin Assignment | PFPAR |
| PIRQL[2:0] | Periodic Interrupt Request Level | PICR |
| PITM[7:0] | Periodic Interrupt Timing Modulus | PITR |
| PIV[7:0] | Periodic Interrupt Vector | PICR |
| POW | Power-Up Reset | RSR |
| PQS | Port QS Data | PORTQS |
| PQSPA[6:0] | Port QS Pin Assignment | PQSPAR |
| PRS[4:0] | Prescaler Rate Selection | ADCTL0 |
| PSCK | Prescaler Clock | TPUMCR |
| PT | Parity Type | SCCR1 |
| PTP | Periodic Timer Prescaler Control | PITR |
| RAF | Receiver Active | SCSR |
| RAMDS | TPURAM Array Disable | TRAMBAR |
| RASP | RAM Array Space | SRAMMCR, TRAMMCR |
| RDRF | Receive Data Register Full | SCSR |
| RE | Receiver Enable | SCCR1 |
| RES10 | 10-Bit Resolution | ADCTL0 |
| RIE | Receiver Interrupt Enable | SCCR1 |

## Table D–9. Register Bit and Field Mnemonics (Continued)

| Mnemonic | Name | Register Location |
|---|---|---|
| RJURR | Unsigned Right-Justified Result | RSLT[0:7] |
| RLCK | SRAM Base Address Lock | SRAMMCR |
| RR[0:F] | Receive Data RAM | QSPI RAM |
| RSTEN | Reset Enable | SYNCR |
| R/$\overline{W}$ | Read/Write | CSOR[0:10], CSORBT |
| RWU | Receiver Wakeup | SCCR1 |
| R[8:0]/T[8:0] | SCI Receive/Transmit Data | SCDR |
| S | Stop Enable | SR |
| S8CM | Select 8-Conversion Sequence Mode | ADCTL1 |
| SBK | Send Break | SCCR1 |
| SCAN | Scan Mode Selection | ADCTL1 |
| SCBR | SCI Baud Rate | SCCR0 |
| SCF | Sequence Complete Flag | ADSTAT |
| SCK | Serial Clock | DDRQS |
| SHEN[1:0] | Show Cycle Enable | SCIMCR |
| SLIMP | Limp Mode | SYNCR |
| SLOCK | Synthesizer Lock | SYNCR |
| SLVEN | Factory Test Mode Enabled | SCIMCR |
| SPACE | Address Space Select | CSOR[0:10], CSORBT |
| SPBR | Serial Clock Baud Rate | SPCR0 |
| SPE | QSPI Enable | SPCR1 |
| SPIF | QSPI Finished Flag | SPSR |
| SPIFIE | SPI Finished Interrupt Enable | SPCR2 |
| SRBK | Service Request Breakpoint Flag | DSSR |
| STEXT | Stop Mode External Clock | SYNCR |
| STF | Stop Flag | TPUMCR |
| STOP | Stop Mode | ADCMCR, QSMCR, SRAMMCR, TRAMMCR, FEE1MCR, FEE2MCR |
| STRB | Address Strobe/Data Strobe | CSOR[0:10], CSORBT |
| STSCIM | Stop Mode SCIM Clock | SYNCR |
| STS[1:0] | Sample Time Selection | ADCTL0 |
| SUPV | Supervisor/Unrestricted | ADCMCR, QSMCR, SCIMCR, TPUMCR |
| SW | Software Watchdog Reset | RSR |
| SWE | Software Watchdog Enable | SYPCR |
| SWP | Software Watchdog Prescale | SYPCR |
| SWT[1:0] | Software Watchdog Timing | SYPCR |
| SYS | System Reset | RSR |
| T2CG | TCR2 Clock/Gate Control | TPUMCR |
| TC | Transmit Complete | SCSR |

## Table D–9. Register Bit and Field Mnemonics (Continued)

| Mnemonic | Name | Register Location |
|---|---|---|
| TCIE | Transmit Complete Interrupt Enable | SCCR1 |
| TCR1P | TCR1 Prescaler Control | TPUMCR |
| TCR2P | TCR2 Prescaler Control | TPUMCR |
| TDRE | Transmit Data Register Empty | SCSR |
| TE | Transmitter Enable | SCCR1 |
| TIE | Transmit Interrupt Enable | SCCR1 |
| TPUF | TPU Freeze Flag | DSSR |
| TST | Test Submodule Reset | RSR |
| V | Overflow Flag | SR |
| VFPE | Verify Program/Erase | FEE1CTL, FEE2CTL |
| W | Frequency Control (VCO) | SYNCR |
| WAIT | Wait States | FEE1MCR, FEE2MCR |
| WAKE | Wakeup by Address Mark | SCCR1 |
| WOMQ | Wired-OR Mode for QSPI Pins | SPCR0 |
| WOMS | Wired-OR Mode for SCI Pins | SCCR1 |
| WREN | Wrap Enable | SPCR2 |
| WRTO | Wrap To | SPCR2 |
| X | Frequency Control Bit (Prescale) | SYNCR |
| Y[5:0] | Frequency Control (Counter) | SYNCR |
| Z | Zero Flag | SR |

# INDEX


– A –

ABD, 3–5, D–18
ABIU, 7–4
Access levels. *See* Supervisor/user privilege levels
ADA[7:0], 7–2
ADC, 7–1, D–4
  bus interface unit (ABIU), 7–4
  clock 7–7
  control registers (ADCTL0, ADCTL1), D–6, D–7
  base address, 7–3
  conversion modes, D–7
  module configuration register (ADCMCR), D–5
  pins, 7–1
  result registers, D–10
  status register (ADSTAT), D–9
  test register (ADTEST), D–5
ADCMCR, D–5
ADCTL0, D–6
ADCTL1, D–7
ADDR0, 3–29, 3–31
ADDR[23:0], 3–24, D–29, D–33
Address
  bus, 3–24
    and data bus configuration options, 1–8
    disable bit (ABD), 3–5, D–18
    pin functions, 3–50
  registers, CPU32, 4–5
  space, 3–26, 3–73, D–31
  strobe, 3–24, D–30
  mark wakeup, 6–33
Addressing modes, CPU32, 4–9
ADSTAT, D–9
ADTEST, D–5
Alternate function code registers, 4–7
Analog
  input pins, 7–2
  reference voltages, 7–3
  subsystem, 7–5
  supply pins, 7–3
  to-digital converter module (ADC), 7–1, D–4
AN[7:0], 7–2
Array space (ASPC), 8–4
$\overline{AS}$, 3–24, 3–31, 3–34, 3–73, 3–76
ASPC, 8–4, D–12
Asynchronous
  bus cycles, 3–24
  input hold time, 3–30
  input setup time, 3–30
Autovector ($\overline{AVEC}$)
  bit, 3–73, D–31
  enable, D–31
  signal, 3–27, 3–64

– B –

Background debugging mode (BDM), 3–4, 4–17
Base address
  of MCU module control registers, 1–6
  registers
    flash EEPROM, 8–5
    SRAM, 9–1
    TPURAM, 10–2
Baud rate, SCK, 6–18
BC, D–57
BCD, 4–4
BDM, 3–4, 4–17
$\overline{BERR}$, 3–26, 3–31, 3–32, 3–39, 3–40, 3–50, 3–52
$\overline{BG}$, 3–44, 3–45, 3–46, 3–76
$\overline{BGACK}$, 3–44, 3–45, 3–46, 3–70, 3–76
BH, D–57
Bit-time, 6–28

Bits per transfer (BITS), 6–22, D–41
enable (BITSE), D–45
$\overline{\text{BKPT}}$, 3–37, 3–48, 3–55, 3–61, D–58
instruction, 3–36
BL, D–57
BLC, D–57
BLKSZ, 3–70, D–29
BM, D–57
BME, D–26
BMT, 3–6, D–26
BOOT, 8–3
Bootstrap operation, 3–71, 3–75
flash EEPROM, 8–7
BP, D–57
$\overline{\text{BR}}$, 3–42, 3–43, 3–44, 3–45, 3–51, 3–70
Branch latch control, D–57
Break frames, 6–31
Breakpoints, 4–20
acknowledge cycles, 3–35, 3–36, 3–37
asserted flag, D–58
enable bits, D–57
hardware, 3–37
instruction, 4–17
mode selection, 3–55
operation, 3–38
software, 3–36
BSA, 4–18
BT, D–57
Buffer amplifier, 7–6
Bus
arbitration, 3–42, 3–43, 3–44, 3–45
clock for MC6800 devices, 3–19
error ($\overline{\text{BERR}}$), 3–26
double bus fault, 3–42
exceptions, 3–39, 3–41
termination, 3–39
grant ($\overline{\text{BG}}$), 3–44
acknowledge ($\overline{\text{BGACK}}$), 3–44
interface unit, ADC, 7–4
intermodule, 1–3
master, 3–44
monitor, 3–6, 3–11, 3–26, 3–31, 3–32
external enable, D–26
period, 3–6
timing (BMT), 3–6, D–26
signals, 3–24
state analyzer, 4–18
BYTE field, 3–72, D–30

**– C –**

C (carry) flag, 4–6, D–3
CCF, D–9
CCL, D–57
CCR, 4–6, D–3
CCTR[2:0], D–9
CD:CA, D–7
Central processing unit (CPU32), 4–1
CFSR, D–52
Channels
conditions latch, D–57
function select registers, 5–14, D–52
interrupt base vector (CIBV), 5–6, D–51
interrupt enable register (CIER), 5–6, 5–13, D–51
interrupt request level (CIRL), 5–6, D–51
interrupt status register (CISR), 5–6, 5–13, D–51
orthogonality, 5–4
priority, 5–5
registers (CPR1, CPR2), 5–16, D–55
register breakpoint flag, D–58
selection, ADC, D–7
status register, D–52
TPU, 5–2
CHBK, D–58
Chip selects, 3–66
base address registers, 3–69, 3–70, D–29
emulation support, 3–76
operation, 3–73
option registers, 3–69, 3–71, 3–72, 3–76, D–29
peripheral, 6–23
pin assignment registers, 3–69, D–28
registers, 3–68
reset operation, 3–75
timing, 3–69, 3–71, 3–72
CIBV, 5–6, D–51
CIER, 5–6, 5–13, D–51
CIRL, 5–6, D–51
CISR, 5–6, 5–13, D–52
CLKOUT, 3–19, 3–30, 3–31, 3–45, 3–47

Clock, ADC, 7–7
Clock phase and polarity (CPHA, CPOL), QSPI, D–41
Clock, system
low-power stop operation, 3–11, 3–19
mode selection, 3–54
prescaler control, 7–7
rate, setting, 3–11, 3–12, 3–15, 3–20
signal duty cycle, 3–13
sources, 3–12
synthesizer, 3–13
control register (SYNCR), D–19
CLKS, D–58
Coherency, TPU, 5–5
Command RAM, 6–9, D–44
Comparator, 7–7
Compatibility, M68000 family, 4–13
Completed queue pointer, 6–10, D–43
Condition code register (CCR), 4–6, D–3
Configuration options, 1–7, 3–48
CONT, D–44
Control block, flash EEPROM, 8–2
Control registers
ADC, 7–7
CPU32, 4–6
QSPI, 6–8
SCI, 6–24
Conversion
channel, 7–9
complete flags, D–9
control, ADC 7–8
counter, D–9
modes, ADC, 7–9, D–7
parameters, 7–9
timing, ADC, 7–14
CPHA, 6–18, D–41
CPOL, 6–18, D–41
CPR1, CPR2, 5–16
CPTQP, 6–10, 6–20, D–43
CPU32, 4–1, D–2
address registers, 4–5
addressing modes, 4–9
block diagram, 4–2
control registers, 4–6
data registers, 4–4
development support disable (CPUD), 3–5, D–18
exception processing, 4–13
instructions, 4–10
memory operand addressing, 4–8
privilege levels, 4–10
processing states, 4–10
program counter, 4–6
registers, 4–2
space, 3–35, 3–36, 3–65
address encoding, 3–35
status register, 4–6
CPUD, 3–5
CSBARBT, 3–69, 3–71, 3–75, D–29
CSBAR[10:0], 3–69, D–29
$\overline{\text{CSBOOT}}$, 3–47, 3–51, 3–70, 3–72, 3–75, 3–76
$\overline{\text{CSE}}$, 3–66, 3–76, 3–78
$\overline{\text{CSM}}$, 3–77
CR, D–44
CSOR, 3–69, D–30
CSORBT, 3–69, 3–72, 3–76, D–30
CSPAR, 3–68, D–28

**– D –**

DAC array, 7–6
Data
and size acknowledge ($\overline{\text{DSACK}}$), 3–26, D–30
bus, 3–24
mode selection, 3–49
pin functions, 3–50
frame, 6–28
register, SCI, 6–27
registers, CPU32, 4–4
space, flash EEPROM, 8–4
strobe ($\overline{\text{DS}}$), 3–25
DATA[15:0], 3–24
DBF, D–20
DCNR, D–59
DDA, 3–78
DDB, 3–78
DDRAB, 3–78, D–22
DDRE, 3–78, D–22
DDRF, 3–79, D–23
DDRG, 3–82, D–21
DDRH, 3–82, D–21

DDRQS, 6–4, 6–18, 6–21, D–38
Debugging, 4–17
and HALT, 3–43
Decoded channel number register (DCNR), D–59
Delay after transfer, 6–20, D–44
Delay before SCK, D–41
Deterministic opcode tracking, 4–19
Development support, 4–17
control register, TPU, D–57
status register, TPU, D–58
TPU, 5–16
DFC, 4–7
Digital control subsystem, 7–7
Digital output pins, ADC, 7–3
DIO, 5–7
Discrete input/output, 3–79
TPU function (DIO), 5–7
Double buffering, 6–30, 6–31
Double bus faults, 3–42
reset, D–20
Driver types, 2–2
DS, 3–25, 3–31, 3–73
DSACK, 3–26, 3–27, 3–30, 3–64, 3–67, 3–72
field, 3–72, D–30
option fields, 3–34
DSCK, 6–19, D–44
DSCKL, D–41
DSCR, D–57
DSSR, D–58
DT, D–44
DTL, D–41
Dynamic bus sizing, 3–24, 3–27

– E –

EBI, 3–23
ECLK, 3–19, 3–73
divide rate (EDIV), 3–19, D–19
EDIV, 3–19, D–19
EEPROM, flash, 8–1
Eight-bit
data bus operation, 3–52
port, 3–28
EMU, 5–13, D–51
Emulation
control, TPU (EMU), 5–13, D–50
mode, 3–77
selection, 3–55
support, TPU, 5–5
support chip select (CSE), 3–76
Enable programming/erasure (ENPE), 8–3, 8–6, D–14
End queue pointer (ENDQP), 6–10, D–42
ENPE, 8–3, 8–6, D–14
Erase control bit (ERAS), 8–5, D–14
Erasing the flash EEPROM, 8–8
Error flags, SCI, 6–32
Event timing, TPU, 5–4
Exceptions
bus error, 3–41
processing, 4–13, 4–16
vector table, 6–4
vectors, 3–62, 4–7, 4–13
EXOFF, D–18
EXT, D–20
EXTAL, 3–8, 3–9, 3–10
Extend (X) flag, 4–6, D–3
External
bus
arbitration, 3–44
clock off (EXOFF), D–18
clock signal (ECLK), 3–19
control logic, 3–31
cycles, 3–6, 3–30
interface (EBI), 3–23
monitor, 3–6
clock division bit (EDIV), 3–19
interrupt requests, 3–5, 3–63, 3–64
low-leakage capacitor, 3–13
reset, D–20
system clock signal, 3–12

– F –

Factory test mode, 3–6, D–18
Fast-termination cycles, 3–30, 3–34
FC[2:0], 3–25
FE, 6–32, D–48
FEEBAH, 8–5, D–13
FEEBAL, 8–5, D–13
FEECTL, 8–5, D–14

FEEBS[3:0], 8–6, 8–7, D–15
FEEMCR, 8–3, D–12
Flash EEPROM, 8–1
  array, 8–3, D–12
  base address registers, D–13
  bootstrap operation, 8–7
  bootstrap words (FEEBS[3:0]), 8–6, D–15
  control block, 8–2
  control register (FEECTL), 8–5, D–14
  module configuration register (FEEMCR), D–12
  program/erase operation, 8–8
  reset operation, 8–7
  signal conditioning, 8–13
Frame, 6–28
Framing error, 6–32, D–48
  flag (FE), 6–32
FREEZE assertion, 3–4
  ADC response to FRZ, 7–4, D–5
  flash EEPROM response to FRZ, 8–3, D–12
  QSM response to FRZ, 6–3, D–37
  TPU response to FRZ, D–58
Freeze bus monitor (FRZBM), 3–4, D–18
Freeze software watchdog (FRZSW), 3–4, D–18
Function codes (FC[1:0]), 3–25, 3–26
Function library, TPU, 5–6

– G –

General purpose I/O, 3–79

– H –

$\overline{\text{HALT}}$ 3–27, 3–31, 3–39, 3–43, 3–44, D–43
Halt monitor, 3–6
  enable (HME), 3–6, D–26
  reset, 3–6
Halt operation, 3–43
HALTA, D–43
Handshaking, 3–30
Hang on T4 (HOT4), D–57
Hardware breakpoints, 3–37
HME, 3–6, D–26
HMIE, D–43
Host interface, TPU, 5–3
Host sequence registers, 5–14, D–53
Host service registers, 5–14, D–53
Host service request field, 5–14
HOT4, D–57
HSQR, D–53
HSRR, D–53

– I –

IARB
  QSM, D–37
  SCIM, 3–4, 3–64, D–19
  TPU, D–51
IDLE, D–47
Idle frame, 6–28
Idle line
  detection, 6–32, D–47
  interrupt enable (ILIE), 6–33, D–46
  type (ILT), 6–32, D–45
  wakeup, 6–33
$\overline{\text{IFETCH}}$, 3–5, 4–20
ILIE, 6–33, D–46
ILQSPI, 6–3, D–38
ILSCI, 6–3, D–38
ILT, 6–32, D–45
IMB, 1–3
Input capture/input transition counter (ITC), 5–7
Input/output, discrete, 3–77
Instruction restart, 4–9
Instruction set summary, 4–12
Interchannel communication, 5–5
Intermodule bus (IMB), 1–3
Internal
  loop, 6–34
  oscillator, 3–12
  phase-locked loop, 3–11
  queue pointer, 6–10
Interrupts, 3–61
  acknowledge bus cycles, 3–35, 3–66
  arbitration (IARB) field
    QSM, D–37
    SCIM, 3–4, 3–64, D–19
    TPU, D–51
  exception processing, 3–62
  handler routines, 6–4
  level

QSPI, D–38
SCI, D–38
chip select, D–31
nonmaskable, 3–63
port F, 3–81
priority, 3–62
mask (IP), 4–6, 6–3, D–3
processing summary, 3–65
QSM, 6–3
recognition, 3–62
request signals (IRQ), 3–62
SCI, 6–34
TPU, 5–6, 5–13
vectors, 3–62, 3–64, 6–4, D–38
INTV[7:0], D–38
IP mask, 3–62, 6–3, D–3
IPIPE, 3–5, 4–20
IPL field, 3–73, D–31
IRQ[7:1], 3–62
ITC, 5–7

– L –

Latch control bit (LAT), 8–5, 8–6, D–14
Left-justified result registers (LJURR, LJSRR), D–10
Length of delay after transfer (DTL), D–41
Limp mode status bit (SLIMP), 3–20, D–20
Link register (LR), D–58
LJSRR, D–10
LJURR, D–10
LOC, D–20
LOCK bit, 8–4, D–12
Lock time, clock synthesizer, 3–13
Long idle-line detection, 6–33
Loop mode, 4–20, D–42, D–45
LOOPQ, D–43
LOOPS, 6–34, D–45
LOSCD, 3–20, D–19
Loss of clock, 3–20, 3–21, 3–22
oscillator disable (LOSCD), 3–20, D–19
reset, 3–22, D–20
Low-pass filter, 3–13
Low-power stop (LPSTOP), 3–11, 3–19, 4–13,
ADC, 7–4
broadcast cycle, 3–39
flash EEPROM, 8–3
QSM, 6–3
SRAM, 9–3
TPU, 5–13
TPURAM, 10–3
LR, D–58

– M –

M bit, 6–28, D–46
M68000 family compatibility, 4–13
Mark, 6–31
Mask value, 3–62
Master in slave out (MISO), 6–10, 6–18, 6–21
Master mode, QSPI, 6–11, 6–18
select, D–40
Master out slave in (MOSI), 6–10, 6–18, 6–21
Master wraparound, 6–21
MC68F333
block diagram, 1–4
features, 1–1
memory map, 1–6
pin assignment, 1–5
pin characteristics, 2–1
Memory map, MC68F333, 1–6
Memory organization, 4–7
Memory, virtual, 4–9
Microengine, TPU, 5–3
Micro-PC breakpoint flag, D–58
Misaligned operands, 3–29
MISO, 6–10, 6–18, 6–21
MM, 3–3, D–18
MODCLK, 3–7, 3–9, 3–48, 3–54
MODE bit, 3–75, D–30
Module mapping (MM) bit, 1–6, 3–3, D–18
MOSI, 6–10, 6–18, 6–21
MSTR, 6–11, 6–18, D–40
Multichannel conversions (MULT), ADC, 7–13, D–7
Multimaster operation, QSM, 6–11
Multiplexer, ADC, 7–5

– N –

N (negative) flag, 4–6, D–3
New queue pointer (NEWQP), 6–10, 6–20, 6–23, D–42

Noise flag (NF), 6–32, D–48
Nonmaskable interrupt, 3–63
NRZ, 6–2, 6–23, 6–28

– O –

OC, 5–7
Opcode tracking, 4–19
Operand alignment, 3–28, 3–29
Operand transfer cases, 3–29
Output compare (OC), 5–7
Overflow (V) flag, 4–6
Overrun error (OR), 6–32, D–47

– P –

PAC, 5–4, D–56
PADB[7:0], 7–3
Parameter RAM, TPU, 5–3, D–55
Parity
  checking, 6–29
  enable (PE), 6–29, D–46
  flag (PF), 6–32, D–48
  type (PT), 6–29, D–45
PC, 4–6
PCBK, D–58
PCS0/$\overline{SS}$, 6–10, 6–21
PCS[3:0], 6–10, D–44
PCS to SCK delay, D–44
PE, 6–29, D–46
PEPAR, 3–78, D–22
Period/pulse-width accumulator (PPWA), 5–10
Periodic interrupt timer (PIT), 3–9, 3–10
  control register (PICR), D–26
  modulus (PITM), 3–10, D–27
  modulus counter, 3–9
  prescaler (PTP), 3–9, D–27
  priority, 3–10
  request level (PIRQL), 3–10
  vector (PIV), 3–10, D–26
Peripheral chip selects, 6–23, D–44
PF, 6–32, D–48
PFIVR, 3–81, D–25
PFLVR, 3–81, D–25
PFPAR, 3–80, D–24
PICR, D–26
Pins
  ADC, 7–1
  assignment field encoding, 3–69
  characteristics, 2–1
  control register, QSM, 6–4
  QSPI, 6–10
  SCI, 6–27
  state during reset, 3–56
PIRQL, D–26
PIT, 3–9
PITCLK, 3–10
PITM, D–27
PITR, D–27
PIV, D–26
PMA, 5–8
PMM, 5–9
Ports, I/O, 3–77
  port A, 3–78, D–21
  port ADA, 7–2, D–5
  port ADB, 7–3, D–5
  port B, 3–78, D–21
  port C, 3–68, 3–73, D–28
  port E, 3–78, 3–79, D–22
  port F, 3–79, 3–80, 3–82, D–23
  port G, 3–82, D–21
  port H, 3–82, D–21
  port QS, 6–4, 6–27, D–38
  size, 3–27, 3–70
  width, 3–24, 3–26
Position-synchronized pulse generator (PSP), 5–9
POW, D–20
Power-on reset, 3–59
PPWA, 5–10
PQSPAR, 6–4, 6–18, 6–21, 6–27, D–38
Prescaler clock, D–52
Prescaler rate selection, D–6
Privilege levels. *See* Supervisor/user privilege level
Processing states, CPU32, 4–10
Program counter, 4–6
  indirect with displacement, 4–9
  indirect with index, 4–9
  Program space, flash EEPROM, 8–4
Program/erase voltage pins, 8–2

Programming the flash EEPROM, 8–8
PRS[4:0], D–6
PSC, 5–4, D–56
PSCK, D–51
PT, 6–29, D–45
PTP, D–27
Pulse-width modulation (PWM), 5–8

– Q –

QILR, 6–2, D–37
QIVR, 6–2, D–37
QSMCR, 6–2, D–36
QSM, 6–1, D–36
  configuration register (QSMCR), 6–2, D–36
  global registers, 6–2
  initialization, 6–34
  interrupt level register (QILR), 6–2, D–37
  interrupt vector register (QIVR), 6–2, D–37
  pin control registers, 6–4
QSPI, 6–2, 6–5
  block diagram, 6–7
  control registers, D–40
  enable, D–41
  finished flag, D–43
  initialization operation, 6–12
  loop mode, D–43
  master operation, 6–13
  operating modes, 6–11
  pins, 6–10
  RAM, 6–8
  registers, 6–7
  slave operation, 6–16
  status register (SPSR), 6–8, D–43
Queue entry, 6–10
Queue pointers, 6–10
Queued serial module (QSM), 6–1, D–36
Queued serial peripheral interface (QSPI), 6–5

– R –

R/$\overline{W}$, 3–25, 3–31
  disable bit (RWD), 3–5
  field, 3–72, D–30
RAM disable (RAMDS), 10–2
RAF, D–47
RAM, QSPI, 6–8
  command, 6–9
  receive, 6–8
  transmit, 6–9
RAMDS, 10–2, D–33
RASP, 9–1, 10–2, D–32, D–34
RC DAC array, 7–6
RDR, 6–27, 6–31
RDRF, 6–32, D–47
RE, 6–31, D–46
Read cycle, 3–32
Read/write, 3–25, D–30
  disable (RWD), D–19
Receive data (RXD), 6–27
Receive data register (RDR), 6–27, 6–31
  full (RDRF), 6–32, D–47
Receive RAM, 6–8, D–43
Receive time (RT) clock, 6–29, 6–32
Receiver
  active flag (RAF), D–47
  enable (RE), 6–31, D–46
  interrupt enable (RIE), D–46
  operation, 6–31
  wakeup, 6–33, D–46
Reference crystal, 3–12
Reference voltages, ADC, 7–3
Register direct, 4–9
Register indirect, 4–9
Register indirect with index, 4–9
Regular bus cycles, 3–31
RES10, D–6
$\overline{RESET}$, 3–46, 3–58
Resets, 1–7, 3–48
  chip selects, 3–74
  control logic, 3–47
  eight-bit data bus configuration, 3–52
  enable (RSTEN), 3–20, D–20
  exception processing, 3–47
  flash EEPROM, 8–7
  loss of clock, 3–22
  module pin function, 3–55
  operating configuration out of, 3–48
  pin state, 3–56

power on, 3–59
processing summary, 3–61
single-chip operation, 3–53
sixteen-bit data bus configuration, 3–50
sources, 3–47
SRAM, 9–3
status register (RSR), 3–61, D–20
timing, 3–58
TPURAM, 10–4
vector, 4–14
flash EEPROM, 8–7
Resolution, 7–8
Resolution time, 7–14
Result registers, 7–16
Retry operation, 3–42
Retry termination, 3–39
RIE, D–46
RJURR, D–10
RLCK, D–34
$\overline{\text{RMC}}$, 3–43
RR, D–44
RSLT[7:0], D–10
RSR, 3–61, D–20
RSTEN, 3–20, 3–22, D–20
RT clock, 6–29, 6–32
RTE, 3–42
RWD, 3–5, D–19
RWU, 6–33, D–46
RXD pin, 6–27, 6–32

– S –

S bit, 4–10, D–3
S8CM, D–7
Sample capacitors, 7–6
Sample time, 7–8, 7–14, D–6
SAR, 7–16
SBK, 6–31, D–46
SCAN, D–7
SCBR, D–45
SCCR[1:0], D–45
SCDR, 6–27, 6–31, D–48
SCF, D–9
Scheduler, TPU, 5–3
SCI, 6–2, 6–23
baud clock, 6–29, D–45
control registers (SCCR[1:0]), 6–24, D–45
data register (SCDR), 6–27, 6–31, D–48
error flags, 6–32
pins, 6–27
registers, 6–23
status register (SCSR), 6–27, D–47
SCIM, 3–1
configuration register (SCIMCR), D–17
test register (SCIMTR), D–19
test registers, D–27
SCK, 6–10, 6–18, 6–21
baud rate, 6–19
SCSR, 6–27, D–47
Select 8-conversion sequence mode (S8CM), D–7
Send break (SBK), 6–31, D–46
Sequence complete flag (SCF), D–9
Serial
clock baud rate, D–41
communication interface (SCI), 6–23
formats, 6–28
mode (M) bit, 6–28
Service
grant latch register (SGLR), D–59
request breakpoint flag, D–58
status bits, D–59
SFC, 4–7
SGLR, D–59
Shadow register, 8–2
SHEN, 3–5, 3–46, D–18
Short idle-line detection, 6–33
Show cycles, 3–5, 3–46
enable bits, 3–5, D–18
Signal characteristics, 2–3
Signal conditioning, 8–13
Signal functions, 2–5
Signed left-justified format, 7–16, D–10
Single
bus cycle operation, 3–27
channel conversions, 7–12
chip integration module (SCIM), 3–1, D–16
chip operation, 3–5, 3–53
Sixteen-bit
port, 3–27

data bus operation, 3–50
Size signals (SIZ[1:0]), 3–25, 3–31
Slave (factory test) mode arbitration, 3–46
Slave mode, QSPI, 6–11, 6–21
Slave wraparound, 6–23
SLIMP, 3–20, D–20
SLOCK, D–20
SLVEN, D–18
SM, 5–9
Software
breakpoints, 3–36
watchdog, 3–7
clock rate, 3–7
divide ratio, 3–8
enable (SWE), 3–7, D–25
prescale, D–25
reset, D–20
service register (SWSR), 3–7, D–27
timer, 3–8
timing (SWT), 3–7, D–26
SPACE field, 3–72, D–31
SPBR, 6–18, D–41
SPCR[3:0], D–40
SPE, 6–8, 6–20, D–41
SPI finished (SPIF), D–43
interrupt enable (SPIFIE), D–42
SPSR, 6–8, D–42
Spurious interrupts, 3–7, 3–64
SPWM, 5–8
SR, D–3
SRAM, 9–1, D–34
array address mapping, 9–1
array address space type, 9–1
register block, 9–1
SRAMBAH, SRAMBAL, D–35
SRAMMCR, D–34
SRBK, D–58
$\overline{SS}$, 6–10, 6–11
Standby operation
SRAM, 9–2
TPURAM, 10–3
Standby RAM (SRAM), 9–1, D–34
with TPU emulation (TPURAM), 10–1, D–32
Start bit, 6–28
Status register
ADC, 7–7
CPU (SR), 4–6, D–3
QSPI, 6–8
SCI, 6–27
STEXT, 3–19, D–20
STF, D–50
STOP bit
ADCMCR, 7–4, D–5
FEEMCR, 8–3, D–12
QSMCR, 6–3, D–37
SRAMMCR, 9–3, D–34
TPUMCR, 5–13, D–49
TRAMMCR, D–32
Stop clocks to TCRs, D–57
Stop mode external clock (STEXT), 3–19
Stop mode SCIM clock (STSCIM), 3–19
STOP shadow bit, 8–7
STRB, 3–72, D–30
STS, 7–8, 7–14, D–6
STSCIM, 3–19, D–20
Successive approximation register (SAR), 7–16
Supervisor stack pointer (SSP), 4–10
Supervisor/user privilege levels (SUPV)
ADC, D–5
CPU32, 4–2
QSM, D–37
SCIM, 3–4, D–18
TPU, D–51
SW, D–20
SWE, 3–7, D–25
SWP, 3–7, D–25
SWSR, D–27
SWT, 3–7, D–26
Synchronization to CLKOUT, 3–30
Synchronized pulse-width modulation (SPWM), 5–8
SYNCR, D–19
Synthesizer lock, D–20
SYPCR, D–25
SYS, D–20
System clock, 3–11
block diagram, 3–12
frequencies, 3–17
protection control register (SYPCR), D–25

reset, 3–22, D–20
states, 3–30
System configuration and protection, 3–2

– T –

T[1:0], D–3
T2CG, 5–12, D–50
T2CLK pin, 5–2
Table lookup and interpolate (TBL), 4–13
TBS, 5–4, D–56
TC, 6–30, D–47
TCIE, 6–31, D–46
TCR1, 5–2, 5–11, D–49
TCR2, 5–2, 5–12, D–51
TDRE, 6–31, D–47
TE, 6–30, D–46
Ten-bit resolution, D–6
Test submodule, 3–82
Three-state control pin (TSC), 3–60
TICR, 5–11, D–51
TIE, 6–31, D–46
Time
bases, TPU, 5–2
functions, TPU, 5–7
processor unit (TPU), 5–1, D–49
Timer
channels, 5–2
control register one (TCR1), 5–11, D–49
control register two (TCR2), 5–12, D–50
count registers, 5–2
TPU, 5–1, D–49
configuration control registers, 5–11
emulation support, 5–5
function library, 5–6
host interface, 5–3
interrupts, 5–6
configuration register (TICR), D–51
module configuration register (TPUMCR), D–49
parameter RAM, 5–3
time functions, 5–7
timer channels, 5–2
TPUF, D–58
TPUMCR, 5–11, D–49
TPURAM, 10–1, D–32
array address mapping, 10–2, D–32
base address and status register, D–33
module configuration register (TRAMMCR), 10–1, 10–2, D–32
Tracing, 4–6, 4–17, D–3
TRAMBAR, 10–2, D–33
TRAMMCR, 10–1, 10–2, D–32
Transfer
delay, 6–20
length, 6–19
time, 7–14
Transmission complete (TC), 6–30, D–47
interrupt enable (TCIE), 6–31, D–46
Transmit
data (TXD), 6–27
register empty (TDRE), D–47
interrupt enable (TIE), 6–31, D–46
RAM, 6–9, D–43
Transmitter, 6–30
enable (TE), 6–30
Traps, 4–11
TSC, 3–60
TST, D–20
TXD, 6–27, 6–30

– U –

Uninitialized interrupt vector, 6–4
Unsigned left-justified format, 7–16, D–10
Unsigned right-justified format, 7–16, D–10
Upper/lower byte option, D–30
User privilege level. *See* Supervisor/user privilege levels.
User stack pointer (USP), 4–10

– V –

V (overflow) flag, 4–6, D–3
VBR, 4–7, 4–14
$V_{DD}$, 9–2, 10–3
ramp-up time, 3–59
$V_{DDA}$, 7–3
$V_{DDSYN}$, 3–13, 3–59
Vector base register (VBR), 4–7, 4–14
Verify program/erase bit (VFPE), 8–5, D–14

VFPE bit, 8–5, D–14
$V_{FPE16K}$, 8–2
$V_{FPE48K}$, 8–2
Virtual memory, 4–9
Voltage controlled oscillator (VCO), 3–13
$V_{RH}$, 7–3
$V_{RL}$, 7–3
$V_{SSA}$, 7–3
$V_{STBY}$, 9–2, 10–3

– W –

W bit, D–19
WAIT, 8–4, D–13
Wait states, 3–30, D–13
    flash EEPROM, 8–4
WAKE, 6–33, D–46
Wakeup, receiver, 6–33, D–45
Wired-OR mode
    for QSPI pins (WOMQ), D–41
    for SCI pins (WOMS), 6–30, D–46
Word boundaries, 4–7
Wrap enable (WREN), 6–21, D–42
Wrap to (WRTO), D–42
Write cycle, 3–34
Write-lock protection, 8–4

– X –

X (extend) flag, 4–6, D–3
X frequency control bit, D–19

– Z –

Z (zero) flag, 4–6, D–3

INTRODUCTION 1

SIGNAL DESCRIPTIONS 2

SINGLE-CHIP INTEGRATION MODULE 3

CENTRAL PROCESSING UNIT 4

TIME PROCESSOR UNIT 5

QUEUED SERIAL MODULE 6

ANALOG-TO-DIGITAL CONVERTER 7

FLASH EEPROM 8

STANDBY RAM MODULE 9

STANDBY RAM WITH TPU EMULATION 10

ELECTRICAL CHARACTERISTICS A

MECHANICAL DATA AND ORDERING INFORMATION B

DEVELOPMENT SUPPORT C

REGISTER SUMMARY D

INDEX I

1 INTRODUCTION

2 SIGNAL DESCRIPTIONS

3 SINGLE-CHIP INTEGRATION MODULE

4 CENTRAL PROCESSING UNIT

5 TIME PROCESSOR UNIT

6 QUEUED SERIAL MODULE

7 ANALOG-TO-DIGITAL CONVERTER

8 FLASH EEPROM

9 STANDBY RAM MODULE

10 STANDBY RAM WITH TPU EMULATION

A ELECTRICAL CHARACTERISTICS

B MECHANICAL DATA AND ORDERING INFORMATION

C DEVELOPMENT SUPPORT

D REGISTER SUMMARY

I INDEX

*1ATX31386-0 Printed in USA 9/93 BANTA CO. MOTO #24 18,000 MCU YGACAA*

## *How to Reach Us:*

**Home Page:**
www.freescale.com

**E-mail:**
support@freescale.com

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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

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

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

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

RoHS-compliant and/or Pb- free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb- free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale.s Environmental Products program, go to http://www.freescale.com/epp.

freescale™
semiconductor

MC68F333UM/AD