

MC68HC908AZ60/D

**H C  8**

**MC68HC908AZ60**

**Rev 2.0**

**HCMOS Microcontroller Unit  
TECHNICAL DATA**





# List of Sections

|   |     |
|---|-----|
| List of Sections . . . . .                  | 1   |
| Table of Contents . . . . .                 | 3   |
| General Description . . . . .               | 11  |
| Memory Map . . . . .                        | 23  |
| RAM . . . . .                               | 35  |
| FLASH-1 Memory . . . . .                    | 37  |
| FLASH-2 Memory . . . . .                    | 51  |
| EEPROM-1 . . . . .                          | 63  |
| EEPROM-2 . . . . .                          | 75  |
| Central Processor Unit (CPU) . . . . .      | 87  |
| System Integration Module (SIM) . . . . .   | 105 |
| Clock Generator Module (CGM) . . . . .      | 127 |
| Configuration Register (CONFIG-1) . . . . . | 155 |
| Configuration Register (CONFIG-2) . . . . . | 159 |
| Break Module . . . . .                      | 161 |

## List of Sections

|   |     |
|---|-----|
| Monitor ROM (MON) .....                           | 167 |
| Computer Operating Properly Module (COP) .....    | 179 |
| Low-Voltage Inhibit (LVI) .....                   | 185 |
| External Interrupt Module (IRQ) .....             | 191 |
| Serial Communications Interface Module (SCI)..... | 199 |
| Serial Peripheral Interface Module (SPI) .....    | 237 |
| Timer Interface Module B (TIMB) .....             | 269 |
| Programmable Interrupt Timer (PIT).....           | 295 |
| I/O Ports .....                                   | 305 |
| MSCAN Controller (MSCAN08).....                   | 331 |
| Keyboard Module (KBD).....                        | 381 |
| Timer Interface Module A (TIMA-6) .....           | 389 |
| Analog-to-Digital Converter (ADC-15) .....        | 421 |
| Specifications .....                              | 433 |
| Appendix A: Future EEPROM Registers .....         | 449 |
| Glossary .....                                    | 453 |
| Literature Updates .....                          | 465 |
| Revision History .....                            | 469 |

# Table of Contents

## List of Sections

## Table of Contents

|                     |   |    |
|---------------------|---|----|
| General Description | Contents .....                            | 11 |
|                     | Introduction .....                        | 11 |
|                     | Features .....                            | 12 |
|                     | MCU Block Diagram .....                   | 13 |
|                     | Pin Assignments .....                     | 15 |
|                     | Ordering Information .....                | 22 |
| Memory Map          | Contents .....                            | 23 |
|                     | Introduction .....                        | 23 |
|                     | I/O Section .....                         | 26 |
| RAM                 | Contents .....                            | 35 |
|                     | Introduction .....                        | 35 |
|                     | Functional Description .....              | 35 |
| FLASH-1 Memory      | Contents .....                            | 37 |
|                     | Introduction .....                        | 37 |
|                     | Future FLASH Memory .....                 | 37 |
|                     | Functional Description .....              | 38 |
|                     | FLASH-1 Control Register .....            | 39 |
|                     | FLASH Charge Pump Frequency Control ..... | 41 |
|                     | FLASH Erase Operation .....               | 41 |
|                     | FLASH Program/Margin Read Operation ..... | 43 |
|                     | FLASH Block Protection .....              | 46 |
|                     | FLASH-1 Block Protect Register .....      | 47 |
|                     | FLASH-2 Block Protect Register .....      | 48 |
|                     | Low-Power Modes .....                     | 50 |
| FLASH-2 Memory      | Contents .....                            | 51 |
|                     | Introduction .....                        | 51 |
|                     | Future FLASH Memory .....                 | 51 |

## Table of Contents

|                                    |  |      |
|------------------------------------|--|------|
|                                    | Functional Description .....               | .52  |
|                                    | FLASH Control Register .....               | .53  |
|                                    | FLASH Charge Pump Frequency Control .....  | .55  |
|                                    | FLASH Erase Operation .....                | .55  |
|                                    | FLASH Program/Margin Read Operation .....  | .57  |
|                                    | FLASH Block Protection .....               | .60  |
|                                    | FLASH Block Protect Register .....         | .60  |
|                                    | Low-Power Modes .....                      | .61  |
| EEPROM-1                           | Contents .....                             | .63  |
|                                    | Introduction .....                         | .63  |
|                                    | Future EEPROM Memory .....                 | .64  |
|                                    | Features .....                             | .64  |
|                                    | Functional Description .....               | .65  |
|                                    | Low-Power Modes .....                      | .74  |
| EEPROM-2                           | Contents .....                             | .75  |
|                                    | Introduction .....                         | .75  |
|                                    | Future EEPROM Memory .....                 | .76  |
|                                    | Features .....                             | .76  |
|                                    | Functional Description .....               | .77  |
|                                    | Low-Power Modes .....                      | .86  |
| Central Processor<br>Unit (CPU)    | Contents .....                             | .87  |
|                                    | Introduction .....                         | .87  |
|                                    | Features .....                             | .88  |
|                                    | CPU registers .....                        | .89  |
|                                    | Arithmetic/logic unit (ALU) .....          | .94  |
|                                    | Low-power modes .....                      | .94  |
|                                    | CPU during break interrupts .....          | .95  |
|                                    | Instruction Set Summary .....              | .96  |
|                                    | Opcode Map .....                           | .103 |
| System Integration<br>Module (SIM) | Contents .....                             | .105 |
|                                    | Introduction .....                         | .106 |
|                                    | SIM Bus Clock Control and Generation ..... | .109 |
|                                    | Reset and System Initialization .....      | .110 |
|                                    | SIM Counter .....                          | .115 |
|                                    | Program Exception Control .....            | .116 |
|                                    | Low-Power Modes .....                      | .120 |
|                                    | SIM Registers .....                        | .123 |

|   |  |     |
|---|--|-----|
| Clock Generator<br>Module (CGM)                   | Contents .....                             | 127 |
|   | Introduction .....                         | 128 |
|   | Features .....                             | 128 |
|   | Functional Description .....               | 129 |
|   | I/O Signals .....                          | 139 |
|   | CGM Registers .....                        | 141 |
|   | Interrupts .....                           | 147 |
|   | Low-Power Modes .....                      | 147 |
|   | CGM During Break Interrupts .....          | 148 |
|   | Acquisition/Lock Time Specifications ..... | 149 |
| Configuration<br>Register<br>(CONFIG-1)           | Contents .....                             | 155 |
|   | Introduction .....                         | 155 |
|   | Functional Description .....               | 156 |
| Configuration<br>Register<br>(CONFIG-2)           | Contents .....                             | 159 |
|   | Introduction .....                         | 159 |
|   | Functional Description .....               | 159 |
| Break Module                                      | Contents .....                             | 161 |
|   | Introduction .....                         | 161 |
|   | Features .....                             | 161 |
|   | Functional Description .....               | 162 |
|   | Low-Power Modes .....                      | 164 |
|   | Break Module Registers .....               | 165 |
| Monitor ROM<br>(MON)                              | Contents .....                             | 167 |
|   | Introduction .....                         | 167 |
|   | Features .....                             | 168 |
|   | Functional Description .....               | 168 |
| Computer<br>Operating<br>Properly Module<br>(COP) | Contents .....                             | 179 |
|   | Introduction .....                         | 179 |
|   | Functional Description .....               | 180 |
|   | I/O Signals .....                          | 181 |
|   | COP Control Register .....                 | 183 |
|   | Interrupts .....                           | 183 |
|   | Monitor Mode .....                         | 183 |
|   | Low-Power Modes .....                      | 184 |
|   | COP Module During Break Interrupts .....   | 184 |

## Table of Contents

|   |  |     |
|---|--|-----|
| Low-Voltage<br>Inhibit (LVI)                          | Contents . . . . .                               | 185 |
|   | Introduction . . . . .                           | 186 |
|   | Features . . . . .                               | 186 |
|   | Functional Description . . . . .                 | 186 |
|   | LVI Status Register . . . . .                    | 189 |
|   | LVI Interrupts . . . . .                         | 190 |
|   | Low-Power Modes . . . . .                        | 190 |
| External Interrupt<br>Module (IRQ)                    | Contents . . . . .                               | 191 |
|   | Introduction . . . . .                           | 191 |
|   | Features . . . . .                               | 191 |
|   | Functional Description . . . . .                 | 192 |
|   | $\overline{\text{IRQ}}$ Pin . . . . .            | 195 |
|   | IRQ Module During Break Interrupts . . . . .     | 196 |
|   | IRQ Status and Control Register . . . . .        | 197 |
| Serial<br>Communications<br>Interface Module<br>(SCI) | Contents . . . . .                               | 199 |
|   | Introduction . . . . .                           | 200 |
|   | Features . . . . .                               | 200 |
|   | Pin Name Conventions . . . . .                   | 201 |
|   | Functional Description . . . . .                 | 201 |
|   | Low-Power Modes . . . . .                        | 218 |
|   | SCI During Break Module Interrupts . . . . .     | 219 |
|   | I/O Signals . . . . .                            | 219 |
|   | I/O Registers . . . . .                          | 220 |
| Serial Peripheral<br>Interface Module<br>(SPI)        | Contents . . . . .                               | 237 |
|   | Introduction . . . . .                           | 238 |
|   | Features . . . . .                               | 238 |
|   | Pin Name and Register Name Conventions . . . . . | 239 |
|   | Functional Description . . . . .                 | 240 |
|   | Transmission Formats . . . . .                   | 244 |
|   | Error Conditions . . . . .                       | 249 |
|   | Interrupts . . . . .                             | 253 |
|   | Queuing Transmission Data . . . . .              | 255 |
|   | Resetting the SPI . . . . .                      | 256 |
|   | Low-Power Modes . . . . .                        | 257 |
|   | SPI During Break Interrupts . . . . .            | 258 |
|   | I/O Signals . . . . .                            | 259 |
|   | I/O Registers . . . . .                          | 262 |



|                                       |                                     |     |
|---------------------------------------|-------------------------------------|-----|
| Timer Interface<br>Module B (TIMB)    | Contents .....                      | 269 |
|                                       | Introduction .....                  | 270 |
|                                       | Features .....                      | 270 |
|                                       | Functional Description .....        | 273 |
|                                       | Interrupts .....                    | 280 |
|                                       | Low-Power Modes .....               | 281 |
|                                       | TIMB During Break Interrupts .....  | 282 |
|                                       | I/O Signals .....                   | 283 |
|                                       | I/O Registers .....                 | 284 |
| Programmable<br>Interrupt Timer (PIT) | Contents .....                      | 295 |
|                                       | Introduction .....                  | 295 |
|                                       | Features .....                      | 296 |
|                                       | Functional Description .....        | 296 |
|                                       | PIT Counter Prescaler .....         | 298 |
|                                       | Low-Power Modes .....               | 298 |
|                                       | PIT During Break Interrupts .....   | 299 |
|                                       | I/O Registers .....                 | 300 |
| I/O Ports                             | Contents .....                      | 305 |
|                                       | Introduction .....                  | 306 |
|                                       | Port A .....                        | 307 |
|                                       | Port B .....                        | 309 |
|                                       | Port C .....                        | 312 |
|                                       | Port D .....                        | 315 |
|                                       | Port E .....                        | 318 |
|                                       | Port F .....                        | 322 |
|                                       | Port G .....                        | 325 |
| Port H .....                          | 327                                 |     |
| MSCAN Controller<br>(MSCAN08)         | Contents .....                      | 331 |
|                                       | Introduction .....                  | 332 |
|                                       | Features .....                      | 333 |
|                                       | External Pins .....                 | 334 |
|                                       | Message Storage .....               | 335 |
|                                       | Identifier Acceptance Filter .....  | 340 |
|                                       | Interrupts .....                    | 344 |
|                                       | Protocol Violation Protection ..... | 346 |
|                                       | Low Power Modes .....               | 346 |
|                                       | Timer Link .....                    | 350 |
| Clock System .....                    | 351                                 |     |

## Table of Contents

|   |   |     |
|---|---|-----|
|   | Memory Map .....                                | 355 |
|   | Programmer's Model of Message Storage .....     | 355 |
|   | Programmer's Model of Control Registers .....   | 360 |
| Keyboard Module<br>(KBD)                | Contents .....                                  | 381 |
|   | Introduction .....                              | 381 |
|   | Features .....                                  | 382 |
|   | Functional Description .....                    | 382 |
|   | Keyboard Initialization .....                   | 385 |
|   | Low-Power Modes .....                           | 386 |
|   | Keyboard Module During Break Interrupts .....   | 386 |
|   | I/O Registers .....                             | 387 |
| Timer Interface<br>Module A (TIMA-6)    | Contents .....                                  | 389 |
|   | Introduction .....                              | 390 |
|   | Features .....                                  | 390 |
|   | Functional Description .....                    | 393 |
|   | Interrupts .....                                | 402 |
|   | Low-Power Modes .....                           | 403 |
|   | TIMA During Break Interrupts .....              | 404 |
|   | I/O Signals .....                               | 405 |
|   | I/O Registers .....                             | 406 |
| Analog-to-Digital<br>Converter (ADC-15) | Contents .....                                  | 421 |
|   | Introduction .....                              | 421 |
|   | Features .....                                  | 422 |
|   | Functional Description .....                    | 422 |
|   | Interrupts .....                                | 425 |
|   | Low-Power Modes .....                           | 425 |
|   | I/O Signals .....                               | 426 |
|   | I/O Registers .....                             | 427 |
| Specifications                          | Contents .....                                  | 433 |
|   | Electrical Specifications .....                 | 434 |
|   | Mechanical Specifications .....                 | 447 |
| Appendix A: Future<br>EEPROM Registers  | EEPROM Timebase Divider Control Registers ..... | 449 |
|   | EEDIVH and EEDIVL Registers .....               | 450 |
|   | EEDIV Non-volatile Registers .....              | 451 |

Glossary

|                    |   |     |
|--------------------|---|-----|
| Literature Updates | Literature Distribution Centers . . . . .     | 465 |
|                    | Customer Focus Center . . . . .               | 466 |
|                    | Microcontroller Division's Web Site . . . . . | 466 |

|                  |   |     |
|------------------|---|-----|
| Revision History | Major Changes Between Revision 2.0 and Revision 1.0 . . . . . | 469 |
|------------------|---|-----|

## Table of Contents

# General Description

---

---

## Contents

- Introduction ..... 11
- Features ..... 12
- MCU Block Diagram ..... 13
- Pin Assignments ..... 15
  - Power Supply Pins (VDD and VSS) ..... 16
  - Oscillator Pins (OSC1 and OSC2) ..... 16
  - External Reset Pin (RST) ..... 17
  - External Interrupt Pin (IRQ) ..... 17
  - Analog Power Supply Pin (VDDA) ..... 17
  - Analog Ground Pin (VSSA) ..... 17
  - External Filter Capacitor Pin (CGMXFC) ..... 17
  - Port A Input/Output (I/O) Pins (PTA7–PTA0) ..... 17
  - Port B I/O Pins (PTB7/ATD7–PTB0/ATD0) ..... 17
  - Port C I/O Pins (PTC5–PTC0) ..... 17
  - Port D I/O Pins (PTD7–PTD0/ATD8) ..... 18
  - Port E I/O Pins (PTE7/SPSCK–PTE0/TxD) ..... 18
  - Port F I/O Pins (PTF6–PTF0/TACH2) ..... 18
  - Port G I/O Pins (PTG2/KBD2–PTG0/KBD0) ..... 18
  - Port H I/O Pins (PTH1/KBD4–PTH0/KBD3) ..... 18
  - CAN Transmit Pin (CANTx) ..... 19
  - CAN Receive Pin (CANRx) ..... 19
- Ordering Information ..... 22
  - MC Order Numbers ..... 22

---

---

## Introduction

The MC68HC908AZ60 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). The M68HC08 Family is based on the customer-specified integrated circuit (CSIC) design strategy. All MCUs in the family use the enhanced M68HC08

## General Description

central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

This part is designed to emulate the MC68HC08AZxx automotive family.

*In AZxx mode the MC68HC908AZ60 offers extra features which are not available on the MC68HC08AZ32 device. It is the user's responsibility to ensure compatibility between the features used on the MC68HC908AZ60 and those which are available on the device which will ultimately be used in the application.*

---

---

## Features

Features of the MC68HC908AZ60 include:

- High-Performance M68HC08 Architecture
- Fully Upward-Compatible Object Code with M6805, M146805, and M68HC05 Families
- 8.4 MHz Internal Bus Frequency
- 60 Kbytes of FLASH Electrically Erasable Read-Only Memory (FLASH)
- FLASH Data Security
- 1 Kbyte of On-Chip Electrically Erasable Programmable Read-Only Memory with Security Option (EEPROM)
- 2 Kbyte of On-Chip RAM
- Clock Generator Module (CGM)
- Serial Peripheral Interface Module (SPI)
- Serial Communications Interface Module (SCI)
- 8-Bit, 15-Channel Analog-to-Digital Converter (ADC-15)
- 16-Bit, 6-Channel Timer Interface Module (TIMA-6)
- Programmable Interrupt Timer (PIT)
- System Protection Features

- Computer Operating Properly (COP) with Optional Reset
- Low-Voltage Detection with Optional Reset
- Illegal Opcode Detection with Optional Reset
- Illegal Address Detection with Optional Reset
- Low-Power Design (Fully Static with Stop and Wait Modes)
- Master Reset Pin and Power-On Reset
- 16-Bit, 2-Channel Timer Interface Module (TIMB)
- 5-Bit Keyboard Interrupt Module
- MSCAN Controller (Motorola Scalable CAN) implements CAN 2.0b Protocol as Defined in BOSCH Specification September 1991

Features of the CPU08 include:

- Enhanced HC05 Programming Model
- Extensive Loop Control Functions
- 16 Addressing Modes (Eight More Than the HC05)
- 16-Bit Index Register and Stack Pointer
- Memory-to-Memory Data Transfers
- Fast  $8 \times 8$  Multiply Instruction
- Fast 16/8 Divide Instruction
- Binary-Coded Decimal (BCD) Instructions
- Optimization for Controller Applications
- C Language Support

---

---

## MCU Block Diagram

**Figure 1** shows the structure of the MC68HC908AZ60.

# General Description

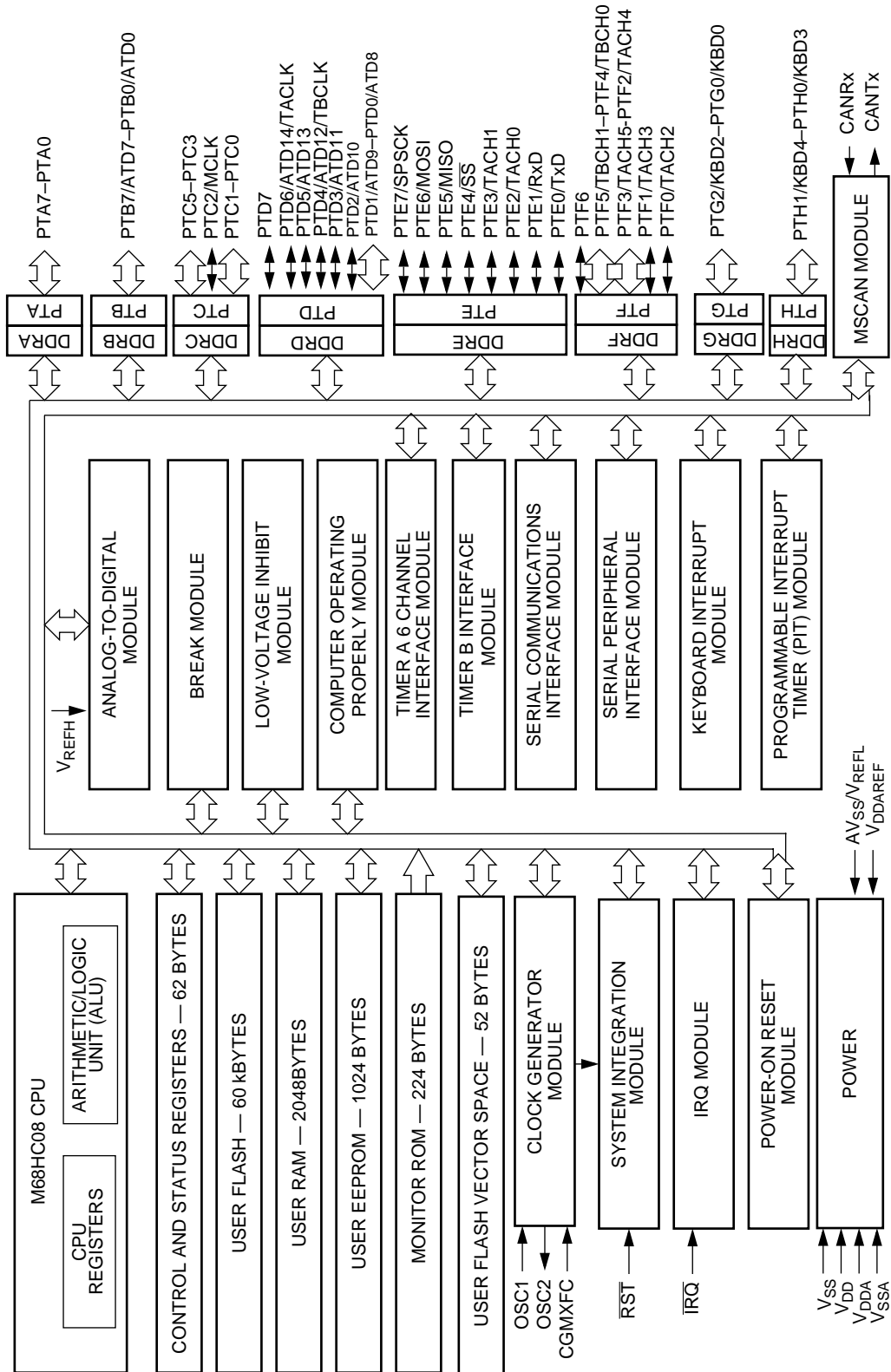


Figure 1. MCU Block Diagram for the MC68HC908AZ60 (64-Pin QFP)



## Pin Assignments

Figure 2 shows the MC68HC908AZ60 pin assignments.

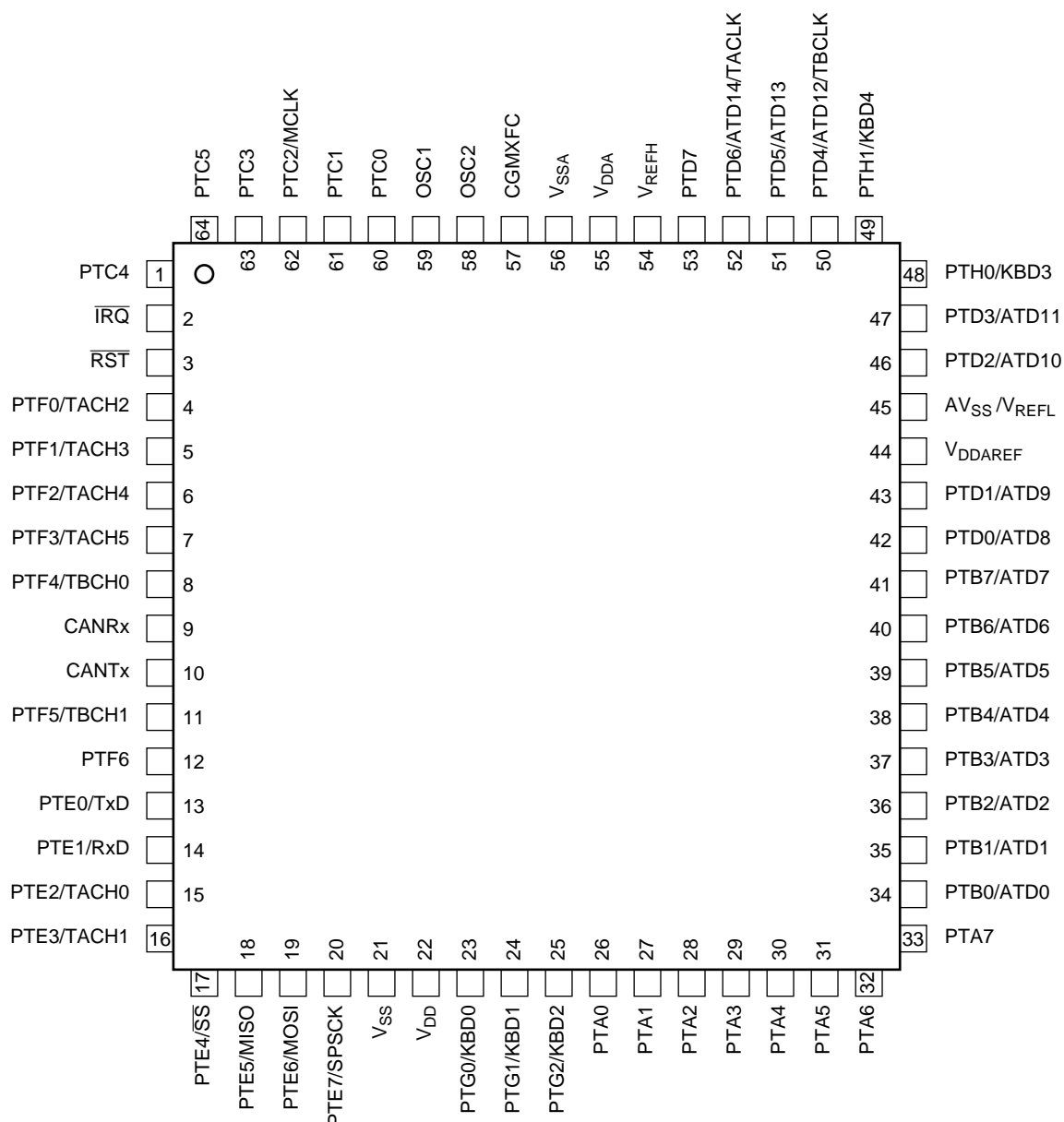


Figure 2. MC68HC908AZ60 (64-Pin QFP)

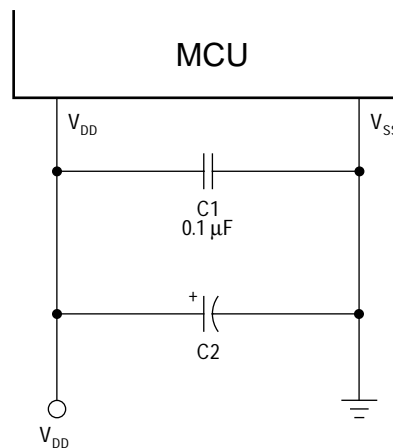
**NOTE:** The following pin descriptions are just a quick reference. For a more detailed representation, see [I/O Ports](#) on page 305.

## General Description

### Power Supply Pins ( $V_{DD}$ and $V_{SS}$ )

$V_{DD}$  and  $V_{SS}$  are the power supply and ground pins. The MCU operates from a single power supply.

Fast signal transitions on MCU pins place high, short-duration current demands on the power supply. To prevent noise problems, take special care to provide power supply bypassing at the MCU as shown in [Figure 3](#). Place the C1 bypass capacitor as close to the MCU as possible. Use a high-frequency response ceramic capacitor for C1. C2 is an optional bulk current bypass capacitor for use in applications that require the port pins to source high current levels.



NOTE: Component values shown represent typical applications.

**Figure 3. Power supply bypassing**

$V_{SS}$  is also the ground for the port output buffers and the ground return for the serial clock in the serial peripheral interface module (SPI). See [Serial Peripheral Interface Module \(SPI\)](#) on page 237.

**NOTE:**  $V_{SS}$  must be grounded for proper MCU operation.

### Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. See [Clock Generator Module \(CGM\)](#) on page 127.

|  |   |
|--|---|
| <b>External Reset Pin (<math>\overline{\text{RST}}</math>)</b>     | A logic 0 on the $\overline{\text{RST}}$ pin forces the MCU to a known startup state. $\overline{\text{RST}}$ is bidirectional, allowing a reset of the entire system. It is driven low when any internal reset source is asserted. See <a href="#">System Integration Module (SIM)</a> on page 105 for more information. |
| <b>External Interrupt Pin (<math>\overline{\text{IRQ}}</math>)</b> | $\overline{\text{IRQ}}$ is an asynchronous external interrupt pin. See <a href="#">External Interrupt Module (IRQ)</a> on page 191.   |
| <b>Analog Power Supply Pin (<math>V_{\text{DDA}}</math>)</b>       | $V_{\text{DDA}}$ is the power supply pin for the analog portion of the chip. This pin will supply the clock generator module (CGM). See <a href="#">Clock Generator Module (CGM)</a> on page 127.   |
| <b>Analog Ground Pin (<math>V_{\text{SSA}}</math>)</b>             | The $V_{\text{SSA}}$ analog ground pin is used only for the ground connections for the analog sections of the circuit and should be decoupled as per the $V_{\text{SS}}$ digital ground pin. The analog sections consist of a clock generator module (CGM). See <a href="#">Clock Generator Module (CGM)</a> on page 127. |
| <b>External Filter Capacitor Pin (CGMXFC)</b>                      | CGMXFC is an external filter capacitor connection for the CGM. See <a href="#">Clock Generator Module (CGM)</a> on page 127   |
| <b>Port A Input/Output (I/O) Pins (PTA7–PTA0)</b>                  | PTA7–PTA0 are general-purpose bidirectional I/O port pins. See <a href="#">I/O Ports</a> on page 305.   |
| <b>Port B I/O Pins (PTB7/ATD7–PTB0/ATD0)</b>                       | Port B is an 8-bit special function port that shares all eight pins with the analog-to-digital converter (ADC). See <a href="#">Analog-to-Digital Converter (ADC-15)</a> on page 421 and <a href="#">I/O Ports</a> on page 305.   |
| <b>Port C I/O Pins (PTC5–PTC0)</b>                                 | PTC5–PTC3 and PTC1–PTC0 are general-purpose bidirectional I/O port pins. PTC2/MCLK is a special function port that shares its pin with  |

## General Description

the system clock which has a frequency equivalent to the system clock. See [I/O Ports](#) on page 305.

### Port D I/O Pins (PTD7–PTD0/ATD8)

Port D is an 8-bit special-function port that shares seven of its pins with the analog-to-digital converter module (ADC-15), one of its pins with the timer interface module (TIMA), and one more of its pins with the timer interface module (TIMB). See [Timer Interface Module A \(TIMA-6\)](#) on page 389, [Analog-to-Digital Converter \(ADC-15\)](#) on page 421 and [I/O Ports](#) on page 305.

### Port E I/O Pins (PTE7/SPSCK–PTE0/ TxD)

Port E is an 8-bit special function port that shares two of its pins with the timer interface module (TIMA), four of its pins with the serial peripheral interface module (SPI), and two of its pins with the serial communication interface module (SCI). See [Serial Communications Interface Module \(SCI\)](#) on page 199, [Serial Peripheral Interface Module \(SPI\)](#) on page 237, [Timer Interface Module A \(TIMA-6\)](#) on page 389, and [I/O Ports](#) on page 305.

### Port F I/O Pins (PTF6–PTF0/TACH2)

Port F is a 7-bit special function port that shares its pins with the timer interface module (TIMB). Six of its pins are shared with the timer interface module (TIMA-6). See [Timer Interface Module A \(TIMA-6\)](#) on page 389, [Timer Interface Module B \(TIMB\)](#) on page 269, and [I/O Ports](#) on page 305.

### Port G I/O Pins (PTG2/KBD2–PTG0/ KBD0)

Port G is a 3-bit special function port that shares all of its pins with the keyboard interrupt module (KBD). See [Keyboard Module \(KBD\)](#) on page 381 and [I/O Ports](#) on page 305.

### Port H I/O Pins (PTH1/KBD4–PTH0/ KBD3)

Port H is a 2-bit special-function port that shares all of its pins with the keyboard interrupt module (KBD). See [Keyboard Module \(KBD\)](#) on page 381 and [I/O Ports](#) on page 305.

**CAN Transmit Pin (CANTx)** This pin is the digital output from the CAN module (CANTx). See [MSCAN Controller \(MSCAN08\)](#) on page 331.

**CAN Receive Pin (CANRx)** This pin is the digital input to the CAN module (CANRx). See [MSCAN Controller \(MSCAN08\)](#) on page 331.

**Table 1. External Pins Summary**

| Pin Name                          | Function   | Driver Type              | Hysteresis (note 1) | Reset State |
|-----------------------------------|--|--------------------------|---------------------|-------------|
| PTA7–PTA0                         | General-Purpose I/O  | Dual State               | No                  | Input Hi-Z  |
| PTB7/ATD7–PTB0/ATD0               | General-Purpose I/O<br>ADC Channel                               | Dual State               | No                  | Input Hi-Z  |
| PTC5–PTC0                         | General-Purpose I/O  | Dual State               | No                  | Input Hi-Z  |
| PTD7                              | General Purpose I/O/   | Dual State               | No                  | Input Hi-Z  |
| PTD6/ATD14/TACLK ADC Channel      | General-Purpose I/O<br>ADC Channel/Timer<br>External Input Clock | Dual State               | No                  | Input Hi-Z  |
| PTD5/ATD13 ADC Channel            | General-Purpose I/O<br>ADC Channel                               | Dual State               | No                  | Input Hi-Z  |
| PTD4/ATD12/TBCLK ADC Channel      | General-Purpose I/O<br>ADC Channel/Timer<br>External Input Clock | Dual State               | No                  | Input Hi-Z  |
| PTD3/ATD11–PTD0/ATD8 ADC Channels | General-Purpose I/O<br>ADC Channel                               | Dual State               | No                  | Input Hi-Z  |
| PTE7/SPSCK                        | General-Purpose I/O<br>SPI Clock                                 | Dual State<br>Open Drain | Yes                 | Input Hi-Z  |
| PTE6/MOSI                         | General-Purpose I/O<br>SPI Data Path                             | Dual State<br>Open Drain | Yes                 | Input Hi-Z  |
| PTE5/MISO                         | General-Purpose I/O<br>SPI Data Path                             | Dual State<br>Open Drain | Yes                 | Input Hi-Z  |
| PTE4/ $\overline{SS}$             | General-Purpose I/O<br>SPI Slave Select                          | Dual State               | Yes                 | Input Hi-Z  |
| PTE3/TACH1                        | General-Purpose I/O<br>Timer Channel 1                           | Dual State               | Yes                 | Input Hi-Z  |
| PTE2/TACH0                        | General-Purpose I/O<br>Timer Channel 0                           | Dual State               | Yes                 | Input Hi-Z  |
| PTE1/RxD                          | General-Purpose I/O<br>SCI Receive Data                          | Dual State               | Yes                 | Input Hi-Z  |

## General Description

**Table 1. External Pins Summary (Continued)**

| Pin Name                            | Function                                      | Driver Type | Hysteresis (note 1) | Reset State |
|-------------------------------------|---|-------------|---------------------|-------------|
| PTE0/TxD                            | General-Purpose I/O<br>SCI Transmit Data      | Dual State  | No                  | Input Hi-Z  |
| PTF6                                | General-Purpose I/O                           | Dual State  | No                  | Input Hi-Z  |
| PTF5/TBCH1–PTF4/TBCH0               | General-Purpose<br>I/O/Timer B Channel        | Dual State  | Yes                 | Input Hi-Z  |
| PTF3/TACH5                          | General-Purpose I/O<br>Timer A Channel 5      | Dual State  | Yes                 | Input Hi-Z  |
| PTF2/TACH4                          | General-Purpose I/O<br>Timer A Channel 4      | Dual State  | Yes                 | Input Hi-Z  |
| PTF1/TACH3                          | General-Purpose I/O<br>Timer A Channel 3      | Dual State  | Yes                 | Input Hi-Z  |
| PTF0/TACH2                          | General-Purpose I/O<br>Timer A Channel 2      | Dual State  | Yes                 | Input Hi-Z  |
| PTG2/KBD2–PTG0/KBD0                 | General-Purpose I/O/<br>Keyboard Wakeup Pin   | Dual State  | Yes                 | Input Hi-Z  |
| PTH1/KBD4 –PTH0/KBD3                | General-Purpose I/O/<br>Keyboard Wakeup Pin   | Dual State  | Yes                 | Input Hi-Z  |
| V <sub>DD</sub>                     | Chip Power Supply                             | N/A         | N/A                 | N/A         |
| V <sub>SS</sub>                     | Chip Ground                                   | N/A         | N/A                 | N/A         |
| V <sub>DDAREF</sub>                 | ADC Power Supply/<br>ADC Reference<br>Voltage | N/A         | N/A                 | N/A         |
| A <sub>VSS</sub> /V <sub>REFL</sub> | ADC Ground/ADC<br>Reference Voltage           | N/A         | N/A                 | N/A         |
| V <sub>REFH</sub>                   | A/D Reference<br>Voltage                      | N/A         | N/A                 | N/A         |
| OSC1                                | External Clock In                             | N/A         | N/A                 | Input Hi-Z  |
| OSC2                                | External Clock Out                            | N/A         | N/A                 | Output      |
| CGMXFC                              | PLL Loop Filter Cap                           | N/A         | N/A                 | N/A         |
| IRQ                                 | External Interrupt<br>Request                 | N/A         | N/A                 | Input Hi-Z  |
| RST                                 | Reset   | N/A         | N/A                 | Output Low  |
| CANRx                               | CAN Serial Input                              | N/A         | Yes                 | Input Hi-Z  |
| CANTx                               | CAN Serial Output                             | Output      | No                  | Output      |

**Table 2. Clock Source Summary**

| <b>Module</b> | <b>Clock Source</b>           |
|---------------|-------------------------------|
| ADC           | CGMXCLK or Bus Clock          |
| CAN           | CGMXCLK or CGMOUT             |
| COP           | CGMXCLK                       |
| CPU           | Bus Clock                     |
| EEPROM        | RC OSC or Bus Clock           |
| SPI           | Bus Clock/SPSCK               |
| SCI           | CGMXCLK                       |
| TIMA-6        | Bus Clock or PTD6/ATD14/TACLK |
| TIMB          | Bus Clock or PTD4/TBCLK       |
| PIT           | Bus Clock                     |
| SIM           | CGMOUT and CGMXCLK            |
| IRQ           | Bus Clock                     |
| BRK           | Bus Clock                     |
| LVI           | Bus Clock                     |
| CGM           | OSC1 and OSC2                 |

*Note 1: Hysteresis is not 100% tested but is typically a minimum of 300mV*

## General Description

---

---

### Ordering Information

This section contains instructions for ordering the MC68HC908AZ60.

#### MC Order Numbers

**Table 3. MC Order Numbers**

| MC Order Number  | Operating Temperature Range |
|------------------|-----------------------------|
| MC68HC908AZ60CFU | -40 °C to + 85°C            |
| MC68HC908AZ60VFU | -40 °C to + 105 °C          |
| MC68HC908AZ60MFU | -40 °C to + 125 °C          |



# Memory Map

---

---

## Contents

|                    |    |
|--------------------|----|
| Introduction ..... | 23 |
| I/O Section .....  | 26 |

---

---

## Introduction

The CPU08 can address 64 Kbytes of memory space. The memory map, shown in [Figure 1](#), includes:

- 60 Kbytes of FLASH EEPROM
- 2048 Bytes of RAM
- 1024 Bytes of EEPROM with Protect Option
- 52 Bytes of User-Defined Vectors
- 224 Bytes of Monitor ROM

The following definitions apply to the memory map representation of reserved and unimplemented locations.

- **Reserved** — Accessing a reserved location can have unpredictable effects on MCU operation.
- **Unimplemented** — Accessing an unimplemented location causes an illegal address reset if illegal address resets are enabled.

# Memory Map

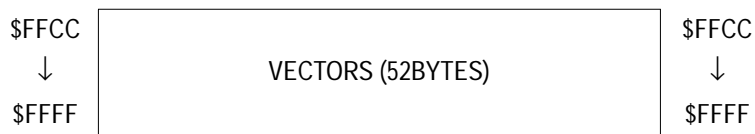
**Figure 1. Memory Map**

|        |  |        |
|--------|--|--------|
| \$0000 | I/O REGISTERS (64 BYTES)                   | \$0000 |
| ↓      |  | ↓      |
| \$003F | I/O REGISTERS, 16 BYTES                    | \$003F |
| ↓      |  | ↓      |
| \$0040 | RAM-1, 1024 BYTES                          | \$0040 |
| ↓      |  | ↓      |
| \$004F | FLASH-2, 176 BYTES                         | \$004F |
| ↓      |  | ↓      |
| \$0050 | CAN CONTROL AND MESSAGE BUFFERS, 128 BYTES | \$0050 |
| ↓      |  | ↓      |
| \$044F | FLASH-2, 128 BYTES                         | \$044F |
| ↓      |  | ↓      |
| \$0450 | EEPROM-2, 512 BYTES                        | \$0450 |
| ↓      |  | ↓      |
| \$04FF | EEPROM-1, 512 BYTES                        | \$04FF |
| ↓      |  | ↓      |
| \$0500 | RAM-2, 1024 BYTES                          | \$0500 |
| ↓      |  | ↓      |
| \$057F | FLASH-2, 29,184 BYTES                      | \$057F |
| ↓      |  | ↓      |
| \$0580 | FLASH-1, 32,256 BYTES                      | \$0580 |
| ↓      |  | ↓      |
| \$05FF | SIM BREAK STATUS REGISTER (SBSR)           | \$05FF |
| ↓      |  | ↓      |
| \$0600 | SIM RESET STATUS REGISTER (SRSR)           | \$0600 |
| ↓      |  | ↓      |
| \$07FF | RESERVED                                   | \$07FF |
| ↓      |  | ↓      |
| \$0800 | SIM BREAK FLAG CONTROL REGISTER (SBFCR)    | \$0800 |
| ↓      |  | ↓      |
| \$09FF |  | \$09FF |
| \$0A00 |  | \$0A00 |
| ↓      |  | ↓      |
| \$0DFF |  | \$0DFF |
| ↓      |  | ↓      |
| \$0E00 |  | \$0E00 |
| ↓      |  | ↓      |
| \$7FFF |  | \$7FFF |
| ↓      |  | ↓      |
| \$8000 |  | \$8000 |
| ↓      |  | ↓      |
| \$FDFF |  | \$FDFF |
| ↓      |  | ↓      |
| \$FE00 |  | \$FE00 |
| ↓      |  | ↓      |
| \$FE01 |  | \$FE01 |
| ↓      |  | ↓      |
| \$FE02 |  | \$FE02 |
| ↓      |  | ↓      |
| \$FE03 |  | \$FE03 |

**Figure 1. Memory Map (Continued)**

|        |   |        |
|--------|---|--------|
| \$FE04 | RESERVED  | \$FE04 |
| \$FE05 | RESERVED  | \$FE05 |
| \$FE06 | UNIMPLEMENTED                                   | \$FE06 |
| \$FE07 | RESERVED  | \$FE07 |
| \$FE08 | RESERVED  | \$FE08 |
| \$FE09 | CONFIGURATION WRITE-ONCE REGISTER<br>(CONFIG-2) | \$FE09 |
| \$FE0A | RESERVED  | \$FE0A |
| \$FE0B | FLASH CONTROL REGISTER (FLCR1)                  | \$FE0B |
| \$FE0C | BREAK ADDRESS REGISTER HIGH (BRKH)              | \$FE0C |
| \$FE0D | BREAK ADDRESS REGISTER LOW (BRKL)               | \$FE0D |
| \$FE0E | BREAK STATUS AND CONTROL REGISTER (BSCR)        | \$FE0E |
| \$FE0F | LVI STATUS REGISTER (LVISR)                     | \$FE0F |
| \$FE10 | RESERVED  | \$FE10 |
| \$FE11 | FLASH CONTROL REGISTER (FLCR2)                  | \$FE11 |
| \$FE12 | UNIMPLEMENTED (5BYTES)                          | \$FE12 |
| ↓      |   | ↓      |
| \$FE17 | EEPROM NON-VOLATILE REGISTER (EENVR2)           | \$FE17 |
| \$FE18 | EEPROM CONTROL REGISTER (EECR2)                 | \$FE18 |
| \$FE19 | RESERVED  | \$FE19 |
| \$FE1A | EEPROM ARRAY CONFIGURATION (EEACR2)             | \$FE1A |
| \$FE1B | EEPROM NON-VOLATILE REGISTER (EENVR1)           | \$FE1B |
| \$FE1C | EEPROM CONTROL REGISTER (EECR1)                 | \$FE1C |
| \$FE1D | RESERVED  | \$FE1D |
| \$FE1E | EEPROM ARRAY CONFIGURATION (EEACR1)             | \$FE1E |
| \$FE1F | MONITOR ROM (224 BYTES)                         | \$FE1F |
| \$FE20 |   | ↓      |
| \$FEFF | UNIMPLEMENTED (128 BYTES)                       | \$FEFF |
| \$FF00 |   | ↓      |
| ↓      |   | ↓      |
| \$FF7F | FLASH BLOCK PROTECT REGISTER (FLBPR1)           | \$FF7F |
| \$FF80 | FLASH BLOCK PROTECT REGISTER (FLBPR2)           | \$FF80 |
| \$FF81 | RESERVED (75 BYTES)                             | \$FF81 |
| \$FF82 |   | ↓      |
| ↓      |   | ↓      |
| \$FFCB |   | \$FFCB |

**Figure 1. Memory Map (Continued)**



## I/O Section

Addresses \$0000–\$003F, shown in [Figure 2](#), contain most of the control, status, and data registers. Additional I/O registers have these addresses:

- \$FE00 (SIM break status register, SBSR)
- \$FE01 (SIM reset status register, SRSR)
- \$FE03 (SIM break flag control register, SBFCR)
- \$FE09 (configuration write-once register, CONFIG-2)
- \$FE0B (FLASH control register, FLCR1)
- \$FE0C and \$FE0D (break address registers, BRKH and BRKL)
- \$FE0E (break status and control register, BRKSCR)
- \$FE0F (LVI status register, LVISR)
- \$FE11 (FLASH control register, FLCR2)
- \$FE18 (EEPROM non-volatile register, EENVR2)
- \$FE19 (EEPROM control register, EECR2)
- \$FE1B (EEPROM array configuration register, EEACR2)
- \$FE1C (EEPROM non-volatile register, EENVR1)
- \$FE1D (EEPROM control register, EECR1)
- \$FE1F (EEPROM array configuration register, EEACR1)
- \$FF80 (FLASH block protect register, FLBPR1)
- \$FF81 (FLASH block protect register, FLBPR2)
- \$FFFF (COP control register, COPCTL)

[Table 1](#) is a list of vector locations.

| Addr.  | Register Name                    | Bit 7  | 6      | 5     | 4      | 3     | 2     | 1     | Bit 0 |       |
|--------|----------------------------------|--------|--------|-------|--------|-------|-------|-------|-------|-------|
| \$0000 | Port A Data Register (PTA)       | Read:  | PTA7   | PTA6  | PTA5   | PTA4  | PTA3  | PTA2  | PTA1  | PTA0  |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$0001 | Port B Data Register (PTB)       | Read:  | PTB7   | PTB6  | PTB5   | PTB4  | PTB3  | PTB2  | PTB1  | PTB0  |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$0002 | Port C Data Register (PTC)       | Read:  | 0      | 0     | PTC5   | PTC4  | PTC3  | PTC2  | PTC1  | PTC0  |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$0003 | Port D Data Register (PTD)       | Read:  | PTD7   | PTD6  | PTD5   | PTD4  | PTD3  | PTD2  | PTD1  | PTD0  |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$0004 | Data Direction Register A (DDRA) | Read:  | DDRA7  | DDRA6 | DDRA5  | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$0005 | Data Direction Register B (DDRB) | Read:  | DDRB7  | DDRB6 | DDRB5  | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$0006 | Data Direction Register C (DDRC) | Read:  | MCLKEN | 0     | DDRC5  | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$0007 | Data Direction Register D (DDRD) | Read:  | DDRD7  | DDRD6 | DDRD5  | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$0008 | Port E Data Register (PTE)       | Read:  | PTE7   | PTE6  | PTE5   | PTE4  | PTE3  | PTE2  | PTE1  | PTE0  |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$0009 | Port F Data Register (PTF)       | Read:  | 0      | PTF6  | PTF5   | PTF4  | PTF3  | PTF2  | PTF1  | PTF0  |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$000A | Port G Data Register (PTG)       | Read:  | 0      | 0     | 0      | 0     | 0     | PTG2  | PTG1  | PTG0  |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$000B | Port H Data Register (PTH)       | Read:  | 0      | 0     | 0      | 0     | 0     | 0     | PTH1  | PTH0  |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$000C | Data Direction Register E (DDRE) | Read:  | DDRE7  | DDRE6 | DDRE5  | DDRE4 | DDRE3 | DDRE2 | DDRE1 | DDRE0 |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$000D | Data Direction Register F (DDRF) | Read:  | 0      | DDRF6 | DDRF5  | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$000E | Data Direction Register G (DDRG) | Read:  | 0      | 0     | 0      | 0     | 0     | DDRG2 | DDRG1 | DDRG0 |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$000F | Data Direction Register H (DDRH) | Read:  | 0      | 0     | 0      | 0     | 0     | 0     | DDRH1 | DDRH0 |
|        |                                  | Write: |        |       |        |       |       |       |       |       |
| \$0010 | SPI Control Register (SPCR)      | Read:  | SPRIE  | R     | SPMSTR | CPOL  | CPHA  | SPWOM | SPE   | SPTIE |
|        |                                  | Write: |        |       |        |       |       |       |       |       |

**Figure 2. Control, Status, and Data Registers (Sheet 1 of 6)**

## Memory Map

| Addr.  | Register Name                                |        | Bit 7   | 6               | 5                | 4      | 3     | 2          | 1      | Bit 0 |
|--------|--|--------|---------|-----------------|------------------|--------|-------|------------|--------|-------|
|        |  |        |         | = Unimplemented |                  |        | R     | = Reserved |        |       |
| \$0011 | SPI Status and Control Register (SPSCR)      | Read:  | SPRF    | ERRIE           | OVRF             | MODF   | SPTF  | MODFEN     | SPR1   | SPR0  |
|        |  | Write: |         |                 |                  |        |       |            |        |       |
| \$0012 | SPI Data Register (SPDR)                     | Read:  | R7      | R6              | R5               | R4     | R3    | R2         | R1     | R0    |
|        |  | Write: | T7      | T6              | T5               | T4     | T3    | T2         | T1     | T0    |
| \$0013 | SCI Control Register 1 (SCC1)                | Read:  | LOOPS   | ENSCI           | TXINV            | M      | WAKE  | ILTY       | PEN    | PTY   |
|        |  | Write: |         |                 |                  |        |       |            |        |       |
| \$0014 | SCI Control Register 2 (SCC2)                | Read:  | SCTIE   | TCIE            | SCRIE            | ILIE   | TE    | RE         | RWU    | SBK   |
|        |  | Write: |         |                 |                  |        |       |            |        |       |
| \$0015 | SCI Control Register 3 (SCC3)                | Read:  | R8      | T8              | R                | R      | ORIE  | NEIE       | FEIE   | PEIE  |
|        |  | Write: |         |                 |                  |        |       |            |        |       |
| \$0016 | SCI Status Register 1 (SCS1)                 | Read:  | SCTE    | TC              | SCRf             | IDLE   | OR    | NF         | FE     | PE    |
|        |  | Write: |         |                 |                  |        |       |            |        |       |
| \$0017 | SCI Status Register 2 (SCS2)                 | Read:  | 0       | 0               | 0                | 0      | 0     | 0          | BKF    | RPF   |
|        |  | Write: |         |                 |                  |        |       |            |        |       |
| \$0018 | SCI Data Register (SCDR)                     | Read:  | R7      | R6              | R5               | R4     | R3    | R2         | R1     | R0    |
|        |  | Write: | T7      | T6              | T5               | T4     | T3    | T2         | T1     | T0    |
| \$0019 | SCI Baud Rate Register (SCBR)                | Read:  | 0       | 0               | SCP1             | SCP0   |       | SCR2       | SCR1   | SCR0  |
|        |  | Write: |         |                 |                  |        |       |            |        |       |
| \$001A | IRQ Status and Control Register (ISCR)       | Read:  | 0       | 0               | 0                | 0      | IRQF  | 0          | IMASK1 | MODE1 |
|        |  | Write: |         |                 |                  |        |       | ACK1       |        |       |
| \$001B | Keyboard Status and Control Register (KBSCR) | Read:  | 0       | 0               | 0                | 0      | KEYF  | 0          | IMASKK | MODEK |
|        |  | Write: |         |                 |                  |        |       | ACKK       |        |       |
| \$001C | PLL Control Register (PCTL)                  | Read:  | PLLIE   | PLLF            | PLLON            | BCS    | 1     | 1          | 1      | 1     |
|        |  | Write: |         |                 |                  |        |       |            |        |       |
| \$001D | PLL Bandwidth Control Register (PBWC)        | Read:  | AUTO    | LOCK            | $\overline{ACQ}$ | XLD    | 0     | 0          | 0      | 0     |
|        |  | Write: |         |                 |                  |        |       |            |        |       |
| \$001E | PLL Programming Register (PPG)               | Read:  | MUL7    | MUL6            | MUL5             | MUL4   | VRS7  | VRS6       | VRS5   | VRS4  |
|        |  | Write: |         |                 |                  |        |       |            |        |       |
| \$001F | Configuration Write-Once Register (CONFIG-1) | Read:  | LVISTOP | R               | LVIRST           | LVIPWR | SSREC | COPL       | STOP   | COPD  |
|        |  | Write: |         |                 |                  |        |       |            |        |       |
| \$0020 | Timer A Status and Control Register (TASC)   | Read:  | TOF     | TOIE            | TSTOP            | 0      | 0     | PS2        | PS1    | PS0   |
|        |  | Write: | 0       |                 |                  | TRST   |       |            |        |       |
| \$0021 | Keyboard Interrupt Enable Register (KBIE)    | Read:  | 0       | 0               | 0                | KBIE4  | KBIE3 | KBIE2      | KBIE1  | KBIE0 |
|        |  | Write: |         |                 |                  |        |       |            |        |       |

Figure 2. Control, Status, and Data Registers (Sheet 2 of 6)

| Addr.  | Register Name   |        | Bit 7  | 6     | 5    | 4    | 3     | 2     | 1    | Bit 0  |
|--------|---|--------|--------|-------|------|------|-------|-------|------|--------|
| \$0022 | Timer A Counter Register High (TACNTH)                | Read:  | Bit 15 | 14    | 13   | 12   | 11    | 10    | 9    | Bit 8  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$0023 | Timer A Counter Register Low (TACNTL)                 | Read:  | Bit 7  | 6     | 5    | 4    | 3     | 2     | 1    | Bit 0  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$0024 | Timer A Modulo Register High (TAMODH)                 | Read:  | Bit 15 | 14    | 13   | 12   | 11    | 10    | 9    | Bit 8  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$0025 | Timer A Modulo Register Low (TAMODL)                  | Read:  | Bit 7  | 6     | 5    | 4    | 3     | 2     | 1    | Bit 0  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$0026 | Timer A Channel 0 Status and Control Register (TASC0) | Read:  | CH0F   | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
|        |   | Write: | 0      |       |      |      |       |       |      |        |
| \$0027 | Timer A Channel 0 Register High (TACH0H)              | Read:  | Bit 15 | 14    | 13   | 12   | 11    | 10    | 9    | Bit 8  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$0028 | Timer A Channel 0 Register Low (TACH0L)               | Read:  | Bit 7  | 6     | 5    | 4    | 3     | 2     | 1    | Bit 0  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$0029 | Timer A Channel 1 Status and Control Register (TASC1) | Read:  | CH1F   | CH1IE | 0    | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
|        |   | Write: | 0      |       |      |      |       |       |      |        |
| \$002A | Timer A Channel 1 Register High (TACH1H)              | Read:  | Bit 15 | 14    | 13   | 12   | 11    | 10    | 9    | Bit 8  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$002B | Timer A Channel 1 Register Low (TACH1L)               | Read:  | Bit 7  | 6     | 5    | 4    | 3     | 2     | 1    | Bit 0  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$002C | Timer A Channel 2 Status and Control Register (TASC2) | Read:  | CH2F   | CH2IE | MS2B | MS2A | ELS2B | ELS2A | TOV2 | CH2MAX |
|        |   | Write: | 0      |       |      |      |       |       |      |        |
| \$002D | Timer A Channel 2 Register High (TACH2H)              | Read:  | Bit 15 | 14    | 13   | 12   | 11    | 10    | 9    | Bit 8  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$002E | Timer A Channel 2 Register Low (TACH2L)               | Read:  | Bit 7  | 6     | 5    | 4    | 3     | 2     | 1    | Bit 0  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$002F | Timer A Channel 3 Status and Control Register (TASC3) | Read:  | CH3F   | CH3IE | 0    | MS3A | ELS3B | ELS3A | TOV3 | CH3MAX |
|        |   | Write: | 0      |       |      |      |       |       |      |        |
| \$0030 | Timer A Channel 3 Register High (TACH3H)              | Read:  | Bit 15 | 14    | 13   | 12   | 11    | 10    | 9    | Bit 8  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$0031 | Timer A Channel 3 Register Low (TACH3L)               | Read:  | Bit 7  | 6     | 5    | 4    | 3     | 2     | 1    | Bit 0  |
|        |   | Write: |        |       |      |      |       |       |      |        |
| \$0032 | Timer A Channel 4 Status and Control Register (TASC4) | Read:  | CH4F   | CH4IE | MS4B | MS4A | ELS4B | ELS4A | TOV4 | CH4MAX |
|        |   | Write: | 0      |       |      |      |       |       |      |        |
| \$0033 | Timer A Channel 4 Register High (TACH4H)              | Read:  | Bit 15 | 14    | 13   | 12   | 11    | 10    | 9    | Bit 8  |
|        |   | Write: |        |       |      |      |       |       |      |        |

**Figure 2. Control, Status, and Data Registers (Sheet 3 of 6)**

## Memory Map

| Addr.  | Register Name   |              | Bit 7       | 6     | 5     | 4         | 3     | 2     | 1     | Bit 0  |
|--------|---|--------------|-------------|-------|-------|-----------|-------|-------|-------|--------|
| \$0034 | Timer A Channel 4 Register Low (TACH4L)               | Read: Write: | Bit 7       | 6     | 5     | 4         | 3     | 2     | 1     | Bit 0  |
| \$0035 | Timer A Channel 5 Status and Control Register (TASC5) | Read: Write: | CH5F<br>0   | CH5IE | 0     | MS5A      | ELS5B | ELS5A | TOV5  | CH5MAX |
| \$0036 | Timer A Channel 5 Register High (TACH5H)              | Read: Write: | Bit 15      | 14    | 13    | 12        | 11    | 10    | 9     | Bit 8  |
| \$0037 | Timer A Channel 5 Register Low (TACH5L)               | Read: Write: | Bit 7       | 6     | 5     | 4         | 3     | 2     | 1     | Bit 0  |
| \$0038 | Analog-to-Digital Status and Control Register (ADSCR) | Read: Write: | COCO        | AIEN  | ADCO  | ADCH4     | ADCH3 | ADCH2 | ADCH1 | ADCH0  |
| \$0039 | Analog-to-Digital Data Register (ADR)                 | Read: Write: | AD7         | AD6   | AD5   | AD4       | AD3   | AD2   | AD1   | AD0    |
| \$003A | Analog-to-Digital Input Clock Register (ADICLK)       | Read: Write: | ADIV2       | ADIV1 | ADIV0 | ADICLK    | 0     | 0     | 0     | 0      |
| \$0040 | Timer B Status and Control Register (TBSCR)           | Read: Write: | TOF<br>TOIE | TOIE  | TSTOP | 0<br>TRST | 0     | PS2   | PS1   | PS0    |
| \$0041 | Timer B Counter Register High (TBCNTH)                | Read: Write: | Bit 15      | 14    | 13    | 12        | 11    | 10    | 9     | Bit 8  |
| \$0042 | Timer B Counter Register Low (TBCNTL)                 | Read: Write: | Bit 7       | 6     | 5     | 4         | 3     | 2     | 1     | Bit 0  |
| \$0043 | Timer B Modulo Register High (TBMODH)                 | Read: Write: | Bit 15      | 14    | 13    | 12        | 11    | 10    | 9     | Bit 8  |
| \$0044 | Timer B Modulo Register Low (TBMODL)                  | Read: Write: | Bit 7       | 6     | 5     | 4         | 3     | 2     | 1     | Bit 0  |
| \$0045 | Timer B CH0 Status and Control Register (TBSC0)       | Read: Write: | CH0F<br>0   | CH0IE | MS0B  | MS0A      | ELS0B | ELS0A | TOV0  | CH0MAX |
| \$0046 | Timer B CH0 Register High (TBCH0H)                    | Read: Write: | Bit 15      | 14    | 13    | 12        | 11    | 10    | 9     | Bit 8  |
| \$0047 | Timer B CH0 Register Low (TBCH0L)                     | Read: Write: | Bit 7       | 6     | 5     | 4         | 3     | 2     | 1     | Bit 0  |
| \$0048 | Timer B CH1 Status and Control Register (TBSC1)       | Read: Write: | CH1F<br>0   | CH1IE | 0     | MS1A      | ELS1B | ELS1A | TOV1  | CH1MAX |
| \$0049 | Timer B CH1 Register High (TBCH1H)                    | Read: Write: | Bit 15      | 14    | 13    | 12        | 11    | 10    | 9     | Bit 8  |
| \$004A | Timer B CH1 Register Low (TBCH1L)                     | Read: Write: | Bit 7       | 6     | 5     | 4         | 3     | 2     | 1     | Bit 0  |

Figure 2. Control, Status, and Data Registers (Sheet 4 of 6)



| Addr.  | Register Name                                |        | Bit 7  | 6     | 5     | 4       | 3      | 2     | 1     | Bit 0 |
|--------|--|--------|--------|-------|-------|---------|--------|-------|-------|-------|
| \$004B | PIT Status and Control Register (PSC)        | Read:  | POF    | PIE   | PSTOP | 0       | 0      | PPS2  | PPS1  | PPS0  |
|        |  | Write: |        |       |       | PRST    |        |       |       |       |
| \$004C | PIT Counter Register High (PCNTH)            | Read:  | Bit 15 | 14    | 13    | 12      | 11     | 10    | 9     | Bit 8 |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$004D | PIT Counter Register Low (PCNTL)             | Read:  | Bit 7  | 6     | 5     | 4       | 3      | 2     | 1     | Bit 0 |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$004E | PIT Modulo Register High (PMODH)             | Read:  | Bit 15 | 14    | 13    | 12      | 11     | 10    | 9     | Bit 8 |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$004F | PIT Modulo Register Low (PMODL)              | Read:  | Bit 7  | 6     | 5     | 4       | 3      | 2     | 1     | Bit 0 |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE00 | SIM Break Status Register (SBSR)             | Read:  | R      | R     | R     | R       | R      | R     | BW    | R     |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE01 | SIM Reset Status Register (SRSR)             | Read:  | POR    | PIN   | COP   | ILOP    | ILAD   | 0     | LVI   | 0     |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE03 | SIM Break Flag Control Register (SBFCR)      | Read:  | BCFE   | R     | R     | R       | R      | R     | R     | R     |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE09 | Configuration Write-Once Register (CONFIG-2) | Read:  | 0      | 0     | 0     | MSCAND  | 0      | 0     | 0     | AZxx  |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE0B | Flash Control Register (FLCR1)               | Read:  | FDIV1  | FDIV0 | BLK1  | BLK0    | HVEN   | VERF  | ERASE | PGM   |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE0C | Break Address Register High (BRKH)           | Read:  | Bit 15 | 14    | 13    | 12      | 11     | 10    | 9     | Bit 8 |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE0D | Break Address Register Low (BRKL)            | Read:  | Bit 7  | 6     | 5     | 4       | 3      | 2     | 1     | Bit 0 |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE0E | Break Status and Control Register (BRKSCR)   | Read:  | BRKE   | BRKA  | 0     | 0       | 0      | 0     | 0     | 0     |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE0F | LVI Status Register (LVISR)                  | Read:  | LVIOUT | 0     | 0     | 0       | 0      | 0     | 0     | 0     |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE11 | Flash Control Register (FLCR2)               | Read:  | FDIV1  | FDIV0 | BLK1  | BLK0    | HVEN   | VERF  | ERASE | PGM   |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE18 | EEPROM Nonvolatile Register (EENVR2)         | Read:  | EERA   | CON2  | CON1  | EEPRTCT | EEBP3  | EEBP2 | EEBP1 | EEBP0 |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE19 | EEPROM Control Register (EECR2)              | Read:  | EEBCLK | 0     | EEOFF | EERAS1  | EERAS0 | EELAT | 0     | EEPGM |
|        |  | Write: |        |       |       |         |        |       |       |       |
| \$FE1A | Reserved                                     | Read:  | R      | R     | R     | R       | R      | R     | R     | R     |
|        |  | Write: |        |       |       |         |        |       |       |       |

Figure 2. Control, Status, and Data Registers (Sheet 5 of 6)

## Memory Map

| Addr.  | Register Name                          |        | Bit 7                                | 6    | 5     | 4       | 3      | 2     | 1     | Bit 0 |
|--------|--|--------|--------------------------------------|------|-------|---------|--------|-------|-------|-------|
| \$FE1B | EEPROM Array Control Register (EEACR2) | Read:  | EERA                                 | CON2 | CON1  | EEPRTCT | EEBP3  | EEBP2 | EEBP1 | EEBP0 |
|        |  | Write: |                                      |      |       |         |        |       |       |       |
| \$FE1C | EEPROM Nonvolatile Register (EENVR1)   | Read:  | EERA                                 | CON2 | CON1  | EEPRTCT | EEBP3  | EEBP2 | EEBP1 | EEBP0 |
|        |  | Write: |                                      |      |       |         |        |       |       |       |
| \$FE1D | EEPROM Control Register (EECR1)        | Read:  | EEBCLK                               | 0    | EEOFF | EERAS1  | EERAS0 | EELAT | 0     | EPPGM |
|        |  | Write: |                                      |      |       |         |        |       |       |       |
| \$FE1E | Reserved                               | Read:  | R                                    | R    | R     | R       | R      | R     | R     | R     |
|        |  | Write: |                                      |      |       |         |        |       |       |       |
| \$FE1F | EEPROM Array Control Register (EEACR1) | Read:  | EERA                                 | CON2 | CON1  | EEPRTCT | EEBP3  | EEBP2 | EEBP1 | EEBP0 |
|        |  | Write: |                                      |      |       |         |        |       |       |       |
| \$FF80 | FLASH Block Protect Register (FLBPR1)  | Read:  |                                      |      |       |         | BPR3   | BPR2  | BPR1  | BPR0  |
|        |  | Write: |                                      |      |       |         |        |       |       |       |
| \$FF81 | FLASH Block Protect Register (FLBPR2)  | Read:  |                                      |      |       |         | BPR3   | BPR2  | BPR1  | BPR0  |
|        |  | Write: |                                      |      |       |         |        |       |       |       |
| \$FFFF | COP Control Register (COPCTL)          | Read:  | LOW BYTE OF RESET VECTOR             |      |       |         |        |       |       |       |
|        |  | Write: | WRITING TO \$FFFF CLEARS COP COUNTER |      |       |         |        |       |       |       |

**Figure 2. Control, Status, and Data Registers (Sheet 6 of 6)**

**Table 1. Vector Addresses**

| Address | Vector                       |
|---------|------------------------------|
| \$FFCC  | TIMA Channel 5 Vector (High) |
| \$FFCD  | TIMA Channel 5 Vector (Low)  |
| \$FFCE  | TIMA Channel 4 Vector (High) |
| \$FFCF  | TIMA Channel 4 Vector (Low)  |
| \$FFD0  | ADC Vector (High)            |
| \$FFD1  | ADC Vector (Low)             |
| \$FFD2  | Keyboard Vector (High)       |
| \$FFD3  | Keyboard Vector (Low)        |
| \$FFD4  | SCI Transmit Vector (High)   |
| \$FFD5  | SCI Transmit Vector (Low)    |
| \$FFD6  | SCI Receive Vector (High)    |
| \$FFD7  | SCI Receive Vector (Low)     |
| \$FFD8  | SCI Error Vector (High)      |
| \$FFD9  | SCI Error Vector (Low)       |
| \$FFDA  | CAN Transmit Vector (High)   |
| \$FFDB  | CAN Transmit Vector (Low)    |
| \$FFDC  | CAN Receive Vector (High)    |
| \$FFDD  | CAN Receive Vector (Low)     |
| \$FFDE  | CAN Error Vector (High)      |
| \$FFDF  | CAN Error Vector (Low)       |
| \$FFE0  | CAN Wakeup Vector (High)     |
| \$FFE1  | CAN Wakeup Vector (Low)      |
| \$FFE2  | SPI Transmit Vector (High)   |
| \$FFE3  | SPI Transmit Vector (Low)    |
| \$FFE4  | SPI Receive Vector (High)    |
| \$FFE5  | SPI Receive Vector (Low)     |
| \$FFE6  | TIMB Overflow Vector (High)  |
| \$FFE7  | TIMB Overflow Vector (Low)   |
| \$FFE8  | TIMB CH1 Vector (High)       |
| \$FFE9  | TIMB CH1 Vector (Low)        |
| \$FFEA  | TIMB CH0 Vector (High)       |
| \$FFEB  | TIMB CH0 Vector (Low)        |
| \$FFEC  | TIMA Overflow Vector (High)  |
| \$FFED  | TIMA Overflow Vector (Low)   |
| \$FFEE  | TIMA CH3 Vector (High)       |
| \$FFEF  | TIMA CH3 Vector (Low)        |

Priority ↑ Low

**Table 1. Vector Addresses (Continued)**

| Address | Vector                 |
|---------|------------------------|
| \$FFF0  | TIMA CH2 Vector (High) |
| \$FFF1  | TIMA CH2 Vector (Low)  |
| \$FFF2  | TIMA CH1 Vector (High) |
| \$FFF3  | TIMA CH1 Vector (Low)  |
| \$FFF4  | TIMA CH0 Vector (High) |
| \$FFF5  | TIMA CH0 Vector (Low)  |
| \$FFF6  | PIT Vector (High)      |
| \$FFF7  | PIT Vector (Low)       |
| \$FFF8  | PLL Vector (High)      |
| \$FFF9  | PLL Vector (Low)       |
| \$FFFA  | IRQ1 Vector (High)     |
| \$FFFB  | IRQ1 Vector (Low)      |
| \$FFFC  | SWI Vector (High)      |
| \$FFFD  | SWI Vector (Low)       |
| \$FFFE  | Reset Vector (High)    |
| \$FFFF  | Reset Vector (Low)     |

It is recommended that all vector addresses are defined.

---

---

## Contents

|                                  |    |
|----------------------------------|----|
| Introduction . . . . .           | 35 |
| Functional Description . . . . . | 35 |

---

---

## Introduction

This section describes the 2048 bytes of random-access memory (RAM).

---

---

## Functional Description

Addresses \$0050 through \$044F and \$0A00 through \$0DFF are RAM locations. The location of the stack RAM is programmable with the reset stack pointer instruction (RSP). The 16-bit stack pointer allows the stack RAM to be anywhere in the 64K-byte memory space.

**NOTE:** *For correct operation, the stack pointer must point only to RAM locations.*

Within page zero are 176 bytes of RAM. Because the location of the stack RAM is programmable, all page zero RAM locations can be used for input/output (I/O) control and user data or code. When the stack pointer is moved from its reset location at \$00FF, direct addressing mode instructions can access all page zero RAM locations efficiently. Page zero RAM, therefore, provides ideal locations for frequently accessed global variables.

Before processing an interrupt, the CPU uses five bytes of the stack to save the contents of the CPU registers.

## RAM

**NOTE:** *For M68HC05, M6805, and M146805 compatibility, the H register is not stacked.*

During a subroutine call, the CPU uses two bytes of the stack to store the return address. The stack pointer decrements during pushes and increments during pulls.

**NOTE:** *Be careful when using nested subroutines. The CPU could overwrite data in the RAM during a subroutine or during the interrupt stacking operation.*

# FLASH-1 Memory

---

---

## Contents

|   |    |
|---|----|
| Introduction .....                        | 37 |
| Future FLASH Memory .....                 | 37 |
| Functional Description .....              | 38 |
| FLASH-1 Control Register .....            | 39 |
| FLASH Charge Pump Frequency Control ..... | 41 |
| FLASH Erase Operation .....               | 41 |
| FLASH Program/Margin Read Operation ..... | 43 |
| Smart Programming Algorithm .....         | 45 |
| FLASH Block Protection .....              | 46 |
| FLASH-1 Block Protect Register .....      | 47 |
| FLASH-2 Block Protect Register .....      | 48 |
| Low-Power Modes .....                     | 50 |
| WAIT Mode .....                           | 50 |
| STOP Mode .....                           | 50 |

---

---

## Introduction

This section describes the operation of the embedded FLASH-1 memory. This memory can be read, programmed, and erased from a single external supply. The program and erase operations are enabled through the use of an internal charge pump.

---

---

## Future FLASH Memory

Design is underway to introduce an improved Flash memory module. The new module will offer improved write erase cycling and faster programming times. However Flash program and erase algorithms will

## FLASH-1 Memory

change, as will the block protection. The new silicon can be identified by an 'A' suffix, i.e. 68HC908AZ60A, and by mask set.

**NOTE:** *In order that current software is compatible and also to prevent problems if code should runaway, Flash program and erase algorithms should not be embedded in software.*

---

---

### Functional Description

The FLASH memory physically consists of two independent arrays of 32K bytes with an additional 52 bytes of user vectors and two bytes of block protection. An erased bit reads as a logic 0 and a programmed bit reads as a logic 1. Program and erase operations are facilitated through control bits in a memory mapped register. Details for these operations appear later in this section. Memory in the FLASH array is organized into pages within rows. There are 8 pages of memory per row with 8 bytes per page. The minimum erase block size is a single row, 64 bytes. Programming is performed on a per page basis; eight bytes at a time. The address ranges for the user memory, control register and vectors are:

- \$8000–\$FDFF
- \$FF80–FF81 (Block Protect Registers)
- \$FE0B FLASH Control Register
- \$FFCC–\$FFFF (These locations are reserved for user-defined interrupt and reset vectors.)

When programming the FLASH, just enough program time must be used to program a page. Too much program time can result in a program disturb condition; in which case an erased bit on the row being programmed becomes unintentionally programmed. Program disturb is avoided by using an iterative program and margin read technique known as the smart programming algorithm. The smart programming algorithm is required whenever programming the FLASH (See [FLASH Program/Margin Read Operation](#) on page 43). As well, to avoid the program disturb issue, each storage page of the row should not be programmed more than once before it is erased. The 8 program cycle maximum per row aligns with the architecture's 8 pages of storage per



row. The margin read step of the smart programming algorithm is used to insure programmed bits are programmed to sufficient margin for data retention over the device lifetime. The following is the row architecture for this array:

- \$8000–\$803F (Row0)
- \$8040–\$807F (Row1)
- \$8080–\$80BF (Row2)
- -----
- \$FFC0–\$FFFF (Row 511)

Programming tools are available from Motorola. Contact your local Motorola representative for more information.

**NOTE:** *A security feature prevents viewing of the FLASH contents.<sup>1</sup>*

## FLASH-1 Control Register

The FLASH-1 control register controls FLASH-1 program, erase, and margin read operations.

Address: \$FE0B

|        | Bit 7 | 6     | 5    | 4    | 3    | 2      | 1     | Bit 0 |
|--------|-------|-------|------|------|------|--------|-------|-------|
| Read:  | FDIV1 | FDIV0 | BLK1 | BLK0 | HVEN | MARGIN | ERASE | PGM   |
| Write: |       |       |      |      |      |        |       |       |
| Reset: | 0     | 0     | 0    | 0    | 0    | 0      | 0     | 0     |

**Figure 1. FLASH-1 Control Register (FLCR1)**

### FDIV1 — Frequency Divide Control Bit

This read/write bit together with FDIV0 selects the factor by which the charge pump clock is divided from the system clock. See [FLASH Charge Pump Frequency Control](#) on page 41.

1. No security feature is absolutely secure. However, Motorola's strategy is to make reading or copying the FLASH difficult for unauthorized users.

### FDIV0 — Frequency Divide Control Bit

This read/write bit together with FDIV1 selects the factor by which the charge pump clock is divided from the system clock. See [FLASH Charge Pump Frequency Control](#) on page 41.

### BLK1— Block Erase Control Bit

This read/write bit together with BLK0 allows erasing of blocks of varying size. See [FLASH Erase Operation](#) on page 41 for a description of available block sizes.

### BLK0 — Block Erase Control Bit

This read/write bit together with BLK1 allows erasing of blocks of varying size. See [FLASH Erase Operation](#) on page 41 for a description of available block sizes.

### HVEN — High-Voltage Enable Bit

This read/write bit enables the charge pump to drive high voltages for program and erase operations in the array. HVEN can only be set if either PGM = 1 or ERASE = 1 and the proper sequence for program/margin read or erase is followed.

1 = High voltage enabled to array and charge pump on

0 = High voltage disabled to array and charge pump off

### MARGIN — Margin Read Control Bit

This read/write bit configures the memory for margin read operation. MARGIN cannot be set if the HVEN = 1. MARGIN will automatically return to unset (0) if asserted when HVEN = 1.

1 = Margin read operation selected

0 = Margin read operation unselected

### ERASE — Erase Control Bit

This read/write bit configures the memory for erase operation. ERASE is interlocked with the PGM bit such that both bits cannot be set at the same time.

1 = Erase operation selected

0 = Erase operation unselected

### PGM — Program Control Bit

This read/write bit configures the memory for program operation. PGM is interlocked with the ERASE bit such that both bits cannot be set at the same time.

- 1 = Program operation selected
- 0 = Program operation unselected

## FLASH Charge Pump Frequency Control

The internal charge pump, required for program, margin read, and erase operations, is designed to operate most efficiently with a 2MHz clock. The charge pump clock is derived from the bus clock. [Table 1](#) shows how the FDIV bits are used to select a charge pump frequency based on the bus clock frequency. Program, margin read and erase operations cannot be performed if the bus clock frequency is below 2 MHz.

**Table 1. Charge Pump Clock Frequency**

| FDIV1 | FDIV0 | Pump Clock Frequency | Bus Clock Frequency |
|-------|-------|----------------------|---------------------|
| 0     | 0     | Bus Frequency ÷ 1    | 2 MHz ± 10%         |
| 0     | 1     | Bus Frequency ÷ 2    | 4 MHz ± 10%         |
| 1     | 0     | Bus Frequency ÷ 2    | 4 MHz ± 10%         |
| 1     | 1     | Bus Frequency ÷ 4    | 8 MHz ± 10%         |

**NOTE:** *FDIV0 and FDIV1 must be set to the same value in both flash arrays.*

## FLASH Erase Operation

[Memory Characteristics](#) on page 445 has a detailed description of the times used in this algorithm. Use the following procedure to erase a block of FLASH memory:

1. Set the ERASE bit, the BLK0, BLK1, FDIV0, and FDIV1 bits in the FLASH-1 control register. See [Table 2](#) for block sizes. See [Table 1](#) for FDIV settings.

## FLASH-1 Memory

2. Insure target portion of array is unprotected, read the block protect register: address \$FF80. See [FLASH Block Protection](#) on page 46 and [FLASH-1 Block Protect Register](#) on page 47 for more information.
3. Write to any FLASH address with any data within the block address range desired.
4. Set the HVEN bit.
5. Wait for a time,  $t_{ERASE}$ .
6. Clear the HVEN bit.
7. Wait for a time,  $t_{KILL}$ , for the high voltages to dissipate.
8. Clear the ERASE bit.
9. After time  $t_{HVD}$ , the memory can be accessed in read mode again.

**NOTE:** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps.*

**Table 2** shows the various block sizes which can be erased in one erase operation.

**Table 2. Erase Block Sizes**

| BLK1 | BLK0 | Block Size, Addresses Cared      |
|------|------|----------------------------------|
| 0    | 0    | Full Array: 24 Kbytes            |
| 0    | 1    | One-Half Array: 16 Kbytes (A14 ) |
| 1    | 0    | Eight Rows: 512 Bytes (A14–A9)   |
| 1    | 1    | Single Row: 64 Bytes (A14–A6)    |

In step 2 of the erase operation, the cared addresses are latched and used to determine the location of the block to be erased. For instance, with  $BLK0 = BLK1 = 0$ , writing to any Flash address in the range \$8000 to \$FFFF will enable the full-array erase.

---

---

## FLASH Program/Margin Read Operation

**NOTE:** *After a total of 8 program operations have been applied to a row, the row must be erased before further programming in order to avoid program disturb. An erased byte will read \$00.*

Programming of the FLASH memory is done on a page basis. A page consists of eight consecutive bytes starting from address \$XXX0 or \$XXX8. The purpose of the margin read mode is to ensure that data has been programmed with sufficient margin for long-term data retention. While performing a margin read the operation is the same as for ordinary read mode except that a built-in counter stretches the data access for an additional eight cycles to allow sensing of the lower cell current. Margin read mode imposes a more stringent read condition on the bitcell to insure the bitcell is programmed with enough margin for long-term data retention. During these eight cycles the COP counter continues to run. The user must account for these extra cycles within COP feed loops. A margin read cycle can only follow a page programming operation. To program and margin read the FLASH memory, use the following algorithm. [Memory Characteristics](#) on page 445, has a detailed description of the times used in this algorithm.

1. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
2. Read from the block protect register.
3. Write data to the eight bytes of the page being programmed. This requires eight separate write operations.
4. Set the HVEN bit.
5. Wait for time,  $t_{\text{PROG}}$ .
6. Clear the HVEN bit.
7. Wait for time,  $t_{\text{HVTV}}$ .
8. Set the MARGIN bit.
9. Wait for time,  $t_{\text{VTP}}$ .
10. Clear the PGM bit.
11. Wait for time,  $t_{\text{HVD}}$ .

12. Read back data in margin read mode. This is done in eight separate read operations which are each stretched by eight cycles.
13. Clear the MARGIN bit.

**NOTE:** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. It is highly recommended that the interrupt mask is set during programming. Under very controlled situations it may be possible to omit this step, but it remains the user's responsibility to ensure that any interrupts do not interfere with the Flash being programmed properly.*

This program/margin read sequence is repeated throughout the memory until all data is programmed. The Smart Programming Algorithm is always required when programming any part of the array. This algorithm insures the minimum possible program time and avoids the deleterious program disturb effect. (See [FLASH Erase Operation](#) on page 41).

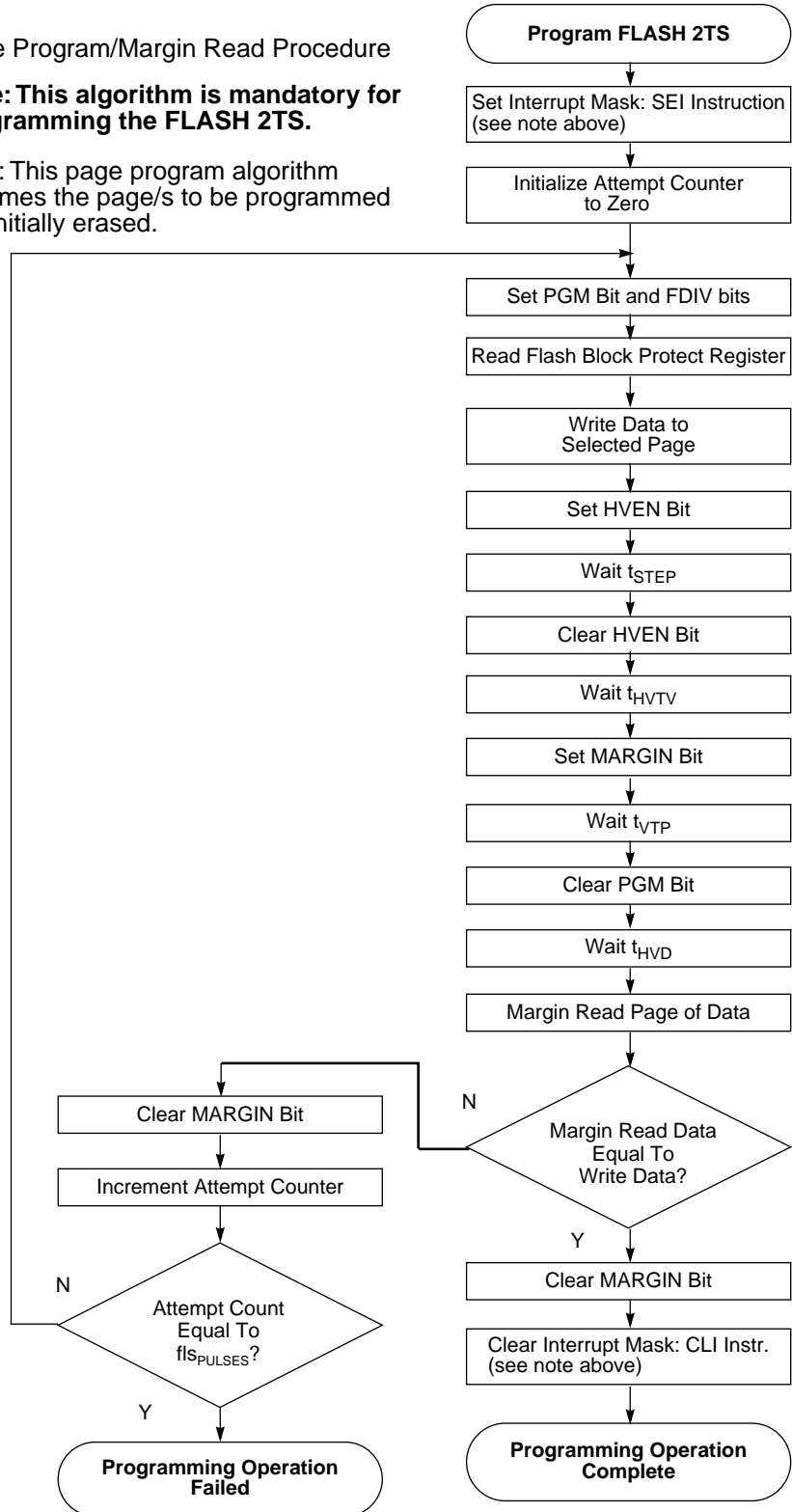
**NOTE:** *In order to ensure proper FLASH read operation after completion of the smart programming algorithm, a series of 500 dummy FLASH reads must be performed of any address before accurate data is read from the FLASH.*

**Smart  
Programming  
Algorithm**

Page Program/Margin Read Procedure

**Note: This algorithm is mandatory for programming the FLASH 2TS.**

Note: This page program algorithm assumes the page/s to be programmed are initially erased.



---

---

### FLASH Block Protection

**NOTE:** *In performing a program or erase operation the FLASH Block Protect Register must be read after setting the PGM or ERASE bit and before asserting the HVEN bit.*

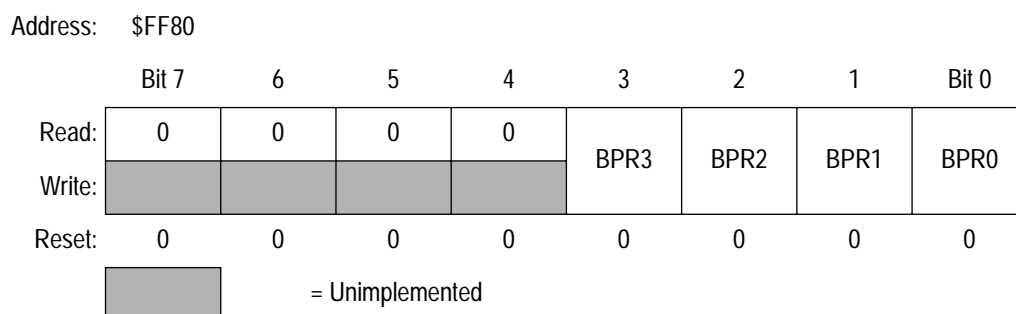
Due to the ability of the on-board charge pump to erase and program the FLASH memory in the target application, provision is made for protecting blocks of memory from unintentional erase or program operations due to system malfunction. This protection is done by reserving a location in the memory for block protect information and requiring that this location be read from to enable setting of the HVEN bit. When the block protect register is read, its contents are latched by the FLASH control logic. If the address range for an erase or program operation includes a protected block, the PGM or ERASE bit is cleared which prevents the HVEN bit in the FLASH control register from being set so that no high voltage is allowed in the array.

When the block protect register is erased (all 0s), the entire memory is accessible for program and erase. When bits within the register are programmed, they lock blocks of memory address ranges as shown in [FLASH-1 Block Protect Register](#) on page 47. The block protect register itself can be erased or programmed only with an external voltage  $V_{HI}$  present on the  $\overline{IRQ}$  pin. The presence of  $V_{HI}$  on the  $\overline{IRQ}$  pin also allows entry in to monitor mode out of reset. Therefore, the ability to change the block protect register is voltage dependent and can occur in either user or monitor modes.



## FLASH-1 Block Protect Register

The block protect register is implemented as a byte within the FLASH-1 memory. Each bit, when programmed, protects a range of addresses in the FLASH-1 array.



**Figure 2. FLASH-1 Block Protect Register (FLBPR1)**

### BPR3 — Block Protect Register Bit 3

This bit protects the memory contents in the address range \$C000 to \$FFFF.

- 1 = Address range protected from erase or program
- 0 = Address range open to erase or program

### BPR2 — Block Protect Register Bit 2

This bit protects the memory contents in the address range \$A000 to \$FFFF.

- 1 = Address range protected from erase or program
- 0 = Address range open to erase or program

### BPR1 — Block Protect Register Bit 1

This bit protects the memory contents in the address range \$9000 to \$FFFF.

- 1 = Address range protected from erase or program
- 0 = Address range open to erase or program

## FLASH-1 Memory

### BPR0 — Block Protect Register Bit 0

This bit protects the memory contents in the address range \$8000 to \$FFFF.

1 = Address range protected from erase or program

0 = Address range open to erase or program

By programming the block protect bits, a portion of the memory will be locked so that no further erase or program operations may be performed. Programming more than one bit at a time is redundant. If both bit 1 and bit 2 are set, for instance, the address range \$9000 through \$FFFF is locked. If all bits are erased, then all of the memory is available for erase and program. The presence of a voltage  $V_{HI}$  on the  $\overline{IRQ}$  pin will bypass the block protection so that all of the memory, including the block protect register, is open for program and erase operations.

---



---

## FLASH-2 Block Protect Register

**NOTE:** This block protect register controls the FLASH-2 array block protection. However, since it is physically located in FLASH-1 array, the FLASH-1 control register must be used to program/erase this register.

The block protect register is implemented as a byte within the FLASH-1 memory. Each bit, when programmed, protects a range of addresses in the FLASH-2 array.

Address: \$FF81

|        | Bit 7 | 6 | 5 | 4 | 3    | 2    | 1    | Bit 0 |
|--------|-------|---|---|---|------|------|------|-------|
| Read:  | 0     | 0 | 0 | 0 | BPR3 | BPR2 | BPR1 | BPR0  |
| Write: |       |   |   |   |      |      |      |       |
| Reset: | 0     | 0 | 0 | 0 | 0    | 0    | 0    | 0     |

= Unimplemented

**Figure 3. FLASH-2 Block Protect Register (FLBPR2)**

### BPR3 — Block Protect Register Bit 3

This bit protects the memory contents in the address range \$4000 to \$7FFF.

- 1 = Address range protected from erase or program
- 0 = Address range open to erase or program

### BPR2 — Block Protect Register Bit 2

This bit protects the memory contents in the address range \$2000 to \$7FFF.

- 1 = Address range protected from erase or program
- 0 = Address range open to erase or program

### BPR1 — Block Protect Register Bit 1

This bit protects the memory contents in the address range \$1000 to \$7FFF.

- 1 = Address range protected from erase or program
- 0 = Address range open to erase or program

### BPR0 — Block Protect Register Bit 0

This bit protects the memory contents in the address range \$0450 to \$7FFF.

- 1 = Address range protected from erase or program
- 0 = Address range open to erase or program

By programming the block protect bits, a portion of the memory will be locked so that no further erase or program operations may be performed. Programming more than one bit at a time is redundant. If both bit 1 and bit 2 are set, for instance, the address range \$1000 through \$FFFF is locked. If all bits are erased, then all of the memory is available for erase and program. The presence of a voltage  $V_{HI}$  on the  $\overline{IRQ}$  pin will bypass the block protection so that all of the memory, including the block protect register, is open for program and erase operations.

---

---

### Low-Power Modes

The WAIT and STOP instructions put the MCU in low power consumption standby modes.

#### WAIT Mode

Putting the MCU into wait mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly, but there will not be any memory activity since the CPU is inactive.

The WAIT instruction should not be executed while performing a program or erase operation on the FLASH. When the MCU is put into wait mode, the charge pump for the FLASH is disabled so that either a program or erase operation will not continue. If the memory is in either program mode (PGM = 1, HVEN = 1) or erase mode (ERASE = 1, HVEN = 1), then it will remain in that mode during wait. Exit from wait must now be done with a reset rather than an interrupt because if exiting wait with an interrupt, the memory will not be in read mode and the interrupt vector cannot be read from the memory.

#### STOP Mode

When the MCU is put into stop mode, if the FLASH is in read mode, it will be put into low power standby.

The STOP instruction should not be executed while performing a program or erase operation on the FLASH. When the MCU is put into stop mode, the charge pump for the FLASH is disabled so that either a program or erase operation will not continue. If the memory is in either program mode (PGM = 1, HVEN = 1) or erase mode (ERASE = 1, HVEN = 1), then it will remain in that mode during stop. Exit from stop must now be done with a reset rather than an interrupt because if exiting stop with an interrupt, the memory will not be in read mode and the interrupt vector cannot be read from the memory.

# FLASH-2 Memory

---

---

## Contents

|   |    |
|---|----|
| Introduction .....                        | 51 |
| Future FLASH Memory .....                 | 51 |
| Functional Description .....              | 52 |
| FLASH Control Register .....              | 53 |
| FLASH Charge Pump Frequency Control ..... | 55 |
| FLASH Erase Operation .....               | 55 |
| FLASH Program/Margin Read Operation ..... | 57 |
| Smart Programming Algorithm .....         | 59 |
| FLASH Block Protection .....              | 60 |
| FLASH Block Protect Register .....        | 60 |
| Low-Power Modes .....                     | 61 |
| WAIT Mode .....                           | 61 |
| STOP Mode .....                           | 61 |

---

---

## Introduction

This section describes the operation of the embedded FLASH-2 memory. This memory can be read, programmed, and erased from a single external supply. The program and erase operations are enabled through the use of an internal charge pump.

---

---

## Future FLASH Memory

Design is underway to introduce an improved Flash memory module. The new module will offer improved write erase cycling and faster programming times. However Flash program and erase algorithms will

## FLASH-2 Memory

change, as will the block protection. The new silicon can be identified by mask set.

**NOTE:** *In order that current software is compatible and also to prevent problems if code should runaway, Flash program and erase algorithms should not be embedded in software.*

---

---

### Functional Description

The FLASH-2 memory is an array of up to 29,488 bytes. An erased bit reads as a logic 0 and a programmed bit reads as a logic 1. Program and erase operations are facilitated through control bits in a memory mapped register. Details for these operations appear later in this section. Memory in the FLASH array is organized into pages within rows. There are 8 pages of memory per row with 8 bytes per page. The minimum erase block size is a single row, 64 bytes. Programming is performed on a per page basis; eight bytes at a time. The address ranges for the user memory and the control register are:

- \$0450–\$04FF
- \$0580–\$05FF
- \$0E00–\$7FFF
- \$FE11 FLASH-2 Control Register

When programming the FLASH, just enough program time must be used to program a page. Too much program time can result in a program disturb condition; in which case an erased bit on the row being programmed becomes unintentionally programmed. Program disturb is avoided by using an iterative program and margin read technique known as the smart programming algorithm. The smart programming algorithm is required whenever programming the FLASH (See [FLASH Program/Margin Read Operation](#) on page 57). As well, to avoid the program disturb issue each storage page of the row should not be programmed more than once before it is erased. The 8 program cycle maximum per row aligns with the architecture's 8 pages of storage per row. The margin read step of the smart programming algorithm is used to insure programmed bits are programmed to sufficient margin for data

retention over the device lifetime. The following is the row architecture for this array:

- \$7F40–\$7F7F (Row 509)
- \$7F80–\$7FBF (Row 510)
- \$7FC0–\$7FFF (Row 511)

Programming tools are available from Motorola. Contact your local Motorola representative for more information.

**NOTE:** A security feature prevents viewing of the FLASH contents.<sup>1</sup>

## FLASH Control Register

The FLASH-2 control register controls FLASH-2 program, erase, and margin read operations.

Address: \$FE11

|        | Bit 7 | 6     | 5    | 4    | 3    | 2      | 1     | Bit 0 |
|--------|-------|-------|------|------|------|--------|-------|-------|
| Read:  |       |       |      |      |      |        |       |       |
| Write: | FDIV1 | FDIV0 | BLK1 | BLK0 | HVEN | MARGIN | ERASE | PGM   |
| Reset: | 0     | 0     | 0    | 0    | 0    | 0      | 0     | 0     |

**Figure 1. FLASH-2 Control Register (FLCR2)**

### FDIV1 — Frequency Divide Control Bit

This read/write bit together with FDIV0 selects the factor by which the charge pump clock is divided from the system clock. See [FLASH Charge Pump Frequency Control](#) on page 55.

### FDIV0 — Frequency Divide Control Bit

This read/write bit together with FDIV1 selects the factor by which the charge pump clock is divided from the system clock. See [FLASH Charge Pump Frequency Control](#) on page 55.

1. No security feature is absolutely secure. However, Motorola's strategy is to make reading or copying the FLASH difficult for unauthorized users.

## FLASH-2 Memory

### BLK1— Block Erase Control Bit

This read/write bit together with BLK0 allows erasing of blocks of varying size. See [FLASH Erase Operation](#) on page 55 for a description of available block sizes.

### BLK0 — Block Erase Control Bit

This read/write bit together with BLK1 allows erasing of blocks of varying size. See [FLASH Erase Operation](#) on page 55 for a description of available block sizes.

### HVEN — High-Voltage Enable Bit

This read/write bit enables the charge pump to drive high voltages for program and erase operations in the array. HVEN can only be set if either PGM = 1 or ERASE = 1 and the proper sequence for program/margin read or erase is followed.

1 = High voltage enabled to array and charge pump on

0 = High voltage disabled to array and charge pump off

### MARGIN — Margin Read Control Bit

This read/write bit configures the memory for margin read operation. MARGIN cannot be set if the HVEN = 1. MARGIN will automatically return to unset (0) if asserted when HVEN = 1.

1 = Margin read operation selected

0 = Margin read operation unselected

### ERASE — Erase Control Bit

This read/write bit configures the memory for erase operation. ERASE is interlocked with the PGM bit such that both bits cannot be set at the same time.

1 = Erase operation selected

0 = Erase operation unselected

### PGM — Program Control Bit

This read/write bit configures the memory for program operation. PGM is interlocked with the ERASE bit such that both bits cannot be set at the same time.

1 = Program operation selected

0 = Program operation unselected



---

## FLASH Charge Pump Frequency Control

The internal charge pump, required for program, margin read, and erase operations, is designed to operate most efficiently with a 2MHz clock. The charge pump clock is derived from the bus clock. [Table 1](#) shows how the FDIV bits are used to select a charge pump frequency based on the bus clock frequency. Program, margin read and erase operations cannot be performed if the bus clock frequency is below 2 MHz.

**Table 1. Charge Pump Clock Frequency**

| FDIV1 | FDIV0 | Pump Clock Frequency   | Bus Clock Frequency |
|-------|-------|------------------------|---------------------|
| 0     | 0     | Bus Frequency $\div$ 1 | 2 MHz $\pm$ 10%     |
| 0     | 1     | Bus Frequency $\div$ 2 | 4 MHz $\pm$ 10%     |
| 1     | 0     | Bus Frequency $\div$ 2 | 4 MHz $\pm$ 10%     |
| 1     | 1     | Bus Frequency $\div$ 4 | 8 MHz $\pm$ 10%     |

**NOTE:** *FDIV0 and FDIV1 must be set to the same value in both flash arrays.*

---

## FLASH Erase Operation

[Memory Characteristics](#) on page 445 has a detailed description of the times used in this algorithm. Use the following procedure to erase a block of FLASH-2 memory:

1. Set the ERASE bit, the BLK0, BLK1, FDIV0, and FDIV1 bits in the FLASH control register. See [Table 2](#) for block sizes. See [Table 1](#) for FDIV settings.
2. Insure target portion of array is unprotected, read the block protect register: address \$FF81. See Section [FLASH Block Protection](#) on page 60 and Section [FLASH Block Protect Register](#) on page 60 for more information.
3. Write to any FLASH address with any data within the block address range desired.
4. Set the HVEN bit.
5. Wait for a time,  $t_{ERASE}$ .

## FLASH-2 Memory

6. Clear the HVEN bit.
7. Wait for a time,  $t_{KILL}$ , for the high voltages to dissipate.
8. Clear the ERASE bit.
9. After time  $t_{HVD}$ , the memory can be accessed in read mode again.

**NOTE:** While these operations must be performed in the order shown, other unrelated operations may occur between the steps.

**Table 2** shows the various block sizes which can be erased in one erase operation.

**Table 2. Erase Block Sizes**

| BLK1 | BLK0 | Block Size, Addresses Cared      |
|------|------|----------------------------------|
| 0    | 0    | Full Array: 24 Kbytes            |
| 0    | 1    | One-Half Array: 16 Kbytes (A14 ) |
| 1    | 0    | Eight Rows: 512 Bytes (A14–A9)   |
| 1    | 1    | Single Row: 64 Bytes (A14–A6)    |

In step 2 of the erase operation, the cared addresses are latched and used to determine the location of the block to be erased. For instance, with  $BLK0 = BLK1 = 0$ , writing to any Flash address in the range \$0450-\$05FF or \$0E00-\$7FFF will enable the full-array erase.

---

---

## FLASH Program/Margin Read Operation

**NOTE:** *After a total of 8 program operations have been applied to a row, the row must be erased before further programming in order to avoid program disturb. An erased byte will read \$00.*

Programming of the FLASH memory is done on a page basis. A page consists of eight consecutive bytes starting from address \$XXX0 or \$XXX8. The purpose of the margin read mode is to ensure that data has been programmed with sufficient margin for long-term data retention. While performing a margin read the operation is the same as for ordinary read mode except that a built-in counter stretches the data access for an additional eight cycles to allow sensing of the lower cell current. Margin read mode imposes a more stringent read condition on the bitcell to insure the bitcell is programmed with enough margin for long-term data retention. During these eight cycles the COP counter continues to run. The user must account for these extra cycles within COP feed loops. A margin read cycle can only follow a page programming operation. To program and margin read the FLASH memory, use the following algorithm. [Memory Characteristics](#) on page 445 has a detailed description of the times used in this algorithm.

1. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
2. Read from the block protect register.
3. Write data to the eight bytes of the page being programmed. This requires eight separate write operations.
4. Set the HVEN bit.
5. Wait for time,  $t_{\text{PROG}}$ .
6. Clear the HVEN bit.
7. Wait for time,  $t_{\text{HVTV}}$ .
8. Set the MARGIN bit.
9. Wait for time,  $t_{\text{VTP}}$ .
10. Clear the PGM bit.
11. Wait for time,  $t_{\text{HVD}}$ .

12. Read back data in margin read mode. This is done in eight separate read operations which are each stretched by eight cycles.
13. Clear the MARGIN bit.

**NOTE:** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. It is highly recommended that the interrupt mask is set during programming. Under very controlled situations it may be possible to omit this step, but it remains the user's responsibility to ensure that any interrupts do not interfere with the Flash being programmed properly.*

This program/margin read sequence is repeated throughout the memory until all data is programmed. The Smart Programming Algorithm is always required when programming any part of the array. This algorithm insures the minimum possible program time and avoids the deleterious program disturb effect. (See [FLASH Erase Operation](#) on page 55).

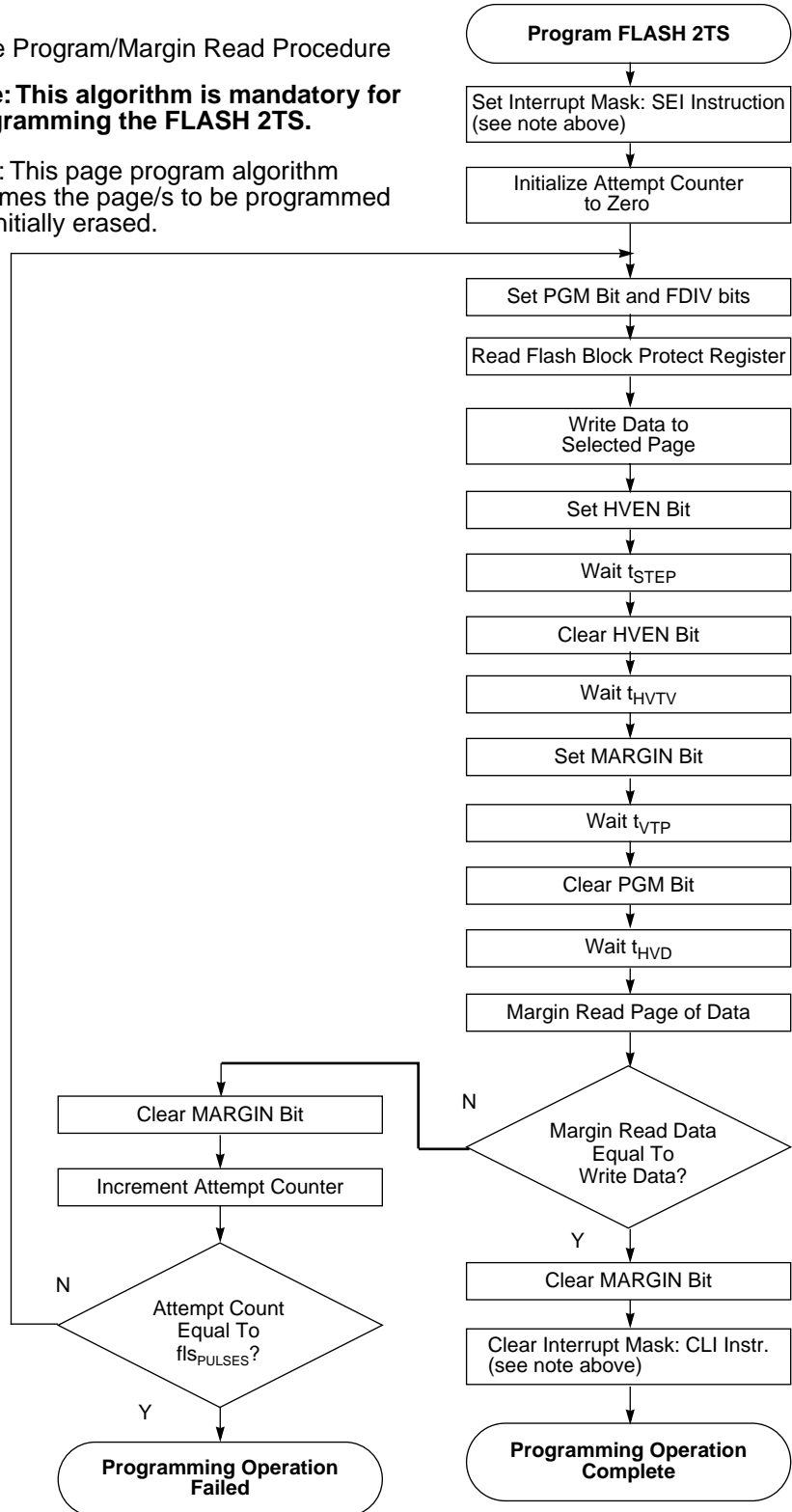
**NOTE:** *In order to ensure proper FLASH read operation after completion of the smart programming algorithm, a series of 500 dummy FLASH reads must be performed of any address before accurate data is read from the FLASH.*

**Smart  
Programming  
Algorithm**

Page Program/Margin Read Procedure

**Note: This algorithm is mandatory for programming the FLASH 2TS.**

Note: This page program algorithm assumes the page/s to be programmed are initially erased.



---

---

### FLASH Block Protection

**NOTE:** *In performing a program or erase operation the FLASH Block Protect Register must be read after setting the PGM or ERASE bit and before asserting the HVEN bit.*

Due to the ability of the on-board charge pump to erase and program the FLASH memory in the target application, provision is made for protecting blocks of memory from unintentional erase or program operations due to system malfunction. This protection is done by reserving a location in the memory for block protect information and requiring that this location be read before setting the HVEN bit. When the block protect register is read, its contents are latched by the FLASH control logic. If the address range for an erase or program operation includes a protected block, the PGM or ERASE bit is cleared which prevents the HVEN bit in the FLASH control register from being set so that no high voltage is allowed in the array.

When the block protect register is erased (all 0s), the entire memory is accessible for program and erase. When bits within the register are programmed, they lock blocks of memory address ranges as shown in [FLASH Block Protect Register](#) on page 60. The block protect register itself can be erased or programmed only with an external voltage  $V_{HI}$  present on the  $\overline{IRQ}$  pin. The presence of  $V_{HI}$  on the  $\overline{IRQ}$  pin also allows entry in to monitor mode out of reset. Therefore, the ability to change the block protect register is voltage dependent and can occur in either user or monitor modes.

---

---

### FLASH Block Protect Register

The block protect register for FLASH-2 is physically implemented as a byte within the FLASH-1 memory. Please refer to the FLASH-1 memory section, [FLASH-2 Block Protect Register](#) on page 48 for definition of this register. Each bit, when programmed, protects a range of addresses in the FLASH-2 array.

---

---

## Low-Power Modes

The WAIT and STOP instructions put the MCU in low power consumption standby modes.

### WAIT Mode

Putting the MCU into wait mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly, but there will not be any memory activity since the CPU is inactive.

The WAIT instruction should not be executed while performing a program or erase operation on the FLASH. When the MCU is put into wait mode, the charge pump for the FLASH is disabled so that either a program or erase operation will not continue. If the memory is in either program mode (PGM = 1, HVEN = 1) or erase mode (ERASE = 1, HVEN = 1), then it will remain in that mode during wait. Exit from wait must now be done with a reset rather than an interrupt because if exiting wait with an interrupt, the memory will not be in read mode and the interrupt vector cannot be read from the memory.

### STOP Mode

When the MCU is put into stop mode, if the FLASH is in read mode, it will be put into low power standby.

The STOP instruction should not be executed while performing a program or erase operation on the FLASH. When the MCU is put into stop mode, the charge pump for the FLASH is disabled so that either a program or erase operation will not continue. If the memory is in either program mode (PGM = 1, HVEN = 1) or erase mode (ERASE = 1, HVEN = 1), then it will remain in that mode during wait. Exit from stop must now be done with a reset rather than an interrupt because if exiting stop with an interrupt, the memory will not be in read mode and the interrupt vector cannot be read from the memory.

## FLASH-2 Memory



---

---

## Contents

|  |    |
|--|----|
| Introduction . . . . .                   | 63 |
| Future EEPROM Memory . . . . .           | 64 |
| Features . . . . .                       | 64 |
| Functional Description. . . . .          | 65 |
| EEPROM Programming . . . . .             | 65 |
| EEPROM Erasing . . . . .                 | 67 |
| EEPROM Block Protection . . . . .        | 68 |
| MCU Configuration . . . . .              | 69 |
| MC68HC908AZ60 EEPROM Protection. . . . . | 69 |
| EEPROM Control Register . . . . .        | 70 |
| EEPROM Nonvolatile Register . . . . .    | 72 |
| Low-Power Modes . . . . .                | 74 |
| Wait Mode. . . . .                       | 74 |
| Stop Mode. . . . .                       | 74 |

---

---

## Introduction

This section describes the electrically erasable programmable read-only memory (EEPROM). The 1024 bytes available on the MC68HC908AZ60 are physically located in two 512byte arrays. This chapter details the array covering the address range \$0800 to \$09FF. For information relating to the array covering address range \$0600 to \$07FF (see [EEPROM-2](#) on page 75).

---

---

### Future EEPROM Memory

Design is underway to introduce an improved EEPROM module, which will simplify programming and erase. Current read, write and erase algorithms are fully compatible with the new EEPROM design. The new EEPROM module requires a constant timebase through the set up of new timebase control registers. If more information is required for code compatibility please contact the factory. The silicon differences will be identified by mask set. Please read [Appendix A: Future EEPROM Registers](#) for preliminary details.

**NOTE:** *This new silicon will not allow multiple writes before erase. EEPROM bytes must be erased before reprogramming.*

---

---

### Features

EEPROM features include:

- Byte, Block, or Bulk Erasable
- Nonvolatile Block Protection Option
- Nonvolatile MCU Configuration Bits
- On-Chip Charge Pump for Programming/Erasing
- Security Option

---

---

## Functional Description

The 512 bytes of EEPROM-1 can be programmed or erased without an external voltage supply. The EEPROM has a lifetime of 10,000 write-erase cycles. EEPROM cells are protected with a nonvolatile block protection option. These options are stored in the EEPROM nonvolatile register (EENVR1) and are loaded into the EEPROM array configuration register after reset (EEACR1) or a read of EENVR1. Hardware interlocks are provided to protect stored data corruption from accidental programming/erasing.

The EEPROM-1 array will leave the factory in the erased state. All addresses will be logic 1 and bit 4 of the EENVR1 register will be programmed to 1 such that the full array can be available and unprotected.

### EEPROM Programming

The unprogrammed state is a logic 1. Programming changes the state to a logic 0. Only valid EEPROM bytes in the non-protected blocks and EENVR1 can be programmed. **It is recommended that all bits should be erased before being programmed.**

Follow this procedure to program a byte of EEPROM after first ensuring the block protect feature is not set on the address block of the byte to be programmed:

1. Clear EERAS1 and EERAS0 and set EELAT in the EECR1. (See note A and B.)
2. Write the desired data to any user EEPROM address.
3. Set the EEPGM bit. (See note C.)
4. Wait for a time,  $t_{\text{EEPGM}}$ , to program the byte.
5. Clear EEPGM bit.
6. Wait for a time,  $t_{\text{EEFPV}}$ , for the programming voltage to fall.
7. Clear EELAT bits. (See note D.)
8. Repeat steps 1 to 7 for more EEPROM programming.

### NOTES:

- a. EERAS1 and EERAS0 must be cleared for programming. Otherwise, the part will be in erase mode.
- b. Setting EELAT bit configures the address and data buses to latch data for programming the array. Only data with valid EEPROM address will be latched. If another consecutive valid EEPROM write occurs, this address and data will override the previous address and data. Any attempts to read other EEPROM data will read the latched data. If EELAT is set, other writes to the EECR1 will only be allowed after a valid EEPROM write.
- c. To ensure proper programming sequence, the EEPGM bit cannot be set if the EELAT bit is cleared and a non-EEPROM write has occurred. When EEPGM is set, the onboard charge pump generates the program voltage and applies it to the user EEPROM array. When the EEPGM bit is cleared, the program voltage is removed from the array and the internal charge pump is turned off.
- d. Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM array.

## EEPROM Erasing

The unprogrammed state is a logic 1. Only the valid EEPROM bytes in the nonprotected blocks and EENVR1 can be erased.

Use this procedure to erase EEPROM after first ensuring the block protect feature is not set on the address block of the byte to be erased:

1. Clear/set EERAS1 and EERAS0 to select byte/block/bulk erase, and set EELAT in EECR1. (See note A.)
2. Write any data to the desired address for byte erase, to any address in the desired block for block erase, or to any array address for bulk erase.
3. Set the EEPGM bit. (See note B.)
4. Wait for a time,  $t_{EEPGM}$ ,  $t_{EEBLOCK}$ ,  $t_{EEBULK}$ .
5. Clear EEPGM bit.
6. Wait for a time,  $t_{EEFPV}$ , for the erasing voltage to fall.
7. Clear EELAT bits. (See note C.)
8. Repeat steps 1 to 7 for more EEPROM byte/block erasing.

EEBPx bit must be cleared to erase EEPROM data in the corresponding block. If any EEBPx is set, the corresponding block can not be erased and bulk erase mode does not apply.

### NOTES:

- a. Setting EELAT bit configures the address and data buses to latch data for erasing the array. Only valid EEPROM addresses with their data will be latched. If another consecutive valid EEPROM write occurs, this address and data will override the previous address and data. In block erase mode, any EEPROM address in the block may be used in step 2. All locations within this block will be erased. In bulk erase mode, any EEPROM address may be used to erase the whole EEPROM. EENVR1 is not affected with block or bulk erase. Any attempts to read other EEPROM data will read the latched data. If EELAT is set, other writes to the EECR1 will only be allowed after a valid EEPROM write.
- b. The EEPGM bit cannot be set if the EELAT bit is cleared and a non-EEPROM write has occurred. This is to ensure proper erasing sequence. Once EEPGM is set, the type of erase

mode cannot be modified. If EEPGM is set, the onboard charge pump generates the erase voltage and applies it to the user EEPROM array. When the EEPGM bit is cleared, the erase voltage is removed from the array and the internal charge pump is turned off.

- c. Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM array.

All bits should be erased before being programmed.

## EEPROM Block Protection

The 512 bytes of EEPROM are divided into four 128-byte blocks. Each of these blocks can be separately protected by EEBPx bit. Any attempt to program or erase memory locations within the protected block will not allow the program/erase voltage to be applied to the array. [Table 1](#) shows the address ranges within the blocks.

**Table 1. EEPROM Array Address Blocks**

| Block Number (EEBPx) | Address Range |
|----------------------|---------------|
| EEBP0                | \$0800–\$087F |
| EEBP1                | \$0880–\$08FF |
| EEBP2                | \$0900–\$097F |
| EEBP3                | \$0980–\$09FF |

If EEBPx bit is set, that corresponding address block is protected. These bits are effective after a reset or a read to EENVR1 register. The block protect configuration can be modified by erasing/programming the corresponding bits in the EENVR1 register and then reading the EENVR1 register.

**MCU  
Configuration**

The EEPROM nonvolatile register (EENVR1) also contains general-purpose bits which can be used to enable/disable functions within the MCU which, for safety reasons, need to be controlled from nonvolatile memory. On reset, this special register loads the MCU configuration into the volatile EEPROM array configuration register (EEACR1). Thereafter, all reads to the EENVR1 will reload EEACR1.

The MCU configuration can be changed by programming/erasing the EENVR1 like a normal EEPROM byte. **Please note that it is the user's responsibility to erase and program the EENVR1 register to the correct system requirements and verify it prior to use.** The new array configuration will take affect after a system reset or a read of the EENVR1.

**MC68HC908AZ60  
EEPROM  
Protection**

The MC68HC908AZ60 has a special protection option which prevents program/erase access to memory locations \$08F0 to \$08FF. This protect function is enabled by programming the EEPRTCT bit in the EENVR to 0.

In addition to the disabling of the program and erase operations on memory locations \$08F0 to \$08FF the enabling of the protect option has the following effects.

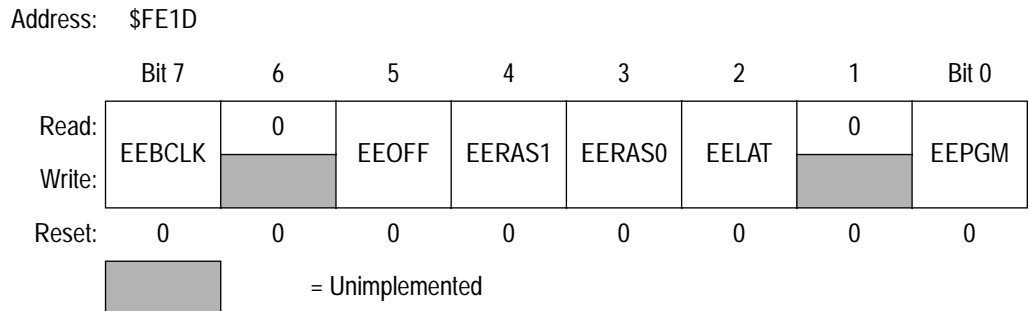
- Bulk and block erase modes are disabled.
- Programming and erasing of the EENVR is disabled.
- Unsecure locations (\$0800–\$08EF) can be erased using the single byte erase function as normal.
- Secured locations can be read as normal.
- Writing to a secure location no longer qualifies as a “valid EEPROM write” as detailed in (see EEPROM Programming) Note B and (see EEPROM Erasing) Note A.

**NOTE:** *Once armed, the protect option is permanently enabled. As a consequence, all functions in the EENVR will remain in the state they were in immediately before the security was enabled.*

## EEPROM-1

### EEPROM Control Register

This read/write register controls programming/erasing of the array.



**Figure 1. EEPROM-1 Control Register (EECR1)**

#### EEBCLK — EEPROM Bus Clock Enable

This read/write bit determines which clock will be used to drive the internal charge pump for programming/erasing. Reset clears this bit.

1 = Bus clock drives charge pump

0 = Internal RC oscillator drives charge pump

**NOTE:** *It is recommended that the internal RC oscillator is used to drive the internal charge pump for applications that have a bus frequency of less than 8 MHz.*

#### EEOFF — EEPROM Power Down

This read/write bit disables the EEPROM module for lower power consumption. Any attempts to access the array will give unpredictable results. Reset clears this bit.

1 = Disable EEPROM array

0 = Enable EEPROM array

**NOTE:** *The EEPROM requires a recovery time,  $t_{EEOFF}$ , to stabilize after clearing the EEOFF bit.*



### EERAS1 and EERAS0 — Erase Bits

These read/write bits set the erase modes. Reset clears these bits.

**Table 2. EEPROM Program/Erase Mode Select**

| EEDPx | EERAS1 | EERAS0 | MODE             |
|-------|--------|--------|------------------|
| 0     | 0      | 0      | Byte Program     |
| 0     | 0      | 1      | Byte Erase       |
| 0     | 1      | 0      | Block Erase      |
| 0     | 1      | 1      | Bulk Erase       |
| 1     | X      | X      | No Erase/Program |

X = don't care

### EELAT — EEPROM Latch Control

This read/write bit latches the address and data buses for programming the EEPROM array. EELAT cannot be cleared if EEPGM is still set. Reset clears this bit.

- 1 = Buses configured for EEPROM programming
- 0 = Buses configured for normal read operation

### EEPGM — EEPROM Program/Erase Enable

This read/write bit enables the internal charge pump and applies the programming/erasing voltage to the EEPROM array if the EELAT bit is set and a write to a valid EEPROM location has occurred. Reset clears the EEPGM bit.

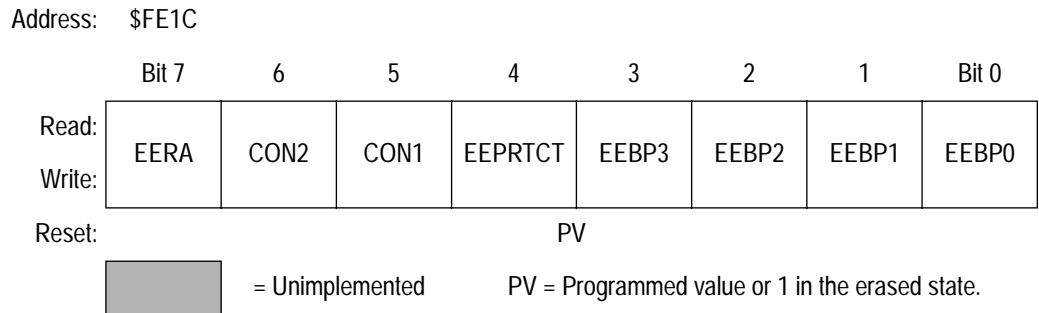
- 1 = EEPROM programming/erasing power switched on
- 0 = EEPROM programming/erasing power switched off

**NOTE:** *Writing logic 0s to both the EELAT and EEPGM bits with a single instruction will clear EEPGM only to allow time for the removal of high voltage.*

## EEPROM-1

### EEPROM Nonvolatile Register

The EEPROM nonvolatile register (EENVR1) is shown in [Figure 2](#).



**Figure 2. EEPROM-1 Nonvolatile Register (EENVR1)**

EERA — EEPROM Redundant Array

This bit is reserved for future use and should always be equal to 0.

CONx — MCU Configuration Bits

These read/write bits can be used to enable/disable functions within the MCU. Reset loads CONx from EENVR1 to EEACR1.

CON2 — Unused

CON1 — Unused

**NOTE:** *This feature is a write-once feature. Once the protection is enabled it may not be disabled.*

EEPRTCT — EEPROM Protection

This one-time programmable bit can be used to protect 16 bytes (\$8F0–\$8FF) from being erased or programmed.

1 = EEPROM protection disabled

0 = EEPROM protection enabled

EEBP3–EEBP0 — EEPROM Block Protection Bits

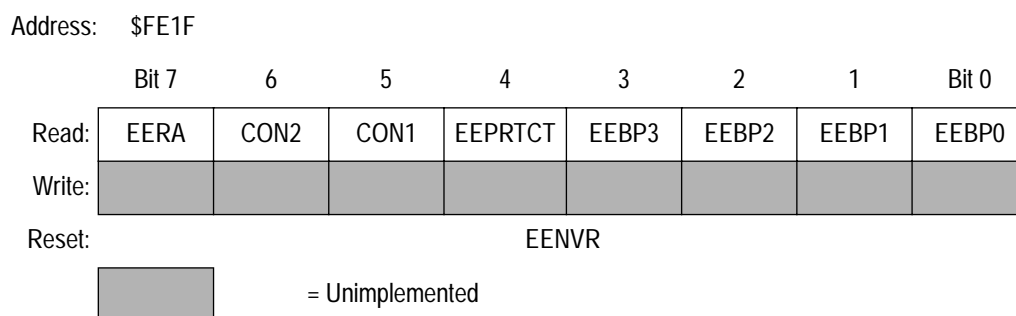
These read/write bits select blocks of EEPROM array from being programmed or erased. Reset loads EEBP[3:0] from EENVR1 to EEACR1.

1 = EEPROM array block is protected

0 = EEPROM array block is unprotected

## EEPROM Array Configuration Register

The EEPROM array configuration register (EEACR1) is shown in [Figure 3](#).



**Figure 3. EEPROM-1 Array Control Register (EEACR1)**

**EERA** — EEPROM Redundant Array

This bit is reserved for future use and should always be equal to 0.

**CONx** — MCU Configuration Bits

These read/write bits can be used to enable/disable functions within the MCU. Reset loads CONx from EENVR1 to EEACR1.

**CON2** — Unused

**CON1** — Unused

**NOTE:** *This feature is a write-once feature. Once the protection is enabled it may not be disabled.*

**EEPRTCT** — EEPROM Protection

This one-time programmable bit can be used to protect 16 bytes (\$8F0–\$8FF) from being erased or programmed.

1 = EEPROM protection disabled

0 = EEPROM protection enabled

**EEBP3–EEBP0** — EEPROM Block Protection Bits

These read/write bits select blocks of EEPROM array from being programmed or erased. Reset loads EEBP[3:0] from EENVR1 to EEACR1.

1 = EEPROM array block is protected

0 = EEPROM array block is unprotected

---

---

## Low-Power Modes

The WAIT and STOP instructions can put the MCU in low power-consumption standby modes.

### Wait Mode

The WAIT instruction does not affect the EEPROM. It is possible to program the EEPROM and put the MCU in wait mode. However, if the EEPROM is inactive, power can be reduced by setting the EEOFF bit before executing the WAIT instruction.

### Stop Mode

The STOP instruction reduces the EEPROM power consumption to a minimum. The STOP instruction should not be executed while the high voltage is turned on (EEPGM = 1).

If stop mode is entered while program/erase is in progress, high voltage will be automatically turned off. However, the EEPGM bit will remain set. When stop mode is terminated, and if EEPGM is still set, the high voltage will be automatically turned back on. Program/erase time will need to be extended if program/erase is interrupted by entering stop mode.

The module requires a recovery time,  $t_{EESTOP}$ , to stabilize after leaving stop mode. Attempts to access the array during the recovery time will result in unpredictable behavior.

---

---

## Contents

|  |    |
|--|----|
| Introduction . . . . .                   | 75 |
| Future EEPROM Memory . . . . .           | 76 |
| Features . . . . .                       | 76 |
| Functional Description. . . . .          | 77 |
| EEPROM Programming . . . . .             | 77 |
| EEPROM Erasing . . . . .                 | 79 |
| EEPROM Block Protection . . . . .        | 80 |
| MCU Configuration . . . . .              | 81 |
| MC68HC908AZ60 EEPROM Protection. . . . . | 81 |
| EEPROM Control Register . . . . .        | 82 |
| EEPROM Nonvolatile Register . . . . .    | 84 |
| Low-Power Modes . . . . .                | 86 |
| Wait Mode. . . . .                       | 86 |
| Stop Mode. . . . .                       | 86 |

---

---

## Introduction

This section describes the electrically erasable programmable read-only memory (EEPROM). The 1024 bytes available on the MC68HC908AZ60 are physically located in two 512byte arrays. This chapter details the array covering the address range \$0600 to \$07FF. For information relating to the array covering address range \$0800 to \$09FF (see [EEPROM-1](#) on page 63).

---

---

### Future EEPROM Memory

Design is underway to introduce an improved EEPROM module, which will simplify programming and erase. Current read, write and erase algorithms are fully compatible with the new EEPROM design. The new EEPROM module requires a constant timebase through the set up of new timebase control registers. If more information is required for code compatibility please contact the factory. The silicon differences will be identified by mask set. Please read [Appendix A: Future EEPROM Registers](#) for preliminary details.

**NOTE:** *This new silicon will not allow multiple writes before erase. EEPROM bytes must be erased before reprogramming.*

---

---

### Features

EEPROM features include:

- Byte, Block, or Bulk Erasable
- Nonvolatile Block Protection Option
- Nonvolatile MCU Configuration Bits
- On-Chip Charge Pump for Programming/Erasing
- Security Option

---

---

## Functional Description

The 512 bytes of EEPROM-2 can be programmed or erased without an external voltage supply. The EEPROM has a lifetime of 10,000 write-erase cycles. EEPROM cells are protected with a nonvolatile block protection option. These options are stored in the EEPROM nonvolatile register (EENVR2) and are loaded into the EEPROM array configuration register after reset (EEACR2) or a read of EENVR2. Hardware interlocks are provided to protect stored data corruption from accidental programming/erasing.

The EEPROM-2 array will leave the factory in the erased state. All addresses will be logic 1 and bit 4 of the EENVR2 register will be programmed to 1 such that the full array can be available and unprotected.

### EEPROM Programming

The unprogrammed state is a logic 1. Programming changes the state to a logic 0. Only valid EEPROM bytes in the non-protected blocks and EENVR2 can be programmed. **It is recommended that all bits should be erased before being programmed.**

Follow this procedure to program a byte of EEPROM after first ensuring the block protect feature is not set on the address block of the byte to be programmed:

1. Clear EERAS1 and EERAS0 and set EELAT in the EECR2. (See note A and B.)
2. Write the desired data to any user EEPROM address.
3. Set the EEPGM bit. (See note C.)
4. Wait for a time,  $t_{\text{EEPGM}}$ , to program the byte.
5. Clear EEPGM bit.
6. Wait for a time,  $t_{\text{EEFPV}}$ , for the programming voltage to fall.
7. Clear EELAT bits. (See note D.)
8. Repeat steps 1 to 7 for more EEPROM programming.

### NOTES:

- a. EERAS1 and EERAS0 must be cleared for programming. Otherwise, the part will be in erase mode.
- b. Setting EELAT bit configures the address and data buses to latch data for programming the array. Only data with valid EEPROM address will be latched. If another consecutive valid EEPROM write occurs, this address and data will override the previous address and data. Any attempts to read other EEPROM data will read the latched data. If EELAT is set, other writes to the EECR2 will only be allowed after a valid EEPROM write.
- c. To ensure proper programming sequence, the EEPGM bit cannot be set if the EELAT bit is cleared and a non-EEPROM write has occurred. When EEPGM is set, the onboard charge pump generates the program voltage and applies it to the user EEPROM array. When the EEPGM bit is cleared, the program voltage is removed from the array and the internal charge pump is turned off.
- d. Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM array.



## EEPROM Erasing

The unprogrammed state is a logic 1. Only the valid EEPROM bytes in the nonprotected blocks and EENVR2 can be erased.

Use this procedure to erase EEPROM after first ensuring the block protect feature is not set on the address block of the byte to be erased:

1. Clear/set EERAS1 and EERAS0 to select byte/block/bulk erase, and set EELAT in EECR2. (See note A.)
2. Write any data to the desired address for byte erase, to any address in the desired block for block erase, or to any array address for bulk erase.
3. Set the EEPGM bit. (See note B.)
4. Wait for a time,  $t_{EEPGM}$ ,  $t_{EEBLOCK}$ ,  $t_{EEBULK}$ .
5. Clear EEPGM bit.
6. Wait for a time,  $t_{EEFPV}$ , for the erasing voltage to fall.
7. Clear EELAT bits. (See note C.)
8. Repeat steps 1 to 7 for more EEPROM byte/block erasing.

EEBPx bit must be cleared to erase EEPROM data in the corresponding block. If any EEBPx is set, the corresponding block can not be erased and bulk erase mode does not apply.

### NOTES:

- a. Setting EELAT bit configures the address and data buses to latch data for erasing the array. Only valid EEPROM addresses with their data will be latched. If another consecutive valid EEPROM write occurs, this address and data will override the previous address and data. In block erase mode, any EEPROM address in the block may be used in step 2. All locations within this block will be erased. In bulk erase mode, any EEPROM address may be used to erase the whole EEPROM. EENVR2 is not affected with block or bulk erase. Any attempts to read other EEPROM data will read the latched data. If EELAT is set, other writes to the EECR2 will only be allowed after a valid EEPROM write.
- b. The EEPGM bit cannot be set if the EELAT bit is cleared and a non-EEPROM write has occurred. This is to ensure proper erasing sequence. Once EEPGM is set, the type of erase

mode cannot be modified. If EEPGM is set, the onboard charge pump generates the erase voltage and applies it to the user EEPROM array. When the EEPGM bit is cleared, the erase voltage is removed from the array and the internal charge pump is turned off.

- c. Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM array.

All bits should be erased before being programmed.

### EEPROM Block Protection

The 512 bytes of EEPROM are divided into four 128-byte blocks. Each of these blocks can be separately protected by EEBPx bit. Any attempt to program or erase memory locations within the protected block will not allow the program/erase voltage to be applied to the array. [Table 1](#) shows the address ranges within the blocks.

**Table 1. EEPROM Array Address Blocks**

| Block Number (EEBPx) | Address Range |
|----------------------|---------------|
| EEBP0                | \$0600–\$067F |
| EEBP1                | \$0680–\$06FF |
| EEBP2                | \$0700–\$077F |
| EEBP3                | \$0780–\$07FF |

If EEBPx bit is set, that corresponding address block is protected. These bits are effective after a reset or a read to EENVR2 register. The block protect configuration can be modified by erasing/programming the corresponding bits in the EENVR2 register and then reading the EENVR2 register.

**MCU  
Configuration**

The EEPROM nonvolatile register (EENVR2) also contains general-purpose bits which can be used to enable/disable functions within the MCU which, for safety reasons, need to be controlled from nonvolatile memory. On reset, this special register loads the MCU configuration into the volatile EEPROM array configuration register (EEACR2). Thereafter, all reads to the EENVR2 will reload EEACR2.

The MCU configuration can be changed by programming/erasing the EENVR2 like a normal EEPROM byte. **Please note that it is the user's responsibility to erase and program the EENVR2 register to the correct system requirements and verify it prior to use.** The new array configuration will take affect after a system reset or a read of the EENVR2.

**MC68HC908AZ60  
EEPROM  
Protection**

The MC68HC908AZ60 has a special protection option which prevents program/erase access to memory locations \$08F0 to \$08FF. This protect function is enabled by programming the EEPRTCT bit in the EENVR to 0.

In addition to the disabling of the program and erase operations on memory locations \$08F0 to \$08FF the enabling of the protect option has the following effects.

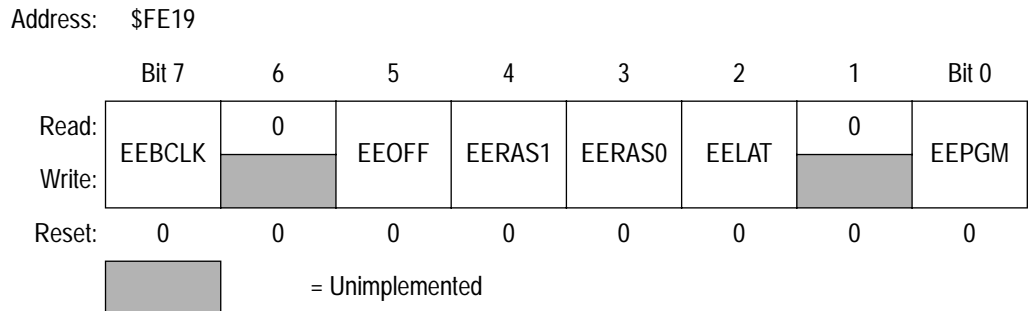
- Bulk and block erase modes are disabled.
- Programming and erasing of the EENVR is disabled.
- Unsecure locations (\$0800–\$08EF) can be erased using the single byte erase function as normal.
- Secured locations can be read as normal.
- Writing to a secure location no longer qualifies as a “valid EEPROM write” as detailed in (see EEPROM Programming) Note B and (see EEPROM Erasing) Note A.

**NOTE:** *Once armed, the protect option is permanently enabled. As a consequence, all functions in the EENVR will remain in the state they were in immediately before the security was enabled.*

## EEPROM-2

### EEPROM Control Register

This read/write register controls programming/erasing of the array.



**Figure 1. EEPROM-2 Control Register (EECR2)**

#### EEBCLK — EEPROM Bus Clock Enable

This read/write bit determines which clock will be used to drive the internal charge pump for programming/erasing. Reset clears this bit.

1 = Bus clock drives charge pump

0 = Internal RC oscillator drives charge pump

**NOTE:** *It is recommended that the internal RC oscillator is used to drive the internal charge pump for applications that have a bus frequency of less than 8 MHz.*

#### EEOFF — EEPROM Power Down

This read/write bit disables the EEPROM module for lower power consumption. Any attempts to access the array will give unpredictable results. Reset clears this bit.

1 = Disable EEPROM array

0 = Enable EEPROM array

**NOTE:** *The EEPROM requires a recovery time,  $t_{EEOFF}$ , to stabilize after clearing the EEOFF bit.*

### EERAS1 and EERAS0 — Erase Bits

These read/write bits set the erase modes. Reset clears these bits.

**Table 2. EEPROM Program/Erase Mode Select**

| EEDPx | EERAS1 | EERAS0 | MODE             |
|-------|--------|--------|------------------|
| 0     | 0      | 0      | Byte Program     |
| 0     | 0      | 1      | Byte Erase       |
| 0     | 1      | 0      | Block Erase      |
| 0     | 1      | 1      | Bulk Erase       |
| 1     | X      | X      | No Erase/Program |

X = don't care

### EELAT — EEPROM Latch Control

This read/write bit latches the address and data buses for programming the EEPROM array. EELAT cannot be cleared if EEPGM is still set. Reset clears this bit.

- 1 = Buses configured for EEPROM programming
- 0 = Buses configured for normal read operation

### EEPGM — EEPROM Program/Erase Enable

This read/write bit enables the internal charge pump and applies the programming/erasing voltage to the EEPROM array if the EELAT bit is set and a write to a valid EEPROM location has occurred. Reset clears the EEPGM bit.

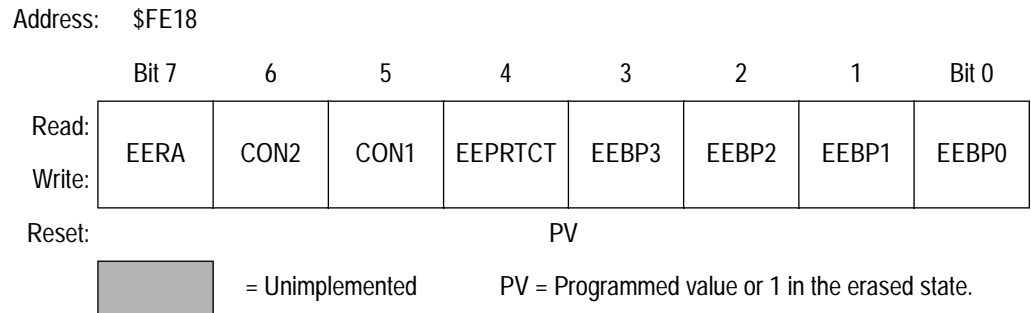
- 1 = EEPROM programming/erasing power switched on
- 0 = EEPROM programming/erasing power switched off

**NOTE:** *Writing logic 0s to both the EELAT and EEPGM bits with a single instruction will clear EEPGM only to allow time for the removal of high voltage.*

## EEPROM-2

### EEPROM Nonvolatile Register

The EEPROM nonvolatile register (EENVR2) is shown in [Figure 2](#).



**Figure 2. EEPROM-2 Nonvolatile Register (EENVR2)**

**EERA** — EEPROM Redundant Array

This bit is reserved for future use and should always be equal to 0.

**CONx** — MCU Configuration Bits

These read/write bits can be used to enable/disable functions within the MCU. Reset loads CONx from EENVR2 to EEACR2.

**CON2** — Unused

**CON1** — Unused

**NOTE:** *This feature is a write-once feature. Once the protection is enabled it may not be disabled.*

**EEPRTCT** — EEPROM Protection

This one-time programmable bit can be used to protect 16 bytes (\$8F0–\$8FF) from being erased or programmed.

1 = EEPROM protection disabled

0 = EEPROM protection enabled

**EEBP3–EEBP0** — EEPROM Block Protection Bits

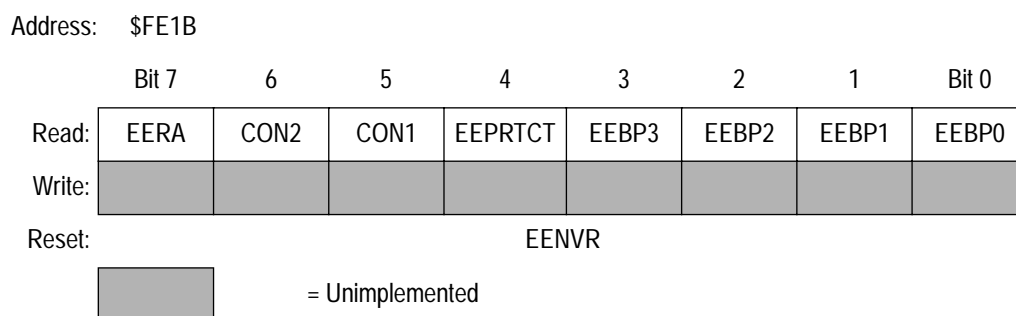
These read/write bits select blocks of EEPROM array from being programmed or erased. Reset loads EEBP[3:0] from EENVR2 to EEACR2.

1 = EEPROM array block is protected

0 = EEPROM array block is unprotected

## EEPROM Array Configuration Register

The EEPROM array configuration register (EEACR2) is shown in [Figure 3](#).



**Figure 3. EEPROM-2 Array Control Register (EEACR2)**

**EERA** — EEPROM Redundant Array

This bit is reserved for future use and should always be equal to 0.

**CONx** — MCU Configuration Bits

These read/write bits can be used to enable/disable functions within the MCU. Reset loads CONx from EENVR2 to EEACR2.

**CON2** — Unused

**CON1** — Unused

**NOTE:** *This feature is a write-once feature. Once the protection is enabled it may not be disabled.*

**EEPRTCT** — EEPROM Protection

This one-time programmable bit can be used to protect 16 bytes (\$8F0–\$8FF) from being erased or programmed.

1 = EEPROM protection disabled

0 = EEPROM protection enabled

**EEBP3–EEBP0** — EEPROM Block Protection Bits

These read/write bits select blocks of EEPROM array from being programmed or erased. Reset loads EEBP[3:0] from EENVR2 to EEACR2.

1 = EEPROM array block is protected

0 = EEPROM array block is unprotected

---

---

## Low-Power Modes

The WAIT and STOP instructions can put the MCU in low power-consumption standby modes.

### Wait Mode

The WAIT instruction does not affect the EEPROM. It is possible to program the EEPROM and put the MCU in wait mode. However, if the EEPROM is inactive, power can be reduced by setting the EEOFF bit before executing the WAIT instruction.

### Stop Mode

The STOP instruction reduces the EEPROM power consumption to a minimum. The STOP instruction should not be executed while the high voltage is turned on (EEPGM = 1).

If stop mode is entered while program/erase is in progress, high voltage will be automatically turned off. However, the EEPGM bit will remain set. When stop mode is terminated, and if EEPGM is still set, the high voltage will be automatically turned back on. Program/erase time will need to be extended if program/erase is interrupted by entering stop mode.

The module requires a recovery time,  $t_{EESTOP}$ , to stabilize after leaving stop mode. Attempts to access the array during the recovery time will result in unpredictable behavior.



# Central Processor Unit (CPU)

---

---

## Contents

|   |     |
|---|-----|
| Introduction . . . . .                  | 87  |
| Features . . . . .                      | 88  |
| CPU registers . . . . .                 | 89  |
| Accumulator (A) . . . . .               | 89  |
| Index register (H:X) . . . . .          | 90  |
| Stack pointer (SP) . . . . .            | 90  |
| Program counter (PC) . . . . .          | 91  |
| Condition code register (CCR) . . . . . | 91  |
| Arithmetic/logic unit (ALU) . . . . .   | 94  |
| Low-power modes . . . . .               | 94  |
| WAIT mode . . . . .                     | 94  |
| STOP mode . . . . .                     | 94  |
| CPU during break interrupts . . . . .   | 95  |
| Instruction Set Summary . . . . .       | 96  |
| Opcode Map . . . . .                    | 103 |

---

---

## Introduction

This section describes the central processor unit (CPU8). The M68HC08 CPU is an enhanced and fully object-code-compatible version of the M68HC05 CPU. The *CPU08 Reference Manual* (Motorola document number CPU08RM/AD) contains a description of the CPU instruction set, addressing modes, and architecture.

## Central Processor Unit (CPU)

---

---

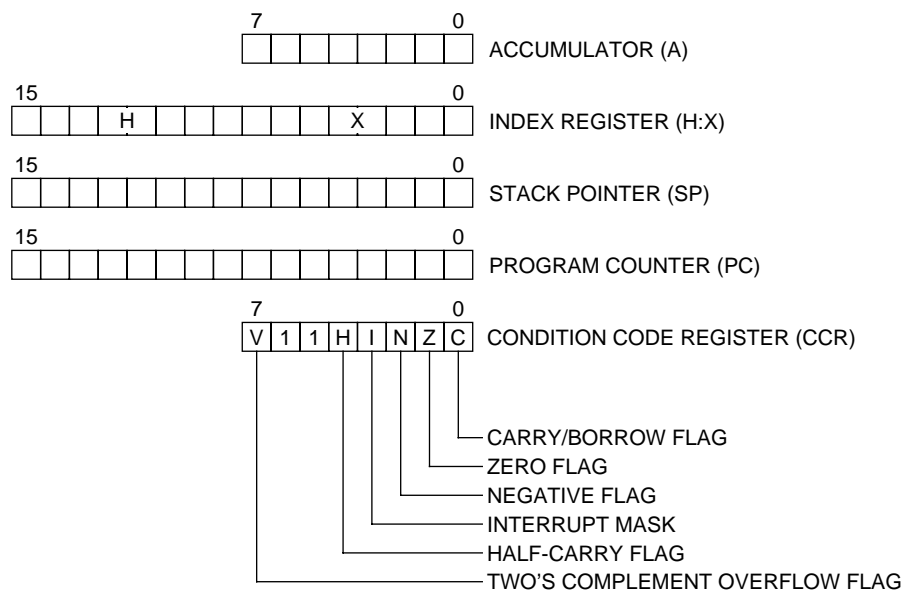
### Features

Features of the CPU include the following:

- Full upward, object-code compatibility with M68HC05 family
- 16-bit stack pointer with stack manipulation instructions
- 16-bit index register with X-register manipulation instructions
- 8.4MHz CPU internal bus frequency
- 64K byte program/data memory space
- 16 addressing modes
- Memory-to-memory data moves without using accumulator
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- Enhanced binary-coded decimal (BCD) data handling
- Low-power STOP and WAIT Modes

## CPU registers

**Figure 1** shows the five CPU registers. CPU registers are not part of the memory map.



**Figure 1. CPU registers**

### Accumulator (A)

The accumulator is a general-purpose 8-bit register. The CPU uses the accumulator to hold operands and the results of arithmetic/logic operations.



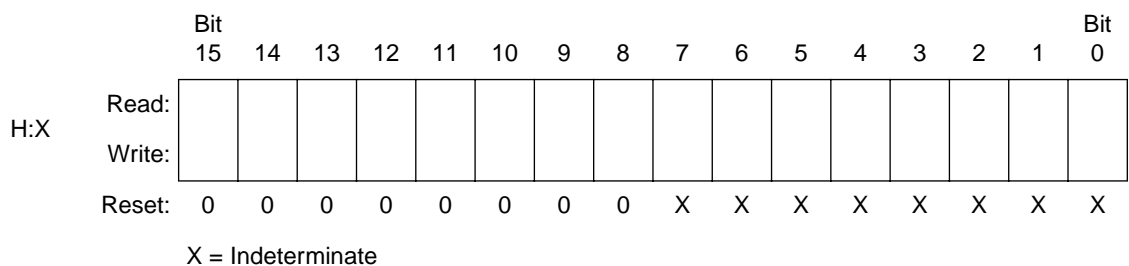
**Figure 2. Accumulator (A)**

## Central Processor Unit (CPU)

### Index register (H:X)

The 16-bit index register allows indexed addressing of a 64K byte memory space. H is the upper byte of the index register and X is the lower byte. H:X is the concatenated 16-bit index register.

In the indexed addressing modes, the CPU uses the contents of the index register to determine the conditional address of the operand.



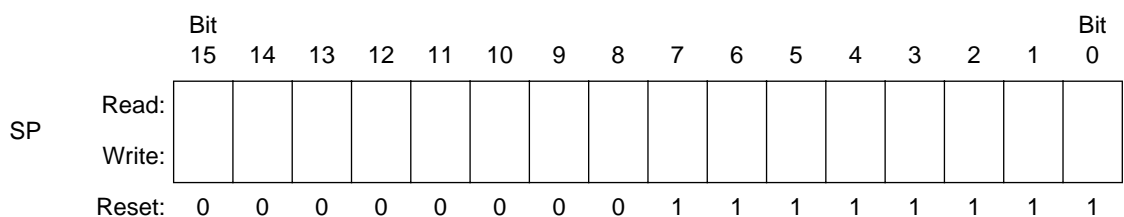
**Figure 3. Index register (H:X)**

The index register can also be used as a temporary data storage location.

### Stack pointer (SP)

The stack pointer is a 16-bit register that contains the address of the next location on the stack. During a reset, the stack pointer is preset to \$00FF. The reset stack pointer (RSP) instruction sets the least significant byte to \$FF and does not affect the most significant byte. The stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

In the stack pointer 8-bit offset and 16-bit offset addressing modes, the stack pointer can function as an index register to access data on the stack. The CPU uses the contents of the stack pointer to determine the conditional address of the operand.



**Figure 4. Stack pointer (SP)**

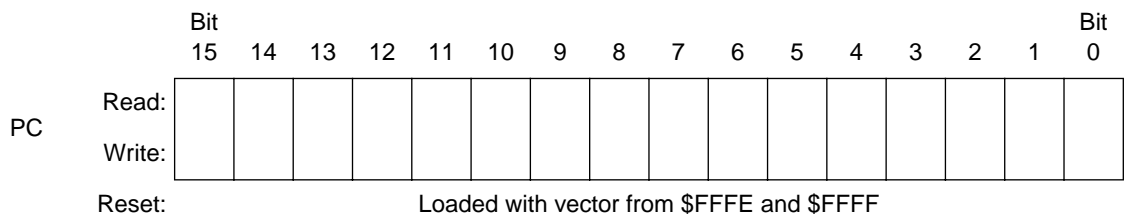
**NOTE:** *The location of the stack is arbitrary and may be relocated anywhere in RAM. Moving the SP out of page zero (\$0000 to \$00FF) frees direct address (page zero) space. For correct operation, the stack pointer must point only to RAM locations.*

**Program counter (PC)**

The program counter is a 16-bit register that contains the address of the next instruction or operand to be fetched.

Normally, the program counter automatically increments to the next sequential memory location every time an instruction or operand is fetched. Jump, branch, and interrupt operations load the program counter with an address other than that of the next sequential location.

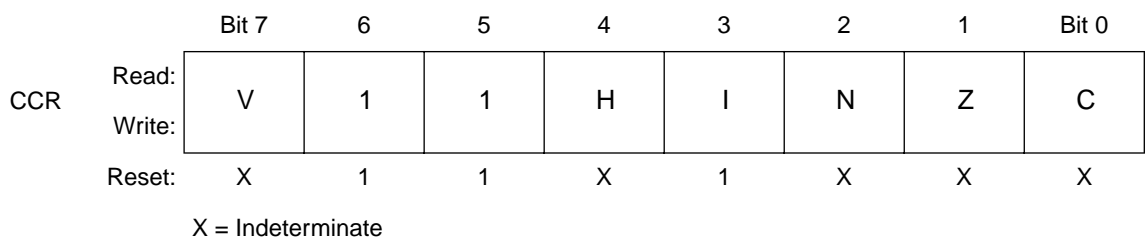
During reset, the program counter is loaded with the reset vector address located at \$FFFE and \$FFFF. The vector address is the address of the first instruction to be executed after exiting the reset state.



**Figure 5. Program counter (PC)**

**Condition code register (CCR)**

The 8-bit condition code register contains the interrupt mask and five flags that indicate the results of the instruction just executed. Bits 6 and 5 are set permanently to '1'. The following paragraphs describe the functions of the condition code register.



**Figure 6. Condition code register (CCR)**

## Central Processor Unit (CPU)

### V — Overflow flag

The CPU sets the overflow flag when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow flag.

- 1 = Overflow
- 0 = No overflow

### H — Half-carry flag

The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an ADD or ADC operation. The half-carry flag is required for binary-coded decimal (BCD) arithmetic operations. The DAA instruction uses the states of the H and C flags to determine the appropriate correction factor.

- 1 = Carry between bits 3 and 4
- 0 = No carry between bits 3 and 4

### I — Interrupt mask

When the interrupt mask is set, all maskable CPU interrupts are disabled. CPU interrupts are enabled when the interrupt mask is cleared. When a CPU interrupt occurs, the interrupt mask is set automatically after the CPU registers are saved on the stack, but before the interrupt vector is fetched.

- 1 = Interrupts disabled
- 0 = Interrupts enabled

**NOTE:** *To maintain M6805 compatibility, the upper byte of the index register (H) is not stacked automatically. If the interrupt service routine modifies H, then the user must stack and unstack H using the PSHH and PULH instructions.*

After the I bit is cleared, the highest-priority interrupt request is serviced first.

A return from interrupt (RTI) instruction pulls the CPU registers from the stack and restores the interrupt mask from the stack. After any reset, the interrupt mask is set and can only be cleared by the clear interrupt mask software instruction (CLI).

**N** — Negative flag

The CPU sets the negative flag when an arithmetic operation, logic operation, or data manipulation produces a negative result, setting bit 7 of the result.

- 1 = Negative result
- 0 = Non-negative result

**Z** — Zero flag

The CPU sets the zero flag when an arithmetic operation, logic operation, or data manipulation produces a result of \$00.

- 1 = Zero result
- 0 = Non-zero result

**C** — Carry/borrow flag

The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some instructions - such as bit test and branch, shift, and rotate - also clear or set the carry/borrow flag.

- 1 = Carry out of bit 7
- 0 = No carry out of bit 7

## Central Processor Unit (CPU)

---

---

### Arithmetic/logic unit (ALU)

The ALU performs the arithmetic and logic operations defined by the instruction set.

Refer to the *CPU08 Reference Manual* (Motorola document number CPU08RM/AD) for a description of the instructions and addressing modes and more detail about CPU architecture.

---

---

### Low-power modes

The WAIT and STOP instructions put the MCU in low-power consumption standby modes.

#### WAIT mode

The WAIT instruction:

- clears the interrupt mask (I bit) in the condition code register, enabling interrupts. After exit from WAIT mode by interrupt, the I bit remains clear. After exit by reset, the I bit is set.
- Disables the CPU clock

#### STOP mode

The STOP instruction:

- clears the interrupt mask (I bit) in the condition code register, enabling external interrupts. After exit from STOP mode by external interrupt, the I bit remains clear. After exit by reset, the I bit is set.
- Disables the CPU clock

After exiting STOP mode, the CPU clock begins running after the oscillator stabilization delay.



## CPU during break interrupts

If the break module is enabled, a break interrupt causes the CPU to execute the software interrupt instruction (SWI) at the completion of the current CPU instruction. See [Break Module](#) on page 161. The program counter vectors to \$FFFC–\$FFFD (\$FEFC–\$FEFD in monitor mode).

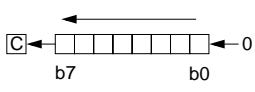
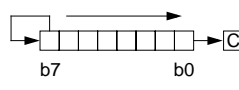
A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation if the break interrupt has been deasserted.

# Central Processor Unit (CPU)

## Instruction Set Summary

Table 1 provides a summary of the M68HC08 instruction set.

Table 1. Instruction Set Summary

| Source Form  | Operation                              | Description   | Effect on CCR |   |   |   |   |   | Address Mode  | Opcode   | Operand   | Cycles                               |
|--|--|---|---------------|---|---|---|---|---|---|--|---|--------------------------------------|
|  |  |   | V             | H | I | N | Z | C |   |  |   |                                      |
| ADC #opr<br>ADC opr<br>ADC opr<br>ADC opr,X<br>ADC opr,X<br>ADC ,X<br>ADC opr,SP<br>ADC opr,SP | Add with Carry                         | $A \leftarrow (A) + (M) + (C)$  | ↑             | ↑ | — | ↑ | ↑ | ↑ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A9<br>B9<br>C9<br>D9<br>E9<br>F9<br>9EE9<br>9ED9 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| ADD #opr<br>ADD opr<br>ADD opr<br>ADD opr,X<br>ADD opr,X<br>ADD ,X<br>ADD opr,SP<br>ADD opr,SP | Add without Carry                      | $A \leftarrow (A) + (M)$  | ↑             | ↑ | — | ↑ | ↑ | ↑ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | AB<br>BB<br>CB<br>DB<br>EB<br>FB<br>9EEB<br>9EDB | ii<br>dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| AIS #opr   | Add Immediate Value (Signed) to SP     | $SP \leftarrow (SP) + (16 \ll M)$   | —             | — | — | — | — | — | IMM   | A7   | ii  | 2                                    |
| AIX #opr   | Add Immediate Value (Signed) to H:X    | $H:X \leftarrow (H:X) + (16 \ll M)$   | —             | — | — | — | — | — | IMM   | AF   | ii  | 2                                    |
| AND #opr<br>AND opr<br>AND opr<br>AND opr,X<br>AND opr,X<br>AND ,X<br>AND opr,SP<br>AND opr,SP | Logical AND                            | $A \leftarrow (A) \& (M)$   | 0             | — | — | ↑ | ↑ | — | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A4<br>B4<br>C4<br>D4<br>E4<br>F4<br>9EE4<br>9ED4 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| ASL opr<br>ASLA<br>ASLX<br>ASL opr,X<br>ASL ,X<br>ASL opr,SP                                   | Arithmetic Shift Left<br>(Same as LSL) |  | ↑             | — | — | ↑ | ↑ | ↑ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1               | 38<br>48<br>58<br>68<br>78<br>9E68               | dd<br>ff<br>ff                                  | 4<br>1<br>1<br>4<br>3<br>5           |
| ASR opr<br>ASRA<br>ASRX<br>ASR opr,X<br>ASR opr,X<br>ASR opr,SP                                | Arithmetic Shift Right                 |  | ↑             | — | — | ↑ | ↑ | ↑ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1               | 37<br>47<br>57<br>67<br>77<br>9E67               | dd<br>ff<br>ff                                  | 4<br>1<br>1<br>4<br>3<br>5           |
| BCC rel  | Branch if Carry Bit Clear              | $PC \leftarrow (PC) + 2 + rel \text{ ? } (C) = 0$                                   | —             | — | — | — | — | — | REL   | 24   | rr  | 3                                    |

**Table 1. Instruction Set Summary (Continued)**

| Source Form  | Operation  | Description   | Effect on CCR |   |   |   |   |   | Address Mode   | Opcode   | Operand   | Cycles                               |
|--|--|---|---------------|---|---|---|---|---|--|--|---|--------------------------------------|
|  |  |   | V             | H | I | N | Z | C |  |  |   |                                      |
| BCLR <i>n, opr</i>   | Clear Bit <i>n</i> in <i>M</i>                       | $M_n \leftarrow 0$                                      | -             | - | - | - | - | - | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 11<br>13<br>15<br>17<br>19<br>1B<br>1D<br>1F     | dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd          | 4<br>4<br>4<br>4<br>4<br>4<br>4<br>4 |
| BCS <i>rel</i>   | Branch if Carry Bit Set (Same as BLO)                | $PC \leftarrow (PC) + 2 + rel ? (C) = 1$                | -             | - | - | - | - | - | REL  | 25   | rr  | 3                                    |
| BEQ <i>rel</i>   | Branch if Equal                                      | $PC \leftarrow (PC) + 2 + rel ? (Z) = 1$                | -             | - | - | - | - | - | REL  | 27   | rr  | 3                                    |
| BGE <i>opr</i>   | Branch if Greater Than or Equal To (Signed Operands) | $PC \leftarrow (PC) + 2 + rel ? (N \oplus V) = 0$       | -             | - | - | - | - | - | REL  | 90   | rr  | 3                                    |
| BGT <i>opr</i>   | Branch if Greater Than (Signed Operands)             | $PC \leftarrow (PC) + 2 + rel ? (Z)   (N \oplus V) = 0$ | -             | - | - | - | - | - | REL  | 92   | rr  | 3                                    |
| BHCC <i>rel</i>  | Branch if Half Carry Bit Clear                       | $PC \leftarrow (PC) + 2 + rel ? (H) = 0$                | -             | - | - | - | - | - | REL  | 28   | rr  | 3                                    |
| BHCS <i>rel</i>  | Branch if Half Carry Bit Set                         | $PC \leftarrow (PC) + 2 + rel ? (H) = 1$                | -             | - | - | - | - | - | REL  | 29   | rr  | 3                                    |
| BHI <i>rel</i>   | Branch if Higher                                     | $PC \leftarrow (PC) + 2 + rel ? (C)   (Z) = 0$          | -             | - | - | - | - | - | REL  | 22   | rr  | 3                                    |
| BHS <i>rel</i>   | Branch if Higher or Same (Same as BCC)               | $PC \leftarrow (PC) + 2 + rel ? (C) = 0$                | -             | - | - | - | - | - | REL  | 24   | rr  | 3                                    |
| BIH <i>rel</i>   | Branch if $\overline{IRQ}$ Pin High                  | $PC \leftarrow (PC) + 2 + rel ? \overline{IRQ} = 1$     | -             | - | - | - | - | - | REL  | 2F   | rr  | 3                                    |
| BIL <i>rel</i>   | Branch if $\overline{IRQ}$ Pin Low                   | $PC \leftarrow (PC) + 2 + rel ? \overline{IRQ} = 0$     | -             | - | - | - | - | - | REL  | 2E   | rr  | 3                                    |
| BIT # <i>opr</i><br>BIT <i>opr</i><br>BIT <i>opr</i><br>BIT <i>opr,X</i><br>BIT <i>opr,X</i><br>BIT , <i>X</i><br>BIT <i>opr,SP</i><br>BIT <i>opr,SP</i> | Bit Test   | (A) & (M)   | 0             | - | - | ↑ | ↑ | - | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2  | A5<br>B5<br>C5<br>D5<br>E5<br>F5<br>9EE5<br>9ED5 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| BLE <i>opr</i>   | Branch if Less Than or Equal To (Signed Operands)    | $PC \leftarrow (PC) + 2 + rel ? (Z)   (N \oplus V) = 1$ | -             | - | - | - | - | - | REL  | 93   | rr  | 3                                    |
| BLO <i>rel</i>   | Branch if Lower (Same as BCS)                        | $PC \leftarrow (PC) + 2 + rel ? (C) = 1$                | -             | - | - | - | - | - | REL  | 25   | rr  | 3                                    |
| BLS <i>rel</i>   | Branch if Lower or Same                              | $PC \leftarrow (PC) + 2 + rel ? (C)   (Z) = 1$          | -             | - | - | - | - | - | REL  | 23   | rr  | 3                                    |
| BLT <i>opr</i>   | Branch if Less Than (Signed Operands)                | $PC \leftarrow (PC) + 2 + rel ? (N \oplus V) = 1$       | -             | - | - | - | - | - | REL  | 91   | rr  | 3                                    |
| BMC <i>rel</i>   | Branch if Interrupt Mask Clear                       | $PC \leftarrow (PC) + 2 + rel ? (I) = 0$                | -             | - | - | - | - | - | REL  | 2C   | rr  | 3                                    |
| BMI <i>rel</i>   | Branch if Minus                                      | $PC \leftarrow (PC) + 2 + rel ? (N) = 1$                | -             | - | - | - | - | - | REL  | 2B   | rr  | 3                                    |
| BMS <i>rel</i>   | Branch if Interrupt Mask Set                         | $PC \leftarrow (PC) + 2 + rel ? (I) = 1$                | -             | - | - | - | - | - | REL  | 2D   | rr  | 3                                    |

# Central Processor Unit (CPU)

Table 1. Instruction Set Summary (Continued)

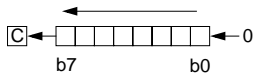
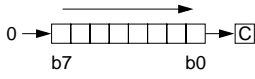
| Source Form   | Operation                         | Description  | Effect on CCR |   |   |   |   |   | Address Mode   | Opcode                                       | Operand  | Cycles                               |
|---|-----------------------------------|--|---------------|---|---|---|---|---|--|--|--|--------------------------------------|
|   |                                   |  | V             | H | I | N | Z | C |  |  |  |                                      |
| BNE <i>rel</i>  | Branch if Not Equal               | $PC \leftarrow (PC) + 2 + rel ? (Z) = 0$   | -             | - | - | - | - | - | REL  | 26   | rr   | 3                                    |
| BPL <i>rel</i>  | Branch if Plus                    | $PC \leftarrow (PC) + 2 + rel ? (N) = 0$   | -             | - | - | - | - | - | REL  | 2A   | rr   | 3                                    |
| BRA <i>rel</i>  | Branch Always                     | $PC \leftarrow (PC) + 2 + rel$   | -             | - | - | - | - | - | REL  | 20   | rr   | 3                                    |
| BRCLR <i>n,opr,rel</i>  | Branch if Bit <i>n</i> in M Clear | $PC \leftarrow (PC) + 3 + rel ? (Mn) = 0$  | -             | - | - | - | - | ↑ | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 01<br>03<br>05<br>07<br>09<br>0B<br>0D<br>0F | dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BRN <i>rel</i>  | Branch Never                      | $PC \leftarrow (PC) + 2$   | -             | - | - | - | - | - | REL  | 21   | rr   | 3                                    |
| BRSET <i>n,opr,rel</i>  | Branch if Bit <i>n</i> in M Set   | $PC \leftarrow (PC) + 3 + rel ? (Mn) = 1$  | -             | - | - | - | - | ↑ | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 00<br>02<br>04<br>06<br>08<br>0A<br>0C<br>0E | dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BSET <i>n,opr</i>   | Set Bit <i>n</i> in M             | $Mn \leftarrow 1$  | -             | - | - | - | - | - | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 10<br>12<br>14<br>16<br>18<br>1A<br>1C<br>1E | dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd                         | 4<br>4<br>4<br>4<br>4<br>4<br>4<br>4 |
| BSR <i>rel</i>  | Branch to Subroutine              | $PC \leftarrow (PC) + 2$ ; push (PCL)<br>$SP \leftarrow (SP) - 1$ ; push (PCH)<br>$SP \leftarrow (SP) - 1$<br>$PC \leftarrow (PC) + rel$   | -             | - | - | - | - | - | REL  | AD   | rr   | 4                                    |
| CBEQ <i>opr,rel</i><br>CBEQA # <i>opr,rel</i><br>CBEQX # <i>opr,rel</i><br>CBEQ <i>opr,X+,rel</i><br>CBEQ <i>X+,rel</i><br>CBEQ <i>opr,SP,rel</i> | Compare and Branch if Equal       | $PC \leftarrow (PC) + 3 + rel ? (A) - (M) = \$00$<br>$PC \leftarrow (PC) + 3 + rel ? (A) - (M) = \$00$<br>$PC \leftarrow (PC) + 3 + rel ? (X) - (M) = \$00$<br>$PC \leftarrow (PC) + 3 + rel ? (A) - (M) = \$00$<br>$PC \leftarrow (PC) + 2 + rel ? (A) - (M) = \$00$<br>$PC \leftarrow (PC) + 4 + rel ? (A) - (M) = \$00$ | -             | - | - | - | - | - | DIR<br>IMM<br>IX1+<br>IX+<br>SP1   | 31<br>41<br>51<br>61<br>71<br>9E61           | dd rr<br>ii rr<br>ii rr<br>ff rr<br>rr<br>ff rr                      | 5<br>4<br>4<br>5<br>4<br>6           |
| CLC   | Clear Carry Bit                   | $C \leftarrow 0$   | -             | - | - | - | - | 0 | INH  | 98   |  | 1                                    |
| CLI   | Clear Interrupt Mask              | $I \leftarrow 0$   | -             | - | 0 | - | - | - | INH  | 9A   |  | 2                                    |
| CLR <i>opr</i><br>CLRA<br>CLR X<br>CLR H<br>CLR <i>opr,X</i><br>CLR , <i>X</i><br>CLR <i>opr,SP</i>   | Clear                             | $M \leftarrow \$00$<br>$A \leftarrow \$00$<br>$X \leftarrow \$00$<br>$H \leftarrow \$00$<br>$M \leftarrow \$00$<br>$M \leftarrow \$00$<br>$M \leftarrow \$00$  | 0             | - | - | 0 | 1 | - | DIR<br>INH<br>INH<br>INH<br>IX1<br>IX<br>SP1   | 3F<br>4F<br>5F<br>8C<br>6F<br>7F<br>9E6F     | dd<br>ff<br>ff<br>ff   | 3<br>1<br>1<br>1<br>3<br>2<br>4      |

Table 1. Instruction Set Summary (Continued)

| Source Form  | Operation                        | Description  | Effect on CCR |   |   |   |   |   | Address Mode  | Opcode   | Operand   | Cycles                               |
|--|----------------------------------|--|---------------|---|---|---|---|---|---|--|---|--------------------------------------|
|  |                                  |  | V             | H | I | N | Z | C |   |  |   |                                      |
| CMP #opr<br>CMP opr<br>CMP opr<br>CMP opr,X<br>CMP opr,X<br>CMP ,X<br>CMP opr,SP<br>CMP opr,SP | Compare A with M                 | (A) – (M)  | ↑             | – | – | ↑ | ↑ | ↑ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A1<br>B1<br>C1<br>D1<br>E1<br>F1<br>9EE1<br>9ED1 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| COM opr<br>COMA<br>COMX<br>COM opr,X<br>COM ,X<br>COM opr,SP                                   | Complement (One's Complement)    | $M \leftarrow (\overline{M}) = \$FF - (M)$<br>$A \leftarrow (\overline{A}) = \$FF - (M)$<br>$X \leftarrow (\overline{X}) = \$FF - (M)$<br>$M \leftarrow (\overline{M}) = \$FF - (M)$<br>$M \leftarrow (\overline{M}) = \$FF - (M)$<br>$M \leftarrow (\overline{M}) = \$FF - (M)$   | 0             | – | – | ↑ | ↑ | 1 | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1               | 33<br>43<br>53<br>63<br>73<br>9E63               | dd<br>ff<br>ff                                  | 4<br>1<br>1<br>4<br>3<br>5           |
| CPHX #opr<br>CPHX opr  | Compare H:X with M               | (H:X) – (M:M + 1)  | ↑             | – | – | ↑ | ↑ | ↑ | IMM<br>DIR  | 65<br>75   | ii ii+1<br>dd                                   | 3<br>4                               |
| CPX #opr<br>CPX opr<br>CPX opr<br>CPX ,X<br>CPX opr,X<br>CPX opr,X<br>CPX opr,SP<br>CPX opr,SP | Compare X with M                 | (X) – (M)  | ↑             | – | – | ↑ | ↑ | ↑ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A3<br>B3<br>C3<br>D3<br>E3<br>F3<br>9EE3<br>9ED3 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| DAA  | Decimal Adjust A                 | (A) <sub>10</sub>  | U             | – | – | ↑ | ↑ | ↑ | INH   | 72   |   | 2                                    |
| DBNZ opr,rel<br>DBNZA rel<br>DBNZX rel<br>DBNZ opr,X,rel<br>DBNZ X,rel<br>DBNZ opr,SP,rel      | Decrement and Branch if Not Zero | $A \leftarrow (A) - 1$ or $M \leftarrow (M) - 1$ or $X \leftarrow (X) - 1$<br>$PC \leftarrow (PC) + 3 + rel ? (result) \neq 0$<br>$PC \leftarrow (PC) + 2 + rel ? (result) \neq 0$<br>$PC \leftarrow (PC) + 2 + rel ? (result) \neq 0$<br>$PC \leftarrow (PC) + 3 + rel ? (result) \neq 0$<br>$PC \leftarrow (PC) + 2 + rel ? (result) \neq 0$<br>$PC \leftarrow (PC) + 4 + rel ? (result) \neq 0$ | –             | – | – | – | – | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1               | 3B<br>4B<br>5B<br>6B<br>7B<br>9E6B               | dd rr<br>rr<br>rr<br>ff rr<br>rr<br>ff rr       | 5<br>3<br>3<br>5<br>4<br>6           |
| DEC opr<br>DECA<br>DECX<br>DEC opr,X<br>DEC ,X<br>DEC opr,SP                                   | Decrement                        | $M \leftarrow (M) - 1$<br>$A \leftarrow (A) - 1$<br>$X \leftarrow (X) - 1$<br>$M \leftarrow (M) - 1$<br>$M \leftarrow (M) - 1$<br>$M \leftarrow (M) - 1$   | ↑             | – | – | ↑ | ↑ | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1               | 3A<br>4A<br>5A<br>6A<br>7A<br>9E6A               | dd<br>ff<br>ff                                  | 4<br>1<br>1<br>4<br>3<br>5           |
| DIV  | Divide                           | $A \leftarrow (H:A)/(X)$<br>H ← Remainder  | –             | – | – | – | ↑ | ↑ | INH   | 52   |   | 7                                    |
| EOR #opr<br>EOR opr<br>EOR opr<br>EOR opr,X<br>EOR opr,X<br>EOR ,X<br>EOR opr,SP<br>EOR opr,SP | Exclusive OR M with A            | $A \leftarrow (A \oplus M)$  | 0             | – | – | ↑ | ↑ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A8<br>B8<br>C8<br>D8<br>E8<br>F8<br>9EE8<br>9ED8 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |

# Central Processor Unit (CPU)

Table 1. Instruction Set Summary (Continued)

| Source Form  | Operation                           | Description   | Effect on CCR |   |   |   |   |   | Address Mode  | Opcode   | Operand  | Cycles                               |
|--|-------------------------------------|---|---------------|---|---|---|---|---|---|--|--|--------------------------------------|
|  |                                     |   | V             | H | I | N | Z | C |   |  |  |                                      |
| INC <i>opr</i><br>INCA<br>INCLX<br>INC <i>opr</i> ,X<br>INC ,X<br>INC <i>opr</i> ,SP   | Increment                           | M ← (M) + 1<br>A ← (A) + 1<br>X ← (X) + 1<br>M ← (M) + 1<br>M ← (M) + 1<br>M ← (M) + 1  | ↑             | - | - | ↑ | ↑ | - | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1               | 3C<br>4C<br>5C<br>6C<br>7C<br>9E6C               | dd<br>ff<br>ff   | 4<br>1<br>1<br>4<br>3<br>5           |
| JMP <i>opr</i><br>JMP <i>opr</i><br>JMP <i>opr</i> ,X<br>JMP <i>opr</i> ,X<br>JMP ,X   | Jump                                | PC ← Jump Address   | -             | - | - | - | - | - | DIR<br>EXT<br>IX2<br>IX1<br>IX                      | BC<br>CC<br>DC<br>EC<br>FC                       | dd hh ll<br>ee ff<br>ff                                  | 2<br>3<br>4<br>3<br>2                |
| JSR <i>opr</i><br>JSR <i>opr</i><br>JSR <i>opr</i> ,X<br>JSR <i>opr</i> ,X<br>JSR ,X   | Jump to Subroutine                  | PC ← (PC) + <i>n</i> ( <i>n</i> = 1, 2, or 3)<br>Push (PCL); SP ← (SP) - 1<br>Push (PCH); SP ← (SP) - 1<br>PC ← Unconditional Address | -             | - | - | - | - | - | DIR<br>EXT<br>IX2<br>IX1<br>IX                      | BD<br>CD<br>DD<br>ED<br>FD                       | dd hh ll<br>ee ff<br>ff                                  | 4<br>5<br>6<br>5<br>4                |
| LDA # <i>opr</i><br>LDA <i>opr</i><br>LDA <i>opr</i><br>LDA <i>opr</i> ,X<br>LDA <i>opr</i> ,X<br>LDA ,X<br>LDA <i>opr</i> ,SP<br>LDA <i>opr</i> ,SP | Load A from M                       | A ← (M)   | 0             | - | - | ↑ | ↑ | - | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A6<br>B6<br>C6<br>D6<br>E6<br>F6<br>9EE6<br>9ED6 | ii dd<br>dd hh ll<br>ee ff<br>ff<br>ff<br>ff<br>ff ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| LDHX # <i>opr</i><br>LDHX <i>opr</i>   | Load H:X from M                     | H:X ← (M:M + 1)   | 0             | - | - | ↑ | ↑ | - | IMM<br>DIR  | 45<br>55   | ii jj<br>dd  | 3<br>4                               |
| LDX # <i>opr</i><br>LDX <i>opr</i><br>LDX <i>opr</i><br>LDX <i>opr</i> ,X<br>LDX <i>opr</i> ,X<br>LDX ,X<br>LDX <i>opr</i> ,SP<br>LDX <i>opr</i> ,SP | Load X from M                       | X ← (M)   | 0             | - | - | ↑ | ↑ | - | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | AE<br>BE<br>CE<br>DE<br>EE<br>FE<br>9EEE<br>9EDE | ii dd<br>dd hh ll<br>ee ff<br>ff<br>ff<br>ff<br>ff ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| LSL <i>opr</i><br>LSLA<br>LSLX<br>LSL <i>opr</i> ,X<br>LSL ,X<br>LSL <i>opr</i> ,SP  | Logical Shift Left<br>(Same as ASL) |    | ↑             | - | - | ↑ | ↑ | ↑ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1               | 38<br>48<br>58<br>68<br>78<br>9E68               | dd<br>ff<br>ff   | 4<br>1<br>1<br>4<br>3<br>5           |
| LSR <i>opr</i><br>LSRA<br>LSRX<br>LSR <i>opr</i> ,X<br>LSR ,X<br>LSR <i>opr</i> ,SP  | Logical Shift Right                 |    | ↑             | - | - | 0 | ↑ | ↑ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1               | 34<br>44<br>54<br>64<br>74<br>9E64               | dd<br>ff<br>ff   | 4<br>1<br>1<br>4<br>3<br>5           |
| MOV <i>opr</i> , <i>opr</i><br>MOV <i>opr</i> ,X+<br>MOV # <i>opr</i> , <i>opr</i><br>MOV X+, <i>opr</i>   | Move                                | (M) <sub>Destination</sub> ← (M) <sub>Source</sub><br>H:X ← (H:X) + 1 (IX+D, DIX+)  | 0             | - | - | ↑ | ↑ | - | DD<br>DIX+<br>IMD<br>IX+D                           | 4E<br>5E<br>6E<br>7E                             | dd dd<br>dd<br>ii dd<br>dd                               | 5<br>4<br>4<br>4                     |
| MUL  | Unsigned multiply                   | X:A ← (X) × (A)   | -             | 0 | - | - | - | 0 | INH   | 42   |  | 5                                    |

**Table 1. Instruction Set Summary (Continued)**

| Source Form  | Operation                  | Description   | Effect on CCR |   |   |   |   |   | Address Mode  | Opcode   | Operand   | Cycles                               |
|--|----------------------------|---|---------------|---|---|---|---|---|---|--|---|--------------------------------------|
|  |                            |   | V             | H | I | N | Z | C |   |  |   |                                      |
| NEG <i>opr</i><br>NEGA<br>NEGX<br>NEG <i>opr,X</i><br>NEG ,X<br>NEG <i>opr,SP</i>  | Negate (Two's Complement)  | $M \leftarrow -(M) = \$00 - (M)$<br>$A \leftarrow -(A) = \$00 - (A)$<br>$X \leftarrow -(X) = \$00 - (X)$<br>$M \leftarrow -(M) = \$00 - (M)$<br>$M \leftarrow -(M) = \$00 - (M)$                      | ↑             | - | - | ↑ | ↑ | ↑ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1                     | 30<br>40<br>50<br>60<br>70<br>9E60               | dd<br>ff<br>ff  | 4<br>1<br>1<br>4<br>3<br>5           |
| NOP  | No Operation               | None  | -             | - | - | - | - | - | INH   | 9D   |   | 1                                    |
| NSA  | Nibble Swap A              | $A \leftarrow (A[3:0]:A[7:4])$  | -             | - | - | - | - | - | INH   | 62   |   | 3                                    |
| ORA # <i>opr</i><br>ORA <i>opr</i><br>ORA <i>opr</i><br>ORA <i>opr,X</i><br>ORA <i>opr,X</i><br>ORA ,X<br>ORA <i>opr,SP</i><br>ORA <i>opr,SP</i> | Inclusive OR A and M       | $A \leftarrow (A)   (M)$  | 0             | - | - | ↑ | ↑ | - | IMM<br>DIR<br>EXT<br>IX2<br>DA<br>IX1<br>IX<br>SP1<br>SP2 | AA<br>BA<br>CA<br>DA<br>EA<br>FA<br>9EEA<br>9EDA | ii<br>dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| PSHA   | Push A onto Stack          | Push (A); $SP \leftarrow (SP) - 1$  | -             | - | - | - | - | - | INH   | 87   |   | 2                                    |
| PSHH   | Push H onto Stack          | Push (H); $SP \leftarrow (SP) - 1$  | -             | - | - | - | - | - | INH   | 8B   |   | 2                                    |
| PSHX   | Push X onto Stack          | Push (X); $SP \leftarrow (SP) - 1$  | -             | - | - | - | - | - | INH   | 89   |   | 2                                    |
| PULA   | Pull A from Stack          | $SP \leftarrow (SP) + 1$ ; Pull (A)   | -             | - | - | - | - | - | INH   | 86   |   | 2                                    |
| PULH   | Pull H from Stack          | $SP \leftarrow (SP) + 1$ ; Pull (H)   | -             | - | - | - | - | - | INH   | 8A   |   | 2                                    |
| PULX   | Pull X from Stack          | $SP \leftarrow (SP) + 1$ ; Pull (X)   | -             | - | - | - | - | - | INH   | 88   |   | 2                                    |
| ROL <i>opr</i><br>ROLA<br>ROLX<br>ROL <i>opr,X</i><br>ROL ,X<br>ROL <i>opr,SP</i>  | Rotate Left through Carry  |   | ↑             | - | - | ↑ | ↑ | ↑ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1                     | 39<br>49<br>59<br>69<br>79<br>9E69               | dd<br>ff<br>ff  | 4<br>1<br>1<br>4<br>3<br>5           |
| ROR <i>opr</i><br>RORA<br>RORX<br>ROR <i>opr,X</i><br>ROR ,X<br>ROR <i>opr,SP</i>  | Rotate Right through Carry |   | ↑             | - | - | ↑ | ↑ | ↑ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1                     | 36<br>46<br>56<br>66<br>76<br>9E66               | dd<br>ff<br>ff  | 4<br>1<br>1<br>4<br>3<br>5           |
| RSP  | Reset Stack Pointer        | $SP \leftarrow \$FF$  | -             | - | - | - | - | - | INH   | 9C   |   | 1                                    |
| RTI  | Return from Interrupt      | $SP \leftarrow (SP) + 1$ ; Pull (CCR)<br>$SP \leftarrow (SP) + 1$ ; Pull (A)<br>$SP \leftarrow (SP) + 1$ ; Pull (X)<br>$SP \leftarrow (SP) + 1$ ; Pull (PCH)<br>$SP \leftarrow (SP) + 1$ ; Pull (PCL) | ↑             | ↑ | ↑ | ↑ | ↑ | ↑ | INH   | 80   |   | 7                                    |
| RTS  | Return from Subroutine     | $SP \leftarrow SP + 1$ ; Pull (PCH)<br>$SP \leftarrow SP + 1$ ; Pull (PCL)  | -             | - | - | - | - | - | INH   | 81   |   | 4                                    |

# Central Processor Unit (CPU)

**Table 1. Instruction Set Summary (Continued)**

| Source Form  | Operation                                    | Description   | Effect on CCR |   |   |   |   |   | Address Mode  | Opcode   | Operand   | Cycles                               |
|--|--|---|---------------|---|---|---|---|---|---|--|---|--------------------------------------|
|  |  |   | V             | H | I | N | Z | C |   |  |   |                                      |
| SBC #opr<br>SBC opr<br>SBC opr<br>SBC opr,X<br>SBC opr,X<br>SBC ,X<br>SBC opr,SP<br>SBC opr,SP | Subtract with Carry                          | $A \leftarrow (A) - (M) - (C)$  | ↑             | - | - | ↑ | ↑ | ↑ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A2<br>B2<br>C2<br>D2<br>E2<br>F2<br>9EE2<br>9ED2 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| SEC  | Set Carry Bit                                | $C \leftarrow 1$  | -             | - | - | - | - | 1 | INH   | 99   |   | 1                                    |
| SEI  | Set Interrupt Mask                           | $I \leftarrow 1$  | -             | - | 1 | - | - | - | INH   | 9B   |   | 2                                    |
| STA opr<br>STA opr<br>STA opr,X<br>STA opr,X<br>STA ,X<br>STA opr,SP<br>STA opr,SP             | Store A in M                                 | $M \leftarrow (A)$  | 0             | - | - | ↑ | ↑ | - | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2        | B7<br>C7<br>D7<br>E7<br>F7<br>9EE7<br>9ED7       | dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ff<br>ee ff       | 3<br>4<br>4<br>3<br>2<br>4<br>5      |
| STHX opr   | Store H:X in M                               | $(M:M + 1) \leftarrow (H:X)$  | 0             | - | - | ↑ | ↑ | - | DIR   | 35   | dd  | 4                                    |
| STOP   | Enable $\overline{IRQ}$ Pin; Stop Oscillator | $I \leftarrow 0$ ; Stop Oscillator  | -             | - | 0 | - | - | - | INH   | 8E   |   | 1                                    |
| STX opr<br>STX opr<br>STX opr,X<br>STX opr,X<br>STX ,X<br>STX opr,SP<br>STX opr,SP             | Store X in M                                 | $M \leftarrow (X)$  | 0             | - | - | ↑ | ↑ | - | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2        | BF<br>CF<br>DF<br>EF<br>FF<br>9EEF<br>9EDF       | dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ff<br>ee ff       | 3<br>4<br>4<br>3<br>2<br>4<br>5      |
| SUB #opr<br>SUB opr<br>SUB opr<br>SUB opr,X<br>SUB opr,X<br>SUB ,X<br>SUB opr,SP<br>SUB opr,SP | Subtract                                     | $A \leftarrow (A) - (M)$  | ↑             | - | - | ↑ | ↑ | ↑ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A0<br>B0<br>C0<br>D0<br>E0<br>F0<br>9EE0<br>9ED0 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| SWI  | Software Interrupt                           | PC $\leftarrow$ (PC) + 1; Push (PCL)<br>SP $\leftarrow$ (SP) - 1; Push (PCH)<br>SP $\leftarrow$ (SP) - 1; Push (X)<br>SP $\leftarrow$ (SP) - 1; Push (A)<br>SP $\leftarrow$ (SP) - 1; Push (CCR)<br>SP $\leftarrow$ (SP) - 1; I $\leftarrow$ 1<br>PCH $\leftarrow$ Interrupt Vector High Byte<br>PCL $\leftarrow$ Interrupt Vector Low Byte | -             | - | 1 | - | - | - | INH   | 83   |   | 9                                    |
| TAP  | Transfer A to CCR                            | $CCR \leftarrow (A)$  | ↑             | ↑ | ↑ | ↑ | ↑ | ↑ | INH   | 84   |   | 2                                    |
| TAX  | Transfer A to X                              | $X \leftarrow (A)$  | -             | - | - | - | - | - | INH   | 97   |   | 1                                    |
| TPA  | Transfer CCR to A                            | $A \leftarrow (CCR)$  | -             | - | - | - | - | - | INH   | 85   |   | 1                                    |



**Table 1. Instruction Set Summary (Continued)**

| Source Form   | Operation                 | Description                            | Effect on CCR |   |   |   |   |   | Address Mode                          | Opcode                             | Operand        | Cycles                     |
|---|---------------------------|--|---------------|---|---|---|---|---|---------------------------------------|------------------------------------|----------------|----------------------------|
|   |                           |  | V             | H | I | N | Z | C |                                       |                                    |                |                            |
| TST <i>opr</i><br>TSTA<br>TSTX<br>TST <i>opr,X</i><br>TST ,X<br>TST <i>opr,SP</i> | Test for Negative or Zero | (A) – \$00 or (X) – \$00 or (M) – \$00 | 0             | – | – | ↑ | ↓ | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3D<br>4D<br>5D<br>6D<br>7D<br>9E6D | dd<br>ff<br>ff | 3<br>1<br>1<br>3<br>2<br>4 |
| TSX   | Transfer SP to H:X        | H:X ← (SP) + 1                         | –             | – | – | – | – | – | INH                                   | 95                                 |                | 2                          |
| TXA   | Transfer X to A           | A ← (X)                                | –             | – | – | – | – | – | INH                                   | 9F                                 |                | 1                          |
| TXS   | Transfer H:X to SP        | (SP) ← (H:X) – 1                       | –             | – | – | – | – | – | INH                                   | 94                                 |                | 2                          |

A Accumulator  
C Carry/borrow bit  
CCR Condition code register  
PC Program counter  
dd Direct address of operand  
dd rr Direct address of operand and relative offset of branch instruction  
DDD Direct to direct addressing mode  
DIR Direct addressing mode  
DIX+ Direct to indexed with post increment addressing mode  
ee ff High and low bytes of offset in indexed, 16-bit offset addressing mode  
EXT Extended addressing mode  
ff Offset byte in indexed, 8-bit offset addressing mode  
H Half-carry bit  
H Index register high byte  
hh ll High and low bytes of operand address in extended addressing mode  
I Interrupt mask  
ii Immediate operand byte  
IMD Immediate source to direct destination addressing mode  
IMM Immediate addressing mode  
INH Inherent addressing mode  
IX Indexed, no offset addressing mode  
IX+ Indexed, no offset, post increment addressing mode  
IX+D Indexed with post increment to direct addressing mode  
IX1 Indexed, 8-bit offset addressing mode  
IX1+ Indexed, 8-bit offset, post increment addressing mode  
IX2 Indexed, 16-bit offset addressing mode  
M Memory location  
N Negative bit

Any bit  
Operand (one or two bytes)  
Program counter  
Program counter high byte  
Program counter low byte  
Relative addressing mode  
Relative program counter offset byte  
Relative program counter offset byte  
Stack pointer, 8-bit offset addressing mode  
Stack pointer 16-bit offset addressing mode  
Stack pointer  
Undefined  
Overflow bit  
Index register low byte  
Zero bit  
Logical AND  
Logical OR  
Logical EXCLUSIVE OR  
Contents of  
Negation (two's complement)  
Immediate value  
Sign extend  
Loaded with  
If  
Concatenated with  
Set or cleared  
Not affected

## Opcode Map

The opcode map is provided in [Table 2](#).

# Central Processor Unit (CPU)

**Table 2. Opcode Map**

| Bit Manipulation |                |               | Read-Modify-Write |                |                |                | Control       |               |               |              | Register/Memory |               |               |               |               |               |               |              |
|------------------|----------------|---------------|-------------------|----------------|----------------|----------------|---------------|---------------|---------------|--------------|-----------------|---------------|---------------|---------------|---------------|---------------|---------------|--------------|
| DIR              | DIR            | REL           | DIR               | INH            | INH            | IX1            | SP1           | IX            | INH           | INH          | IMM             | DIR           | EXT           | IX2           | SP2           | IX1           | SP1           | IX           |
| 0                | 1              | 2             | 3                 | 4              | 5              | 6              | 9E6           | 7             | 8             | 9            | A               | B             | C             | D             | 9ED           | E             | 9EE           | F            |
| BRSET0<br>3 DIR  | BSET0<br>4 DIR | BRA<br>2 REL  | NEG<br>2 DIR      | NEGA<br>1 INH  | NEG<br>1 INH   | NEG<br>2 IX1   | NEG<br>3 SP1  | NEG<br>1 IX   | RTI<br>1 INH  | BGE<br>2 REL | SUB<br>2 IMM    | SUB<br>2 DIR  | SUB<br>3 EXT  | SUB<br>4 IX2  | SUB<br>4 SP2  | SUB<br>3 IX1  | SUB<br>3 SP1  | SUB<br>2 IX  |
| BRCLR0<br>3 DIR  | BCLR0<br>4 DIR | BRN<br>2 REL  | CBEO<br>3 DIR     | CBEOA<br>3 IMM | CBEOX<br>3 IMM | CBEO<br>3 IX1+ | CBEO<br>4 SP1 | CBEO<br>1 IX+ | RTS<br>1 INH  | BLT<br>2 REL | COMP<br>2 IMM   | COMP<br>2 DIR | COMP<br>3 EXT | COMP<br>4 IX2 | COMP<br>4 SP2 | COMP<br>3 IX1 | COMP<br>3 SP1 | COMP<br>2 IX |
| BRSET1<br>3 DIR  | BSET1<br>4 DIR | BHI<br>2 REL  | MUL<br>1 DIR      | MUL<br>1 INH   | DIV<br>1 INH   | NSA<br>3 INH   | NSA<br>3 SP1  | NSA<br>1 INH  | DAI<br>1 INH  | BGT<br>2 REL | SBC<br>2 IMM    | SBC<br>2 DIR  | SBC<br>3 EXT  | SBC<br>4 IX2  | SBC<br>4 SP2  | SBC<br>3 IX1  | SBC<br>3 SP1  | SBC<br>2 IX  |
| BRCLR1<br>3 DIR  | BCLR1<br>4 DIR | BLS<br>2 REL  | COM<br>2 DIR      | COMA<br>1 INH  | COMX<br>1 INH  | COM<br>2 IX1   | COM<br>3 SP1  | COM<br>1 IX   | SWI<br>1 INH  | BLE<br>3 REL | CPX<br>2 IMM    | CPX<br>2 DIR  | CPX<br>3 EXT  | CPX<br>4 IX2  | CPX<br>4 SP2  | CPX<br>3 IX1  | CPX<br>3 SP1  | CPX<br>2 IX  |
| BRSET2<br>3 DIR  | BSET2<br>4 DIR | BCC<br>2 REL  | LSR<br>2 DIR      | LSRA<br>1 INH  | LSRX<br>1 INH  | LSR<br>2 IX1   | LSR<br>3 SP1  | LSR<br>1 IX   | TAP<br>1 INH  | TXS<br>2 REL | AND<br>2 IMM    | AND<br>2 DIR  | AND<br>3 EXT  | AND<br>4 IX2  | AND<br>4 SP2  | AND<br>3 IX1  | AND<br>3 SP1  | AND<br>2 IX  |
| BRCLR2<br>3 DIR  | BCLR2<br>4 DIR | BCS<br>2 REL  | STHX<br>2 DIR     | LDHX<br>3 IMM  | LDHX<br>4 DIR  | CPHX<br>3 IMM  | CPHX<br>2 SP1 | CPHX<br>1 IX  | TPA<br>1 INH  | TSX<br>1 INH | BIT<br>2 IMM    | BIT<br>2 DIR  | BIT<br>3 EXT  | BIT<br>4 IX2  | BIT<br>4 SP2  | BIT<br>3 IX1  | BIT<br>3 SP1  | BIT<br>2 IX  |
| BRSET3<br>3 DIR  | BSET3<br>4 DIR | BNE<br>2 REL  | ROR<br>2 DIR      | RORA<br>1 INH  | RORX<br>1 INH  | ROR<br>2 IX1   | ROR<br>3 SP1  | ROR<br>1 IX   | PULA<br>2 INH | 2            | LDA<br>2 IMM    | LDA<br>2 DIR  | LDA<br>3 EXT  | LDA<br>4 IX2  | LDA<br>4 SP2  | LDA<br>3 IX1  | LDA<br>3 SP1  | LDA<br>2 IX  |
| BRCLR3<br>3 DIR  | BCLR3<br>4 DIR | BEQ<br>2 REL  | ASR<br>2 DIR      | ASRA<br>1 INH  | ASRX<br>1 INH  | ASR<br>2 IX1   | ASR<br>3 SP1  | ASR<br>1 IX   | PSHA<br>1 INH | TAX<br>1 INH | AIS<br>2 IMM    | AIS<br>2 DIR  | AIS<br>3 EXT  | AIS<br>4 IX2  | AIS<br>4 SP2  | AIS<br>3 IX1  | AIS<br>3 SP1  | AIS<br>2 IX  |
| BRSET4<br>3 DIR  | BSET4<br>4 DIR | BHCC<br>2 REL | LSL<br>2 DIR      | LSLA<br>1 INH  | LSLX<br>1 INH  | LSL<br>2 IX1   | LSL<br>3 SP1  | LSL<br>1 IX   | PULX<br>1 INH | CLC<br>1 INH | EOR<br>2 IMM    | EOR<br>2 DIR  | EOR<br>3 EXT  | EOR<br>4 IX2  | EOR<br>4 SP2  | EOR<br>3 IX1  | EOR<br>3 SP1  | EOR<br>2 IX  |
| BRCLR4<br>3 DIR  | BCLR4<br>4 DIR | BHCS<br>2 REL | ROL<br>2 DIR      | ROLA<br>1 INH  | ROLDX<br>1 INH | ROL<br>2 IX1   | ROL<br>3 SP1  | ROL<br>1 IX   | PSHX<br>1 INH | SEC<br>1 INH | ADC<br>2 IMM    | ADC<br>2 DIR  | ADC<br>3 EXT  | ADC<br>4 IX2  | ADC<br>4 SP2  | ADC<br>3 IX1  | ADC<br>3 SP1  | ADC<br>2 IX  |
| BRSET5<br>3 DIR  | BSET5<br>4 DIR | BPL<br>2 REL  | DEC<br>2 DIR      | DECA<br>1 INH  | DECX<br>1 INH  | DEC<br>2 IX1   | DEC<br>3 SP1  | DEC<br>1 IX   | PULH<br>1 INH | CLI<br>1 INH | ORA<br>2 IMM    | ORA<br>2 DIR  | ORA<br>3 EXT  | ORA<br>4 IX2  | ORA<br>4 SP2  | ORA<br>3 IX1  | ORA<br>3 SP1  | ORA<br>2 IX  |
| BRCLR5<br>3 DIR  | BCLR5<br>4 DIR | BML<br>2 REL  | DBNZ<br>3 DIR     | DBNZA<br>3 IMM | DBNZX<br>3 IMM | DBNZ<br>3 IX1  | DBNZ<br>4 SP1 | DBNZ<br>1 IX  | PSHH<br>1 INH | SEI<br>1 INH | ADD<br>2 IMM    | ADD<br>2 DIR  | ADD<br>3 EXT  | ADD<br>4 IX2  | ADD<br>4 SP2  | ADD<br>3 IX1  | ADD<br>3 SP1  | ADD<br>2 IX  |
| BRSET6<br>3 DIR  | BSET6<br>4 DIR | BMC<br>2 REL  | INC<br>2 DIR      | INCA<br>1 INH  | INCX<br>1 INH  | INC<br>2 IX1   | INC<br>3 SP1  | INC<br>1 IX   | CLRH<br>1 INH | RSP<br>1 INH | JMP<br>2 IMM    | JMP<br>2 DIR  | JMP<br>3 EXT  | JMP<br>4 IX2  | JMP<br>4 SP2  | JMP<br>3 IX1  | JMP<br>3 SP1  | JMP<br>2 IX  |
| BRCLR6<br>3 DIR  | BCLR6<br>4 DIR | BMS<br>2 REL  | TST<br>2 DIR      | TSTA<br>1 INH  | TSTX<br>1 INH  | TST<br>2 IX1   | TST<br>3 SP1  | TST<br>1 IX   | NOP<br>1 INH  | 1            | BSR<br>4 IMM    | BSR<br>4 DIR  | BSR<br>5 EXT  | BSR<br>6 IX2  | BSR<br>6 SP2  | BSR<br>5 IX1  | BSR<br>5 SP1  | BSR<br>4 IX  |
| BRSET7<br>3 DIR  | BSET7<br>4 DIR | BIL<br>2 REL  | MOV<br>3 DIR      | MOVA<br>3 IMM  | MOVX<br>3 IMM  | MOV<br>3 IX+D  | MOV<br>4 SP1  | MOV<br>1 IX   | STOP<br>1 INH | *            | LDX<br>2 IMM    | LDX<br>2 DIR  | LDX<br>3 EXT  | LDX<br>4 IX2  | LDX<br>4 SP2  | LDX<br>3 IX1  | LDX<br>3 SP1  | LDX<br>2 IX  |
| BRCLR7<br>3 DIR  | BCLR7<br>4 DIR | BIH<br>2 REL  | CLR<br>2 DIR      | CLRA<br>1 INH  | CLR<br>1 INH   | CLR<br>2 IX1   | CLR<br>3 SP1  | CLR<br>1 IX   | WAIT<br>1 INH | TXA<br>1 INH | AIX<br>2 IMM    | AIX<br>2 DIR  | AIX<br>3 EXT  | AIX<br>4 IX2  | AIX<br>4 SP2  | AIX<br>3 IX1  | AIX<br>3 SP1  | AIX<br>2 IX  |

INH Inherent  
 IMM Immediate  
 DIR Direct  
 EXT Extended  
 DD Direct-Direct  
 IX+D Indexed-Direct  
 \* Pre-byte for stack pointer indexed instructions  
 REL Relative  
 IX Indexed, No Offset  
 SP2 Stack Pointer, 16-Bit Offset  
 IX+ Indexed, No Offset with Post Increment  
 IX1+ Indexed, 16-Bit Offset  
 IMM Immediate-Direct  
 IX+D Direct-Indexed  
 \* Pre-byte for stack pointer indexed instructions  
 SP1 Stack Pointer, 8-Bit Offset  
 SP2 Stack Pointer, 16-Bit Offset  
 IX+ Indexed, No Offset with Post Increment  
 IX1+ Indexed, 16-Bit Offset  
 IMM Immediate-Direct  
 IX+D Direct-Indexed  
 \* Pre-byte for stack pointer indexed instructions  
 MSB  
 0  
 LSB  
 5  
 BRSET0  
 3 DIR  
 High Byte of Opcode in Hexadecimal  
 Cycles  
 Opcode Mnemonic  
 Number of Bytes / Addressing Mode  
 Low Byte of Opcode in Hexadecimal  
 High Byte of Opcode in Hexadecimal

# System Integration Module (SIM)

---

---

## Contents

|   |     |
|---|-----|
| Introduction .....                            | 106 |
| SIM Bus Clock Control and Generation .....    | 109 |
| Bus Timing .....                              | 109 |
| Clock Startup from POR or LVI Reset .....     | 110 |
| Clocks in Stop Mode and Wait Mode .....       | 110 |
| Reset and System Initialization .....         | 110 |
| External Pin Reset .....                      | 110 |
| Active Resets from Internal Sources .....     | 111 |
| Power-On Reset .....                          | 112 |
| Computer Operating Properly (COP) Reset ..... | 113 |
| Illegal Opcode Reset .....                    | 113 |
| Illegal Address Reset .....                   | 114 |
| Low-Voltage Inhibit (LVI) Reset .....         | 114 |
| SIM Counter .....                             | 115 |
| SIM Counter During Power-On Reset .....       | 115 |
| SIM Counter During Stop Mode Recovery .....   | 115 |
| SIM Counter and Reset States .....            | 115 |
| Program Exception Control .....               | 116 |
| Interrupts .....                              | 116 |
| Reset .....                                   | 119 |
| Break Interrupts .....                        | 119 |
| Status Flag Protection in Break Mode .....    | 120 |
| Low-Power Modes .....                         | 120 |
| Wait Mode .....                               | 120 |
| Stop Mode .....                               | 122 |
| SIM Registers .....                           | 123 |
| SIM Break Status Register .....               | 123 |
| SIM Reset Status Register .....               | 124 |
| SIM Break Flag Control Register .....         | 125 |

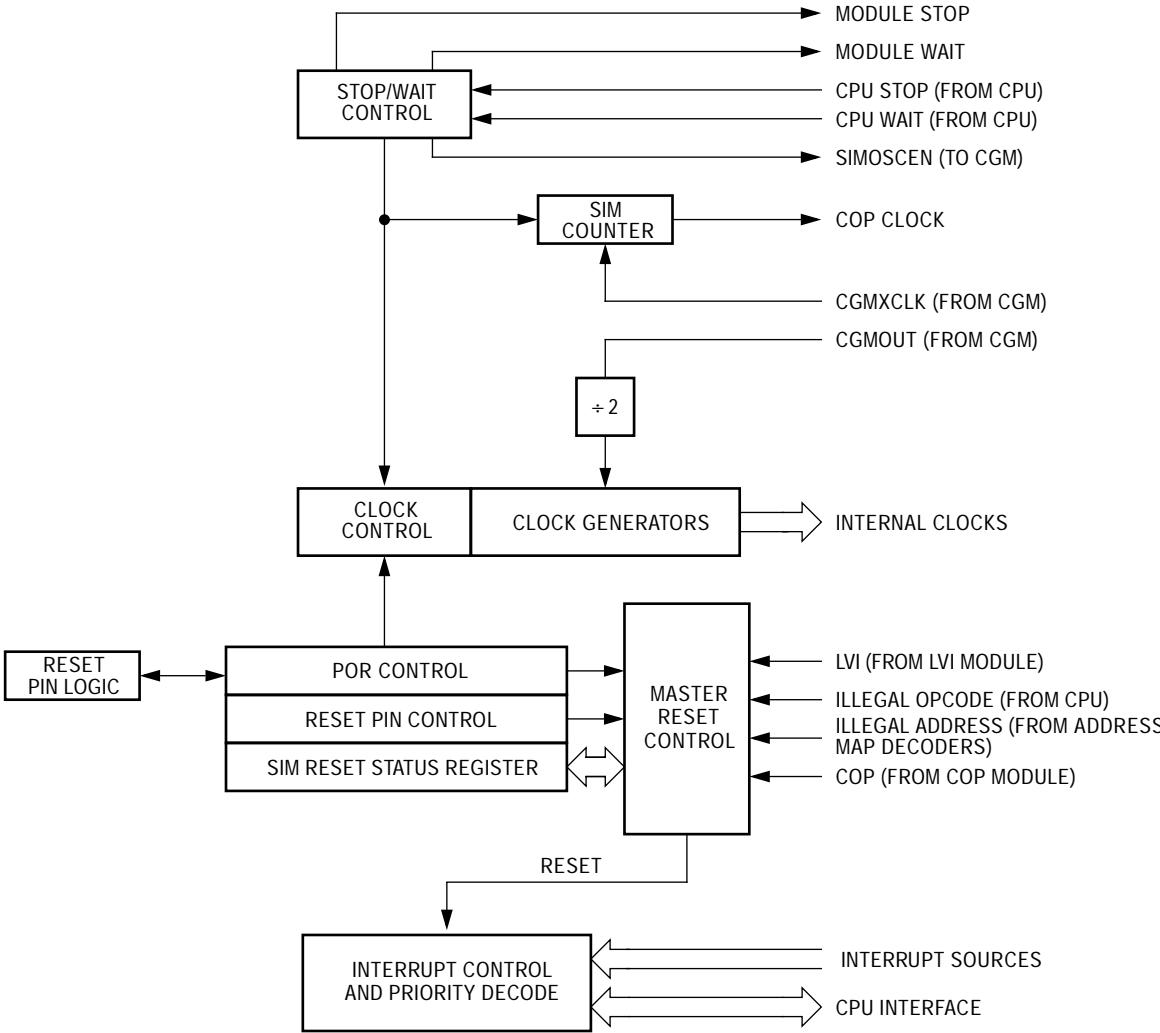
---

---

### Introduction

This section describes the system integration module (SIM), which supports up to 32 external and/or internal interrupts. Together with the central processor unit (CPU), the SIM controls all MCU activities. A block diagram of the SIM is shown in [Figure 1](#). [Figure 2](#) is a summary of the SIM input/output (I/O) registers. The SIM is a system state controller that coordinates CPU and exception timing. The SIM is responsible for:

- Bus clock generation and control for CPU and peripherals
  - Stop/wait/reset/break entry and recovery
  - Internal clock control
- Master reset control, including power-on reset (POR) and computer operating properly (COP) timeout
- Interrupt control:
  - Acknowledge timing
  - Arbitration control timing
  - Vector address generation
- CPU enable/disable timing



**Figure 1. SIM Block Diagram**

## System Integration Module (SIM)

| Register Name                           | Bit 7 | 6   | 5   | 4    | 3    | 2 | 1   | Bit 0 |
|---|-------|-----|-----|------|------|---|-----|-------|
| SIM Break Status Register (SBSR)        | R     | R   | R   | R    | R    | R | BW  | R     |
| SIM Reset Status Register (SRSR)        | POR   | PIN | COP | ILOP | ILAD | 0 | LVI | 0     |
| SIM Break Flag Control Register (SBFCR) | BCFE  | R   | R   | R    | R    | R | R   | R     |

R = Reserved

**Figure 2. SIM I/O Register Summary**

**Table 1. I/O Register Address Summary**

|                 |        |        |        |
|-----------------|--------|--------|--------|
| <b>Register</b> | SBSR   | SRSR   | SBFCR  |
| <b>Address</b>  | \$FE00 | \$FE01 | \$FE03 |

**Table 2** shows the internal signal names used in this section.

**Table 2. Signal Name Conventions**

| Signal Name  | Description  |
|--------------|--|
| CGMXCLK      | Buffered Version of OSC1 from Clock Generator Module (CGM)                               |
| CGMVCLK      | PLL Output   |
| CGMOUT       | PLL-Based or OSC1-Based Clock Output from CGM Module (Bus Clock = CGMOUT Divided by Two) |
| IAB          | Internal Address Bus   |
| IDB          | Internal Data Bus  |
| PORRST       | Signal from the Power-On Reset Module to the SIM   |
| IRST         | Internal Reset Signal  |
| R/ $\bar{W}$ | Read/Write Signal  |

## SIM Bus Clock Control and Generation

The bus clock generator provides system clock signals for the CPU and peripherals on the MCU. The system clocks are generated from an incoming clock, CGMOUT, as shown in [Figure 3](#). This clock can come from either an external oscillator or from the on-chip PLL. (See [Clock Generator Module \(CGM\)](#) on page 127).

### Bus Timing

In user mode, the internal bus frequency is either the crystal oscillator output (CGMXCLK) divided by four or the PLL output (CGMVCLK) divided by four. (See [Clock Generator Module \(CGM\)](#) on page 127).

### Clock Startup from POR or LVI Reset

When the power-on reset module or the low-voltage inhibit module generates a reset, the clocks to the CPU and peripherals are inactive and held in an inactive phase until after 4096 CGMXCLK cycles. The  $\overline{\text{RST}}$  pin is driven low by the SIM during this entire period. The bus clocks start upon completion of the timeout.

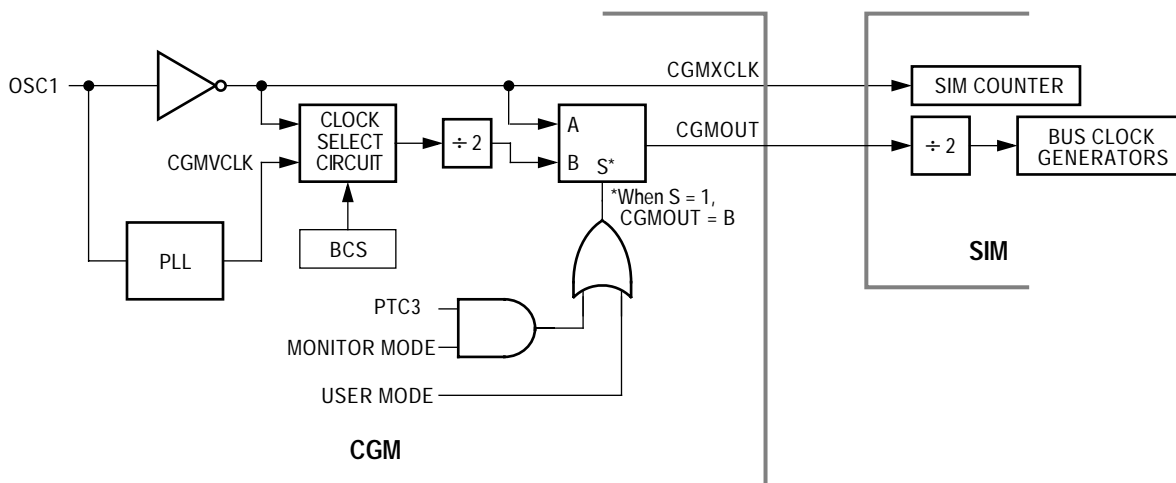


Figure 3. CGM Clock Signals

## System Integration Module (SIM)

### Clocks in Stop Mode and Wait Mode

Upon exit from stop mode by an interrupt, break, or reset, the SIM allows CGMXCLK to clock the SIM counter. The CPU and peripheral clocks do not become active until after the stop delay timeout. This timeout is selectable as 4096 or 32 CGMXCLK cycles. See [Stop Mode](#) on page 122.

In wait mode, the CPU clocks are inactive. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

---

---

## Reset and System Initialization

The MCU has these reset sources:

- Power-on reset module (POR)
- External reset pin ( $\overline{\text{RST}}$ )
- Computer operating properly module (COP)
- Low-voltage inhibit module (LVI)
- Illegal opcode
- Illegal address

All of these resets produce the vector \$FFFE–FFFF (\$FEFE–FEFF in monitor mode) and assert the internal reset signal (IRST). IRST causes all registers to be returned to their default values and all modules to be returned to their reset states.

An internal reset clears the SIM counter (see [SIM Counter](#) on page 115), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR) (see [SIM Registers](#) on page 123).

### External Pin Reset

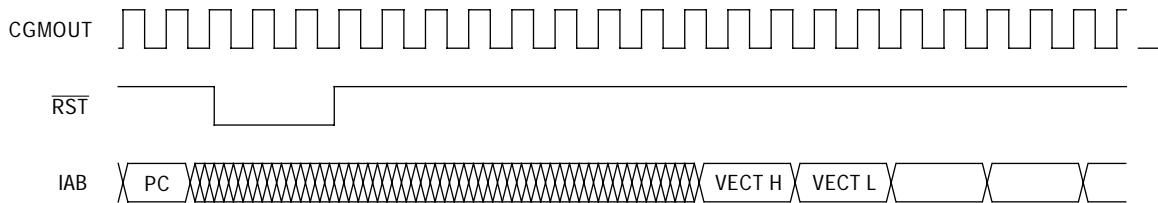
Pulling the asynchronous  $\overline{\text{RST}}$  pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as  $\overline{\text{RST}}$  is held low for a minimum of 67 CGMXCLK cycles, assuming that neither the POR



nor the LVI was the source of the reset. See [Table 3](#) for details. [Figure 4](#) shows the relative timing.

**Table 3. PIN Bit Set Timing**

| Reset Type | Number of Cycles Required to Set PIN |
|------------|--------------------------------------|
| POR/LVI    | 4163 (4096 + 64 + 3)                 |
| All others | 67 (64 + 3)                          |



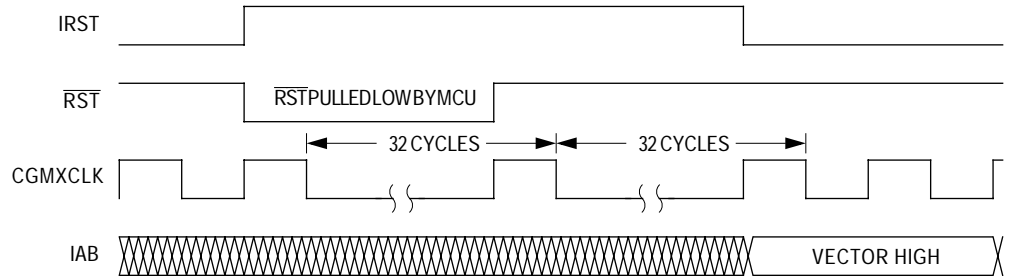
**Figure 4. External Reset Timing**

### Active Resets from Internal Sources

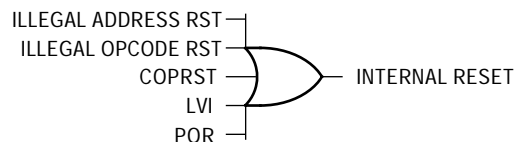
All internal reset sources actively pull the  $\overline{\text{RST}}$  pin low for 32 CGMXCLK cycles to allow resetting of external peripherals. The internal reset signal  $\overline{\text{IRST}}$  continues to be asserted for an additional 32 cycles (see [Figure 5](#)). An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR (see [Figure 6](#)). Note that for LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles during which the SIM forces the  $\overline{\text{RST}}$  pin low. The internal reset signal then follows the sequence from the falling edge of  $\overline{\text{RST}}$  shown in [Figure 5](#).

The COP reset is asynchronous to the bus clock.

The active reset feature allows the part to issue a reset to peripherals and other chips within a system built around the MCU.



**Figure 5. Internal Reset Timing**



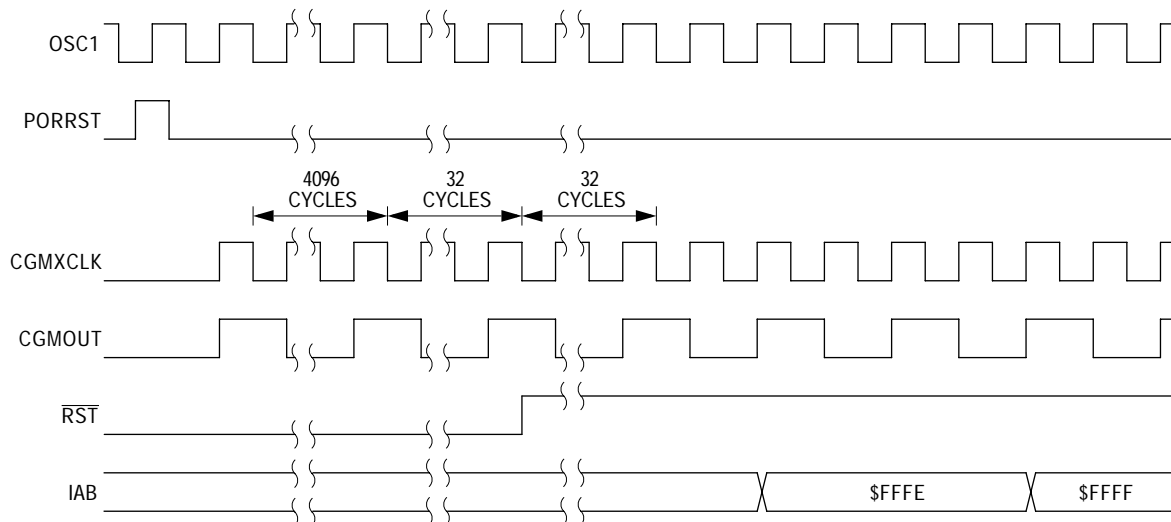
**Figure 6. Sources of Internal Reset**

## Power-On Reset

When power is first applied to the MCU, the power-on reset module (POR) generates a pulse to indicate that power-on has occurred. The external reset pin ( $\overline{RST}$ ) is held low while the SIM counter counts out 4096 CGMXCLK cycles. Another sixty-four CGMXCLK cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur.

At power-on, the following events occur:

- A POR pulse is generated.
- The internal reset signal is asserted.
- The SIM enables CGMOUT.
- Internal clocks to the CPU and modules are held inactive for 4096 CGMXCLK cycles to allow stabilization of the oscillator.
- The  $\overline{RST}$  pin is driven low during the oscillator stabilization time.
- The POR bit of the SIM reset status register (SRSR) is set and all other bits in the register are cleared.



**Figure 7. POR Recovery**

*Computer  
Operating  
Properly (COP)  
Reset*

The overflow of the COP counter causes an internal reset and sets the COP bit in the SIM reset status register (SRSR) if the COPD bit in the CONFIG-1 register is at logic zero.  
See [Computer Operating Properly Module \(COP\)](#) on page 179.

*Illegal Opcode  
Reset*

The SIM decodes signals from the CPU to detect illegal instructions. An illegal instruction sets the ILOP bit in the SIM reset status register (SRSR) and causes a reset.

If the stop enable bit, STOP, in the CONFIG-1 register is logic zero, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset.

## System Integration Module (SIM)

### *Illegal Address Reset*

An opcode fetch from an unmapped address generates an illegal address reset. The SIM verifies that the CPU is fetching an opcode prior to asserting the ILAD bit in the SIM reset status register (SRSR) and resetting the MCU. A data fetch from an unmapped address using indexed addressing and PUL/PSH instructions will also generate an illegal address reset.

**Extra care should be exercised when using this emulation part for development of code to be run in ROM AZ, AB or AS family parts with a smaller memory size, since some legal addresses will become illegal addresses on the smaller ROM memory map device and may, as a result, generate unwanted resets.**

### *Low-Voltage Inhibit (LVI) Reset*

The low-voltage inhibit module (LVI) asserts its output to the SIM when the  $V_{DD}$  voltage falls to the  $V_{LVII}$  voltage. The LVI bit in the SIM reset status register (SRSR) is set and a chip reset is asserted if the LVIPWRD and LVIRSTD bits in the CONFIG-1 register are at logic zero. The  $\overline{RST}$  pin will be held low until the SIM counts 4096 CGMXCLK cycles after  $V_{DD}$  rises above  $V_{LVIR}$ . Another sixty-four CGMXCLK cycles later, the CPU is released from reset to allow the reset vector sequence to occur. See [Low-Voltage Inhibit \(LVI\)](#) on page 185.

---

---

## SIM Counter

The SIM counter is used by the power-on reset module (POR) and in stop mode recovery to allow the oscillator time to stabilize before enabling the internal bus (IBUS) clocks. The SIM counter also serves as a prescaler for the computer operating properly module (COP). The SIM counter overflow supplies the clock for the COP module. The SIM counter is 12 bits long and is clocked by the falling edge of CGMXCLK.

### **SIM Counter During Power-On Reset**

The power-on reset module (POR) detects power applied to the MCU. At power-on, the POR circuit asserts the signal PORRST. Once the SIM is initialized, it enables the clock generation module (CGM) to drive the bus clock state machine.

### **SIM Counter During Stop Mode Recovery**

The SIM counter also is used for stop mode recovery. The STOP instruction clears the SIM counter. After an interrupt, or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the CONFIG-1 register. If the SSREC bit is a logic one, then the stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32 CGMXCLK cycles. This is ideal for applications using canned oscillators that do not require long start-up times from stop mode. External crystal applications should use the full stop recovery time, that is, with SSREC cleared.

### **SIM Counter and Reset States**

External reset has no effect on the SIM counter. See [Stop Mode](#) on page 122 for details. The SIM counter is free-running after all reset states. See [Active Resets from Internal Sources](#) on page 111 for counter control and internal reset recovery sequences.

## Program Exception Control

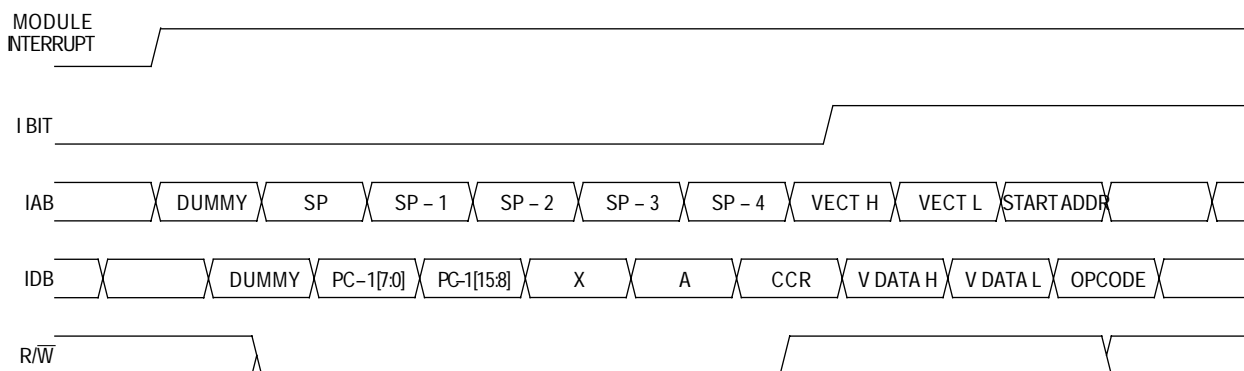
Normal, sequential program execution can be changed in three different ways:

- Interrupts
  - Maskable hardware CPU interrupts
  - Non-maskable software interrupt instruction (SWI)
- Reset
- Break interrupts

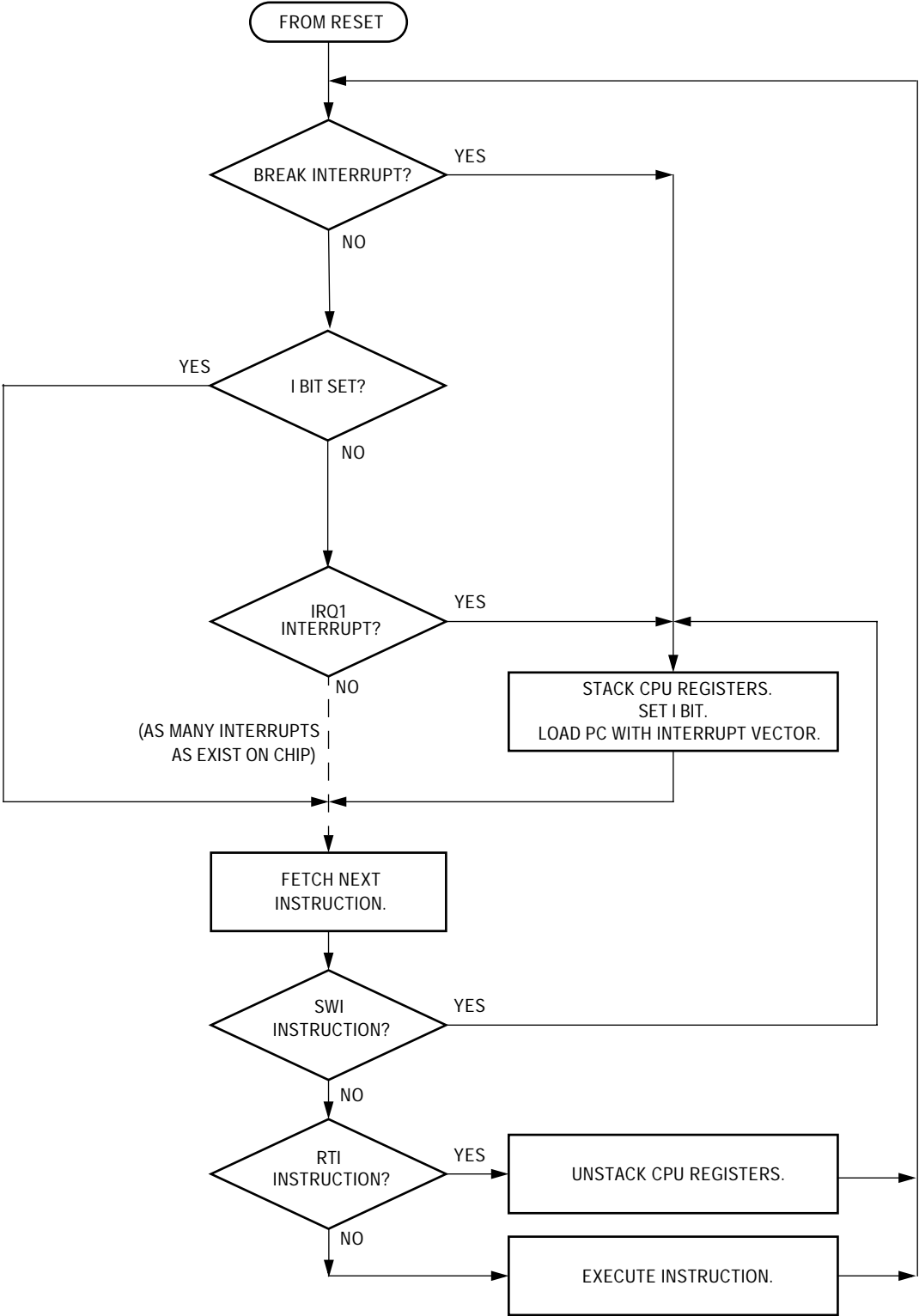
### Interrupts

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. [Figure 8](#) shows interrupt entry timing. [Figure 10](#) shows interrupt recovery timing.

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt can take precedence, regardless of priority, until the latched interrupt is serviced (or the I bit is cleared), see [Figure 9](#).

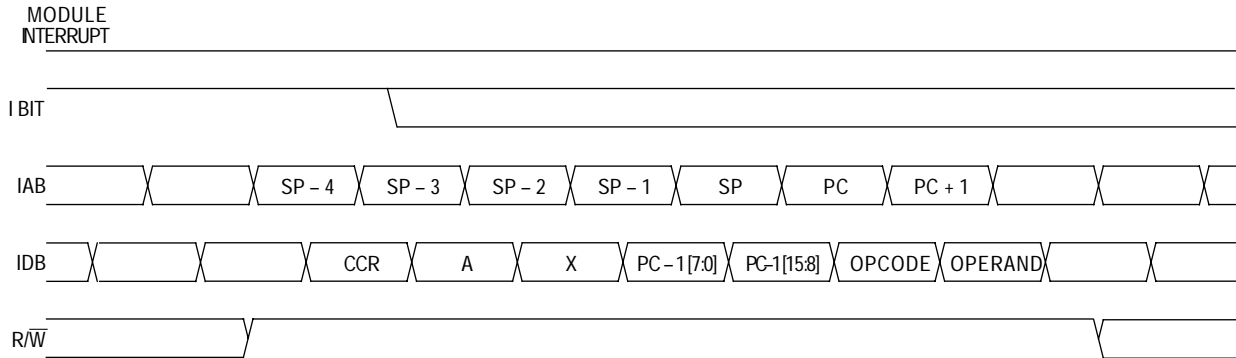


**Figure 8. Interrupt Entry**



**Figure 9. Interrupt Processing**

## System Integration Module (SIM)



**Figure 10. Interrupt Recovery**

### Hardware Interrupts

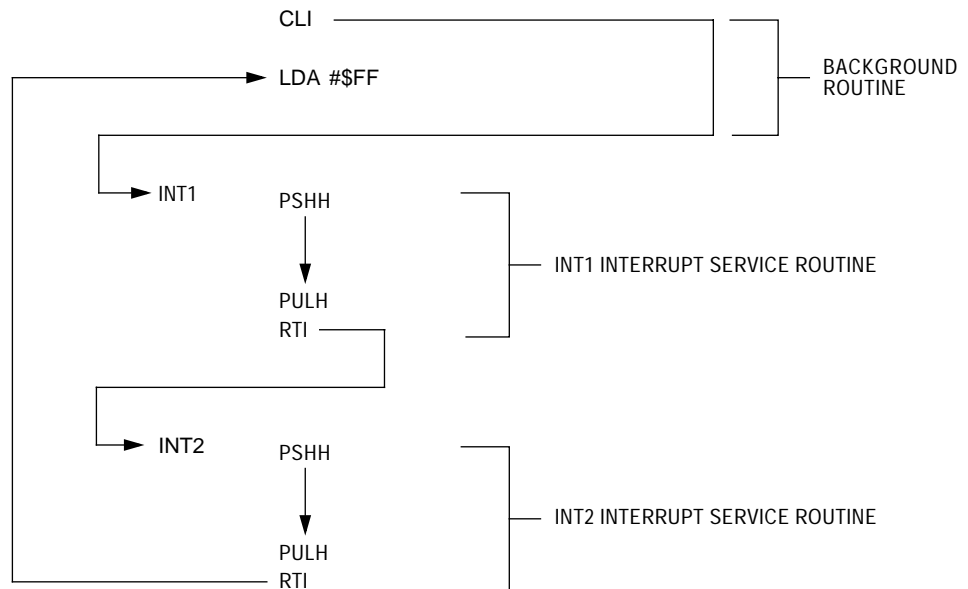
A hardware interrupt does not stop the current instruction. Processing of a hardware interrupt begins after completion of the current instruction. When the current instruction is complete, the SIM checks all pending hardware interrupts. If interrupts are not masked (I bit clear in the condition code register), and if the corresponding interrupt enable bit is set, the SIM proceeds with interrupt processing; otherwise, the next instruction is fetched and executed.

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. [Figure 11](#) demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

The LDA opcode is prefetched by both the INT1 and INT2 RTI instructions. However, in the case of the INT1 RTI prefetch, this is a redundant operation.

**NOTE:** *To maintain compatibility with the M68HC05, M6805 and M146805 Families the H register is not pushed on the stack during interrupt entry. If the interrupt service routine modifies the H register or uses the indexed addressing mode, software should save the H register and then restore it prior to exiting the routine.*





**Figure 11. Interrupt Recognition Example**

### SWI Instruction

The SWI instruction is a non-maskable instruction that causes an interrupt regardless of the state of the interrupt mask (I bit) in the condition code register.

**NOTE:** *A software interrupt pushes PC onto the stack. A software interrupt does **not** push PC – 1, as a hardware interrupt does.*

### Reset

All reset sources always have higher priority than interrupts and cannot be arbitrated.

### Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. See [Break Module](#) on page 161. The SIM puts the CPU into the break state by forcing it to the SWI vector location. Refer to the break interrupt subsection of each module to see how each module is affected by the break state.

## System Integration Module (SIM)

### Status Flag Protection in Break Mode

The SIM controls whether status flags contained in other modules can be cleared during break mode. The user can select whether flags are protected from being cleared by properly initializing the break clear flag enable bit (BCFE) in the SIM break flag control register (SBFCR).

Protecting flags in break mode ensures that set flags will not be cleared while in break mode. This protection allows registers to be freely read and written during break mode without losing status flag information.

Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a two-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first step is accomplished prior to entering break mode. Upon leaving break mode, execution of the second step will clear the flag as normal.

---

---

## Low-Power Modes

Executing the WAIT or STOP instruction puts the MCU in a low power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur.

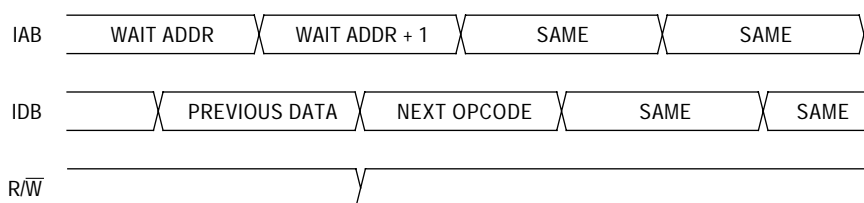
### Wait Mode

In wait mode, the CPU clocks are inactive while one set of peripheral clocks continue to run. [Figure 12](#) shows the timing for wait mode entry.

A module that is active during wait mode can wake up the CPU with an interrupt if the interrupt is enabled. Stacking for the interrupt begins one cycle after the WAIT instruction during which the interrupt occurred. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

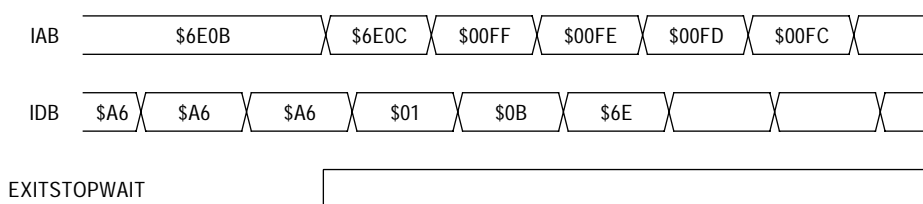
Wait mode can also be exited by a reset or break. A break interrupt during wait mode sets the SIM break wait bit, BW, in the SIM break

status register (SBSR). If the COP disable bit, COPD, in the configuration register is logic 0, then the computer operating properly module (COP) is enabled and remains active in wait mode.



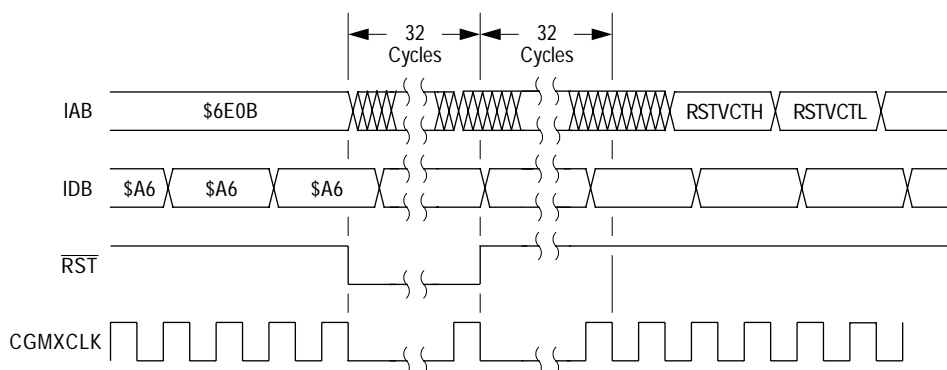
NOTE: Previous data can be operand data or the WAIT opcode, depending on the last instruction.

**Figure 12. Wait Mode Entry Timing**



NOTE: EXITSTOPWAIT =  $\overline{\text{RST}}$  pin OR CPU interrupt OR break interrupt

**Figure 13. Wait Recovery from Interrupt or Break**



**Figure 14. Wait Recovery from Internal Reset**

## System Integration Module (SIM)

### Stop Mode

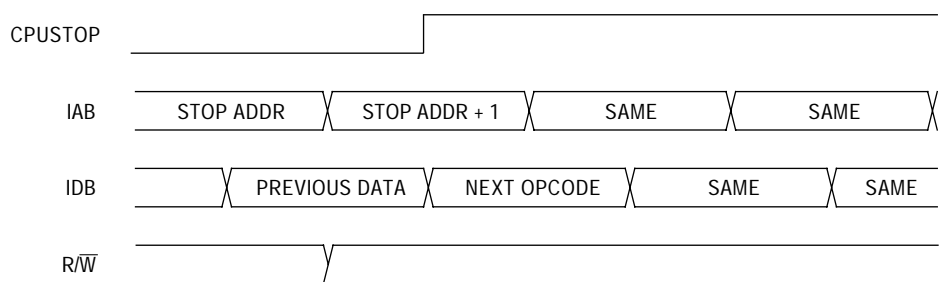
In stop mode, the SIM counter is reset and the system clocks are disabled. An interrupt request from a module can cause an exit from stop mode. Stacking for interrupts begins after the selected stop recovery time has elapsed. Reset also causes an exit from stop mode.

The SIM disables the clock generator module outputs (CGMOUT and CGMXCLK) in stop mode, stopping the CPU and peripherals. Stop recovery time is selectable using the SSREC bit in the configuration register (CONFIG-1). If SSREC is set, stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32. This is ideal for applications using canned oscillators that do not require long startup times from stop mode.

**NOTE:** *External crystal applications should use the full stop recovery time by clearing the SSREC bit.*

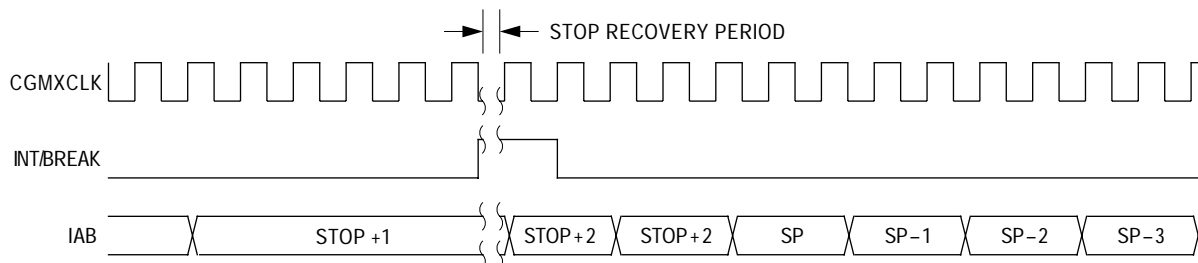
A break module is inactive in Stop mode. The STOP instruction does not affect break module register states.

The SIM counter is held in reset from the execution of the STOP instruction until the beginning of stop recovery. It is then used to time the recovery period. [Figure 15](#) shows stop mode entry timing.



NOTE: Previous data can be operand data or the STOP opcode, depending on the last instruction.

**Figure 15. Stop Mode Entry Timing**



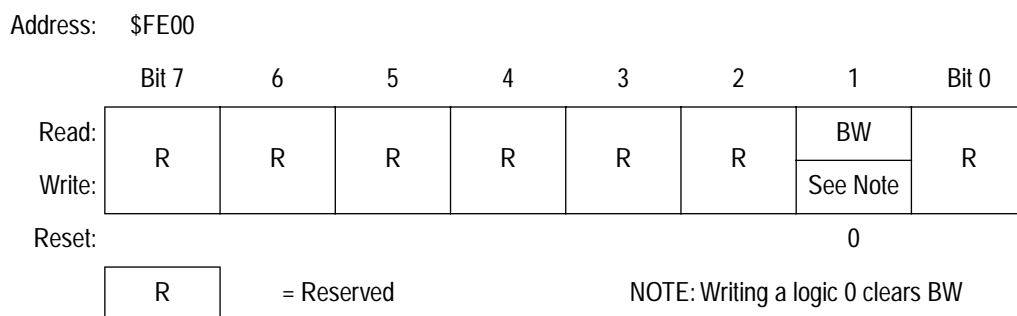
**Figure 16. Stop Mode Recovery from Interrupt or Break**

## SIM Registers

The SIM has three memory mapped registers.

### SIM Break Status Register

The SIM break status register contains a flag to indicate that a break caused an exit from stop or wait mode.



**Figure 17. SIM Break Status Register (SBSR)**

#### BW — SIM Break Wait

This status bit is useful in applications requiring a return to wait mode after exiting from a break interrupt. Clear BW by writing a logic 0 to it. Reset clears BW.

- 1 = Wait mode was exited by break interrupt
- 0 = Wait mode was not exited by break interrupt

## System Integration Module (SIM)

BW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it. The following code is an example of this. Writing zero to the BW bit clears it.

```

; This code works if the H register has been pushed onto the stack in the break
; service routine software. This code should be executed at the end of the
; break service routine software.

HIBYTE EQU 5
LOBYTE EQU 6
; If not BW, do RTI
BRCLR BW,SBSR, RETURN ; See if wait mode or stop mode was exited
; by break.
TST LOBYTE,SP ; If RETURNLO is not zero,
BNE DOLO ; then just decrement low byte.
DEC HIBYTE,SP ; Else deal with high byte, too.
DOLO DEC LOBYTE,SP ; Point to WAIT/STOP opcode.
RETURN PULH ; Restore H register.
RTI


```

### SIM Reset Status Register

This register contains six flags that show the source of the last reset. The status register will automatically clear after reading it. A power-on reset sets the POR bit and clears all other bits in the register.

Address: \$FE01

|        | Bit 7 | 6   | 5   | 4    | 3    | 2 | 1   | Bit 0 |
|--------|-------|-----|-----|------|------|---|-----|-------|
| Read:  | POR   | PIN | COP | ILOP | ILAD | 0 | LVI | 0     |
| Write: |       |     |     |      |      |   |     |       |
| POR:   | 1     | 0   | 0   | 0    | 0    | 0 | 0   | 0     |

 = Unimplemented

**Figure 18. SIM Reset Status Register (SRSR)**

POR — Power-On Reset Bit

1 = Last reset caused by POR circuit

0 = Read of SRSR

**PIN** — External Reset Bit

- 1 = Last reset caused by external reset pin ( $\overline{RST}$ )
- 0 = POR or read of SRSR

**COP** — Computer Operating Properly Reset Bit

- 1 = Last reset caused by COP counter
- 0 = POR or read of SRSR

**ILOP** — Illegal Opcode Reset Bit

- 1 = Last reset caused by an illegal opcode
- 0 = POR or read of SRSR

**ILAD** — Illegal Address Reset Bit (opcode fetches only)

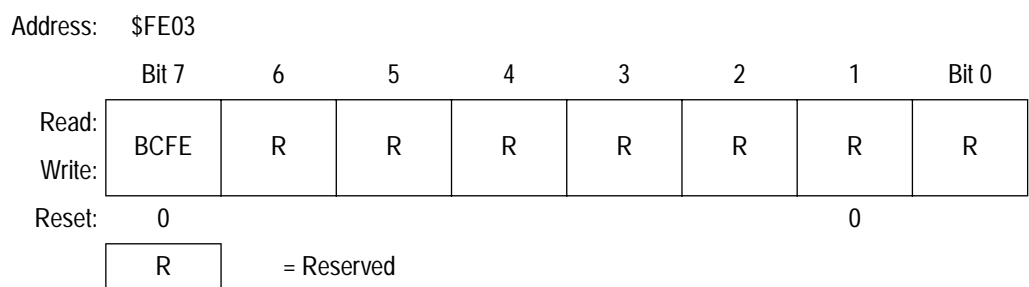
- 1 = Last reset caused by an opcode fetch from an illegal address
- 0 = POR or read of SRSR

**LVI** — Low-Voltage Inhibit Reset Bit

- 1 = Last reset was caused by the LVI circuit
- 0 = POR or read of SRSR

## SIM Break Flag Control Register

The SIM break control register contains a bit that enables software to clear status bits while the MCU is in a break state.



**Figure 19. SIM Break Flag Control Register (SBFCR)**

**BCFE** — Break Clear Flag Enable Bit

This read/write bit enables software to clear status bits by accessing status registers while the MCU is in a break state. To clear status bits during the break state, the BCFE bit must be set.

- 1 = Status bits clearable during break
- 0 = Status bits not clearable during break

## System Integration Module (SIM)



# Clock Generator Module (CGM)

---

---

## Contents

|  |     |
|--|-----|
| Introduction . . . . .                             | 128 |
| Features . . . . .                                 | 128 |
| Functional Description. . . . .                    | 129 |
| Crystal Oscillator Circuit . . . . .               | 129 |
| Phase-Locked Loop Circuit (PLL). . . . .           | 131 |
| Circuits . . . . .                                 | 131 |
| Acquisition and Tracking Modes. . . . .            | 133 |
| Manual and Automatic PLL Bandwidth Modes. . . . .  | 133 |
| Programming the PLL. . . . .                       | 135 |
| Special Programming Exceptions. . . . .            | 137 |
| Base Clock Selector Circuit . . . . .              | 137 |
| CGM External Connections . . . . .                 | 137 |
| I/O Signals. . . . .                               | 139 |
| Crystal Amplifier Input Pin (OSC1) . . . . .       | 139 |
| Crystal Amplifier Output Pin (OSC2). . . . .       | 139 |
| External Filter Capacitor Pin (CGMXFC) . . . . .   | 139 |
| Analog Power Pin (VDDA) . . . . .                  | 140 |
| Oscillator Enable Signal (SIMOSCEN) . . . . .      | 140 |
| Crystal Output Frequency Signal (CGMXCLK). . . . . | 140 |
| CGM Base Clock Output (CGMOUT) . . . . .           | 140 |
| CGM CPU Interrupt (CGMINT). . . . .                | 140 |
| CGM Registers . . . . .                            | 141 |
| PLL Control Register. . . . .                      | 141 |
| PLL Bandwidth Control Register. . . . .            | 143 |
| PLL Programming Register. . . . .                  | 145 |
| Interrupts . . . . .                               | 147 |
| Low-Power Modes . . . . .                          | 147 |
| Wait Mode. . . . .                                 | 147 |
| Stop Mode. . . . .                                 | 148 |
| CGM During Break Interrupts . . . . .              | 148 |
| Acquisition/Lock Time Specifications . . . . .     | 149 |
| Acquisition/Lock Time Definitions. . . . .         | 149 |

## Clock Generator Module (CGM)

|  |     |
|--|-----|
| Parametric Influences on Reaction Time ..... | 150 |
| Choosing a Filter Capacitor .....            | 151 |
| Reaction Time Calculation .....              | 152 |

---

---

### Introduction

The CGM generates the crystal clock signal, CGMXCLK, which operates at the frequency of the crystal. The CGM also generates the base clock signal, CGMOUT, from which the system clocks are derived. CGMOUT is based on either the crystal clock divided by two or the phase-locked loop (PLL) clock, CGMVCLK, divided by two. The PLL is a frequency generator designed for use with 1-MHz to 16-MHz crystals or ceramic resonators. The PLL can generate an 8-MHz bus frequency without using high frequency crystals.

---

---

### Features

Features of the CGM include:

- Phase-Locked Loop with Output Frequency in Integer Multiples of the Crystal Reference
- Programmable Hardware Voltage-Controlled Oscillator (VCO) for Low-Jitter Operation
- Automatic Bandwidth Control Mode for Low-Jitter Operation
- Automatic Frequency Lock Detector
- CPU Interrupt on Entry or Exit from Locked Condition

---

---

## Functional Description

The CGM consists of three major submodules:

- Crystal oscillator circuit — The crystal oscillator circuit generates the constant crystal frequency clock, CGMXCLK.
- Phase-locked loop (PLL) — The PLL generates the programmable VCO frequency clock CGMVCLK.
- Base clock selector circuit — This software-controlled circuit selects either CGMXCLK divided by two or the VCO clock, CGMVCLK, divided by two as the base clock, CGMOUT. The system clocks are derived from CGMOUT.

**Figure 1** shows the structure of the CGM.

### Crystal Oscillator Circuit

The crystal oscillator circuit consists of an inverting amplifier and an external crystal. The OSC1 pin is the input to the amplifier and the OSC2 pin is the output. The SIMOSCEN signal enables the crystal oscillator circuit.

The CGMXCLK signal is the output of the crystal oscillator circuit and runs at a rate equal to the crystal frequency. CGMXCLK is then buffered to produce CGMRCLK, the PLL reference clock.

CGMXCLK can be used by other modules which require precise timing for operation. The duty cycle of CGMXCLK is not guaranteed to be 50% and depends on external factors, including the crystal and related external components.

An externally generated clock also can feed the OSC1 pin of the crystal oscillator circuit. Connect the external clock to the OSC1 pin and let the OSC2 pin float.

# Clock Generator Module (CGM)

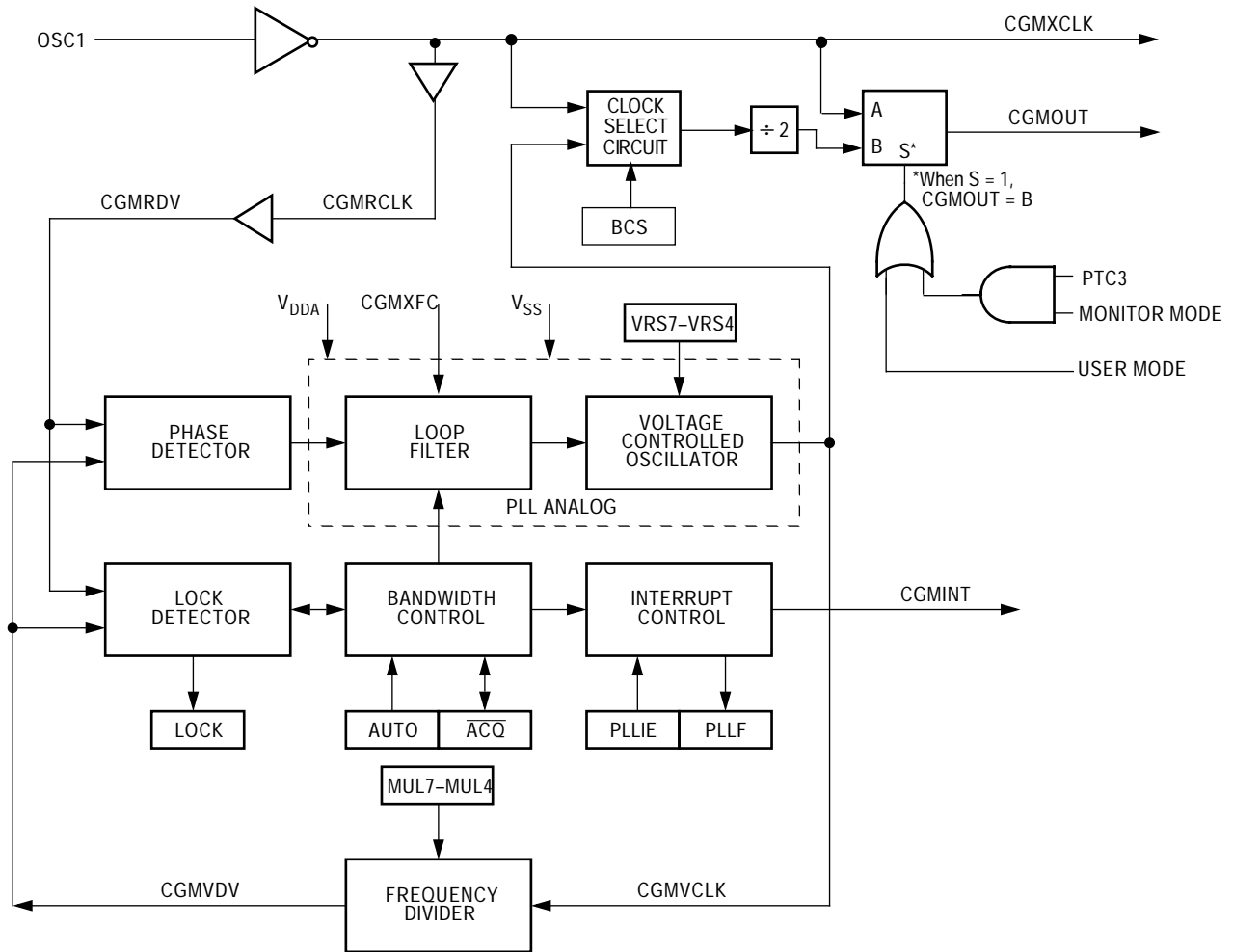



Figure 1. CGM Block Diagram

| Register Name                         | Bit 7  | 6     | 5    | 4                | 3    | 2    | 1    | Bit 0 |
|---------------------------------------|--------|-------|------|------------------|------|------|------|-------|
| PLL Control Register (PCTL)           | Read:  | PLLIE | PLLF | PLLON            | BCS  | 1    | 1    | 1     |
|                                       | Write: |       |      |                  |      |      |      |       |
|                                       | Reset: | 0     | 0    | 1                | 0    | 1    | 1    | 1     |
| PLL Bandwidth Control Register (PBWC) | Read:  | AUTO  | LOCK | $\overline{ACQ}$ | XLD  | 0    | 0    | 0     |
|                                       | Write: |       |      |                  |      |      |      |       |
|                                       | Reset: | 0     | 0    | 0                | 0    | 0    | 0    | 0     |
| PLL Programming Register (PPG)        | Read:  | MUL7  | MUL6 | MUL5             | MUL4 | VRS7 | VRS6 | VRS5  |
|                                       | Write: |       |      |                  |      |      |      |       |
|                                       | Reset: | 0     | 1    | 1                | 0    | 0    | 1    | 1     |

 = Unimplemented

**Figure 2. I/O Register Summary**

**Table 1. I/O Register Address Summary**

| Register | PCTL   | PBWC   | PPG    |
|----------|--------|--------|--------|
| Address  | \$001C | \$001D | \$001E |

### Phase-Locked Loop Circuit (PLL)

The PLL is a frequency generator that can operate in either acquisition mode or tracking mode, depending on the accuracy of the output frequency. The PLL can change between acquisition and tracking modes either automatically or manually.

### Circuits

The PLL consists of these circuits:

- Voltage-controlled oscillator (VCO)
- Modulo VCO frequency divider
- Phase detector
- Loop filter
- Lock detector

## Clock Generator Module (CGM)

The operating range of the VCO is programmable for a wide range of frequencies and for maximum immunity to external noise, including supply and CGMXFC noise. The VCO frequency is bound to a range from roughly one-half to twice the center-of-range frequency,  $f_{VRS}$ . Modulating the voltage on the CGMXFC pin changes the frequency within this range. By design,  $f_{VRS}$  is equal to the nominal center-of-range frequency,  $f_{NOM}$ , (4.9152 MHz) times a linear factor  $L$  or  $(L)f_{NOM}$ .

CGMRCLK is the PLL reference clock, a buffered version of CGMXCLK. CGMRCLK runs at a frequency,  $f_{RCLK}$ , and is fed to the PLL through a buffer. The buffer output is the final reference clock, CGMRDV, running at a frequency  $f_{RDV} = f_{RCLK}$ .

The VCO's output clock, CGMVCLK, running at a frequency  $f_{VCLK}$ , is fed back through a programmable modulo divider. The modulo divider reduces the VCO clock by a factor,  $N$ . The divider's output is the VCO feedback clock, CGMVDV, running at a frequency  $f_{VDV} = f_{VCLK}/N$ . See Programming the PLL for more information.

The phase detector then compares the VCO feedback clock, CGMVDV, with the final reference clock, CGMRDV. A correction pulse is generated based on the phase difference between the two signals. The loop filter then slightly alters the dc voltage on the external capacitor connected to CGMXFC based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, as described in [Acquisition and Tracking Modes](#) on page 133. The value of the external capacitor and the reference frequency determines the speed of the corrections and the stability of the PLL.

The lock detector compares the frequencies of the VCO feedback clock, CGMVDV, and the final reference clock, CGMRDV. Therefore, the speed of the lock detector is directly proportional to the final reference frequency,  $f_{RDV}$ . The circuit determines the mode of the PLL and the lock condition based on this comparison.

### *Acquisition and Tracking Modes*

The PLL filter is manually or automatically configurable into one of two operating modes:

- Acquisition mode — In acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL startup or when the PLL has suffered a severe noise hit and the VCO frequency is far off the desired frequency. When in acquisition mode, the  $\overline{ACQ}$  bit is clear in the PLL bandwidth control register. See [PLL Bandwidth Control Register](#) on page 143.
- Tracking mode — In tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source. See [Base Clock Selector Circuit](#) on page 137. The PLL is automatically in tracking mode when it's not in acquisition mode or when the  $\overline{ACQ}$  bit is set.

### *Manual and Automatic PLL Bandwidth Modes*

The PLL can change the bandwidth or operational mode of the loop filter manually or automatically.

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT. See [PLL Bandwidth Control Register](#) on page 143. If PLL CPU interrupt requests are enabled, the software can wait for a PLL CPU interrupt request and then check the LOCK bit. If CPU interrupts are disabled, software can poll the LOCK bit continuously (during PLL startup, usually) or at periodic intervals. In either case, when the LOCK bit is set, the VCO clock is safe to use as the source for the base clock. See [Base Clock Selector Circuit](#) on page 137. If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application. See [Interrupts](#) on page 147.

These conditions apply when the PLL is in automatic bandwidth control mode:

## Clock Generator Module (CGM)

- The  $\overline{ACQ}$  bit (See PLL Bandwidth Control Register.) is a read-only indicator of the mode of the filter. See [Acquisition and Tracking Modes](#) on page 133.
- The  $\overline{ACQ}$  bit is set when the VCO frequency is within a certain tolerance,  $\Delta_{\text{trk}}$ , and is cleared when the VCO frequency is out of a certain tolerance,  $\Delta_{\text{unt}}$ . See [Electrical Specifications](#) on page 434.
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance,  $\Delta_{\text{Lock}}$ , and is cleared when the VCO frequency is out of a certain tolerance,  $\Delta_{\text{unl}}$ . See [Electrical Specifications](#) on page 434.
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. See [PLL Control Register](#) on page 141.

The PLL also can operate in manual mode (AUTO = 0). Manual mode is used by systems that do not require an indicator of the lock condition for proper operation. Such systems typically operate well below  $f_{\text{busmax}}$  and require fast startup. The following conditions apply when in manual mode:

- $\overline{ACQ}$  is a writable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the  $\overline{ACQ}$  bit must be clear.
- Before entering tracking mode ( $\overline{ACQ} = 1$ ), software must wait a given time,  $t_{\text{acq}}$  (see [Electrical Specifications](#) on page 434), after turning on the PLL by setting PLLON in the PLL control register (PCTL).
- Software must wait a given time,  $t_{\text{al}}$ , after entering tracking mode before selecting the PLL as the clock source to CGMOUT (BCS = 1).
- The LOCK bit is disabled.
- CPU interrupts from the CGM are disabled.



*Programming the PLL*

Use this 9-step procedure to program the PLL. The table below lists the variables used and their meaning.

**Table 2. Variable Definitions**

| Variable             | Definition                         |
|----------------------|------------------------------------|
| $f_{\text{BUSDES}}$  | Desired Bus Clock Frequency        |
| $f_{\text{VCLKDES}}$ | Desired VCO Clock Frequency        |
| $f_{\text{RCLK}}$    | Chosen Reference Crystal Frequency |
| $f_{\text{VCLK}}$    | Calculated VCO Clock Frequency     |
| $f_{\text{BUS}}$     | Calculated Bus Clock Frequency     |
| $f_{\text{NOM}}$     | Nominal VCO Center Frequency       |
| $f_{\text{VRS}}$     | Shifted VCO Center Frequency       |

1. Choose the desired bus frequency,  $f_{\text{BUSDES}}$ .  
Example:  $f_{\text{BUSDES}} = 8 \text{ MHz}$
2. Calculate the desired VCO frequency,  $f_{\text{VCLKDES}}$ .  
$$f_{\text{VCLKDES}} = 4 \times f_{\text{BUSDES}}$$
  
Example:  $f_{\text{VCLKDES}} = 4 \times 8 \text{ MHz} = 32 \text{ MHz}$
3. Using a reference frequency,  $f_{\text{RCLK}}$ , equal to the crystal frequency, calculate the VCO frequency multiplier, N. Round the result to the nearest integer.

$$N = \frac{f_{\text{VCLKDES}}}{f_{\text{RCLK}}}$$

$$\text{Example: } N = \frac{32 \text{ MHz}}{4 \text{ MHz}} = 8$$

4. Calculate the VCO frequency,  $f_{\text{VCLK}}$ .

$$f_{\text{VCLK}} = N \times f_{\text{RCLK}}$$

$$\text{Example: } f_{\text{VCLK}} = 8 \times 4 \text{ MHz} = 32 \text{ MHz}$$

5. Calculate the bus frequency,  $f_{\text{BUS}}$ , and compare  $f_{\text{BUS}}$  with  $f_{\text{BUSDES}}$ .

## Clock Generator Module (CGM)

$$f_{\text{BUS}} = \frac{f_{\text{VCLK}}}{4}$$

$$\text{Example: } f_{\text{BUS}} = \frac{32 \text{ MHz}}{4} = 8 \text{ MHz}$$

6. If the calculated  $f_{\text{bus}}$  is not within the tolerance limits of your application, select another  $f_{\text{BUSDES}}$  or another  $f_{\text{RCLK}}$ .
7. Using the value 4.9152 MHz for  $f_{\text{NOM}}$ , calculate the VCO linear range multiplier, L. The linear range multiplier controls the frequency range of the PLL.

$$L = \text{round}\left(\frac{f_{\text{VCLK}}}{f_{\text{NOM}}}\right)$$

$$\text{Example: } L = \frac{32 \text{ MHz}}{4.9152 \text{ MHz}} = 7$$

8. Calculate the VCO center-of-range frequency,  $f_{\text{VRS}}$ . The center-of-range frequency is the midpoint between the minimum and maximum frequencies attainable by the PLL.

$$f_{\text{VRS}} = L \times f_{\text{NOM}}$$

$$\text{Example: } f_{\text{VRS}} = 7 \times 4.9152 \text{ MHz} = 34.4 \text{ MHz}$$

**NOTE:** For proper operation,  $|f_{\text{VRS}} - f_{\text{VCLK}}| \leq \frac{f_{\text{NOM}}}{2}$ .

*Exceeding the recommended maximum bus frequency or VCO frequency can crash the MCU.*

9. Program the PLL registers accordingly:
  - a. In the upper four bits of the PLL programming register (PPG), program the binary equivalent of N.
  - b. In the lower four bits of the PLL programming register (PPG), program the binary equivalent of L.

### *Special Programming Exceptions*

The programming method described in [Programming the PLL](#) on page 135, does not account for two possible exceptions. A value of 0 for N or L is meaningless when used in the equations given. To account for these exceptions:

- A 0 value for N is interpreted the same as a value of 1.
- A 0 value for L disables the PLL and prevents its selection as the source for the base clock. See [Base Clock Selector Circuit](#) on page 137.

### **Base Clock Selector Circuit**

This circuit is used to select either the crystal clock, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the base clock, CGMOUT. The two input clocks go through a transition control circuit that waits up to three CGMXCLK cycles and three CGMVCLK cycles to change from one clock source to the other. During this time, CGMOUT is held in stasis. The output of the transition control circuit is then divided by two to correct the duty cycle. Therefore, the bus clock frequency, which is one-half of the base clock frequency, is one-fourth the frequency of the selected clock (CGMXCLK or CGMVCLK).

The BCS bit in the PLL control register (PCTL) selects which clock drives CGMOUT. The VCO clock cannot be selected as the base clock source if the PLL is not turned on. The PLL cannot be turned off if the VCO clock is selected. The PLL cannot be turned on or off simultaneously with the selection or deselection of the VCO clock. The VCO clock also cannot be selected as the base clock source if the factor L is programmed to a 0. This value would set up a condition inconsistent with the operation of the PLL, so that the PLL would be disabled and the crystal clock would be forced as the source of the base clock.

### **CGM External Connections**

In its typical configuration, the CGM requires seven external components. Five of these are for the crystal oscillator and two are for the PLL.

The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in [Figure 3](#). [Figure 3](#) shows only the logical

## Clock Generator Module (CGM)

representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

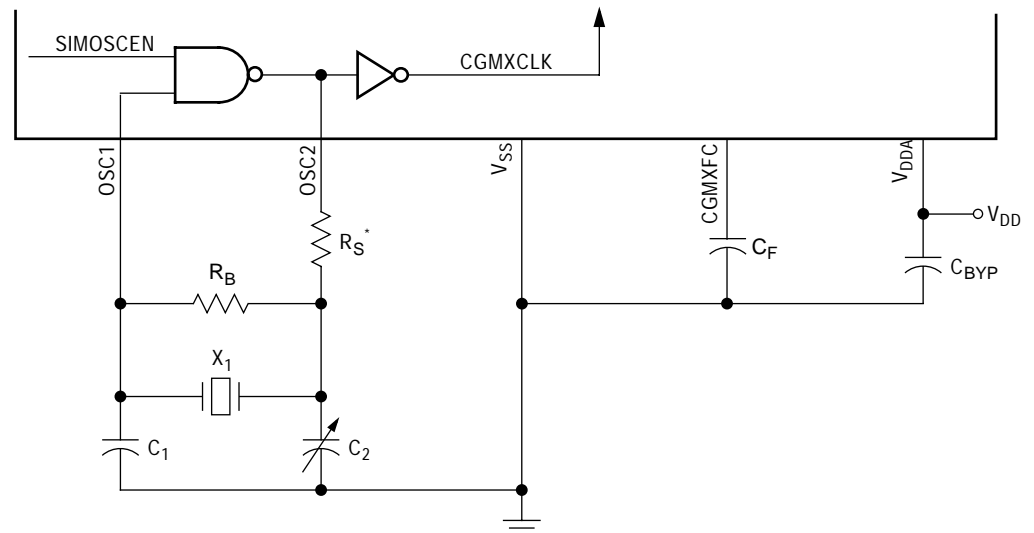
- Crystal,  $X_1$
- Fixed capacitor,  $C_1$
- Tuning capacitor,  $C_2$  (can also be a fixed capacitor)
- Feedback resistor,  $R_B$
- Series resistor,  $R_S$  (optional)

The series resistor ( $R_S$ ) may not be required for all ranges of operation, especially with high-frequency crystals. Refer to the crystal manufacturer's data for more information.

**Figure 3** also shows the external components for the PLL:

- Bypass capacitor,  $C_{BYP}$
- Filter capacitor,  $C_F$

Routing should be done with great care to minimize signal cross talk and noise. (See [Acquisition/Lock Time Specifications](#) on page 149 for routing information and more information on the filter capacitor's value and its effects on PLL performance).



\* $R_S$  can be 0 (shorted) when used with higher-frequency crystals. Refer to manufacturer's data.

**Figure 3. CGM External Connections**

## I/O Signals

The following paragraphs describe the CGM input/output (I/O) signals.

### Crystal Amplifier Input Pin (OSC1)

The OSC1 pin is an input to the crystal oscillator amplifier.

### Crystal Amplifier Output Pin (OSC2)

The OSC2 pin is the output of the crystal oscillator inverting amplifier.

### External Filter Capacitor Pin (CGMXFC)

The CGMXFC pin is required by the loop filter to filter out phase corrections. A small external capacitor is connected to this pin.

**NOTE:** To prevent noise problems,  $C_F$  should be placed as close to the CGMXFC pin as possible with minimum routing distances and no routing of other signals across the  $C_F$  connection.

## Clock Generator Module (CGM)

**Analog Power Pin ( $V_{DDA}$ )**  $V_{DDA}$  is a power pin used by the analog portions of the PLL. Connect the  $V_{DDA}$  pin to the same voltage potential as the  $V_{DD}$  pin.

**NOTE:** Route  $V_{DDA}$  carefully for maximum noise immunity and place bypass capacitors as close as possible to the package.

**Oscillator Enable Signal (SIMOSCEN)** The SIMOSCEN signal enables the oscillator and PLL.

**Crystal Output Frequency Signal (CGMXCLK)** CGMXCLK is the crystal oscillator output signal. It runs at the full speed of the crystal ( $f_{xclk}$ ) and comes directly from the crystal oscillator circuit. [Figure 3](#) shows only the logical relation of CGMXCLK to OSC1 and OSC2 and may not represent the actual circuitry. The duty cycle of CGMXCLK is unknown and may depend on the crystal and other external factors. Also, the frequency and amplitude of CGMXCLK can be unstable at startup.

**CGM Base Clock Output (CGMOUT)** CGMOUT is the clock output of the CGM. This signal is used to generate the MCU clocks. CGMOUT is a 50% duty cycle clock running at twice the bus frequency. CGMOUT is software programmable to be either the oscillator output, CGMXCLK, divided by two or the VCO clock, CGMVCLK, divided by two.

**CGM CPU Interrupt (CGMINT)** CGMINT is the CPU interrupt signal generated by the PLL lock detector.

## CGM Registers

Three registers control and monitor operation of the CGM:


- PLL control register (PCTL)
- PLL bandwidth control register (PBWC)
- PLL programming register (PPG)

### PLL Control Register

The PLL control register contains the interrupt enable and flag bits, the on/off switch, and the base clock selector bit.

Address: \$001C

|        | Bit 7 | 6     | 5      | 4   | 3 | 2 | 1 | Bit 0 |
|--------|-------|-------|--------|-----|---|---|---|-------|
| Read:  | PLLIE | PLL F | PLL ON | BCS | 1 | 1 | 1 | 1     |
| Write: |       |       |        |     |   |   |   |       |
| Reset: | 0     | 0     | 1      | 0   | 1 | 1 | 1 | 1     |

 = Unimplemented

**Figure 4. PLL Control Register (PCTL)**

#### PLLIE — PLL Interrupt Enable Bit

This read/write bit enables the PLL to generate a CPU interrupt request when the LOCK bit toggles, setting the PLL flag, PLLF. When the AUTO bit in the PLL bandwidth control register (PBWC) is clear, PLLIE cannot be written and reads as logic 0. Reset clears the PLLIE bit.

- 1 = PLL CPU interrupt requests enabled
- 0 = PLL CPU interrupt requests disabled

## Clock Generator Module (CGM)

### PLLF — PLL Flag Bit

This read-only bit is set whenever the LOCK bit toggles. PLLF generates a CPU interrupt request if the PLLIE bit also is set. PLLF always reads as logic 0 when the AUTO bit in the PLL bandwidth control register (PBWC) is clear. Clear the PLLF bit by reading the PLL control register. Reset clears the PLLF bit.

1 = Change in lock condition

0 = No change in lock condition

**NOTE:** *Do not inadvertently clear the PLLF bit. Be aware that any read or read-modify-write operation on the PLL control register clears the PLLF bit.*

### PLLON — PLL On Bit

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). See [Base Clock Selector Circuit](#) on page 137. Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on

0 = PLL off

### BCS — Base Clock Select Bit

This read/write bit selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. See [Base Clock Selector Circuit](#) on page 137. Reset and the STOP instruction clear the BCS bit.

1 = CGMVCLK divided by two drives CGMOUT

0 = CGMXCLK divided by two drives CGMOUT

**NOTE:** *PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off*



( $PLLON = 0$ ), selecting  $CGMVCLK$  requires two writes to the PLL control register. See [Base Clock Selector Circuit](#) on page 137.

PCTL3–PCTL0 — Unimplemented

These bits provide no function and always read as logic 1s.


**PLL Bandwidth Control Register**

The PLL bandwidth control register:

- Selects automatic or manual (software-controlled) bandwidth control mode
- Indicates when the PLL is locked
- In automatic bandwidth control mode, indicates when the PLL is in acquisition or tracking mode
- In manual operation, forces the PLL into acquisition or tracking mode

Address: \$001D

|        | Bit 7 | 6    | 5                | 4   | 3 | 2 | 1 | Bit 0 |
|--------|-------|------|------------------|-----|---|---|---|-------|
| Read:  | AUTO  | LOCK | $\overline{ACQ}$ | XLD | 0 | 0 | 0 | 0     |
| Write: |       |      |                  |     |   |   |   |       |
| Reset: | 0     | 0    | 0                | 0   | 0 | 0 | 0 | 0     |

 = Unimplemented

**Figure 5. PLL Bandwidth Control Register (PBWC)**

## Clock Generator Module (CGM)

### AUTO — Automatic Bandwidth Control Bit

This read/write bit selects automatic or manual bandwidth control. When initializing the PLL for manual operation (AUTO = 0), clear the  $\overline{ACQ}$  bit before turning on the PLL. Reset clears the AUTO bit.

1 = Automatic bandwidth control

0 = Manual bandwidth control

### LOCK — Lock Indicator Bit

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock, CGMVCLK, is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as logic 0 and has no meaning. Reset clears the LOCK bit.

1 = VCO frequency correct or locked

0 = VCO frequency incorrect or unlocked

### $\overline{ACQ}$ — Acquisition Mode Bit

When the AUTO bit is set,  $\overline{ACQ}$  is a read-only bit that indicates whether the PLL is in acquisition mode or tracking mode. When the AUTO bit is clear,  $\overline{ACQ}$  is a read/write bit that controls whether the PLL is in acquisition or tracking mode.

In automatic bandwidth control mode (AUTO = 1), the last-written value from manual operation is stored in a temporary location and is recovered when manual operation resumes. Reset clears this bit, enabling acquisition mode.

1 = Tracking mode

0 = Acquisition mode

### XLD — Crystal Loss Detect Bit

When the VCO output, CGMVCLK, is driving CGMOUT, this read/write bit can indicate whether the crystal reference frequency is active or not.

1 = Crystal reference not active

0 = Crystal reference active

To check the status of the crystal reference, do the following:

1. Write a logic 1 to XLD.
2. Wait  $N \times 4$  cycles. N is the VCO frequency multiplier.
3. Read XLD.

The crystal loss detect function works only when the BCS bit is set, selecting CGMVCLK to drive CGMOUT. When BCS is clear, XLD always reads as logic 0.

#### Bits 3–0 — Reserved for Test

These bits enable test functions not available in user mode. To ensure software portability from development systems to user applications, software should write 0s to bits 3–0 when writing to PBWC.

## PLL Programming Register

The PLL programming register contains the programming information for the modulo feedback divider and the programming information for the hardware configuration of the VCO.

Address: \$001E

|        | Bit 7 | 6    | 5    | 4    | 3    | 2    | 1    | Bit 0 |
|--------|-------|------|------|------|------|------|------|-------|
| Read:  | MUL7  | MUL6 | MUL5 | MUL4 | VRS7 | VRS6 | VRS5 | VRS4  |
| Write: |       |      |      |      |      |      |      |       |
| Reset: | 0     | 1    | 1    | 0    | 0    | 1    | 1    | 0     |

**Figure 6. PLL Programming Register (PPG)**

#### MUL7–MUL4 — Multiplier Select Bits

These read/write bits control the modulo feedback divider that selects the VCO frequency multiplier, N. (See [Circuits](#) on page 131 and [Programming the PLL](#) on page 135). A value of \$0 in the multiplier select bits configures the modulo feedback divider the same as a value of \$1. Reset initializes these bits to \$6 to give a default multiply value of 6.

**Table 3. VCO Frequency Multiplier (N) Selection**

| MUL7:MUL6:MUL5:MUL4 | VCO Frequency Multiplier (N) |
|---------------------|------------------------------|
| 0000                | 1                            |
| 0001                | 1                            |
| 0010                | 2                            |
| 0011                | 3                            |
| ↓                   | ↓                            |
| 1101                | 13                           |
| 1110                | 14                           |
| 1111                | 15                           |

**NOTE:** *The multiplier select bits have built-in protection that prevents them from being written when the PLL is on (PLLON = 1).*

#### VRS7–VRS4 — VCO Range Select Bits

These read/write bits control the hardware center-of-range linear multiplier L, which controls the hardware center-of-range frequency,  $f_{VRS}$ . (See [Circuits](#) on page 131, [Programming the PLL](#) on page 135, and [PLL Control Register](#) on page 141.) VRS7–VRS4 cannot be written when the PLLON bit in the PLL control register (PCTL) is set. See [Special Programming Exceptions](#) on page 137. A value of \$0 in the VCO range select bits disables the PLL and clears the BCS bit in the PCTL. (See [Base Clock Selector Circuit](#) on page 137 and [Special Programming Exceptions](#) on page 137 for more information.) Reset initializes the bits to \$6 to give a default range multiply value of 6.

**NOTE:** *The VCO range select bits have built-in protection that prevents them from being written when the PLL is on (PLLON = 1) and prevents selection of the VCO clock as the source of the base clock (BCS = 1) if the VCO range select bits are all clear.*

*The VCO range select bits must be programmed correctly. Incorrect programming can result in failure of the PLL to achieve lock.*

---

---

## Interrupts

When the AUTO bit is set in the PLL bandwidth control register (PBWC), the PLL can generate a CPU interrupt request every time the LOCK bit changes state. The PLLIE bit in the PLL control register (PCTL) enables CPU interrupt requests from the PLL. PLLF, the interrupt flag in the PCTL, becomes set whether CPU interrupt requests are enabled or not. When the AUTO bit is clear, CPU interrupt requests from the PLL are disabled and PLLF reads as logic 0.

Software should read the LOCK bit after a PLL CPU interrupt request to see if the request was due to an entry into lock or an exit from lock. When the PLL enters lock, the VCO clock, CGMVCLK, divided by two can be selected as the CGMOUT source by setting BCS in the PCTL. When the PLL exits lock, the VCO clock frequency is corrupt, and appropriate precautions should be taken. If the application is not frequency sensitive, CPU interrupt requests should be disabled to prevent PLL interrupt service routines from impeding software performance or from exceeding stack limitations.

**NOTE:** *Software can select the CGMVCLK divided by two as the CGMOUT source even if the PLL is not locked (LOCK = 0). Therefore, software should make sure the PLL is locked before setting the BCS bit.*

---

---

## Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

### Wait Mode

The CGM remains active in wait mode. Before entering wait mode, software can disengage and turn off the PLL by clearing the BCS and PLLON bits in the PLL control register (PCTL). Less power-sensitive applications can disengage the PLL without turning it off. Applications that require the PLL to wake the MCU from wait mode also can deselect the PLL output without turning off the PLL.

## Clock Generator Module (CGM)

### Stop Mode

The STOP instruction disables the CGM and holds low all CGM outputs (CGMXCLK, CGMOUT, and CGMINT).

If CGMOUT is being driven by CGMVCLK and a STOP instruction is executed; the PLL will clear the BCS bit in the PLL control register, causing CGMOUT to be driven by CGMXCLK. When the MCU recovers from STOP, the crystal clock divided by two drives CGMOUT and BCS remains clear.

---

---

### CGM During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See [Break Module](#) on page 161.

To allow software to clear status bits during a break interrupt, write a logic 1 to the BCFE bit. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect the PLLF bit during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write the PLL control register during the break state without affecting the PLLF bit.

---

---

## Acquisition/Lock Time Specifications

The acquisition and lock times of the PLL are, in many applications, the most critical PLL design parameters. Proper design and use of the PLL ensures the highest stability and lowest acquisition/lock times.

### Acquisition/Lock Time Definitions

Typical control systems refer to the acquisition time or lock time as the reaction time, within specified tolerances, of the system to a step input. In a PLL, the step input occurs when the PLL is turned on or when it suffers a noise hit. The tolerance is usually specified as a percent of the step input or when the output settles to the desired value plus or minus a percent of the frequency change. Therefore, the reaction time is constant in this definition, regardless of the size of the step input. For example, consider a system with a 5% acquisition time tolerance. If a command instructs the system to change from 0 Hz to 1 MHz, the acquisition time is the time taken for the frequency to reach  $1\text{ MHz} \pm 50\text{ kHz}$ . Fifty kHz = 5% of the 1-MHz step input. If the system is operating at 1 MHz and suffers a  $-100\text{ kHz}$  noise hit, the acquisition time is the time taken to return from 900 kHz to  $1\text{ MHz} \pm 5\text{ kHz}$ . Five kHz = 5% of the 100-kHz step input.

Other systems refer to acquisition and lock times as the time the system takes to reduce the error between the actual output and the desired output to within specified tolerances. Therefore, the acquisition or lock time varies according to the original error in the output. Minor errors may not even be registered. Typical PLL applications prefer to use this definition because the system requires the output frequency to be within a certain tolerance of the desired frequency regardless of the size of the initial error.

The discrepancy in these definitions makes it difficult to specify an acquisition or lock time for a typical PLL. Therefore, the definitions for acquisition and lock times for this module are:

- Acquisition time,  $t_{acq}$ , is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the tracking mode entry tolerance,  $\Delta_{trk}$ . Acquisition time is based on an initial frequency error,

## Clock Generator Module (CGM)

$(f_{\text{des}} - f_{\text{orig}})/f_{\text{des}}$ , of not more than  $\pm 100\%$ . In automatic bandwidth control mode (see [Manual and Automatic PLL Bandwidth Modes](#) on page 133), acquisition time expires when the  $\overline{\text{ACQ}}$  bit becomes set in the PLL bandwidth control register (PBWC).

- Lock time,  $t_{\text{Lock}}$ , is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the lock mode entry tolerance,  $\Delta_{\text{Lock}}$ . Lock time is based on an initial frequency error,  $(f_{\text{des}} - f_{\text{orig}})/f_{\text{des}}$ , of not more than  $\pm 100\%$ . In automatic bandwidth control mode, lock time expires when the LOCK bit becomes set in the PLL bandwidth control register (PBWC). (See [Manual and Automatic PLL Bandwidth Modes](#) on page 133).

Obviously, the acquisition and lock times can vary according to how large the frequency error is and may be shorter or longer in many cases.

### Parametric Influences on Reaction Time

Acquisition and lock times are designed to be as short as possible while still providing the highest possible stability. These reaction times are not constant, however. Many factors directly and indirectly affect the acquisition time.

The most critical parameter which affects the reaction times of the PLL is the reference frequency,  $f_{\text{RDV}}$ . This frequency is the input to the phase detector and controls how often the PLL makes corrections. For stability, the corrections must be small compared to the desired frequency, so several corrections are required to reduce the frequency error. Therefore, the slower the reference the longer it takes to make these corrections. This parameter is also under user control via the choice of crystal frequency  $f_{\text{XCLK}}$ .

Another critical parameter is the external filter capacitor. The PLL modifies the voltage on the VCO by adding or subtracting charge from this capacitor. Therefore, the rate at which the voltage changes for a given frequency error (thus a change in charge) is proportional to the capacitor size. The size of the capacitor also is related to the stability of the PLL. If the capacitor is too small, the PLL cannot make small enough adjustments to the voltage and the system cannot lock. If the capacitor



is too large, the PLL may not be able to adjust the voltage in a reasonable time. See [Choosing a Filter Capacitor](#) on page 151.

Also important is the operating voltage potential applied to  $V_{DDA}$ . The power supply potential alters the characteristics of the PLL. A fixed value is best. Variable supplies, such as batteries, are acceptable if they vary within a known range at very slow speeds. Noise on the power supply is not acceptable, because it causes small frequency errors which continually change the acquisition time of the PLL.

Temperature and processing also can affect acquisition time because the electrical characteristics of the PLL change. The part operates as specified as long as these influences stay within the specified limits. External factors, however, can cause drastic changes in the operation of the PLL. These factors include noise injected into the PLL through the filter capacitor, filter capacitor leakage, stray impedances on the circuit board, and even humidity or circuit board contamination.

## Choosing a Filter Capacitor

As described in [Parametric Influences on Reaction Time](#) on page 150, the external filter capacitor,  $C_F$ , is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency and supply voltage. The value of the capacitor must, therefore, be chosen with supply potential and reference frequency in mind. For proper operation, the external filter capacitor must be chosen according to this equation:

$$C_F = C_{\text{fact}} \left( \frac{V_{DDA}}{f_{rdv}} \right)$$

For acceptable values of  $C_{\text{fact}}$ , (see [Electrical Specifications](#) on page 434). For the value of  $V_{DDA}$ , choose the voltage potential at which the MCU is operating. If the power supply is variable, choose a value near the middle of the range of possible supply values.

This equation does not always yield a commonly available capacitor size, so round to the nearest available size. If the value is between two different sizes, choose the higher value for better stability. Choosing the lower size may seem attractive for acquisition time improvement, but the

## Clock Generator Module (CGM)

PLL may become unstable. Also, always choose a capacitor with a tight tolerance ( $\pm 20\%$  or better) and low dissipation.

### Reaction Time Calculation

The actual acquisition and lock times can be calculated using the equations below. These equations yield nominal values under the following conditions:

- Correct selection of filter capacitor,  $C_F$  (see [Choosing a Filter Capacitor](#) on page 151).
- Room temperature operation
- Negligible external leakage on CGMXFC
- Negligible noise

The K factor in the equations is derived from internal PLL parameters.  $K_{acq}$  is the K factor when the PLL is configured in acquisition mode, and  $K_{trk}$  is the K factor when the PLL is configured in tracking mode. (See [Acquisition and Tracking Modes](#) on page 133).

$$t_{acq} = \left( \frac{V_{DDA}}{f_{RDV}} \right) \left( \frac{8}{K_{ACQ}} \right)$$

$$t_{al} = \left( \frac{V_{DDA}}{f_{RDV}} \right) \left( \frac{4}{K_{TRK}} \right)$$

$$t_{Lock} = t_{ACQ} + t_{AL}$$

Note the inverse proportionality between the lock time and the reference frequency.

In automatic bandwidth control mode, the acquisition and lock times are quantized into units based on the reference frequency. (See [Manual and Automatic PLL Bandwidth Modes](#) on page 133). A certain number of clock cycles,  $n_{ACQ}$ , is required to ascertain that the PLL is within the tracking mode entry tolerance,  $\Delta_{TRK}$ , before exiting acquisition mode. A

certain number of clock cycles,  $n_{\text{TRK}}$ , is required to ascertain that the PLL is within the lock mode entry tolerance,  $\Delta_{\text{Lock}}$ . Therefore, the acquisition time,  $t_{\text{ACQ}}$ , is an integer multiple of  $n_{\text{ACQ}}/f_{\text{RDV}}$ , and the acquisition to lock time,  $t_{\text{AL}}$ , is an integer multiple of  $n_{\text{TRK}}/f_{\text{RDV}}$ . Also, since the average frequency over the entire measurement period must be within the specified tolerance, the total time usually is longer than  $t_{\text{Lock}}$  as calculated above.

In manual mode, it is usually necessary to wait considerably longer than  $t_{\text{Lock}}$  before selecting the PLL clock (see [Base Clock Selector Circuit](#) on page 137), because the factors described in [Parametric Influences on Reaction Time](#) on page 150, may slow the lock time considerably.

## Clock Generator Module (CGM)

# Configuration Register (CONFIG-1)

---

---

## Contents

|                              |     |
|------------------------------|-----|
| Introduction .....           | 155 |
| Functional Description ..... | 156 |

---

---

## Introduction

This section describes the configuration register (CONFIG-1), which contains bits that configure these options:

- Resets caused by the LVI module
- Power to the LVI module
- LVI enabled during stop mode
- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- Computer operating properly module (COP)
- Stop instruction enable/disable.

## Configuration Register (CONFIG-1)

### Functional Description

The configuration register is a write-once register. Out of reset, the configuration register will read the default value. Once the register is written, further writes will have no effect until a reset occurs.

**NOTE:** *If the LVI module and the LVI reset signal are enabled, a reset occurs when  $V_{DD}$  falls to a voltage,  $LVI_{TRIPF}$ , and remains at or below that level for at least nine consecutive CPU cycles. Once an LVI reset occurs, the MCU remains in reset until  $V_{DD}$  rises to a voltage,  $LVI_{TRIPR}$ .*

Address: \$001F

|        | Bit 7   | 6 | 5      | 4      | 3     | 2    | 1    | Bit 0 |
|--------|---------|---|--------|--------|-------|------|------|-------|
| Read:  | LVISTOP | R | LVIRST | LVIPWR | SSREC | COPL | STOP | COPD  |
| Write: |         |   |        |        |       |      |      |       |
| Reset: | 0       | 1 | 1      | 1      | 0     | 0    | 0    | 0     |

|   |
|---|
| R |
|---|

 = Reserved

**Figure 1. Configuration Register (CONFIG-1)**

**LVISTOP** — LVI Stop Mode Enable Bit

LVISTOP enables the LVI module in stop mode. (See [Low-Voltage Inhibit \(LVI\)](#) on page 185).

- 1 = LVI enabled during stop mode
- 0 = LVI disabled during stop mode

**NOTE:** *To have the LVI enabled in stop mode, the LVIPWR must be at a logic 1 and the LVISTOP bit must be at a logic 1. Take note that by enabling the LVI in stop mode, the stop  $I_{DD}$  current will be higher.*

**LVIRST** — LVI Reset Enable Bit

LVIRST enables the reset signal from the LVI module. (See [Low-Voltage Inhibit \(LVI\)](#) on page 185).

- 1 = LVI module resets enabled
- 0 = LVI module resets disabled

LVIPWR — LVI Power Enable Bit

LVIPWR enables the LVI module. (See [Low-Voltage Inhibit \(LVI\)](#) on page 185).

- 1 = LVI module power enabled
- 0 = LVI module power disabled

SSREC — Short Stop Recovery Bit

SSREC enables the CPU to exit stop mode with a delay of 32 CGMXCLK cycles instead of a 4096-CGMXCLK cycle delay. (See [Stop Mode](#) on page 122).

- 1 = Stop mode recovery after 32 CGMXCLK cycles
- 0 = Stop mode recovery after 4096 CGMXCLK cycles

**NOTE:** *If using an external crystal oscillator, do not set the SSREC bit.*

COPL — COP Long Timeout

COPL enables the shorter COP timeout period. (See [Computer Operating Properly Module \(COP\)](#) on page 179).

- 1 = COP timeout period is  $2^{13} - 2^4$  CGMXCLK cycles
- 0 = COP timeout period is  $2^{18} - 2^4$  CGMXCLK cycles

STOP — STOP Instruction Enable Bit

STOP enables the STOP instruction.

- 1 = STOP instruction enabled
- 0 = STOP instruction treated as illegal opcode

COPD — COP Disable Bit

COPD disables the COP module. (See [Computer Operating Properly Module \(COP\)](#) on page 179).

- 1 = COP module disabled
- 0 = COP module enabled

**Extra care should be exercised when using this emulation part for development of code to be run in ROM AZ, AB or AS parts that the options selected by setting the CONFIG-1 register match exactly the options selected on any ROM code request submitted. The enable/disable logic is not necessarily identical in all parts of the AS and AZ families. If in doubt, check with your local field applications representative.**

## Configuration Register (CONFIG-1)



# Configuration Register (CONFIG-2)

---

---

## Contents

|  |                     |
|--|---------------------|
| <a href="#">Introduction . . . . .</a>           | <a href="#">159</a> |
| <a href="#">Functional Description . . . . .</a> | <a href="#">159</a> |

---

---

## Introduction

This section describes the configuration register (CONFIG-2). This register contains bits that configure these options:

- Configures either the MC68HC08AZxx emulator or the MC68HC08ASxx emulator
- Disables the CAN module

---

---

## Functional Description

The configuration register is a write-once register. Out of reset, the configuration register will read the default. Once the register is written, further writes will have no effect until a reset occurs.

Address: \$FE09

|        | Bit 7 | 6 | 5 | 4      | 3 | 2 | 1 | Bit 0 |
|--------|-------|---|---|--------|---|---|---|-------|
| Read:  | 0     | 0 | 0 | MSCAND | 0 | 0 | 0 | AZxx  |
| Write: |       |   |   |        |   |   |   |       |
| Reset: | 0     | 0 | 0 | 1      | 0 | 0 | 0 | 0     |

**Figure 1. Configuration Register (CONFIG-2)**

## Configuration Register (CONFIG-2)

MSCAND — MSCAN Disable Bit

MSCAND disables the MSCAN module. (See [MSCAN Controller \(MSCAN08\)](#) on page 331).

1 = MSCAN module disabled

0 = MSCAN Module enabled

AZxx — AZxx Emulator Enable Bit

AZxx enables the MC68HC08AZxx emulator configuration. This bit will be 0 out of reset.

1 = MC68HC08AZxx emulator enabled

0 = MC68HC08ASxx emulator enabled

**NOTE:** *AZxx bit is reset by a POWER-ON-RESET only.*

# Break Module

---

---

## Contents

|   |     |
|---|-----|
| Introduction .....                            | 161 |
| Features .....                                | 161 |
| Functional Description .....                  | 162 |
| Flag Protection During Break Interrupts ..... | 163 |
| CPU During Break Interrupts .....             | 163 |
| TIM During Break Interrupts .....             | 164 |
| COP During Break Interrupts .....             | 164 |
| Low-Power Modes .....                         | 164 |
| Wait Mode .....                               | 164 |
| Stop Mode .....                               | 164 |
| Break Module Registers .....                  | 165 |
| Break Status and Control Register .....       | 165 |
| Break Address Registers .....                 | 166 |

---

---

## Introduction

The break module can generate a break interrupt that stops normal program flow at a defined address to enter a background program.

---

---

## Features

- Accessible I/O Registers during Break Interrupts
- CPU-Generated Break Interrupts
- Software-Generated Break Interrupts
- COP Disabling during Break Interrupts

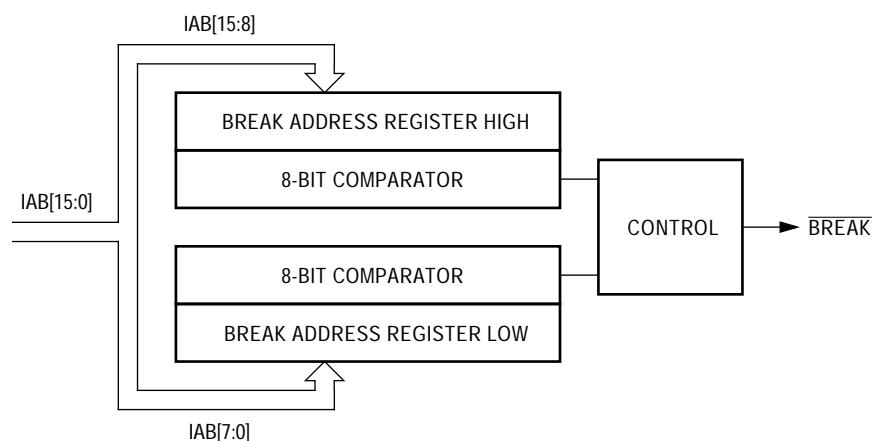
### Functional Description

When the internal address bus matches the value written in the break address registers, the break module issues a breakpoint signal to the CPU. The CPU then loads the instruction register with a software interrupt instruction (SWI) after completion of the current CPU instruction. The program counter vectors to \$FFFC and \$FFFD (\$FEFC and \$FEFD in monitor mode).

The following events can cause a break interrupt to occur:

- A CPU-generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a logic 1 to the BRKA bit in the break status and control register.

When a CPU-generated address matches the contents of the break address registers, the break interrupt begins after the CPU completes its current instruction. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. [Figure 1](#) shows the structure of the break module.



**Figure 1. Break Module Block Diagram**

| Register Name                            | Bit 7  | 6      | 5    | 4  | 3  | 2  | 1  | Bit 0 |       |
|--|--------|--------|------|----|----|----|----|-------|-------|
| Break Address Register High (BRKH)       | Read:  | Bit 15 | 14   | 13 | 12 | 11 | 10 | 9     | Bit 8 |
|  | Write: |        |      |    |    |    |    |       |       |
|  | Reset: | 0      | 0    | 0  | 0  | 0  | 0  | 0     | 0     |
| Break Address Register Low (BRKL)        | Read:  | Bit 7  | 6    | 5  | 4  | 3  | 2  | 1     | Bit 0 |
|  | Write: |        |      |    |    |    |    |       |       |
|  | Reset: | 0      | 0    | 0  | 0  | 0  | 0  | 0     | 0     |
| Break Status and Control Register (BSCR) | Read:  | BRKE   | BRKA | 0  | 0  | 0  | 0  | 0     | 0     |
|  | Write: |        |      |    |    |    |    |       |       |
|  | Reset: | 0      | 0    | 0  | 0  | 0  | 0  | 0     | 0     |

= Unimplemented    R = Reserved

**Figure 2. I/O Register Summary**

**Table 1. I/O Register Address Summary**

|                 |        |        |        |
|-----------------|--------|--------|--------|
| <b>Register</b> | BRKH   | BRKL   | BSCR   |
| <b>Address</b>  | \$FE0C | \$FE0D | \$FE0E |

### Flag Protection During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state.

### CPU During Break Interrupts

The CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with \$FFFC:\$FFFD (\$FEFC:\$FEFD in monitor mode)

The break interrupt begins after completion of the CPU instruction in progress. If the break address register match occurs on the last cycle of a CPU instruction, the break interrupt begins immediately.

## Break Module

|                                    |  |
|------------------------------------|--|
| <b>TIM During Break Interrupts</b> | A break interrupt stops the timer counter.   |
| <b>COP During Break Interrupts</b> | The COP is disabled during a break interrupt when $V_{Hi}$ is present on the $\overline{RST}$ pin. |

---

---

### Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

**Wait Mode** If enabled, the break module is active in wait mode. The SIM break wait bit (BW) in the SIM break status register indicates whether wait was exited by a break interrupt. If so, the user can modify the return address on the stack by subtracting one from it. (See [SIM Break Status Register](#) on page 123).

**Stop Mode** The break module is inactive in stop mode. The STOP instruction does not affect break module register states.

## Break Module Registers

These registers control and monitor operation of the break module:

- Break address register high (BRKH)
- Break address register low (BRKL)
- Break status and control register (BSCR)

### Break Status and Control Register

The break status and control register contains break module enable and status bits.

Address: \$FE0E

|        | Bit 7 | 6    | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|--------|-------|------|---|---|---|---|---|-------|
| Read:  | BRKE  | BRKA | 0 | 0 | 0 | 0 | 0 | 0     |
| Write: |       |      |   |   |   |   |   |       |
| Reset: | 0     | 0    | 0 | 0 | 0 | 0 | 0 | 0     |

= Unimplemented

**Figure 3. Break Status and Control Register (BSCR)**

#### BRKE — Break Enable Bit

This read/write bit enables breaks on break address register matches. Clear BRKE by writing a logic 0 to bit 7. Reset clears the BRKE bit.

- 1 = Breaks enabled on 16-bit address match
- 0 = Breaks disabled on 16-bit address match

#### BRKA — Break Active Bit

This read/write status and control bit is set when a break address match occurs. Writing a logic 1 to BRKA generates a break interrupt. Clear BRKA by writing a logic 0 to it before exiting the break routine. Reset clears the BRKA bit.

- 1 = (When read) Break address match
- 0 = (When read) No break address match

## Break Module

### Break Address Registers

The break address registers contain the high and low bytes of the desired breakpoint address. Reset clears the break address registers.

|           |        |        |    |    |    |    |   |       |
|-----------|--------|--------|----|----|----|----|---|-------|
| Register: | BRKH   | BRKL   |    |    |    |    |   |       |
| Address:  | \$FE0C | \$FE0D |    |    |    |    |   |       |
|           | Bit 7  | 6      | 5  | 4  | 3  | 2  | 1 | Bit 0 |
| Read:     | Bit 15 | 14     | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| Write:    |        |        |    |    |    |    |   |       |
| Reset:    | 0      | 0      | 0  | 0  | 0  | 0  | 0 | 0     |
| Read:     | Bit 7  | 6      | 5  | 4  | 3  | 2  | 1 | Bit 0 |
| Write:    |        |        |    |    |    |    |   |       |
| Reset:    | 0      | 0      | 0  | 0  | 0  | 0  | 0 | 0     |

**Figure 4. Break Address Registers (BRKH and BRKL)**



# Monitor ROM (MON)

---

---

## Contents

|                                  |     |
|----------------------------------|-----|
| Introduction . . . . .           | 167 |
| Features . . . . .               | 168 |
| Functional Description . . . . . | 168 |
| Entering Monitor Mode . . . . .  | 170 |
| Data Format . . . . .            | 171 |
| Echoing . . . . .                | 172 |
| Break Signal . . . . .           | 172 |
| Commands . . . . .               | 173 |
| Baud Rate . . . . .              | 176 |
| Security . . . . .               | 177 |

---

---

## Introduction

This section describes the monitor ROM (MON). The monitor ROM allows complete testing of the MCU through a single-wire interface with a host computer.

---

---

### Features

Features of the monitor ROM include:

- Normal User-Mode Pin Functionality
- One Pin Dedicated to Serial Communication between Monitor ROM and Host Computer
- Standard Mark/Space Non-Return-to-Zero (NRZ) Communication with Host Computer
- Up to 28.8 kBaud Communication with Host Computer
- Execution of Code in RAM or FLASH
- FLASH Security
- FLASH Programming

---

---

### Functional Description

Monitor ROM receives and executes commands from a host computer. [Figure 1](#) shows a sample circuit used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

While simple monitor commands can access any memory address, the MC68HC08AZ60 has a FLASH security feature to prevent external viewing of the contents of FLASH. Proper procedures must be followed to verify FLASH content. Access to the FLASH is denied to unauthorized users of customer specified software (see [Security](#) on page 177).

In monitor mode, the MCU can execute host-computer code in RAM while all MCU pins except PTA0 retain normal operating mode functions. All communication between the host computer and the MCU is through the PTA0 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.

Monitor ROM (MON)  
Functional Description

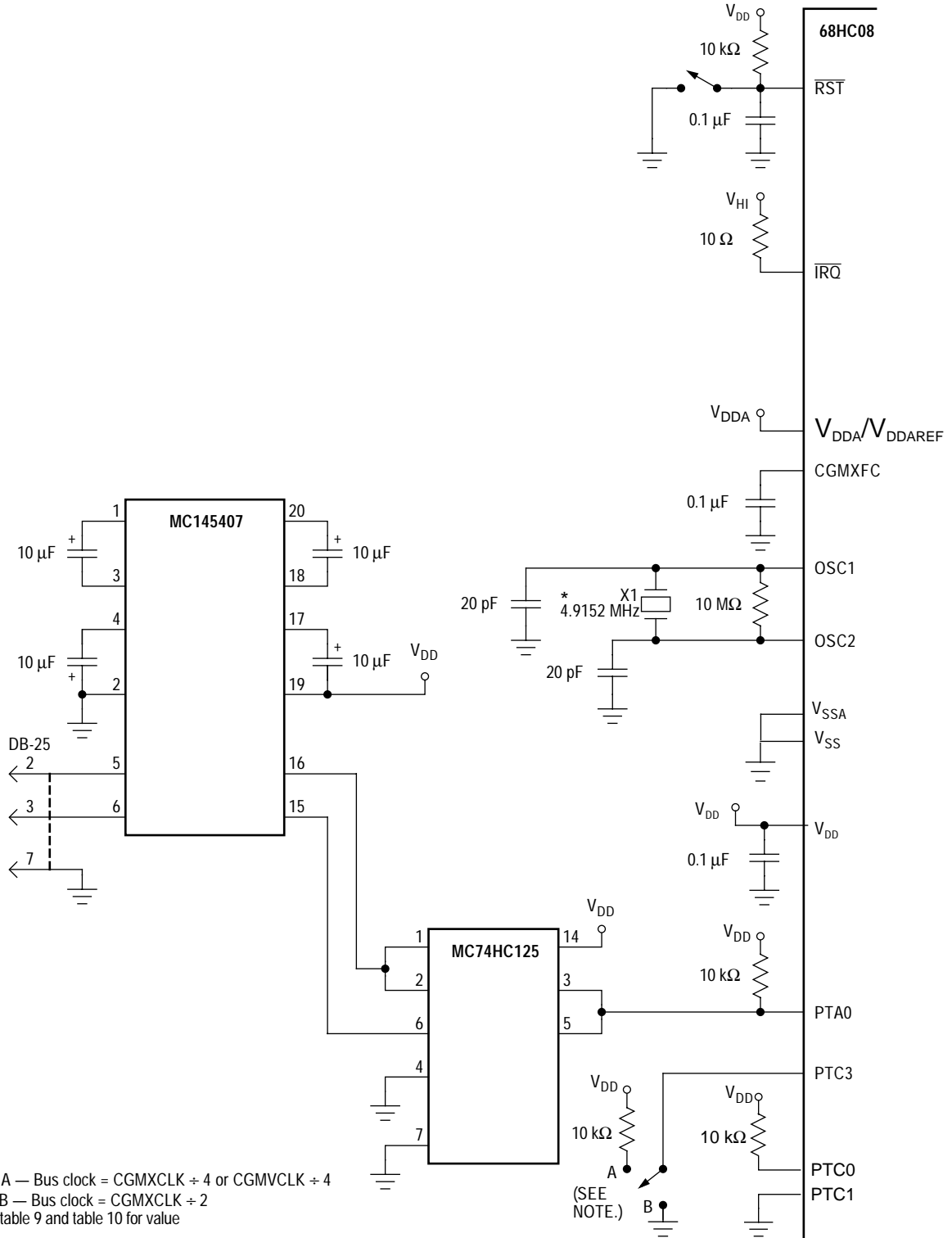


Figure 1. Monitor Mode Circuit

## Monitor ROM (MON)

### Entering Monitor Mode

**Table 1** shows the pin conditions for entering monitor mode.

**Table 1. Mode Selection**

| IRQ Pin        | PTC0 Pin | PTC1 Pin | PTA0 Pin | PTC3 Pin | Mode    | CGMOUT                                     | Bus Frequency      |
|----------------|----------|----------|----------|----------|---------|--|--------------------|
| $V_{HI}^{(1)}$ | 1        | 0        | 1        | 1        | Monitor | $\frac{CGMXCLK}{2}$ or $\frac{CGMVCLK}{2}$ | $\frac{CGMOUT}{2}$ |
| $V_{HI}^{(1)}$ | 1        | 0        | 1        | 0        | Monitor | CGMXCLK                                    | $\frac{CGMOUT}{2}$ |

1. For  $V_{HI}$ , see [5.0 Volt DC Electrical Characteristics](#) on page 436, and [Maximum Ratings](#) on page 434.

Enter monitor mode by either

- Executing a software interrupt instruction (SWI) or
- Applying a logic 0 and then a logic 1 to the  $\overline{RST}$  pin.

Once out of reset, the MCU waits for the host to send eight security bytes (see [Security](#) on page 177). After the security bytes, the MCU sends a break signal (10 consecutive logic 0s) to the host computer, indicating that it is ready to receive a command.

Monitor mode uses alternate vectors for reset, SWI, and break interrupt. The alternate vectors are in the \$FE page instead of the \$FF page and allow code execution from the internal monitor firmware instead of user code. The COP module is disabled in monitor mode as long as  $V_{HI}$  (see [5.0 Volt DC Electrical Characteristics](#) on page 436), is applied to either the  $\overline{IRQ}$  pin or the RESET pin. (See [System Integration Module \(SIM\)](#) on page 105 for more information on modes of operation).

**NOTE:** *Holding the PTC3 pin low when entering monitor mode causes a bypass of a divide-by-two stage at the oscillator. The CGMOUT frequency is equal to the CGMXCLK frequency, and the OSC1 input directly generates internal bus clocks. In this case, the OSC1 signal must have a 50% duty cycle at maximum bus frequency.*

**Table 2** is a summary of the differences between user mode and monitor mode.

**Table 2. Mode Differences**

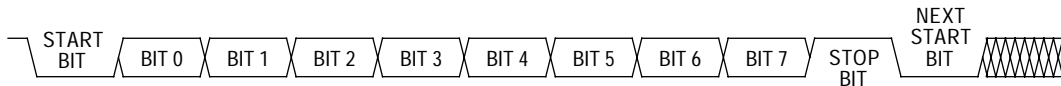
| Modes   | Functions               |                   |                  |                   |                  |                 |                |
|---------|-------------------------|-------------------|------------------|-------------------|------------------|-----------------|----------------|
|         | COP                     | Reset Vector High | Reset Vector Low | Break Vector High | Break Vector Low | SWI Vector High | SWI Vector Low |
| User    | Enabled                 | \$FFFE            | \$FFFF           | \$FFFC            | \$FFFD           | \$FFFC          | \$FFFD         |
| Monitor | Disabled <sup>(1)</sup> | \$FEFE            | \$FEFF           | \$FEFC            | \$FEFD           | \$FEFC          | \$FEFD         |

1. If the high voltage ( $V_{HI}$ ) is removed from the  $IRQ1/V_{PP}$  pin while in monitor mode, the SIM asserts its COP enable output. The COP is a mask option enabled or disabled by the COPD bit in the configuration register. (see [5.0 Volt DC Electrical Characteristics](#) on page 436).

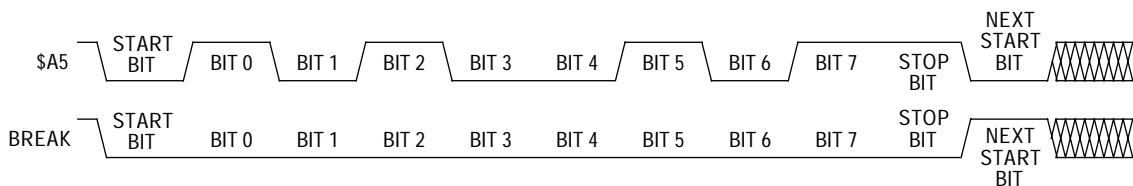
**Data Format**

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. (See [Figure 2](#) and [Figure 3](#).)

The data transmit and receive rate can be anywhere from 4800 baud to 28.8 kBaud. Transmit and receive baud rates must be identical.



**Figure 2. Monitor Data Format**



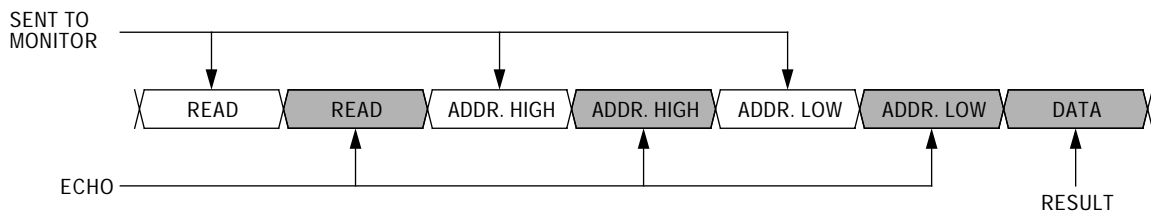
**Figure 3. Sample Monitor Waveforms**

## Monitor ROM (MON)

### Echoing

As shown in [Figure 4](#), the monitor ROM immediately echoes each received byte back to the PTA0 pin for error checking.

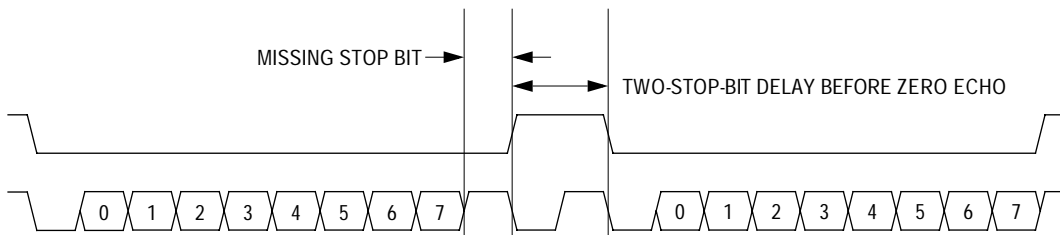
Any result of a command appears after the echo of the last byte of the command.



**Figure 4. Read Transaction**

### Break Signal

A start bit followed by nine low bits is a break signal. (See [Figure 5](#)). When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits before echoing the break signal.



**Figure 5. Break Transaction**

## Commands

The monitor ROM uses these commands:

- READ, read memory
- WRITE, write memory
- IREAD, indexed read
- IWRITE, indexed write
- READSP, read stack pointer
- RUN, run user program

A sequence of IREAD or IWRITE commands can access a block of memory sequentially over the full 64-Kbyte memory map.

**Table 3. READ (Read Memory) Command**

|  |  |
|--|--|
| Description  | Read byte from memory                                |
| Operand  | Specifies 2-byte address in high byte:low byte order |
| Data Returned  | Returns contents of specified address                |
| Opcode   | \$4A   |
| Command Sequence   |  |
| <p>The diagram illustrates the command sequence for the READ command. It consists of seven data bytes: READ, READ, ADDR. HIGH, ADDR. HIGH, ADDR. LOW, ADDR. LOW, and DATA. Arrows indicate the flow of data: the first two 'READ' bytes are sent to the monitor; the next two 'ADDR. HIGH' bytes are echoed back; the next two 'ADDR. LOW' bytes are also echoed back; and finally, the 'DATA' byte is the result of the read operation.</p> |  |

## Monitor ROM (MON)

**Table 4. WRITE (Write Memory) Command**

|   |  |
|---|--|
| Description   | Write byte to memory   |
| Operand   | Specifies 2-byte address in high byte:low byte order; low byte followed by data byte |
| Data Returned   | None   |
| Opcode  | \$49   |
| Command Sequence  |  |
| <p>The diagram illustrates the command sequence for the WRITE command. It consists of eight bytes: WRITE, WRITE, ADDR. HIGH, ADDR. HIGH, ADDR. LOW, ADDR. LOW, DATA, and DATA. Arrows labeled 'SENT TO MONITOR' point to the first two WRITE bytes and the two ADDR. HIGH bytes. Arrows labeled 'ECHO' point to the two ADDR. LOW bytes and the two DATA bytes. The second WRITE byte, both ADDR. HIGH bytes, both ADDR. LOW bytes, and the second DATA byte are shaded gray.</p> |  |

**Table 5. IREAD (Indexed Read) Command**

|  |  |
|--|--|
| Description  | Read next 2 bytes in memory from last address accessed |
| Operand  | Specifies 2-byte address in high byte:low byte order   |
| Data Returned  | Returns contents of next two addresses                 |
| Opcode   | \$1A   |
| Command Sequence   |  |
| <p>The diagram illustrates the command sequence for the IREAD command. It consists of four bytes: IREAD, IREAD, DATA, and DATA. An arrow labeled 'SENT TO MONITOR' points to the first IREAD byte. Arrows labeled 'ECHO' point to the second IREAD byte and the first DATA byte. An arrow labeled 'RESULT' points to the second DATA byte.</p> |  |



**Table 6. IWRITE (Indexed Write) Command**

|                  |                                    |
|------------------|------------------------------------|
| Description      | Write to last address accessed + 1 |
| Operand          | Specifies single data byte         |
| Data Returned    | None                               |
| Opcode           | \$19                               |
| Command Sequence |                                    |

The diagram illustrates the command sequence for the IWRITE command. It consists of four data bytes: IWRITE, IWRITE, DATA, and DATA. The first IWRITE byte is labeled 'SENT TO MONITOR'. The second IWRITE byte and both DATA bytes are labeled 'ECHO'.

**Table 7. READSP (Read Stack Pointer) Command**

|                  |   |
|------------------|---|
| Description      | Reads stack pointer                               |
| Operand          | None  |
| Data Returned    | Returns stack pointer in high byte:low byte order |
| Opcode           | \$0C  |
| Command Sequence |   |

The diagram illustrates the command sequence for the READSP command. It consists of four data bytes: READSP, READSP, SP HIGH, and SP LOW. The first READSP byte is labeled 'SENT TO MONITOR'. The second READSP byte is labeled 'ECHO'. The SP HIGH and SP LOW bytes are labeled 'RESULT'.

## Monitor ROM (MON)

**Table 8. RUN (Run User Program) Command**

|                  |                          |
|------------------|--------------------------|
| Description      | Executes RTI instruction |
| Operand          | None                     |
| Data Returned    | None                     |
| Opcode           | \$28                     |
| Command Sequence |                          |
|                  |                          |

### Baud Rate

With a 4.9152-MHz crystal and the PTC3 pin at logic 1 during reset, data is transferred between the monitor and host at 4800 baud. If the PTC3 pin is at logic 0 during reset, the monitor baud rate is 9600. When the CGM output, CGMOUT, is driven by the PLL, the baud rate is determined by the MUL[7:4] bits in the PLL programming register (PPG). (See [Clock Generator Module \(CGM\)](#) on page 127).

**Table 9. Monitor Baud Rate Selection**

| Monitor Baud Rate | VCO Frequency Multiplier (N) |      |        |        |        |        |
|-------------------|------------------------------|------|--------|--------|--------|--------|
|                   | 1                            | 2    | 3      | 4      | 5      | 6      |
| 4.9152 MHz        | 4800                         | 9600 | 14,400 | 19,200 | 24,000 | 28,800 |
| 4.194 MHz         | 4096                         | 8192 | 12,288 | 16,384 | 20,480 | 24,576 |

Later revisions feature a monitor mode which is optimised to operate with either a 4.1952MHz crystal clock source (or multiples of 4.1952MHz) or a 4MHz crystal (or multiples of 4MHz). This supports designs which use the MSCAN module, which is generally clocked from a 4MHz, 8MHz or 16MHz crystal. The table below outlines the available baud rates for a range of crystals and how they can match to a PC baud rate.

**Table 10**

| Clock freq | Baud rate |          | Closest PC baud PC |        | Error % |        |
|------------|-----------|----------|--------------------|--------|---------|--------|
|            | PTC3=0    | PTC3=1   | PTC3=0             | PTC3=1 | PTC3=0  | PTC3=1 |
| 32kHz      | 57.97     | 28.98    | 57.6               | 28.8   | 0.64    | 0.63   |
| 1MHz       | 1811.59   | 905.80   | 1800               | 900    | 0.64    | 0.64   |
| 2MHz       | 3623.19   | 1811.59  | 3600               | 1800   | 0.64    | 0.64   |
| 4MHz       | 7246.37   | 3623.19  | 7200               | 3600   | 0.64    | 0.64   |
| 4.194MHz   | 7597.83   | 3798.91  | 7680               | 3840   | 1.08    | 1.08   |
| 4.9152MHz  | 8904.35   | 4452.17  | 8861               | 4430   | 0.49    | 0.50   |
| 8MHz       | 14492.72  | 7246.37  | 14400              | 7200   | 0.64    | 0.64   |
| 16MHz      | 28985.51  | 14492.75 | 28800              | 14400  | 0.64    | 0.64   |

**Care should be taken when setting the baud rate since incorrect baud rate setting can result in communications failure.**

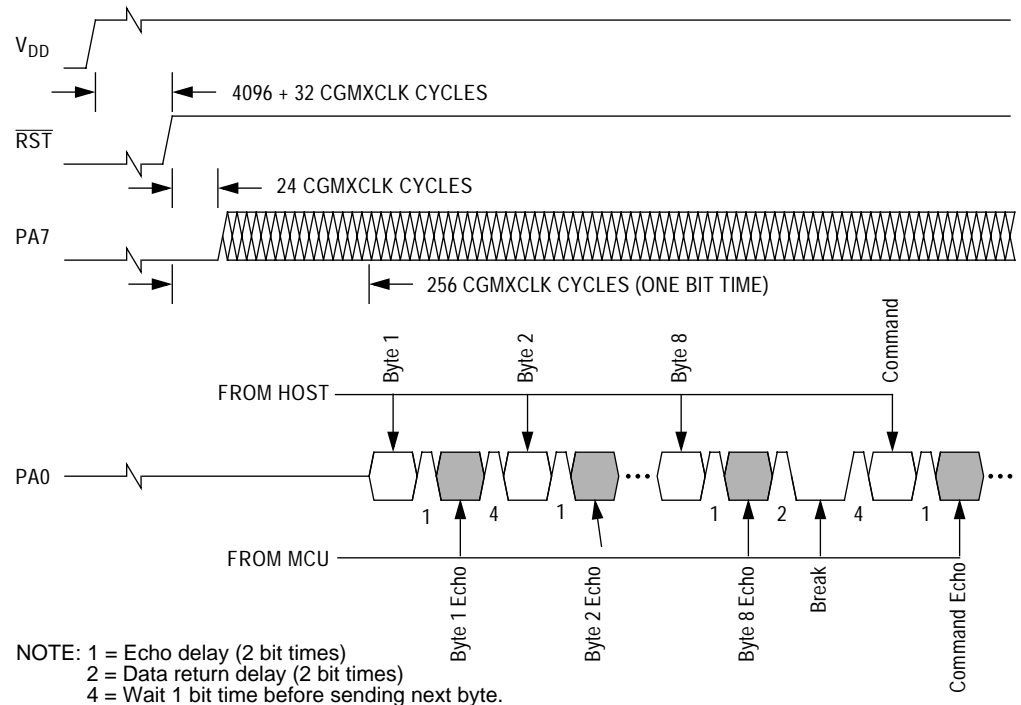
## Security

A security feature discourages unauthorized reading of FLASH locations while in monitor mode. The host can bypass the security feature at monitor mode entry by sending eight security bytes that match the bytes at locations \$FFF6–\$FFFD. Locations \$FFF6–\$FFFD contain user-defined data.

**NOTE:** *Do not leave locations \$FFF6–\$FFFD blank. For security reasons, program locations \$FFF6–\$FFFD even if they are not used for vectors. If FLASH is unprogrammed, the eight security byte values to be sent are \$00, the unprogrammed state of FLASH.*

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PA0.

## Monitor ROM (MON)



**Figure 6. Monitor Mode Entry Timing**

If the received bytes match those at locations \$FFF6–\$FFFD, the host bypasses the security feature and can read all FLASH locations and execute code from FLASH. Security remains bypassed until a power-on reset occurs. After the host bypasses security, any reset other than a power-on reset requires the host to send another eight bytes. If the reset was not a power-on reset, the security remains bypassed regardless of the data that the host sends.

If the received bytes do not match the data at locations \$FFF6–\$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading FLASH locations returns undefined data, and trying to execute code from FLASH causes an illegal address reset. After the host fails to bypass security, any reset other than a power-on reset causes an endless loop of illegal address resets.

After receiving the eight security bytes from the host, the MCU transmits a break character signalling that it is ready to receive a command.

**NOTE:** *The MCU does not transmit a break character until after the host sends the eight security bytes.*

# Computer Operating Properly Module (COP)

---

---

## Contents

|  |     |
|--|-----|
| Introduction . . . . .                       | 179 |
| Functional Description . . . . .             | 180 |
| I/O Signals . . . . .                        | 181 |
| CGMXCLK . . . . .                            | 181 |
| STOP Instruction . . . . .                   | 181 |
| COPCTL Write . . . . .                       | 182 |
| Power-On Reset . . . . .                     | 182 |
| Internal Reset . . . . .                     | 182 |
| Reset Vector Fetch . . . . .                 | 182 |
| COPD . . . . .                               | 182 |
| COPL . . . . .                               | 182 |
| COP Control Register . . . . .               | 183 |
| Interrupts . . . . .                         | 183 |
| Monitor Mode . . . . .                       | 183 |
| Low-Power Modes . . . . .                    | 184 |
| Wait Mode . . . . .                          | 184 |
| Stop Mode . . . . .                          | 184 |
| COP Module During Break Interrupts . . . . . | 184 |

---

---

## Introduction

The COP module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by periodically clearing the COP counter.

---

---

### Functional Description

The COP counter is a free-running 6-bit counter preceded by a 12-bit prescaler. If not cleared by software, the COP counter overflows and generates an asynchronous reset after  $2^{13} - 2^4$  or  $2^{18} - 2^4$  CGMXCLK cycles, depending on the state of the COP long timeout bit, COPL, in the CONFIG-1. When COPL = 1, a 4.9152-MHz crystal gives a COP timeout period of 53.3 ms. Writing any value to location \$FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 4–12 of the SIM counter.

**NOTE:** *Service the COP immediately after reset and before entering or after exiting stop mode to guarantee the maximum time before the first COP counter overflow.*

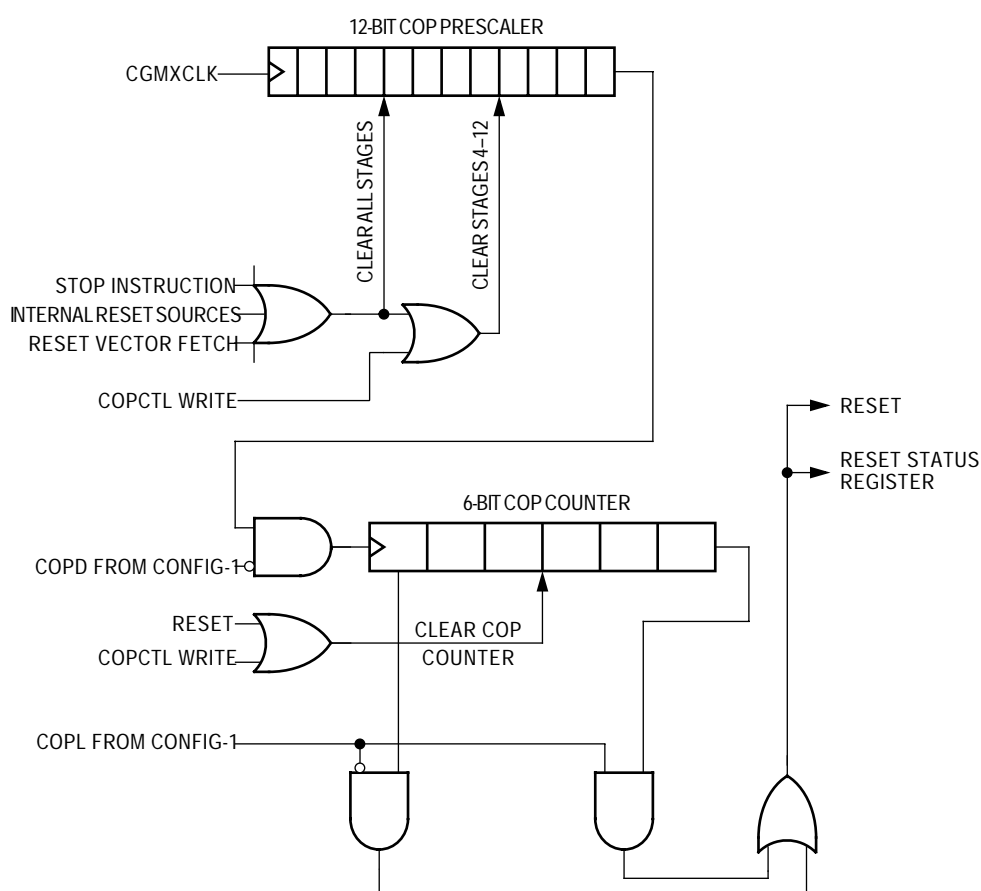
A COP reset pulls the  $\overline{\text{RST}}$  pin low for 32 CGMXCLK cycles and sets the COP bit in the reset status register (RSR).

In monitor mode, the COP is disabled if the  $\overline{\text{RST}}$  pin or the  $\overline{\text{IRQ}}$  pin is held at  $V_{\text{Hi}}$ . During the break state,  $V_{\text{Hi}}$  on the  $\overline{\text{RST}}$  pin disables the COP.

**NOTE:** *Place COP clearing instructions in the main program and not in an interrupt subroutine. Such an interrupt subroutine could keep the COP from generating a reset even while the main program is not working properly.*

## I/O Signals

The following paragraphs describe the signals shown in [Figure 1](#).



**Figure 1. COP Block Diagram**

### **CGMXCLK**

CGMXCLK is the crystal oscillator output signal. CGMXCLK frequency is equal to the crystal frequency.

### **STOP Instruction**

The STOP instruction clears the COP prescaler.

## Computer Operating Properly Module (COP)

|                           |   |
|---------------------------|---|
| <b>COPCTL Write</b>       | Writing any value to the COP control register (COPCTL) (see <a href="#">COP Control Register</a> on page 183), clears the COP counter and clears stages 12 through 4 of the COP prescaler. Reading the COP control register returns the reset vector. |
| <b>Power-On Reset</b>     | The power-on reset (POR) circuit clears the COP prescaler 4096 CGMXCLK cycles after power-up.   |
| <b>Internal Reset</b>     | An internal reset clears the COP prescaler and the COP counter.   |
| <b>Reset Vector Fetch</b> | A reset vector fetch occurs when the vector address appears on the data bus. A reset vector fetch clears the COP prescaler.   |
| <b>COPD</b>               | The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register. (See <a href="#">Configuration Register (CONFIG-1)</a> on page 155).  |
| <b>COPL</b>               | The COPL signal reflects the state of the COP rate select bit. (COPL) in the configuration register. (See <a href="#">Configuration Register (CONFIG-1)</a> on page 155).   |



---



---

## COP Control Register

The COP control register is located at address \$FFFF and overlaps the reset vector. Writing any value to \$FFFF clears the COP counter and starts a new timeout period. Reading location \$FFFF returns the low byte of the reset vector.

|          |  |
|----------|--|
| Address: | \$FFFF   |
|          | Bit 7      6      5      4      3      2      1      Bit 0 |
| Read:    | Low Byte of Reset Vector                                   |
| Write:   | Clear COP Counter  |
| Reset:   | Unaffected by Reset  |

**Figure 2. COP Control Register (COPCTL)**

---



---

## Interrupts

The COP does not generate CPU interrupt requests.

---



---

## Monitor Mode

The COP is disabled in monitor mode when  $V_{Hi}$  is present on the  $\overline{IRQ}/V_{PP}$  pin or on the  $\overline{RST}$  pin.

## Computer Operating Properly Module (COP)

---

---

### Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

#### Wait Mode

The COP remains active in wait mode. To prevent a COP reset during wait mode, periodically clear the COP counter in a CPU interrupt routine.

#### Stop Mode

Stop mode turns off the CGMXCLK input to the COP and clears the COP prescaler. Service the COP immediately before entering or after exiting stop mode to ensure a full COP timeout period after entering or exiting stop mode.

The STOP bit in the configuration register (CONFIG) enables the STOP instruction. To prevent inadvertently turning off the COP with a STOP instruction, disable the STOP instruction by clearing the STOP bit.

---

---

### COP Module During Break Interrupts

The COP is disabled during a break interrupt when  $V_{Hi}$  is present on the  $\overline{RST}$  pin.

# Low-Voltage Inhibit (LVI)

---

---

## Contents

|                              |     |
|------------------------------|-----|
| Introduction .....           | 186 |
| Features .....               | 186 |
| Functional Description ..... | 186 |
| Polled LVI Operation .....   | 188 |
| Forced Reset Operation ..... | 188 |
| False Reset Protection ..... | 188 |
| LVI Status Register .....    | 189 |
| LVI Interrupts .....         | 189 |
| Low-Power Modes .....        | 190 |
| Wait Mode .....              | 190 |
| Stop Mode .....              | 190 |

## Low-Voltage Inhibit (LVI)

---

---

### Introduction

This section describes the low-voltage inhibit module, which monitors the voltage on the  $V_{DD}$  pin and can force a reset when the  $V_{DD}$  voltage falls to the LVI trip voltage.

---

---

### Features

Features of the LVI module include:

- Programmable LVI Reset
- Programmable Power Consumption
- Digital Filtering of  $V_{DD}$  Pin Level

**NOTE:** *If a low voltage interrupt (LVI) occurs during programming of EEPROM or Flash memory, then adequate programming time may not have been allowed to ensure the integrity and retention of the data. It is the responsibility of the user to ensure that in the event of an LVI any addresses being programmed receive specification programming conditions.*

---

---

### Functional Description

**Figure 1** shows the structure of the LVI module. The LVI is enabled out of reset. The LVI module contains a bandgap reference circuit and comparator. The LVI power bit, LVIPWR, enables the LVI to monitor  $V_{DD}$  voltage. The LVI reset bit, LVIRST, enables the LVI module to generate a reset when  $V_{DD}$  falls below a voltage,  $LVI_{TRIPF}$ , and remains at or below that level for nine or more consecutive CPU cycles.

**NOTE:** *Note that short  $V_{DD}$  spikes may not trip the LVI. It is the user's responsibility to ensure a clean  $V_{DD}$  signal within the specified operating voltage range if normal microcontroller operation is to be guaranteed.*

LVISTOP, enables the LVI module during stop mode. This will ensure when the STOP instruction is implemented, the LVI will continue to monitor the voltage level on  $V_{DD}$ . LVIPWR, LVISTOP, and LVIRST are in the configuration register (CONFIG-1). (See [Configuration Register \(CONFIG-1\)](#) on page 155).

Once an LVI reset occurs, the MCU remains in reset until  $V_{DD}$  rises above a voltage,  $LVI_{TRIPR}$ .  $V_{DD}$  must be above  $LVI_{TRIPR}$  for only one CPU cycle to bring the MCU out of reset. (See [Forced Reset Operation](#) on page 188). The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR).

An LVI reset also drives the  $\overline{RST}$  pin low to provide low-voltage protection to external peripheral devices.

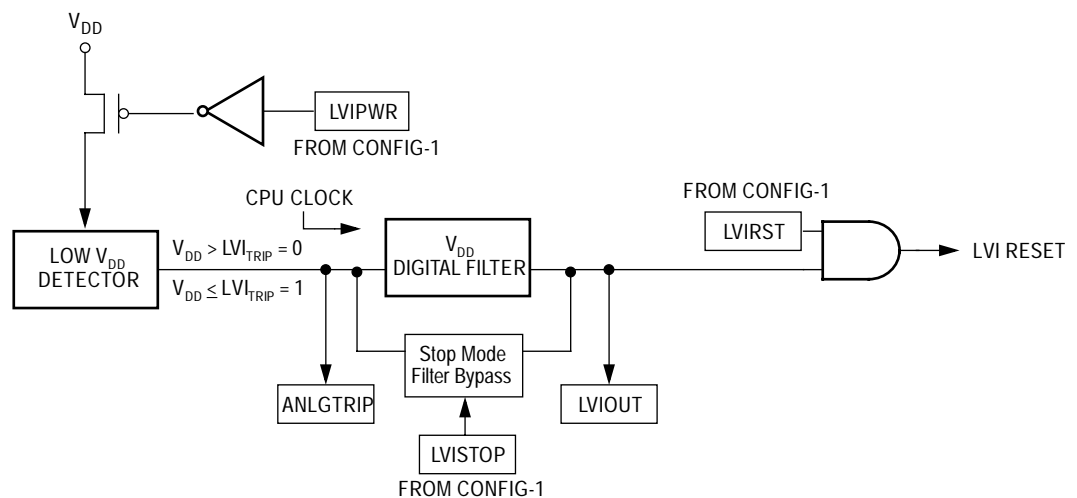



Figure 1. LVI Module Block Diagram

## Low-Voltage Inhibit (LVI)

| Addr.  | Register Name               | Bit 7  | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|--------|-----------------------------|--------|---|---|---|---|---|---|-------|
| \$FE0F | LVI Status Register (LVISR) | LVIOUT |   |   |   |   |   |   |       |

 = Unimplemented

**Figure 2. LVI I/O Register Summary**

### Polled LVI Operation

In applications that can operate at  $V_{DD}$  levels below the  $LVI_{TRIPF}$  level, software can monitor  $V_{DD}$  by polling the LVIOUT bit. In the configuration register, the LVIPWR bit must be at logic 1 to enable the LVI module, and the LVIRST bit must be at logic 0 to disable LVI resets.

### Forced Reset Operation

In applications that require  $V_{DD}$  to remain above the  $LVI_{TRIPF}$  level, enabling LVI resets allows the LVI module to reset the MCU when  $V_{DD}$  falls to the  $LVI_{TRIPF}$  level and remains at or below that level for nine or more consecutive CPU cycles. In the configuration register, the LVIPWR and LVIRST bits must be at logic 1 to enable the LVI module and to enable LVI resets.

### False Reset Protection


The  $V_{DD}$  pin level is digitally filtered to reduce false resets due to power supply noise. In order for the LVI module to reset the MCU,  $V_{DD}$  must remain at or below the  $LVI_{TRIPF}$  level for nine or more consecutive CPU cycles.  $V_{DD}$  must be above  $LVI_{TRIPR}$  for only one CPU cycle to bring the MCU out of reset.

## LVI Status Register

The LVI status register flags  $V_{DD}$  voltages below the  $LVI_{TRIPF}$  level.

Address: \$FE0F

|        | Bit 7  | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|--------|--------|---|---|---|---|---|---|-------|
| Read:  | LVIOUT | 0 | 0 | 0 | 0 | 0 | 0 | 0     |
| Write: |        |   |   |   |   |   |   |       |
| Reset: | 0      | 0 | 0 | 0 | 0 | 0 | 0 | 0     |

 = Unimplemented

**Figure 3. LVI Status Register (LVISR)**

### LVIOUT — LVI Output Bit

This read-only flag becomes set when the  $V_{DD}$  voltage falls below the  $LVI_{TRIPF}$  voltage for 32 to 40  $CGMXCLK$  cycles. (See [Table 1](#)). Reset clears the LVIOUT bit.

**Table 1. LVIOUT Bit Indication**

| $V_{DD}$                             |                                    | LVIOUT         |
|--------------------------------------|------------------------------------|----------------|
| At Level:                            | For Number of $CGMXCLK$ Cycles:    |                |
| $V_{DD} > LVI_{TRIPR}$               | Any                                | 0              |
| $V_{DD} < LVI_{TRIPF}$               | < 32 $CGMXCLK$ Cycles              | 0              |
| $V_{DD} < LVI_{TRIPF}$               | Between 32 and 40 $CGMXCLK$ Cycles | 0 or 1         |
| $V_{DD} < LVI_{TRIPF}$               | > 40 $CGMXCLK$ Cycles              | 1              |
| $LVI_{TRIPF} < V_{DD} < LVI_{TRIPR}$ | Any                                | Previous Value |

## Low-Voltage Inhibit (LVI)

---

---

### LVI Interrupts

The LVI module does not generate interrupt requests.

---

---

### Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

#### Wait Mode

With the LVIPWR bit in the configuration register programmed to logic 1, the LVI module is active after a WAIT instruction.

With the LVIRST bit in the configuration register programmed to logic 1, the LVI module can generate a reset and bring the MCU out of wait mode.

#### Stop Mode

With the LVISTOP and LVIPWR bits in the configuration register programmed to a logic 1, the LVI module will be active after a STOP instruction. Because CPU clocks are disabled during stop mode, the LVI trip must bypass the digital filter to generate a reset and bring the MCU out of stop.

With the LVIPWR bit in the configuration register programmed to logic 1 and the LVISTOP bit at a logic 0, the LVI module will be inactive after a STOP instruction.

**NOTE:** *Note that the LVI feature is intended to provide the safe shutdown of the microcontroller and thus protection of related circuitry prior to any application  $V_{DD}$  voltage collapsing completely to an unsafe level. It is not intended that users operate the microcontroller at lower than specified operating voltage  $V_{DD}$ .*



# External Interrupt Module (IRQ)

---

---

## Contents

|   |     |
|---|-----|
| Introduction .....                      | 191 |
| Features .....                          | 191 |
| Functional Description.....             | 192 |
| IRQ Pin .....                           | 195 |
| IRQ Module During Break Interrupts..... | 196 |
| IRQ Status and Control Register.....    | 197 |

---

---

## Introduction

This section describes the nonmaskable external interrupt (IRQ) input.

---

---

## Features

Features include:

- Dedicated External Interrupt Pin ( $\overline{\text{IRQ}}/V_{PP}$ )
- Hysteresis Buffer
- Programmable Edge-Only or Edge- and Level-Interrupt Sensitivity
- Automatic Interrupt Acknowledge

## External Interrupt Module (IRQ)

### Functional Description

A logic 0 applied to the external interrupt pin can latch a CPU interrupt request. **Figure 1** shows the structure of the IRQ module.

Interrupt signals on the  $\overline{\text{IRQ}}/V_{\text{PP}}$  pin are latched into the IRQ latch. An interrupt latch remains set until one of the following actions occurs:

- Vector fetch — A vector fetch automatically generates an interrupt acknowledge signal that clears the latch that caused the vector fetch.
- Software clear — Software can clear an interrupt latch by writing to the appropriate acknowledge bit in the interrupt status and control register (ISCR). Writing a logic 1 to the ACK1 bit clears the IRQ latch.
- Reset — A reset automatically clears both interrupt latches.

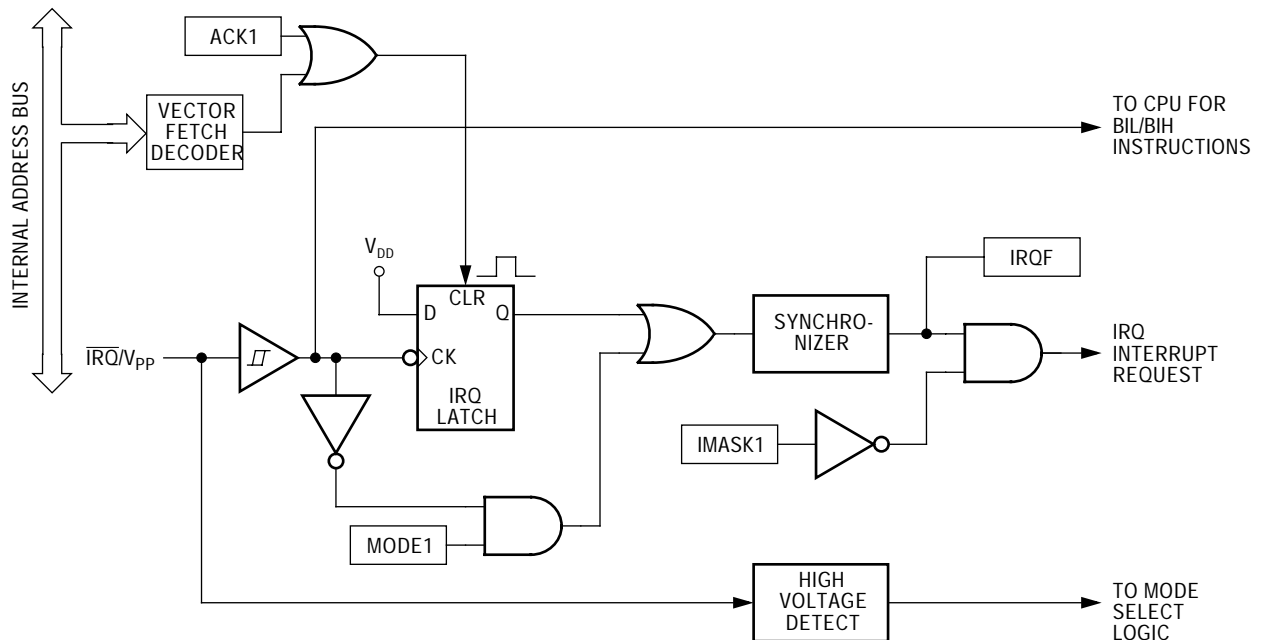


Figure 1. IRQ Block Diagram

**Table 1. IRQ I/O Register Summary**

| Addr.  | Register Name                      | Bit 7  | 6 | 5 | 4 | 3 | 2     | 1    | Bit 0  |       |
|--------|------------------------------------|--------|---|---|---|---|-------|------|--------|-------|
| \$001A | IRQ Status/Control Register (ISCR) | Read:  | 0 | 0 | 0 | 0 | IRQF1 | 0    | IMASK1 | MODE1 |
|        |                                    | Write: | R | R | R | R | R     | ACK1 |        |       |

|   |
|---|
| R |
|---|

 = Reserved

The external interrupt pin is falling-edge triggered and is software-configurable to be both falling-edge and low-level triggered. The MODE1 bit in the ISCR controls the triggering sensitivity of the  $\overline{IRQ}/V_{PP}$  pin.

When an interrupt pin is edge-triggered only, the interrupt latch remains set until a vector fetch, software clear, or reset occurs.

When an interrupt pin is both falling-edge and low-level-triggered, the interrupt latch remains set until both of the following occur:

- Vector fetch or software clear
- Return of the interrupt pin to logic 1

The vector fetch or software clear may occur before or after the interrupt pin returns to logic 1. As long as the pin is low, the interrupt request remains pending. A reset will clear the latch and the MODE1 control bit, thereby clearing the interrupt even if the pin stays low.

When set, the IMASK1 bit in the ISCR masks all external interrupt requests. A latched interrupt request is not presented to the interrupt priority logic unless the corresponding IMASK bit is clear.

**NOTE:** *The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests. (See [Figure 2](#)).*

## External Interrupt Module (IRQ)

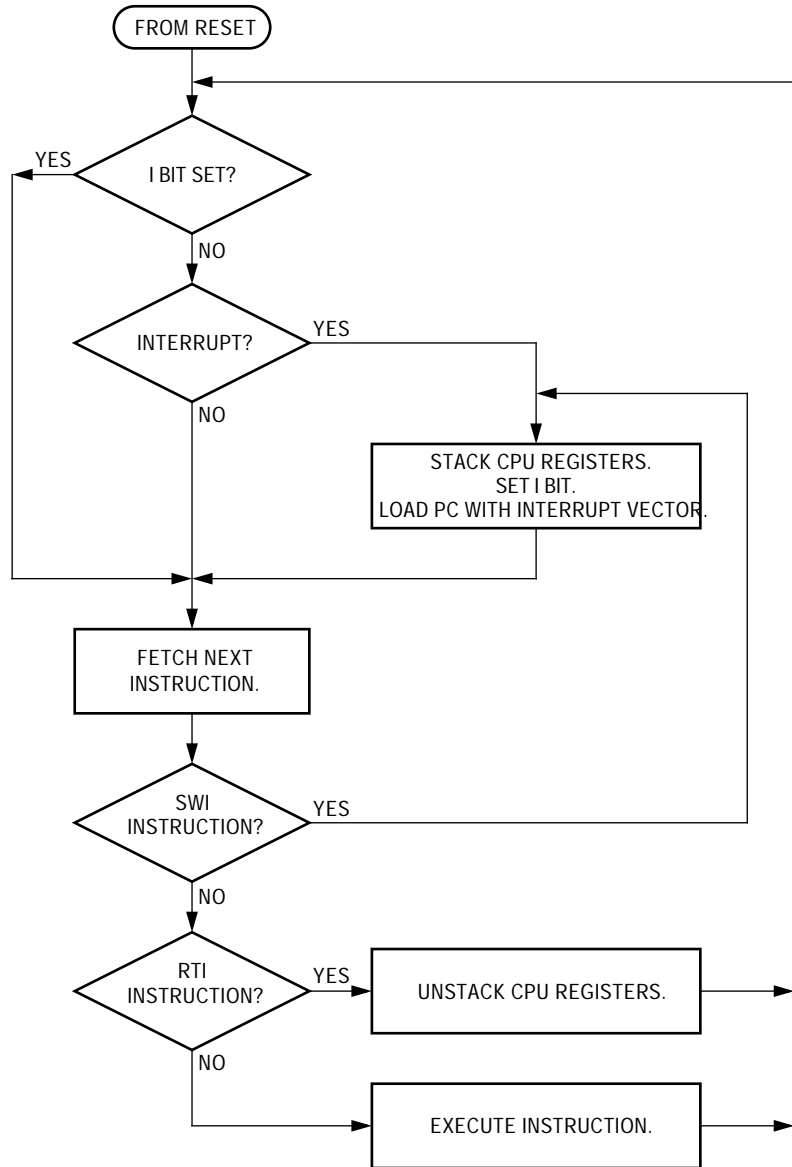


Figure 2. IRQ Interrupt Flowchart

---

---

## IRQ Pin

A logic 0 on the  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  pin can latch an interrupt request into the IRQ latch. A vector fetch, software clear, or reset clears the IRQ latch.

If the MODE1 bit is set, the  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  pin is both falling-edge sensitive and low-level sensitive. With MODE1 set, both of the following actions must occur to clear the IRQ latch:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the latch. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACK1 bit in the interrupt status and control register (ISCR). The ACK1 bit is useful in applications that poll the  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  pin and require software to clear the IRQ latch. Writing to the ACK1 bit can also prevent spurious interrupts due to noise. Setting ACK1 does not affect subsequent transitions on the  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  pin. A falling edge on  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  that occurs after writing to the ACK1 bit latches another interrupt request. If the IRQ mask bit, IMASK1, is clear, the CPU loads the program counter with the vector address at locations \$FFFA and \$FFFB.
- Return of the  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  pin to logic 1 — As long as the  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  pin is at logic 0, the IRQ latch remains set.

The vector fetch or software clear and the return of the  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  pin to logic 1 can occur in any order. The interrupt request remains pending as long as the  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  pin is at logic 0. A reset will clear the latch and the MODE1 control bit, thereby clearing the interrupt even if the pin stays low.

If the MODE1 bit is clear, the  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  pin is falling-edge sensitive only. With MODE1 clear, a vector fetch or software clear immediately clears the IRQ latch.

## External Interrupt Module (IRQ)

The IRQF1 bit in the ISCR register can be used to check for pending interrupts. The IRQF1 bit is not affected by the IMASK1 bit, which makes it useful in applications where polling is preferred.

Use the BIH or BIL instruction to read the logic level on the  $\overline{\text{IRQ}}/V_{\text{PP}}$  pin.

**NOTE:** *When using the level-sensitive interrupt trigger, avoid false interrupts by masking interrupt requests in the interrupt routine.*

---

---

### IRQ Module During Break Interrupts

The system integration module (SIM) controls whether the IRQ interrupt latch can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear the latches during the break state. (See [SIM Break Flag Control Register](#) on page 125)

To allow software to clear the IRQ latch during a break interrupt, write a logic 1 to the BCFE bit. If a latch is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect the latch during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), writing to the ACK1 bit in the IRQ status and control register during the break state has no effect on the IRQ latch.

## IRQ Status and Control Register

The IRQ status and control register (ISCR) controls and monitors operation of the IRQ module. The ISCR has these functions:

- Shows the state of the IRQ interrupt flag
- Clears the IRQ interrupt latch
- Masks IRQ interrupt request
- Controls triggering sensitivity of the  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  interrupt pin

Address: \$001A

|        | Bit 7 | 6 | 5 | 4 | 3     | 2    | 1      | Bit 0 |
|--------|-------|---|---|---|-------|------|--------|-------|
| Read:  | 0     | 0 | 0 | 0 | IRQF1 | 0    | IMASK1 | MODE1 |
| Write: | R     | R | R | R | R     | ACK1 |        |       |
| Reset: | 0     | 0 | 0 | 0 | 0     | 0    | 0      | 0     |

R = Reserved

**Figure 3. IRQ Status and Control Register (ISCR)**

### IRQF — IRQ Flag Bit

This read-only status bit is high when the IRQ interrupt is pending.

1 =  $\overline{\text{IRQ}}$  interrupt pending

0 =  $\overline{\text{IRQ}}$  interrupt not pending

### ACK1 — IRQ Interrupt Request Acknowledge Bit

Writing a logic 1 to this write-only bit clears the IRQ latch. ACK1 always reads as logic 0. Reset clears ACK1.

### IMASK1 — IRQ Interrupt Mask Bit

Writing a logic 1 to this read/write bit disables IRQ interrupt requests. Reset clears IMASK1.

1 = IRQ interrupt requests disabled

0 = IRQ interrupt requests enabled

## External Interrupt Module (IRQ)

### MODE1 — IRQ Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  pin. Reset clears MODE1.

1 =  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  interrupt requests on falling edges and low levels

0 =  $\overline{\text{IRQ}}/\text{V}_{\text{PP}}$  interrupt requests on falling edges only



# Serial Communications Interface Module (SCI)

---

---

## Contents

|  |     |
|--|-----|
| Introduction . . . . .                       | 200 |
| Features . . . . .                           | 200 |
| Pin Name Conventions . . . . .               | 201 |
| Functional Description. . . . .              | 201 |
| Data Format . . . . .                        | 203 |
| Transmitter . . . . .                        | 204 |
| Character Length . . . . .                   | 204 |
| Character Transmission . . . . .             | 204 |
| Break Characters . . . . .                   | 207 |
| Idle Characters . . . . .                    | 207 |
| Inversion of Transmitted Output . . . . .    | 208 |
| Transmitter Interrupts . . . . .             | 208 |
| Receiver . . . . .                           | 208 |
| Character Length . . . . .                   | 211 |
| Character Reception. . . . .                 | 211 |
| Data Sampling . . . . .                      | 211 |
| Framing Errors . . . . .                     | 214 |
| Baud Rate Tolerance . . . . .                | 214 |
| Receiver Wakeup . . . . .                    | 216 |
| Error Interrupts . . . . .                   | 217 |
| Low-Power Modes . . . . .                    | 218 |
| Wait Mode. . . . .                           | 218 |
| Stop Mode. . . . .                           | 218 |
| SCI During Break Module Interrupts . . . . . | 219 |
| I/O Signals. . . . .                         | 219 |
| PTE0/SCTxD (Transmit Data) . . . . .         | 219 |
| PTE1/SCRxD (Receive Data) . . . . .          | 219 |
| I/O Registers . . . . .                      | 220 |
| SCI Control Register 1 . . . . .             | 220 |
| SCI Control Register 2 . . . . .             | 223 |
| SCI Control Register 3 . . . . .             | 227 |

## Serial Communications Interface Module (SCI)

|                              |     |
|------------------------------|-----|
| SCI Status Register 1 .....  | 229 |
| SCI Status Register 2 .....  | 232 |
| SCI Data Register .....      | 233 |
| SCI Baud Rate Register ..... | 234 |

---

---

### Introduction

The SCI allows asynchronous communications with peripheral devices and other MCUs.

---

---

### Features

The SCI module's features include:

- Full Duplex Operation
- Standard Mark/Space Non-Return-to-Zero (NRZ) Format
- 32 Programmable Baud Rates
- Programmable 8-Bit or 9-Bit Character Length
- Separately Enabled Transmitter and Receiver
- Separate Receiver and Transmitter CPU Interrupt Requests
- Programmable Transmitter Output Polarity
- Two Receiver Wakeup Methods:
  - Idle Line Wakeup
  - Address Mark Wakeup
- Interrupt-Driven Operation with Eight Interrupt Flags:
  - Transmitter Empty
  - Transmission Complete
  - Receiver Full
  - Idle Receiver Input
  - Receiver Overrun

- Noise Error
- Framing Error
- Parity Error
- Receiver Framing Error Detection
- Hardware Parity Checking
- 1/16 Bit-Time Noise Detection

---



---

## Pin Name Conventions

The generic names of the SCI input/output (I/O) pins are:

- RxD (receive data)
- TxD (transmit data)

SCI I/O lines are implemented by sharing parallel I/O port pins. The full name of an SCI input or output reflects the name of the shared port pin. **Table 1** shows the full names and the generic names of the SCI I/O pins. The generic pin names appear in the text of this section.

**Table 1. Pin Name Conventions**

|                          |            |            |
|--------------------------|------------|------------|
| <b>Generic Pin Names</b> | RxD        | TxD        |
| <b>Full Pin Names</b>    | PTE1/SCRxD | PTE0/SCTxD |

---



---

## Functional Description

**Figure 1** shows the structure of the SCI module. The SCI allows full-duplex, asynchronous, NRZ serial communication between the MCU and remote devices, including other MCUs. The transmitter and receiver of the SCI operate independently, although they use the same baud rate generator. During normal operation, the CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

# Serial Communications Interface Module (SCI)

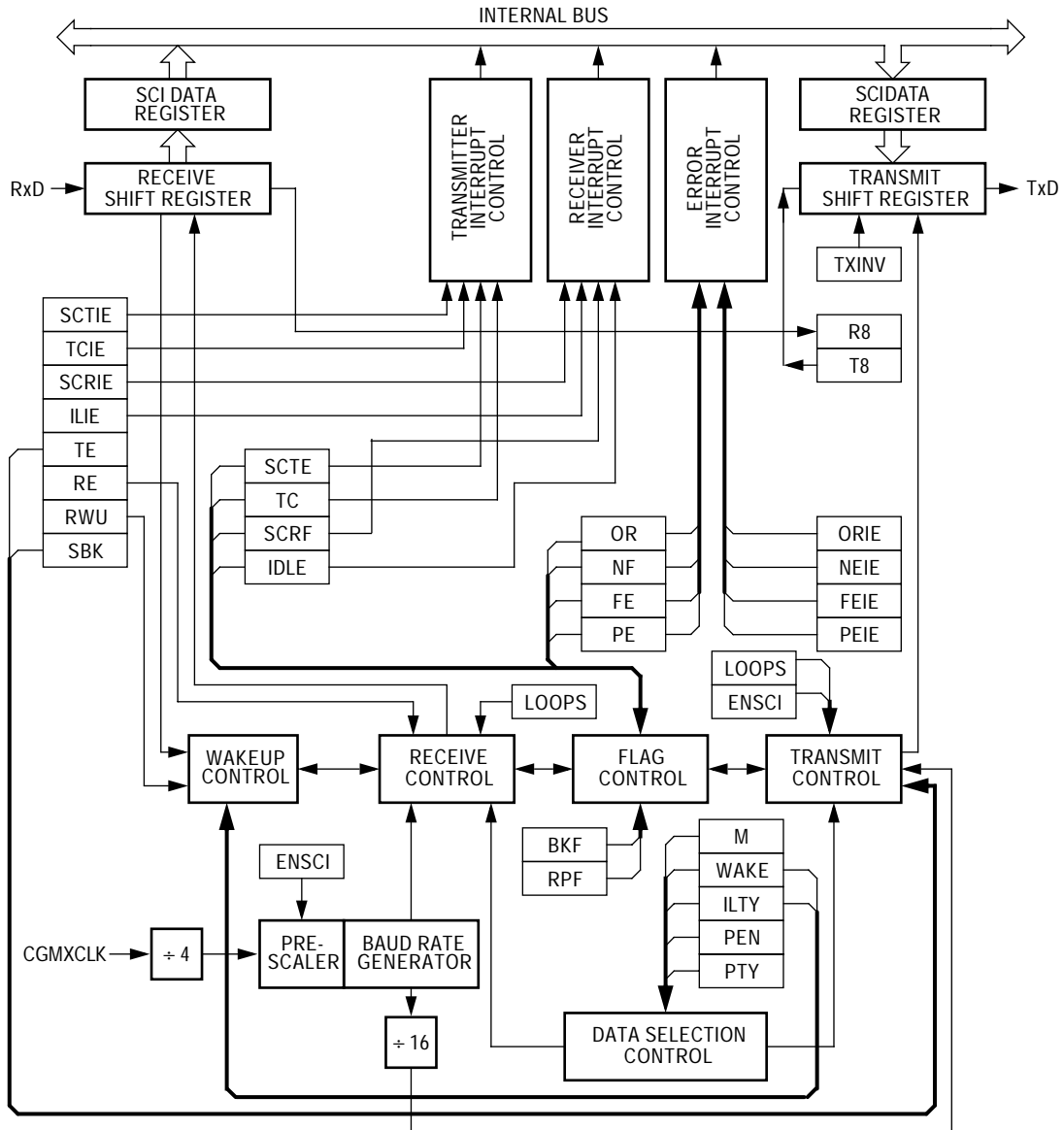


Figure 1. SCI Module Block Diagram

Serial Communications Interface Module (SCI)  
Functional Description

| Register Name                 | Bit 7  | 6                   | 5     | 4     | 3    | 2    | 1    | Bit 0 |      |
|-------------------------------|--------|---------------------|-------|-------|------|------|------|-------|------|
| SCI Control Register 1 (SCC1) | Read:  | LOOPS               | ENSCI | TXINV | M    | WAKE | ILTY | PEN   | PTY  |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 0                   | 0     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Control Register 2 (SCC2) | Read:  | SCTIE               | TCIE  | SCRIE | ILIE | TE   | RE   | RWU   | SBK  |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 0                   | 0     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Control Register 3 (SCC3) | Read:  | R8                  | T8    | R     | R    | ORIE | NEIE | FEIE  | PEIE |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | U                   | U     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Status Register 1 (SCS1)  | Read:  | SCTE                | TC    | SCRF  | IDLE | OR   | NF   | FE    | PE   |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 1                   | 1     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Status Register 2 (SCS2)  | Read:  |                     |       |       |      |      | BKF  | RPF   |      |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 0                   | 0     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Data Register (SCDR)      | Read:  | R7                  | R6    | R5    | R4   | R3   | R2   | R1    | R0   |
|                               | Write: | T7                  | T6    | T5    | T4   | T3   | T2   | T1    | T0   |
|                               | Reset: | Unaffected by Reset |       |       |      |      |      |       |      |
| SCI Baud Rate Register (SCBR) | Read:  |                     |       | SCP1  | SCP0 | R    | SCR2 | SCR1  | SCR0 |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 0                   | 0     | 0     | 0    | 0    | 0    | 0     | 0    |

= Unimplemented    U = Unaffected    R = Reserved

**Figure 2. SCI I/O Register Summary**

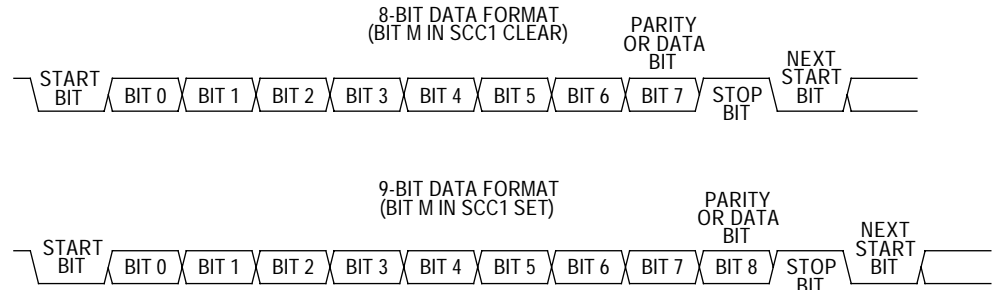
**Table 2. SCI I/O Register Address Summary**

|                 |        |        |        |        |        |        |        |
|-----------------|--------|--------|--------|--------|--------|--------|--------|
| <b>Register</b> | SCC1   | SCC2   | SCC3   | SCS1   | SCS2   | SCDR   | SCBR   |
| <b>Address</b>  | \$0013 | \$0014 | \$0015 | \$0016 | \$0017 | \$0018 | \$0019 |

**Data Format**

The SCI uses the standard non-return-to-zero mark/space data format illustrated in [Figure 3](#).

## Serial Communications Interface Module (SCI)



**Figure 3. SCI Data Formats**

### Transmitter

**Figure 4** shows the structure of the SCI transmitter.

#### *Character Length*

The transmitter can accommodate either 8-bit or 9-bit data. The state of the M bit in SCI control register 1 (SCC1) determines character length. When transmitting 9-bit data, bit T8 in SCI control register 3 (SCC3) is the ninth bit (bit 8).

#### *Character Transmission*

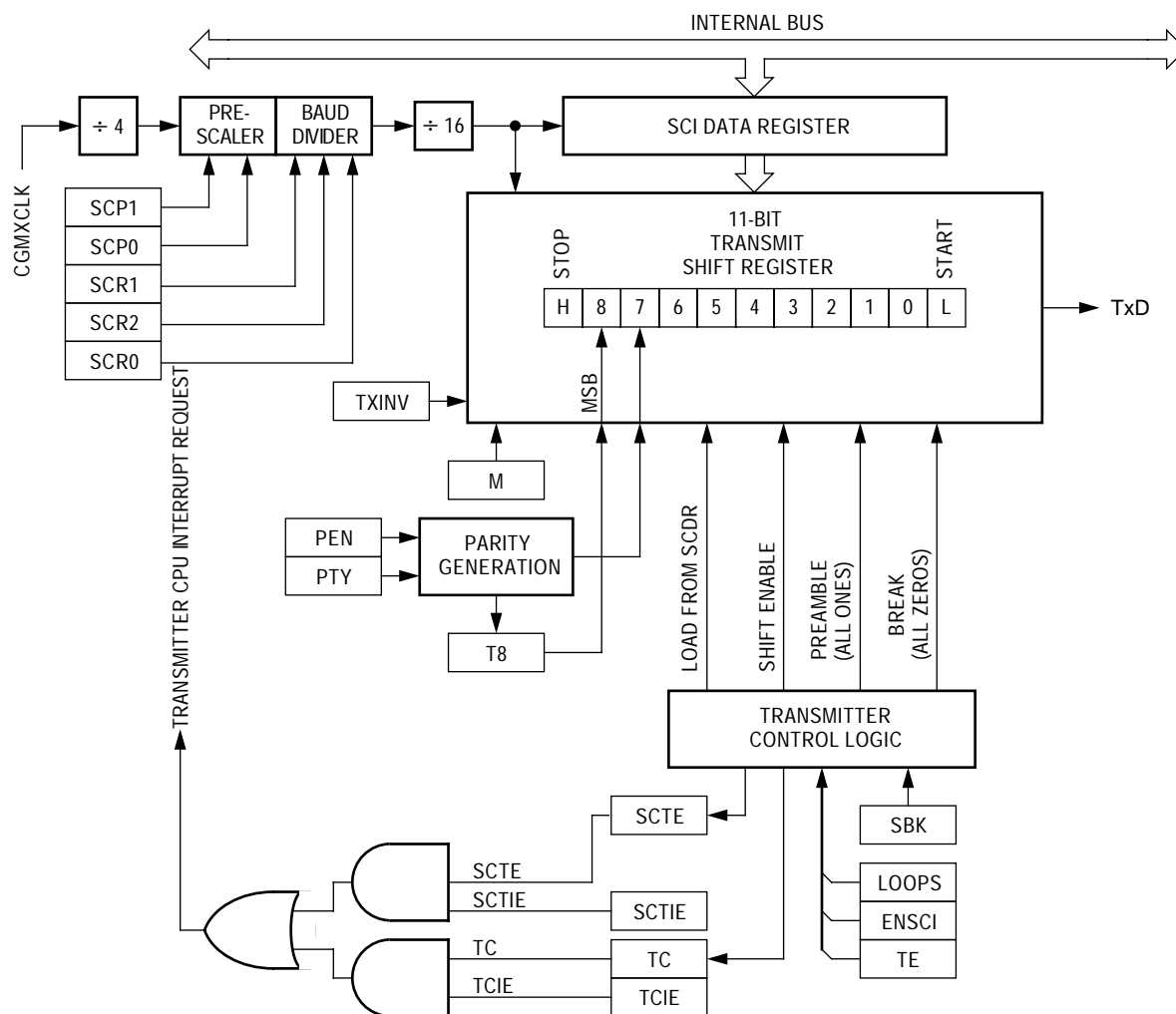
During an SCI transmission, the transmit shift register shifts a character out to the TxD pin. The SCI data register (SCDR) is the write-only buffer between the internal data bus and the transmit shift register. To initiate an SCI transmission:

1. Enable the SCI by writing a logic 1 to the enable SCI bit (ENSCI) in SCI control register 1 (SCC1).
2. Enable the transmitter by writing a logic 1 to the transmitter enable bit (TE) in SCI control register 2 (SCC2).
3. Clear the SCI transmitter empty bit (SCTE) by first reading SCI status register 1 (SCS1) and then writing to the SCDR.
4. Repeat step 3 for each subsequent transmission.

At the start of a transmission, transmitter control logic automatically loads the transmit shift register with a preamble of logic 1s. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

The SCI transmitter empty bit, SCTE, in SCS1 becomes set when the SCDR transfers a byte to the transmit shift register. The SCTE bit indicates that the SCDR can accept new data from the internal data bus. If the SCI transmit interrupt enable bit, SCTIE, in SCC2 is also set, the SCTE bit generates a transmitter CPU interrupt request.

When the transmit shift register is not transmitting a character, the TxD pin goes to the idle condition, logic 1. If at any time software clears the ENSCI bit in SCI control register 1 (SCC1), the transmitter and receiver relinquish control of the port E pins.



**Figure 4. SCI Transmitter**

## Serial Communications Interface Module (SCI)

| Register Name                 | Bit 7  | 6                   | 5     | 4     | 3    | 2    | 1    | Bit 0 |      |
|-------------------------------|--------|---------------------|-------|-------|------|------|------|-------|------|
| SCI Control Register 1 (SCC1) | Read:  | LOOPS               | ENSCI | TXINV | M    | WAKE | ILTY | PEN   | PTY  |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 0                   | 0     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Control Register 2 (SCC2) | Read:  | SCTIE               | TCIE  | SCRIE | ILIE | TE   | RE   | RWU   | SBK  |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 0                   | 0     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Control Register 3 (SCC3) | Read:  | R8                  | T8    | R     | R    | ORIE | NEIE | FEIE  | PEIE |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | U                   | U     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Status Register 1 (SCS1)  | Read:  | SCTE                | TC    | SCRF  | IDLE | OR   | NF   | FE    | PE   |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 1                   | 1     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Data Register (SCDR)      | Read:  | R7                  | R6    | R5    | R4   | R3   | R2   | R1    | R0   |
|                               | Write: | T7                  | T6    | T5    | T4   | T3   | T2   | T1    | T0   |
|                               | Reset: | Unaffected by Reset |       |       |      |      |      |       |      |
| SCI Baud Rate Register (SCBR) | Read:  |                     |       | SCP1  | SCP0 | R    | SCR2 | SCR1  | SCR0 |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 0                   | 0     | 0     | 0    | 0    | 0    | 0     | 0    |

= Unimplemented    U = Unaffected    R = Reserved

**Figure 5. SCI Transmitter I/O Register Summary**

**Table 3. SCI Transmitter I/O Address Summary**

| Register | SCC1   | SCC2   | SCC3   | SCS1   | SCDR   | SCBR   |
|----------|--------|--------|--------|--------|--------|--------|
| Address  | \$0013 | \$0014 | \$0015 | \$0016 | \$0018 | \$0019 |



### *Break Characters*

Writing a logic 1 to the send break bit, SBK, in SCC2 loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit in SCC1. As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next character.

The SCI recognizes a break character when a start bit is followed by eight or nine logic 0 data bits and a logic 0 where the stop bit should be. Receiving a break character has the following effects on SCI registers:

- Sets the framing error bit (FE) in SCS1
- Sets the SCI receiver full bit (SCRF) in SCS1
- Clears the SCI data register (SCDR)
- Clears the R8 bit in SCC3
- Sets the break flag bit (BKF) in SCS2
- May set the overrun (OR), noise flag (NF), parity error (PE), or reception in progress flag (RPF) bits

### *Idle Characters*

An idle character contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in SCC1. The preamble is a synchronizing idle character that begins every transmission.

If the TE bit is cleared during a transmission, the TxD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the character currently being transmitted.

## Serial Communications Interface Module (SCI)

**NOTE:** When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current character shifts out to the TxD pin. Setting TE after the stop bit appears on TxD causes data previously written to the SCDR to be lost.

A good time to toggle the TE bit for a queued idle character is when the SCTE bit becomes set and just before writing the next byte to the SCDR.

### *Inversion of Transmitted Output*

The transmit inversion bit (TXINV) in SCI control register 1 (SCC1) reverses the polarity of transmitted data. All transmitted values, including idle, break, start, and stop bits, are inverted when TXINV is at logic 1. (See SCI Control Register 1.)

### *Transmitter Interrupts*

The following conditions can generate CPU interrupt requests from the SCI transmitter:

- SCI transmitter empty (SCTE) — The SCTE bit in SCS1 indicates that the SCDR has transferred a character to the transmit shift register. SCTE can generate a transmitter CPU interrupt request. Setting the SCI transmit interrupt enable bit, SCTIE, in SCC2 enables the SCTE bit to generate transmitter CPU interrupt requests.
- Transmission complete (TC) — The TC bit in SCS1 indicates that the transmit shift register and the SCDR are empty and that no break or idle character has been generated. The transmission complete interrupt enable bit, TCIE, in SCC2 enables the TC bit to generate transmitter CPU interrupt requests.

### **Receiver**

**Figure 6** shows the structure of the SCI receiver.

Serial Communications Interface Module (SCI)  
Functional Description

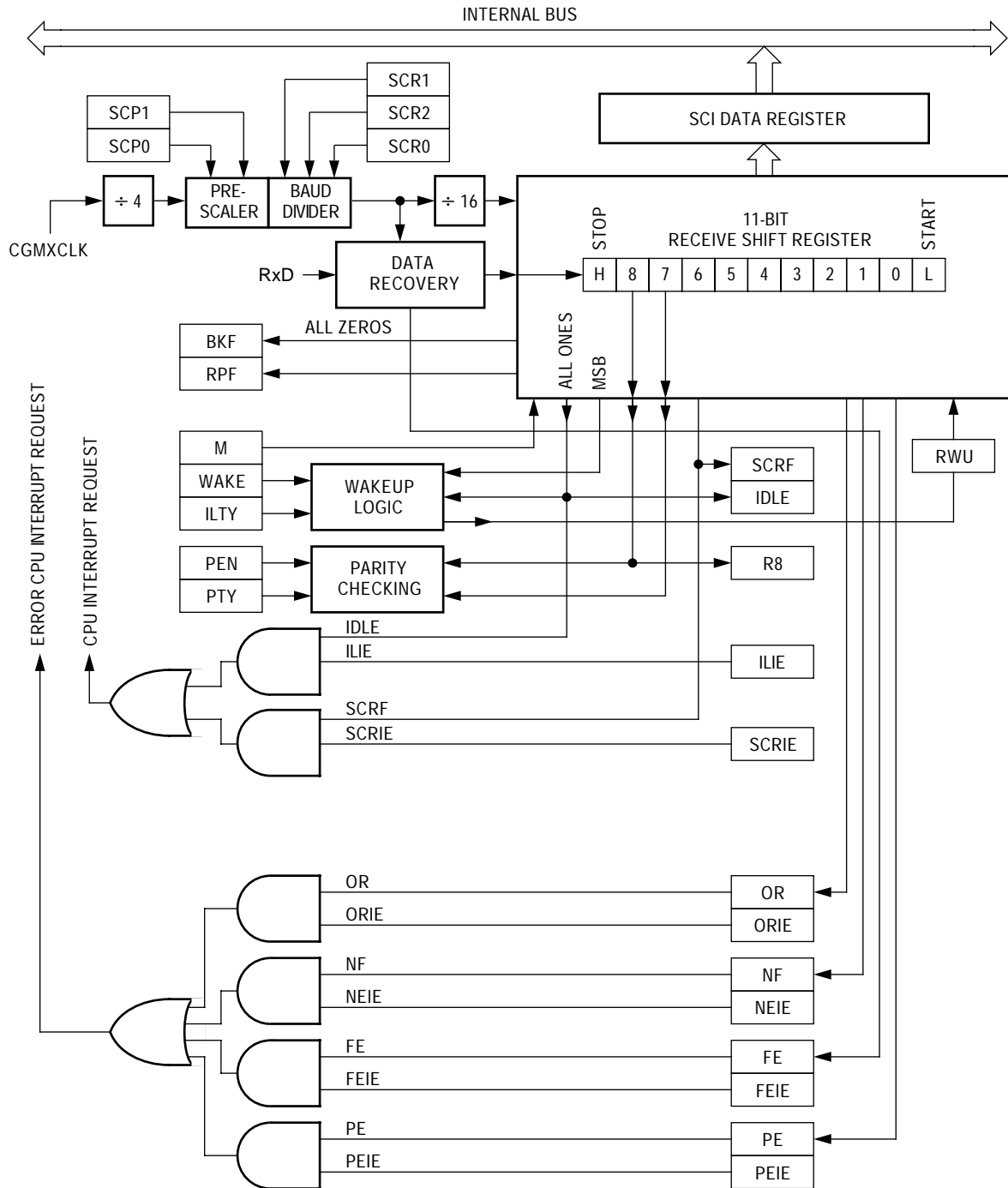


Figure 6. SCI Receiver Block Diagram

## Serial Communications Interface Module (SCI)

| Register Name                 | Bit 7  | 6                   | 5     | 4     | 3    | 2    | 1    | Bit 0 |      |
|-------------------------------|--------|---------------------|-------|-------|------|------|------|-------|------|
| SCI Control Register 1 (SCC1) | Read:  | LOOPS               | ENSCI | TXINV | M    | WAKE | ILTY | PEN   | PTY  |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 0                   | 0     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Control Register 2 (SCC2) | Read:  | SCTIE               | TCIE  | SCRIE | ILIE | TE   | RE   | RWU   | SBK  |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 0                   | 0     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Control Register 3 (SCC3) | Read:  | R8                  | T8    | R     | R    | ORIE | NEIE | FEIE  | PEIE |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | U                   | U     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Status Register 1 (SCS1)  | Read:  | SCTE                | TC    | SCRF  | IDLE | OR   | NF   | FE    | PE   |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 1                   | 1     | 0     | 0    | 0    | 0    | 0     | 0    |
| SCI Status Register 2 (SCS2)  | Read:  |                     |       |       |      |      | BKF  | RPF   |      |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 0                   | 0     | 0     | 0    | 0    | 0    | 0     |      |
| SCI Data Register (SCDR)      | Read:  | R7                  | R6    | R5    | R4   | R3   | R2   | R1    | R0   |
|                               | Write: | T7                  | T6    | T5    | T4   | T3   | T2   | T1    | T0   |
|                               | Reset: | Unaffected by Reset |       |       |      |      |      |       |      |
| SCI Baud Rate Register (SCBR) | Read:  |                     |       | SCP1  | SCP0 | R    | SCR2 | SCR1  | SCR0 |
|                               | Write: |                     |       |       |      |      |      |       |      |
|                               | Reset: | 0                   | 0     | 0     | 0    | 0    | 0    | 0     | 0    |

= Unimplemented    
 U = Unaffected    
 R = Reserved

**Figure 7. SCI I/O Receiver Register Summary**

**Table 4. SCI Receiver I/O Address Summary**

| Register | SCC1   | SCC2   | SCC3   | SCS1   | SCS2   | SCDR   | SCBR   |
|----------|--------|--------|--------|--------|--------|--------|--------|
| Address  | \$0013 | \$0014 | \$0015 | \$0016 | \$0017 | \$0018 | \$0019 |

*Character Length*      The receiver can accommodate either 8-bit or 9-bit data. The state of the M bit in SCI control register 1 (SCC1) determines character length. When receiving 9-bit data, bit R8 in SCI control register 2 (SCC2) is the ninth bit (bit 8). When receiving 8-bit data, bit R8 is a copy of the eighth bit (bit 7).

*Character Reception*      During an SCI reception, the receive shift register shifts characters in from the RxD pin. The SCI data register (SCDR) is the read-only buffer between the internal data bus and the receive shift register.

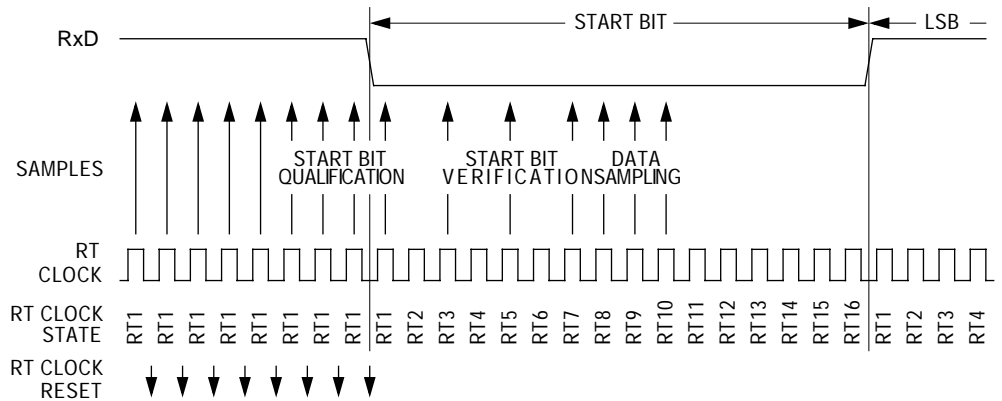
After a complete character shifts into the receive shift register, the data portion of the character transfers to the SCDR. The SCI receiver full bit, SCRF, in SCI status register 1 (SCS1) becomes set, indicating that the received byte can be read. If the SCI receive interrupt enable bit, SCRIE, in SCC2 is also set, the SCRF bit generates a receiver CPU interrupt request.

*Data Sampling*      The receiver samples the RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock is resynchronized at the following times (see [Figure 8](#)):

- After every start bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0)

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

## Serial Communications Interface Module (SCI)



**Figure 8. Receiver Data Sampling**

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. [Table 5](#) summarizes the results of the start bit verification samples.

**Table 5. Start Bit Verification**

| RT3, RT5, and RT7 Samples | Start Bit Verification | Noise Flag |
|---------------------------|------------------------|------------|
| 000                       | Yes                    | 0          |
| 001                       | Yes                    | 1          |
| 010                       | Yes                    | 1          |
| 011                       | No                     | 0          |
| 100                       | Yes                    | 1          |
| 101                       | No                     | 0          |
| 110                       | No                     | 0          |
| 111                       | No                     | 0          |

If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. **Table 6** summarizes the results of the data bit samples.

**Table 6. Data Bit Recovery**

| RT8, RT9, and RT10 Samples | Data Bit Determination | Noise Flag |
|----------------------------|------------------------|------------|
| 000                        | 0                      | 0          |
| 001                        | 0                      | 1          |
| 010                        | 0                      | 1          |
| 011                        | 1                      | 1          |
| 100                        | 0                      | 1          |
| 101                        | 1                      | 1          |
| 110                        | 1                      | 1          |
| 111                        | 1                      | 0          |

**NOTE:** *The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit.*

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. **Table 7** summarizes the results of the stop bit samples.

**Table 7. Stop Bit Recovery**

| RT8, RT9, and RT10 Samples | Framing Error Flag | Noise Flag |
|----------------------------|--------------------|------------|
| 000                        | 1                  | 0          |
| 001                        | 1                  | 1          |
| 010                        | 1                  | 1          |
| 011                        | 0                  | 1          |
| 100                        | 1                  | 1          |
| 101                        | 0                  | 1          |
| 110                        | 0                  | 1          |
| 111                        | 0                  | 0          |

## Serial Communications Interface Module (SCI)

### Framing Errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming character, it sets the framing error bit, FE, in SCS1. A break character also sets the FE bit because a break character has no stop bit. The FE bit is set at the same time that the SCRF bit is set.

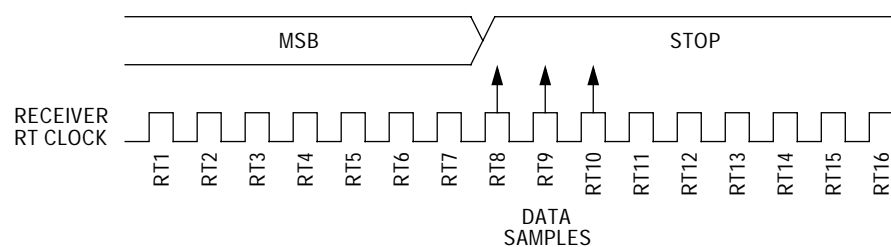
### Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples to fall outside the actual stop bit. Then a noise error occurs. If more than one of the samples is outside the stop bit, a framing error occurs. In most applications, the baud rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming character, it resynchronizes the RT clock on any valid falling edge within the character. Resynchronization within characters corrects misalignments between transmitter bit times and receiver bit times.

### Slow Data Tolerance

**Figure 9** shows how much a slow received character can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.



**Figure 9. Slow Data**

For an 8-bit character, data sampling of the stop bit takes the receiver  $9 \text{ bit times} \times 16 \text{ RT cycles} + 10 \text{ RT cycles} = 154 \text{ RT cycles}$ .



With the misaligned character shown in **Figure 9**, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 9 bit times  $\times$  16 RT cycles + 3 RT cycles = 147 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit character with no errors is

$$\left| \frac{154 - 147}{154} \right| \times 100 = 4.54\%$$

For a 9-bit character, data sampling of the stop bit takes the receiver 10 bit times  $\times$  16 RT cycles + 10 RT cycles = 170 RT cycles.

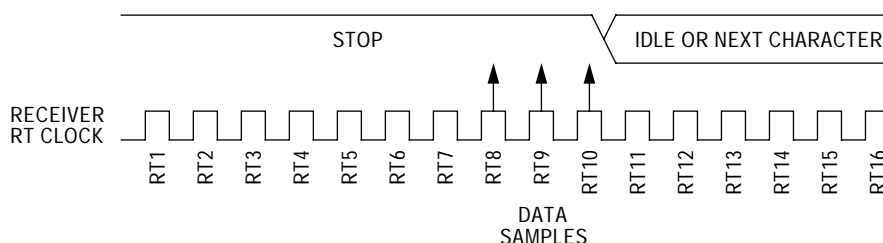
With the misaligned character shown in **Figure 9**, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 10 bit times  $\times$  16 RT cycles + 3 RT cycles = 163 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is

$$\left| \frac{170 - 163}{170} \right| \times 100 = 4.12\%$$

### Fast Data Tolerance

**Figure 10** shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.



**Figure 10. Fast Data**

For an 8-bit character, data sampling of the stop bit takes the receiver 9 bit times  $\times$  16 RT cycles + 10 RT cycles = 154 RT cycles.

## Serial Communications Interface Module (SCI)

With the misaligned character shown in [Figure 10](#), the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times  $\times$  16 RT cycles = 160 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is

$$\left| \frac{154 - 160}{154} \right| \times 100 = 3.90\%.$$

For a 9-bit character, data sampling of the stop bit takes the receiver 10 bit times  $\times$  16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in [Figure 10](#), the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times  $\times$  16 RT cycles = 176 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is

$$\left| \frac{170 - 176}{170} \right| \times 100 = 3.53\%.$$

### *Receiver Wakeup*

So that the MCU can ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in SCC2 puts the receiver into a standby state during which receiver interrupts are disabled.

Depending on the state of the WAKE bit in SCC1, either of two conditions on the RxD pin can bring the receiver out of the standby state:

- Address mark — An address mark is a logic 1 in the most significant bit position of a received character. When the WAKE bit is set, an address mark wakes the receiver from the standby state by clearing the RWU bit. The address mark also sets the SCI receiver full bit, SCRF. Software can then compare the character containing the address mark to the user-defined address of the receiver. If they are the same, the receiver remains awake and processes the characters that follow. If they are not the same, software can set the RWU bit and put the receiver back into the standby state.

- Idle input line condition — When the WAKE bit is clear, an idle character on the RxD pin wakes the receiver from the standby state by clearing the RWU bit. The idle character that wakes the receiver does not set the receiver idle bit, IDLE, or the SCI receiver full bit, SCRF. The idle line type bit, ILTY, determines whether the receiver begins counting logic 1s as idle character bits after the start bit or after the stop bit.

**NOTE:** *With the WAKE bit clear, setting the RWU bit after the RxD pin has been idle may cause the receiver to wake up immediately.*

*Receiver Interrupts*      The following sources can generate CPU interrupt requests from the SCI receiver:

- SCI receiver full (SCRF) — The SCRF bit in SCS1 indicates that the receive shift register has transferred a character to the SCDR. SCRF can generate a receiver CPU interrupt request. Setting the SCI receive interrupt enable bit, SCRIE, in SCC2 enables the SCRF bit to generate receiver CPU interrupts.
- Idle input (IDLE) — The IDLE bit in SCS1 indicates that 10 or 11 consecutive logic 1s shifted in from the RxD pin. The idle line interrupt enable bit, ILIE, in SCC2 enables the IDLE bit to generate CPU interrupt requests.

*Error Interrupts*      The following receiver error flags in SCS1 can generate CPU interrupt requests:

- Receiver overrun (OR) — The OR bit indicates that the receive shift register shifted in a new character before the previous character was read from the SCDR. The previous character remains in the SCDR, and the new character is lost. The overrun interrupt enable bit, ORIE, in SCC3 enables OR to generate SCI error CPU interrupt requests.
- Noise flag (NF) — The NF bit is set when the SCI detects noise on incoming data or break characters, including start, data, and stop bits. The noise error interrupt enable bit, NEIE, in SCC3 enables NF to generate SCI error CPU interrupt requests.

## Serial Communications Interface Module (SCI)

- Framing error (FE) — The FE bit in SCS1 is set when a logic 0 occurs where the receiver expects a stop bit. The framing error interrupt enable bit, FEIE, in SCC3 enables FE to generate SCI error CPU interrupt requests.
- Parity error (PE) — The PE bit in SCS1 is set when the SCI detects a parity error in incoming data. The parity error interrupt enable bit, PEIE, in SCC3 enables PE to generate SCI error CPU interrupt requests.

---

---

### Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

#### Wait Mode

The SCI module remains active in wait mode. Any enabled CPU interrupt request from the SCI module can bring the MCU out of wait mode.

If SCI module functions are not required during wait mode, reduce power consumption by disabling the module before executing the WAIT instruction.

#### Stop Mode

The SCI module is inactive in stop mode. The STOP instruction does not affect SCI register states. Any enabled CPU interrupt request from the SCI module does not bring the MCU out of stop mode. SCI module operation resumes after the MCU exits stop mode.

Because the internal clock is inactive during stop mode, entering stop mode during an SCI transmission or reception results in invalid data.

---

---

## SCI During Break Module Interrupts

The BCFE bit in the break flag control register (BF CR) enables software to clear status bits during the break state. (See [Break Module](#) on page 161).

To allow software to clear status bits during a break interrupt, write a logic 1 to the BCFE bit. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a two-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

---

---

## I/O Signals

Port E shares two of its pins with the SCI module. The two SCI I/O pins are:

- PTE0/SCTxD — Transmit data
- PTE1/SCRxD — Receive data

### **PTE0/SCTxD (Transmit Data)**

The PTE0/SCTxD pin is the serial data output from the SCI transmitter. The SCI shares the PTE0/SCTxD pin with port E. When the SCI is enabled, the PTE0/SCTxD pin is an output regardless of the state of the DDRE2 bit in data direction register E (DDRE).

### **PTE1/SCRxD (Receive Data)**

The PTE1/SCRxD pin is the serial data input to the SCI receiver. The SCI shares the PTE1/SCRxD pin with port E. When the SCI is enabled, the PTE1/SCRxD pin is an input regardless of the state of the DDRE1 bit in data direction register E (DDRE).

## Serial Communications Interface Module (SCI)

---

---

### I/O Registers

The following I/O registers control and monitor SCI operation:

- SCI control register 1 (SCC1)
- SCI control register 2 (SCC2)
- SCI control register 3 (SCC3)
- SCI status register 1 (SCS1)
- SCI status register 2 (SCS2)
- SCI data register (SCDR)
- SCI baud rate register (SCBR)

#### SCI Control Register 1

SCI control register 1:

- Enables loop mode operation
- Enables the SCI
- Controls output polarity
- Controls character length
- Controls SCI wakeup method
- Controls idle character detection
- Enables parity function
- Controls parity type

Address: \$0013

|        | Bit 7 | 6     | 5     | 4 | 3    | 2     | 1   | Bit 0 |
|--------|-------|-------|-------|---|------|-------|-----|-------|
| Read:  | LOOPS | ENSCI | TXINV | M | WAKE | ILLTY | PEN | PTY   |
| Write: |       |       |       |   |      |       |     |       |
| Reset: | 0     | 0     | 0     | 0 | 0    | 0     | 0   | 0     |

**Figure 11. SCI Control Register 1 (SCC1)**

#### LOOPS — Loop Mode Select Bit

This read/write bit enables loop mode operation. In loop mode the RxD pin is disconnected from the SCI, and the transmitter output goes into the receiver input. Both the transmitter and the receiver must be enabled to use loop mode. Reset clears the LOOPS bit.

1 = Loop mode enabled

0 = Normal operation enabled

#### ENSCI — Enable SCI Bit

This read/write bit enables the SCI and the SCI baud rate generator. Clearing ENSCI sets the SCTE and TC bits in SCI status register 1 and disables transmitter interrupts. Reset clears the ENSCI bit.

1 = SCI enabled

0 = SCI disabled

#### TXINV — Transmit Inversion Bit

This read/write bit reverses the polarity of transmitted data. Reset clears the TXINV bit.

1 = Transmitter output inverted

0 = Transmitter output not inverted

**NOTE:** *Setting the TXINV bit inverts all transmitted values, including idle, break, start, and stop bits.*

## Serial Communications Interface Module (SCI)

### M — Mode (Character Length) Bit

This read/write bit determines whether SCI characters are eight or nine bits long. (See [Table 8](#)). The ninth bit can serve as an extra stop bit, as a receiver wakeup signal, or as a parity bit. Reset clears the M bit.

1 = 9-bit SCI characters

0 = 8-bit SCI characters

### WAKE — Wakeup Condition Bit

This read/write bit determines which condition wakes up the SCI: a logic 1 (address mark) in the most significant bit position of a received character or an idle condition on the RxD pin. Reset clears the WAKE bit.

1 = Address mark wakeup

0 = Idle line wakeup

### ILTY — Idle Line Type Bit

This read/write bit determines when the SCI starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. Reset clears the ILTY bit.

1 = Idle character bit count begins after stop bit

0 = Idle character bit count begins after start bit

### PEN — Parity Enable Bit

This read/write bit enables the SCI parity function. (See [Table 8](#)). When enabled, the parity function inserts a parity bit in the most significant bit position. (See [Table 7](#)). Reset clears the PEN bit.

1 = Parity function enabled

0 = Parity function disabled



### PTY — Parity Bit

This read/write bit determines whether the SCI generates and checks for odd parity or even parity. (See [Table 8](#)). Reset clears the PTY bit.

- 1 = Odd parity
- 0 = Even parity

**NOTE:** *Changing the PTY bit in the middle of a transmission or reception can generate a parity error.*

**Table 8. Character Format Selection**

| Control Bits |         | Character Format |           |        |           |                  |
|--------------|---------|------------------|-----------|--------|-----------|------------------|
| M            | PEN:PTY | Start Bits       | Data Bits | Parity | Stop Bits | Character Length |
| 0            | 0X      | 1                | 8         | None   | 1         | 10 Bits          |
| 1            | 0X      | 1                | 9         | None   | 1         | 11 Bits          |
| 0            | 10      | 1                | 7         | Even   | 1         | 10 Bits          |
| 0            | 11      | 1                | 7         | Odd    | 1         | 10 Bits          |
| 1            | 10      | 1                | 8         | Even   | 1         | 11 Bits          |
| 1            | 11      | 1                | 8         | Odd    | 1         | 11 Bits          |

### SCI Control Register 2

SCI control register 2:

- Enables the following CPU interrupt requests:
  - Enables the SCTE bit to generate transmitter CPU interrupt requests
  - Enables the TC bit to generate transmitter CPU interrupt requests
  - Enables the SCRF bit to generate receiver CPU interrupt requests
  - Enables the IDLE bit to generate receiver CPU interrupt requests

## Serial Communications Interface Module (SCI)

- Enables the transmitter
- Enables the receiver
- Enables SCI wakeup
- Transmits SCI break characters

Address: \$0014

|        | Bit 7 | 6    | 5     | 4    | 3  | 2  | 1   | Bit 0 |
|--------|-------|------|-------|------|----|----|-----|-------|
| Read:  | SCTIE | TCIE | SCRIE | ILIE | TE | RE | RWU | SBK   |
| Write: |       |      |       |      |    |    |     |       |
| Reset: | 0     | 0    | 0     | 0    | 0  | 0  | 0   | 0     |

**Figure 12. SCI Control Register 2 (SCC2)**

### SCTIE — SCI Transmit Interrupt Enable Bit

This read/write bit enables the SCTE bit to generate SCI transmitter CPU interrupt requests. Setting the SCTIE bit in SCC3 enables the SCTE bit to generate CPU interrupt requests. Reset clears the SCTIE bit.

- 1 = SCTE enabled to generate CPU interrupt
- 0 = SCTE not enabled to generate CPU interrupt

### TCIE — Transmission Complete Interrupt Enable Bit

This read/write bit enables the TC bit to generate SCI transmitter CPU interrupt requests. Reset clears the TCIE bit.

- 1 = TC enabled to generate CPU interrupt requests
- 0 = TC not enabled to generate CPU interrupt requests

#### SCRIE — SCI Receive Interrupt Enable Bit

This read/write bit enables the SCRF bit to generate SCI receiver CPU interrupt requests. Setting the SCRIE bit in SCC3 enables the SCRF bit to generate CPU interrupt requests. Reset clears the SCRIE bit.

- 1 = SCRF enabled to generate CPU interrupt
- 0 = SCRF not enabled to generate CPU interrupt

#### ILIE — Idle Line Interrupt Enable Bit

This read/write bit enables the IDLE bit to generate SCI receiver CPU interrupt requests. Reset clears the ILIE bit.

- 1 = IDLE enabled to generate CPU interrupt requests
- 0 = IDLE not enabled to generate CPU interrupt requests

#### TE — Transmitter Enable Bit

Setting this read/write bit begins the transmission by sending a preamble of 10 or 11 logic 1s from the transmit shift register to the TxD pin. If software clears the TE bit, the transmitter completes any transmission in progress before the TxD returns to the idle condition (logic 1). Clearing and then setting TE during a transmission queues an idle character to be sent after the character currently being transmitted. Reset clears the TE bit.

- 1 = Transmitter enabled
- 0 = Transmitter disabled

**NOTE:** *Writing to the TE bit is not allowed when the enable SCI bit (ENSCI) is clear. ENSCI is in SCI control register 1.*

## Serial Communications Interface Module (SCI)

### RE — Receiver Enable Bit

Setting this read/write bit enables the receiver. Clearing the RE bit disables the receiver but does not affect receiver interrupt flag bits.

Reset clears the RE bit.

1 = Receiver enabled

0 = Receiver disabled

**NOTE:** *Writing to the RE bit is not allowed when the enable SCI bit (ENSCI) is clear. ENSCI is in SCI control register 1.*

### RWU — Receiver Wakeup Bit

This read/write bit puts the receiver in a standby state during which receiver interrupts are disabled. The WAKE bit in SCC1 determines whether an idle input or an address mark brings the receiver out of the standby state and clears the RWU bit. Reset clears the RWU bit.

1 = Standby state

0 = Normal operation

### SBK — Send Break Bit

Setting and then clearing this read/write bit transmits a break character followed by a logic 1. The logic 1 after the break character guarantees recognition of a valid start bit. If SBK remains set, the transmitter continuously transmits break characters with no logic 1s between them. Reset clears the SBK bit.

1 = Transmit break characters

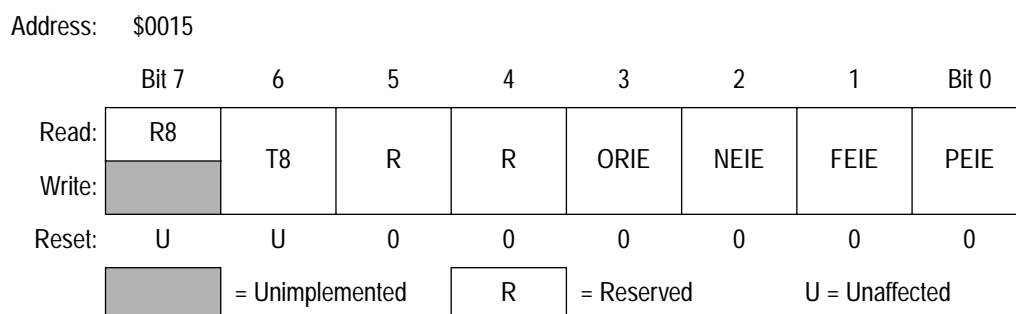
0 = No break characters being transmitted

**NOTE:** *Do not toggle the SBK bit immediately after setting the SCTE bit. Toggling SBK before the preamble begins causes the SCI to send a break character instead of a preamble.*

### SCI Control Register 3

SCI control register 3:

- Stores the ninth SCI data bit received and the ninth SCI data bit to be transmitted.
- Enables the following interrupts:
  - Receiver overrun interrupts
  - Noise error interrupts
  - Framing error interrupts
  - Parity error interrupts



**Figure 13. SCI Control Register 3 (SCC3)**

#### R8 — Received Bit 8

When the SCI is receiving 9-bit characters, R8 is the read-only ninth bit (bit 8) of the received character. R8 is received at the same time that the SCDR receives the other 8 bits.

When the SCI is receiving 8-bit characters, R8 is a copy of the eighth bit (bit 7). Reset has no effect on the R8 bit.

#### T8 — Transmitted Bit 8

When the SCI is transmitting 9-bit characters, T8 is the read/write ninth bit (bit 8) of the transmitted character. T8 is loaded into the transmit shift register at the same time that the SCDR is loaded into the transmit shift register. Reset has no effect on the T8 bit.

## Serial Communications Interface Module (SCI)

### ORIE — Receiver Overrun Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the receiver overrun bit, OR.

- 1 = SCI error CPU interrupt requests from OR bit enabled
- 0 = SCI error CPU interrupt requests from OR bit disabled

### NEIE — Receiver Noise Error Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the noise error bit, NE. Reset clears NEIE.

- 1 = SCI error CPU interrupt requests from NE bit enabled
- 0 = SCI error CPU interrupt requests from NE bit disabled

### FEIE — Receiver Framing Error Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the framing error bit, FE. Reset clears FEIE.

- 1 = SCI error CPU interrupt requests from FE bit enabled
- 0 = SCI error CPU interrupt requests from FE bit disabled

### PEIE — Receiver Parity Error Interrupt Enable Bit

This read/write bit enables SCI receiver CPU interrupt requests generated by the parity error bit, PE. Reset clears PEIE.

- 1 = SCI error CPU interrupt requests from PE bit enabled
- 0 = SCI error CPU interrupt requests from PE bit disabled

**SCI Status Register 1** SCI status register 1 contains flags to signal the following conditions:

- Transfer of SCDR data to transmit shift register complete
- Transmission complete
- Transfer of receive shift register data to SCDR complete
- Receiver input idle
- Receiver overrun
- Noisy data
- Framing error
- Parity error

Address: \$0016

|        | Bit 7 | 6  | 5    | 4    | 3  | 2  | 1  | Bit 0 |
|--------|-------|----|------|------|----|----|----|-------|
| Read:  | SCTE  | TC | SCRF | IDLE | OR | NF | FE | PE    |
| Write: |       |    |      |      |    |    |    |       |
| Reset: | 1     | 1  | 0    | 0    | 0  | 0  | 0  | 0     |

= Unimplemented

**Figure 14. SCI Status Register 1 (SCS1)**

#### SCTE — SCI Transmitter Empty Bit

This clearable, read-only bit is set when the SCDR transfers a character to the transmit shift register. SCCTE can generate an SCI transmitter CPU interrupt request. When the SCTIE bit in SCC2 is set, SCCTE generates an SCI transmitter CPU interrupt request. In normal operation, clear the SCCTE bit by reading SCS1 with SCCTE set and then writing to SCDR. Reset sets the SCCTE bit.

- 1 = SCDR data transferred to transmit shift register
- 0 = SCDR data not transferred to transmit shift register

## Serial Communications Interface Module (SCI)

### TC — Transmission Complete Bit

This read-only bit is set when the SCTE bit is set, and no data, preamble, or break character is being transmitted. TC generates an SCI transmitter CPU interrupt request if the TCIE bit in SCC2 is also set. TC is cleared automatically when data, preamble, or break is queued and ready to be sent. There may be up to 1.5 transmitter clocks of latency between queuing data, preamble, and break and the transmission actually starting. Reset sets the TC bit.

1 = No transmission in progress

0 = Transmission in progress

### SCRF — SCI Receiver Full Bit

This clearable, read-only bit is set when the data in the receive shift register transfers to the SCI data register. SCRF can generate an SCI receiver CPU interrupt request. When the SCRIE bit in SCC2 is set the SCRF generates a CPU interrupt request. In normal operation, clear the SCRF bit by reading SCS1 with SCRF set and then reading the SCDR. Reset clears SCRF.

1 = Received data available in SCDR

0 = Data not available in SCDR

### IDLE — Receiver Idle Bit

This clearable, read-only bit is set when 10 or 11 consecutive logic 1s appear on the receiver input. IDLE generates an SCI error CPU interrupt request if the ILIE bit in SCC2 is also set. Clear the IDLE bit by reading SCS1 with IDLE set and then reading the SCDR. After the receiver is enabled, it must receive a valid character that sets the SCRF bit before an idle condition can set the IDLE bit. Also, after the IDLE bit has been cleared, a valid character must again set the SCRF bit before an idle condition can set the IDLE bit. Reset clears the IDLE bit.

1 = Receiver input idle

0 = Receiver input active (or idle since the IDLE bit was cleared)

### OR — Receiver Overrun Bit

This clearable, read-only bit is set when software fails to read the SCDR before the receive shift register receives the next character. The OR bit generates an SCI error CPU interrupt request if the ORIE

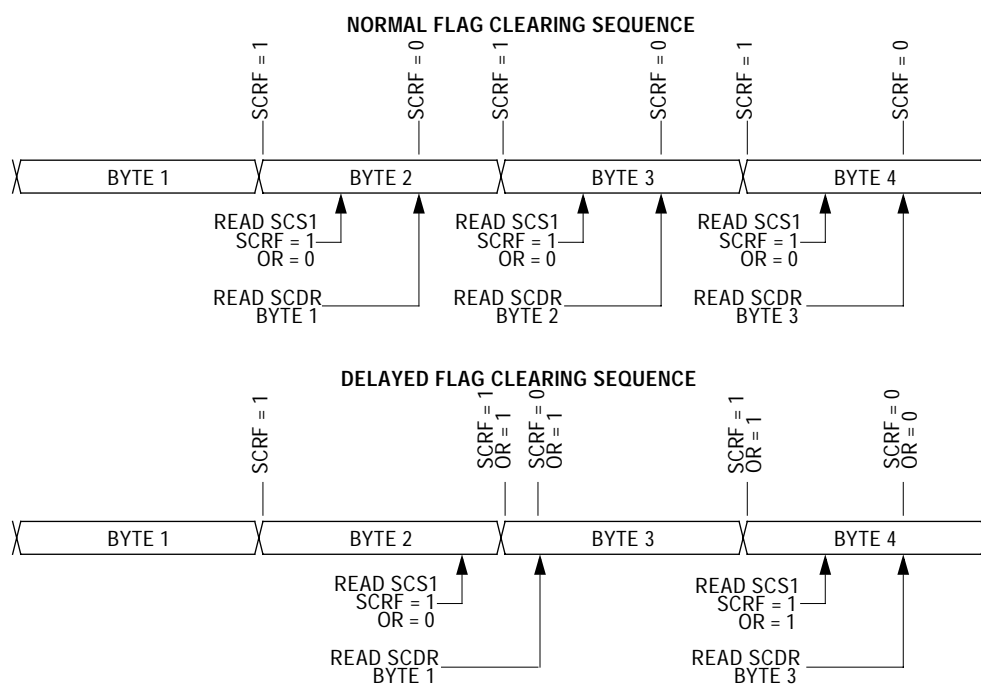


bit in SCC3 is also set. The data in the shift register is lost, but the data already in the SCDR is not affected. Clear the OR bit by reading SCS1 with OR set and then reading the SCDR. Reset clears the OR bit.

- 1 = Receive shift register full and SCRF = 1
- 0 = No receiver overrun

Software latency may allow an overrun to occur between reads of SCS1 and SCDR in the flag-clearing sequence. **Figure 15** shows the normal flag-clearing sequence and an example of an overrun caused by a delayed read of SCDR. The delayed read of SCDR does not clear the OR bit because OR was not set when SCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the SCDR instead of byte 2.

In applications that are subject to software latency or in which it is important to know which byte is lost due to an overrun, the flag-clearing routine can check the OR bit in a second read of SCS1 after reading the data register.



**Figure 15. Flag Clearing Sequence**

## Serial Communications Interface Module (SCI)

### NF — Receiver Noise Flag Bit

This clearable, read-only bit is set when the SCI detects noise on the RxD pin. NF generates an NF CPU interrupt request if the NEIE bit in SCC3 is also set. Clear the NF bit by reading SCS1 and then reading the SCDR. Reset clears the NF bit.

1 = Noise detected

0 = No noise detected

### FE — Receiver Framing Error Bit

This clearable, read-only bit is set when a logic 0 is accepted as the stop bit. FE generates an SCI error CPU interrupt request if the FEIE bit in SCC3 also is set. Clear the FE bit by reading SCS1 with FE set and then reading the SCDR. Reset clears the FE bit.

1 = Framing error detected

0 = No framing error detected

### PE — Receiver Parity Error Bit

This clearable, read-only bit is set when the SCI detects a parity error in incoming data. PE generates a PE CPU interrupt request if the PEIE bit in SCC3 is also set. Clear the PE bit by reading SCS1 with PE set and then reading the SCDR. Reset clears the PE bit.

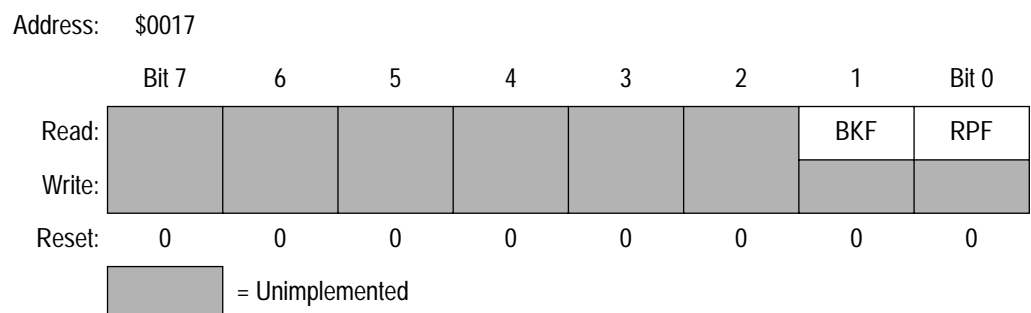
1 = Parity error detected

0 = No parity error detected

### SCI Status Register 2

SCI status register 2 contains flags to signal the following conditions:

- Break character detected
- Incoming data



**Figure 16. SCI Status Register 2 (SCS2)**

### BKF — Break Flag Bit

This clearable, read-only bit is set when the SCI detects a break character on the RxD pin. In SCS1, the FE and SCRF bits are also set. In 9-bit character transmissions, the R8 bit in SCC3 is cleared. BKF does not generate a CPU interrupt request. Clear BKF by reading SCS2 with BKF set and then reading the SCDR. Once cleared, BKF can become set again only after logic 1s again appear on the RxD pin followed by another break character. Reset clears the BKF bit.

- 1 = Break character detected
- 0 = No break character detected

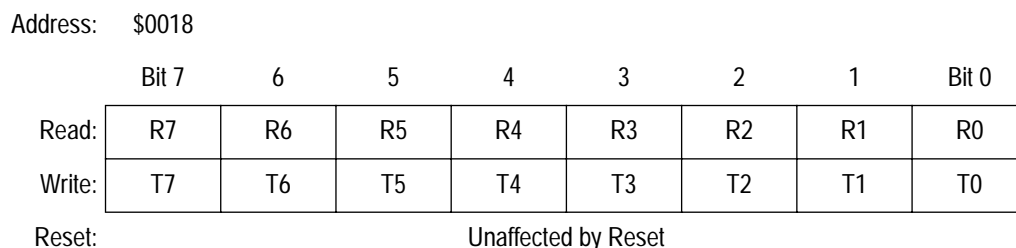
### RPF — Reception in Progress Flag Bit

This read-only bit is set when the receiver detects a logic 0 during the RT1 time period of the start bit search. RPF does not generate an interrupt request. RPF is reset after the receiver detects false start bits (usually from noise or a baud rate mismatch), or when the receiver detects an idle character. Polling RPF before disabling the SCI module or entering stop mode can show whether a reception is in progress.

- 1 = Reception in progress
- 0 = No reception in progress

## SCI Data Register

The SCI data register is the buffer between the internal data bus and the receive and transmit shift registers. Reset has no effect on data in the SCI data register.



**Figure 17. SCI Data Register (SCDR)**

## Serial Communications Interface Module (SCI)

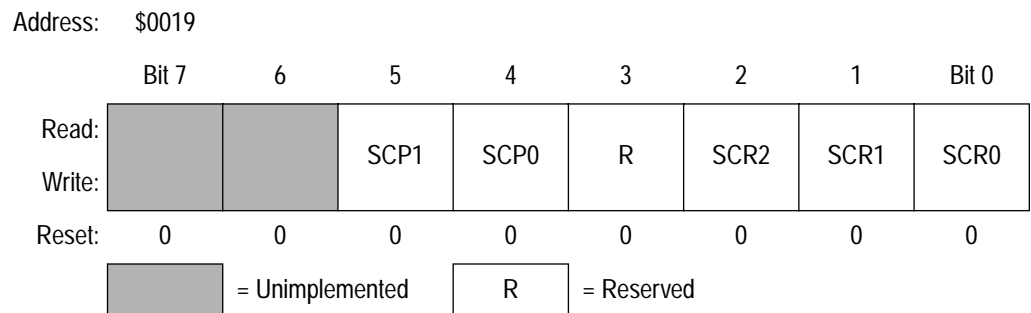
### R7/T7:R0/T0 — Receive/Transmit Data Bits

Reading address \$0018 accesses the read-only received data bits, R7:R0. Writing to address \$0018 writes the data to be transmitted, T7:T0. Reset has no effect on the SCI data register.

**NOTE:** Do not use read-modify-write instructions on the SCI data register.

### SCI Baud Rate Register

The baud rate register selects the baud rate for both the receiver and the transmitter.



**Figure 18. SCI Baud Rate Register (SCBR)**

### SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in [Table 9](#). Reset clears SCP1 and SCP0.

**Table 9. SCI Baud Rate Prescaling**

| SCP[1:0] | Prescaler Divisor (PD) |
|----------|------------------------|
| 00       | 1                      |
| 01       | 3                      |
| 10       | 4                      |
| 11       | 13                     |

SCR2 – SCR0 — SCI Baud Rate Select Bits

These read/write bits select the SCI baud rate divisor as shown in [Table 10](#). Reset clears SCR2–SCR0.

**Table 10. SCI Baud Rate Selection**

| SCR[2:1:0] | Baud Rate Divisor (BD) |
|------------|------------------------|
| 000        | 1                      |
| 001        | 2                      |
| 010        | 4                      |
| 011        | 8                      |
| 100        | 16                     |
| 101        | 32                     |
| 110        | 64                     |
| 111        | 128                    |

Use the following formula to calculate the SCI baud rate:

$$\text{Baud rate} = \frac{f_{\text{Crystal}}}{64 \times \text{PD} \times \text{BD}}$$

where:

- $f_{\text{Crystal}}$  = crystal frequency
- PD = prescaler divisor
- BD = baud rate divisor

[Table 11](#) shows the SCI baud rates that can be generated with a 4.9152-MHz crystal.

## Serial Communications Interface Module (SCI)

**Table 11. SCI Baud Rate Selection Examples**

| SCP[1:0] | Prescaler Divisor (PD) | SCR[2:1:0] | Baud Rate Divisor (BD) | Baud Rate (f <sub>Crystal</sub> = 4.9152 MHz) |
|----------|------------------------|------------|------------------------|---|
| 00       | 1                      | 000        | 1                      | 76,800  |
| 00       | 1                      | 001        | 2                      | 38,400  |
| 00       | 1                      | 010        | 4                      | 19,200  |
| 00       | 1                      | 011        | 8                      | 9600  |
| 00       | 1                      | 100        | 16                     | 4800  |
| 00       | 1                      | 101        | 32                     | 2400  |
| 00       | 1                      | 110        | 64                     | 1200  |
| 00       | 1                      | 111        | 128                    | 600   |
| 01       | 3                      | 000        | 1                      | 25,600  |
| 01       | 3                      | 001        | 2                      | 12,800  |
| 01       | 3                      | 010        | 4                      | 6400  |
| 01       | 3                      | 011        | 8                      | 3200  |
| 01       | 3                      | 100        | 16                     | 1600  |
| 01       | 3                      | 101        | 32                     | 800   |
| 01       | 3                      | 110        | 64                     | 400   |
| 01       | 3                      | 111        | 128                    | 200   |
| 10       | 4                      | 000        | 1                      | 19,200  |
| 10       | 4                      | 001        | 2                      | 9600  |
| 10       | 4                      | 010        | 4                      | 4800  |
| 10       | 4                      | 011        | 8                      | 2400  |
| 10       | 4                      | 100        | 16                     | 1200  |
| 10       | 4                      | 101        | 32                     | 600   |
| 10       | 4                      | 110        | 64                     | 300   |
| 10       | 4                      | 111        | 128                    | 150   |
| 11       | 13                     | 000        | 1                      | 5908  |
| 11       | 13                     | 001        | 2                      | 2954  |
| 11       | 13                     | 010        | 4                      | 1477  |
| 11       | 13                     | 011        | 8                      | 739   |
| 11       | 13                     | 100        | 16                     | 369   |
| 11       | 13                     | 101        | 32                     | 185   |
| 11       | 13                     | 110        | 64                     | 92  |
| 11       | 13                     | 111        | 128                    | 46  |

# Serial Peripheral Interface Module (SPI)

---

---

## Contents

|  |     |
|--|-----|
| Introduction . . . . .                           | 238 |
| Features . . . . .                               | 238 |
| Pin Name and Register Name Conventions . . . . . | 239 |
| Functional Description. . . . .                  | 240 |
| Master Mode. . . . .                             | 242 |
| Slave Mode. . . . .                              | 243 |
| Transmission Formats. . . . .                    | 244 |
| Clock Phase and Polarity Controls . . . . .      | 244 |
| Transmission Format When CPHA = 0. . . . .       | 245 |
| Transmission Format When CPHA = 1. . . . .       | 246 |
| Transmission Initiation Latency. . . . .         | 247 |
| Error Conditions . . . . .                       | 249 |
| Overflow Error. . . . .                          | 249 |
| Mode Fault Error. . . . .                        | 251 |
| Interrupts . . . . .                             | 253 |
| Queuing Transmission Data . . . . .              | 255 |
| Resetting the SPI . . . . .                      | 256 |
| Low-Power Modes . . . . .                        | 257 |
| Wait Mode. . . . .                               | 257 |
| Stop Mode. . . . .                               | 257 |
| SPI During Break Interrupts . . . . .            | 258 |
| I/O Signals. . . . .                             | 259 |
| MISO (Master In/Slave Out) . . . . .             | 259 |
| MOSI (Master Out/Slave In) . . . . .             | 260 |
| SPSCK (Serial Clock) . . . . .                   | 260 |
| SS (Slave Select) . . . . .                      | 260 |
| VSS (Clock Ground) . . . . .                     | 261 |
| I/O Registers . . . . .                          | 262 |
| SPI Control Register . . . . .                   | 262 |
| SPI Status and Control Register. . . . .         | 264 |
| SPI Data Register . . . . .                      | 267 |

# Serial Peripheral Interface Module (SPI)

---

---

## Introduction

This section describes the serial peripheral interface (SPI) module, which allows full-duplex, synchronous, serial communications with peripheral devices.

---

---

## Features

Features of the SPI module include:

- Full-Duplex Operation
- Master and Slave Modes
- Double-Buffered Operation with Separate Transmit and Receive Registers
- Four Master Mode Frequencies (Maximum = Bus Frequency  $\div$  2)
- Maximum Slave Mode Frequency = Bus Frequency
- Serial Clock with Programmable Polarity and Phase
- Two Separately Enabled Interrupts with CPU Service:
  - SPRF (SPI Receiver Full)
  - SPTE (SPI Transmitter Empty)
- Mode Fault Error Flag with CPU Interrupt Capability
- Overflow Error Flag with CPU Interrupt Capability
- Programmable Wired-OR Mode
- I<sup>2</sup>C (Inter-Integrated Circuit) Compatibility



---



---

## Pin Name and Register Name Conventions

The generic names of the SPI input/output (I/O) pins are:

- $\overline{SS}$  (slave select)
- SPSCCK (SPI serial clock)
- MOSI (master out slave in)
- MISO (master in slave out)

The SPI shares four I/O pins with a parallel I/O port. The full name of an SPI pin reflects the name of the shared port pin. [Table 1](#) shows the full names of the SPI I/O pins. The generic pin names appear in the text that follows.

**Table 1. Pin Name Conventions**

|                             |             |             |                                   |               |
|-----------------------------|-------------|-------------|-----------------------------------|---------------|
| <b>SPI Generic Pin Name</b> | <b>MISO</b> | <b>MOSI</b> | <b><math>\overline{SS}</math></b> | <b>SPSCCK</b> |
| <b>Full SPI Pin Name</b>    | PTE5/MISO   | PTE6/MOSI   | PTE4/ $\overline{SS}$             | PTE7/SPSCCK   |

The generic names of the SPI I/O registers are:

- SPI control register (SPCR)
- SPI status and control register (SPSCR)
- SPI data register (SPDR)

[Table 2](#) shows the names and the addresses of the SPI I/O registers.

**Table 2. I/O Register Addresses**

| <b>Register Name</b>                    | <b>Address</b> |
|---|----------------|
| SPI Control Register (SPCR)             | \$0010         |
| SPI Status and Control Register (SPSCR) | \$0011         |
| SPI Data Register (SPDR)                | \$0012         |

# Serial Peripheral Interface Module (SPI)

## Functional Description

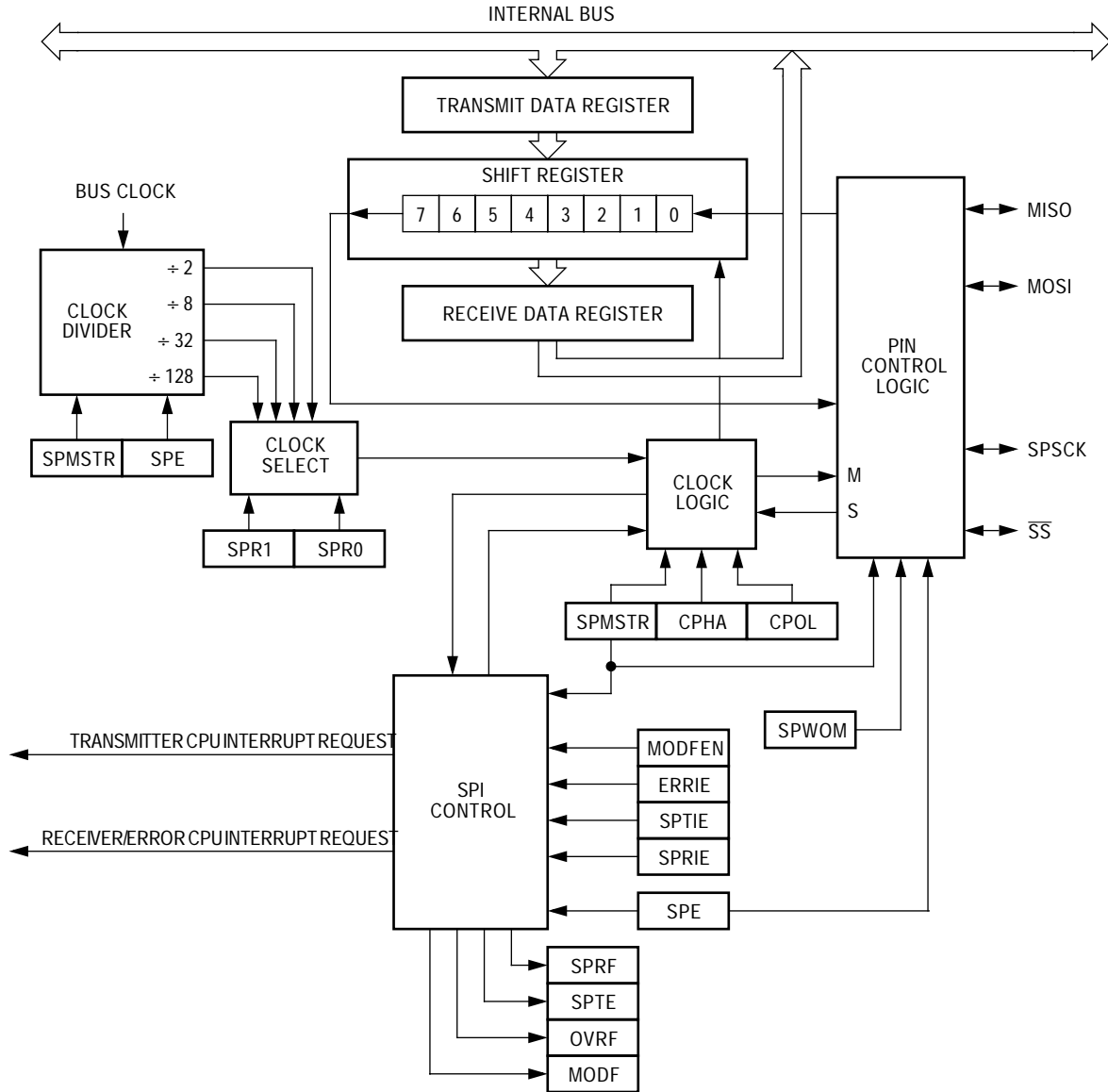
**Table 3** summarizes the SPI I/O registers and **Figure 1** shows the structure of the SPI module.

**Table 3. SPI I/O Register Summary**

| Addr   | Register Name                           | R/W    | Bit 7               | 6     | 5      | 4    | 3     | 2      | 1    | Bit 0 |
|--------|---|--------|---------------------|-------|--------|------|-------|--------|------|-------|
| \$0010 | SPI Control Register (SPCR)             | Read:  | SPRIE               | R     | SPMSTR | CPOL | CPHA  | SPWOM  | SPE  | SPTIE |
|        |   | Write: |                     |       |        |      |       |        |      |       |
|        |   | Reset: | 0                   | 0     | 1      | 0    | 1     | 0      | 0    | 0     |
| \$0011 | SPI Status and Control Register (SPSCR) | Read:  | SPRF                | ERRIE | OVRF   | MODF | SPTIE | MODFEN | SPR1 | SPR0  |
|        |   | Write: | R                   |       | R      | R    |       |        |      |       |
|        |   | Reset: | 0                   | 0     | 0      | 0    | 1     | 0      | 0    | 0     |
| \$0012 | SPI Data Register (SPDR)                | Read:  | R7                  | R6    | R5     | R4   | R3    | R2     | R1   | R0    |
|        |   | Write: | T7                  | T6    | T5     | T4   | T3    | T2     | T1   | T0    |
|        |   | Reset: | Unaffected by Reset |       |        |      |       |        |      |       |

R = Reserved

Serial Peripheral Interface Module (SPI)  
Functional Description



**Figure 1. SPI Module Block Diagram**

The SPI module allows full-duplex, synchronous, serial communication between the MCU and peripheral devices, including other MCUs. Software can poll the SPI status flags or SPI operation can be interrupt driven. All SPI interrupts can be serviced by the CPU.

The following paragraphs describe the operation of the SPI module.

## Serial Peripheral Interface Module (SPI)

### Master Mode

The SPI operates in master mode when the SPI master bit, SPMSTR (SPCR \$0010), is set.

**NOTE:** Configure the SPI modules as master and slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. See [SPI Control Register](#) on page 262.

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the SPI data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE (SPSCR \$0011). The byte begins shifting out on the MOSI pin under the control of the serial clock. (See [Table 4](#)).

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See [SPI Status and Control Register](#) on page 264). Through the SPSCCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

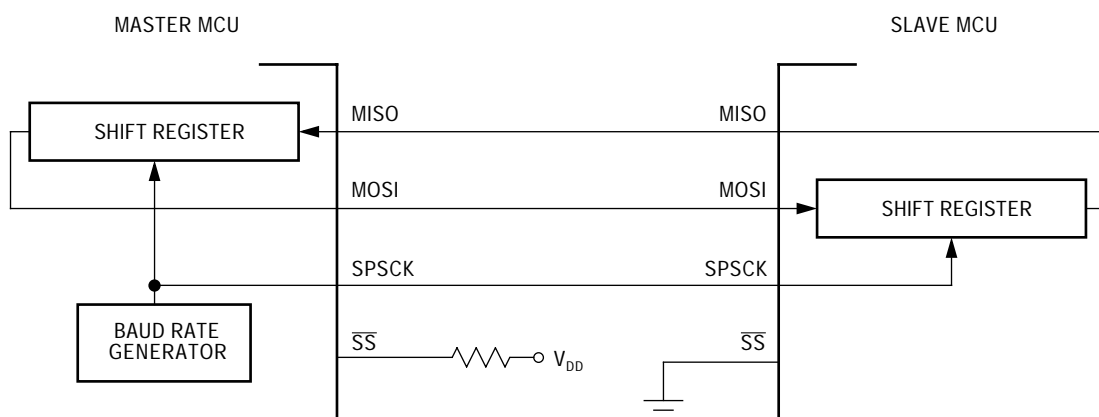


Figure 2. Full-Duplex Master-Slave Connections

As the byte shifts out on the MOSI pin of the master, another byte shifts in from the slave on the master's MISO pin. The transmission ends when the receiver full bit, SPRF (SPSCR), becomes set. At the same time that SPRF becomes set, the byte from the slave transfers to the receive data register. In normal operation, SPRF signals the end of a transmission. Software clears SPRF by reading the SPI status and control register and then reading the SPI data register. Writing to the SPI data register clears the SPTIE bit.

## Slave Mode

The SPI operates in slave mode when the SPMSTR bit (SPCR, \$0010) is clear. In slave mode the SPSCCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the  $\overline{SS}$  pin of the slave MCU must be at logic 0.  $\overline{SS}$  must remain low until the transmission is complete. (See [Mode Fault Error](#) on page 251).

In a slave SPI module, data enters the shift register under the control of the serial clock from the master SPI module. After a byte enters the shift register of a slave SPI, it is transferred to the receive data register, and the SPRF bit (SPSCR) is set. To prevent an overflow condition, slave software then must read the SPI data register before another byte enters the shift register.

The maximum frequency of the SPSCCK for an SPI configured as a slave is the bus clock speed, which is twice as fast as the fastest master SPSCCK clock that can be generated. The frequency of the SPSCCK for an SPI configured as a slave does not have to correspond to any SPI baud rate. The baud rate only controls the speed of the SPSCCK generated by an SPI configured as a master. Therefore, the frequency of the SPSCCK for an SPI configured as a slave can be any frequency less than or equal to the bus speed.

When the master SPI starts a transmission, the data in the slave shift register begins shifting out on the MISO pin. The slave can load its shift register with a new byte for the next transmission by writing to its transmit data register. The slave must write to its transmit data register at least one bus cycle before the master starts the next transmission. Otherwise the byte already in the slave shift register shifts out on the MISO pin.

## Serial Peripheral Interface Module (SPI)

Data written to the slave shift register during a transmission remains in a buffer until the end of the transmission.

When the clock phase bit (CPHA) is set, the first edge of SPSCCK starts a transmission. When CPHA is clear, the falling edge of  $\overline{SS}$  starts a transmission. (See [Transmission Formats](#) on page 244).

If the write to the data register is late, the SPI transmits the data already in the shift register from the previous transmission.

**NOTE:** *To prevent SPSCCK from appearing as a clock edge, SPSCCK must be in the proper idle state before the slave is enabled.*

---

---

### Transmission Formats

During an SPI transmission, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line synchronizes shifting and sampling on the two serial data lines. A slave select line allows individual selection of a slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. On a master SPI device, the slave select line can be used optionally to indicate a multiple-master bus contention.

#### Clock Phase and Polarity Controls

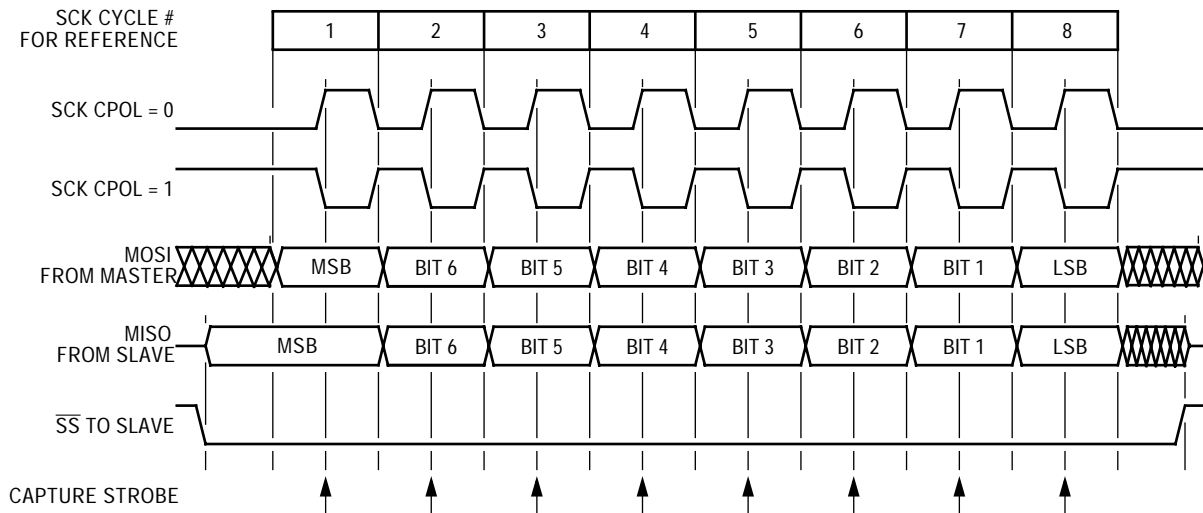
Software can select any of four combinations of serial clock (SCK) phase and polarity using two bits in the SPI control register (SPCR). The clock polarity is specified by the CPOL control bit, which selects an active high or low clock and has no significant effect on the transmission format.

The clock phase (CPHA) control bit (SPCR) selects one of two fundamentally different transmission formats. The clock phase and polarity should be identical for the master SPI device and the communicating slave device. In some cases, the phase and polarity are changed between transmissions to allow a master device to communicate with peripheral slaves having different requirements.

**NOTE:** *Before writing to the CPOL bit or the CPHA bit (SPCR), disable the SPI by clearing the SPI enable bit (SPE).*

**Transmission  
Format When  
CPHA = 0**

**Figure 3** shows an SPI transmission in which CPHA (SPCR) is logic 0. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The  $\overline{SS}$  line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ( $\overline{SS}$ ) is at logic 0, so that only the selected slave drives to the master. The  $\overline{SS}$  pin of the master is not shown but is assumed to be inactive. The  $\overline{SS}$  pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI (see [Mode Fault Error](#) on page 251). When CPHA = 0, the first SPSCCK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SPSCCK edge, and a falling edge on the  $\overline{SS}$  pin is used to start the transmission. The  $\overline{SS}$  pin must be toggled high and then low again between each byte transmitted.

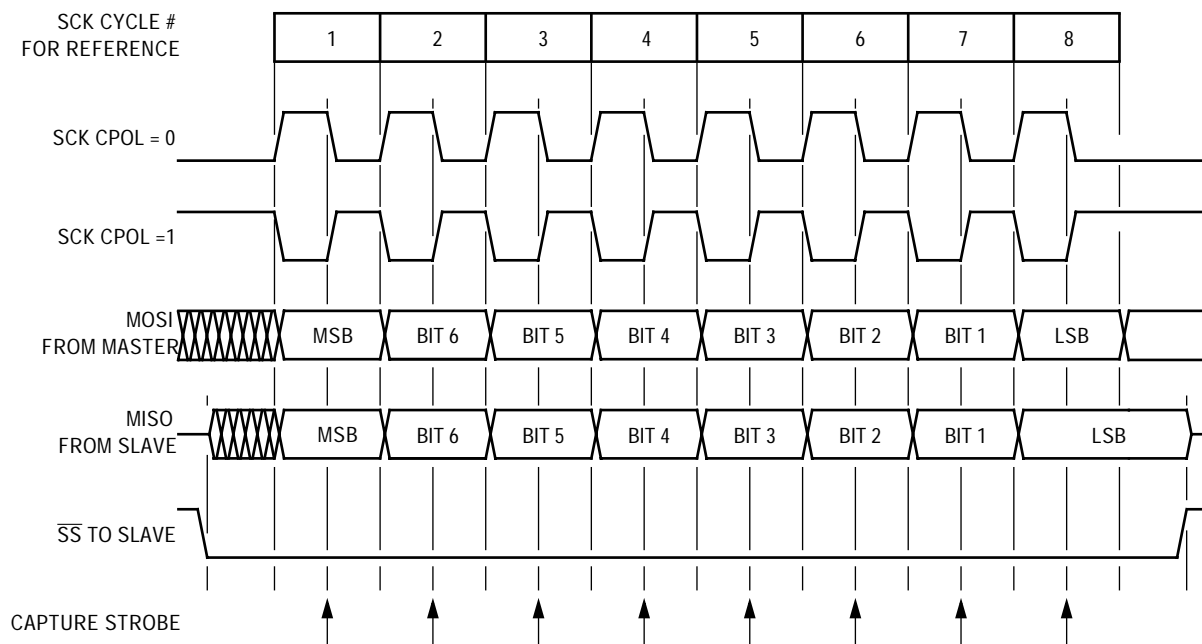


**Figure 3. Transmission Format (CPHA = 0)**

## Serial Peripheral Interface Module (SPI)

### Transmission Format When CPHA = 1

**Figure 4** shows an SPI transmission in which CPHA (SPCR) is logic 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The  $\overline{SS}$  line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ( $\overline{SS}$ ) is at logic 0, so that only the selected slave drives to the master. The  $\overline{SS}$  pin of the master is not shown but is assumed to be inactive. The  $\overline{SS}$  pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See [Mode Fault Error](#) on page 251). When CPHA = 1, the master begins driving its MOSI pin on the first SPSCK edge. Therefore, the slave uses the first SPSCK edge as a start transmission signal. The  $\overline{SS}$  pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.



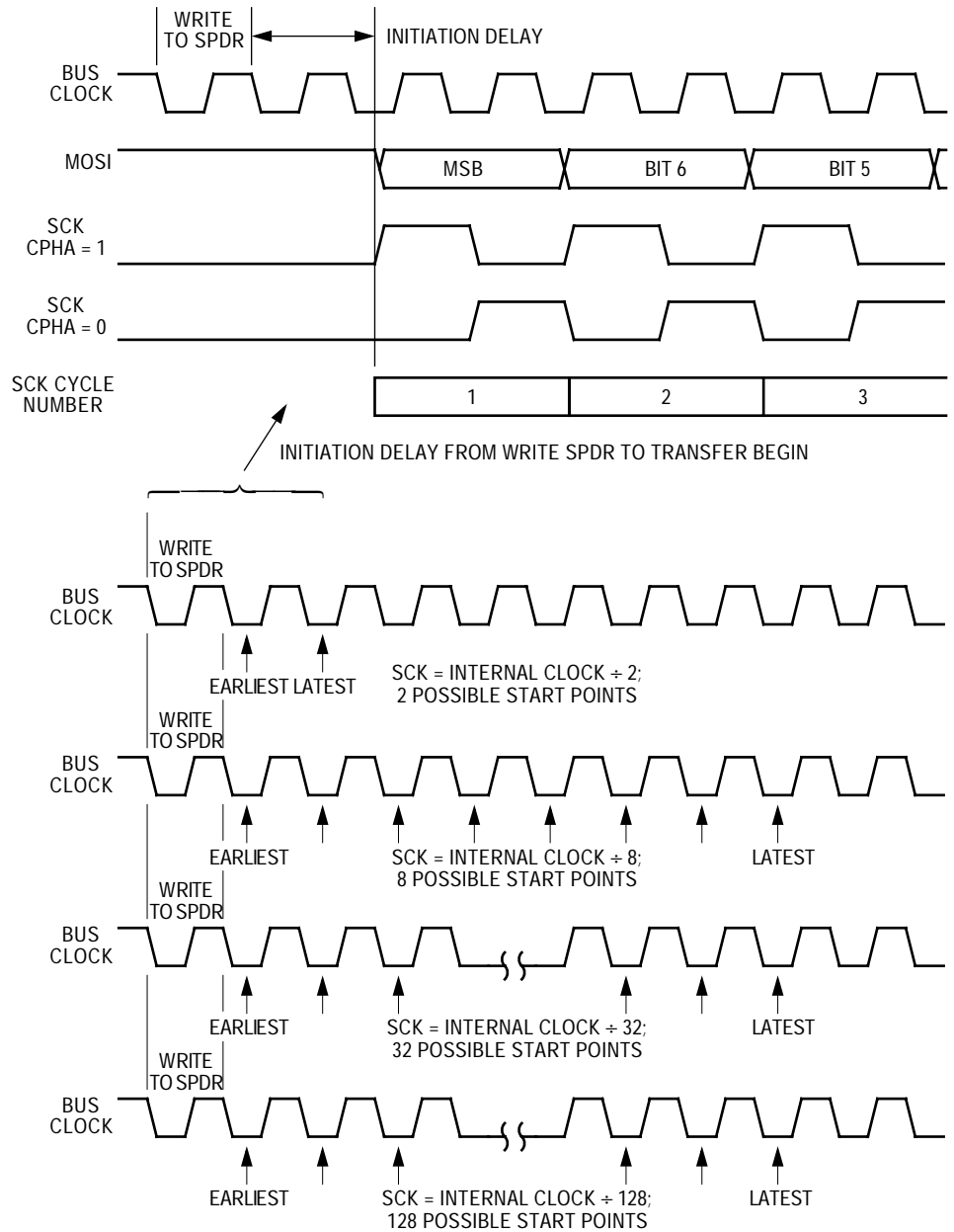
**Figure 4. Transmission Format (CPHA = 1)**



**Transmission  
Initiation Latency**

When the SPI is configured as a master ( $SPMSTR = 1$ ), transmissions are started by a software write to the SPDR (\$0012). CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SCK signal. When  $CPHA = 0$ , the SCK signal remains inactive for the first half of the first SCK cycle. When  $CPHA = 1$ , the first SCK cycle begins with an edge on the SCK line from its inactive to its active level. The SPI clock rate (selected by  $SPR1$ – $SPR0$ ) affects the delay from the write to SPDR and the start of the SPI transmission. (See [Figure 5](#)). The internal SPI clock in the master is a free-running derivative of the internal MCU clock. It is only enabled when both the SPE and SPMSTR bits (SPCR) are set to conserve power. SCK edges occur half way through the low time of the internal MCU clock. Since the SPI clock is free-running, it is uncertain where the write to the SPDR will occur relative to the slower SCK. This uncertainty causes the variation in the initiation delay shown in [Figure 5](#). This delay will be no longer than a single SPI bit time. That is, the maximum delay between the write to SPDR and the start of the SPI transmission is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

# Serial Peripheral Interface Module (SPI)



**Figure 5. Transmission Start Delay (Master)**

---

---

## Error Conditions

Two flags signal SPI error conditions:

1. Overflow (OVRFin SPSCR) — Failing to read the SPI data register before the next byte enters the shift register sets the OVRF bit. The new byte does not transfer to the receive data register, and the unread byte still can be read by accessing the SPI data register. OVRF is in the SPI status and control register.
2. Mode fault error (MODF in SPSCR) — The MODF bit indicates that the voltage on the slave select pin ( $\overline{SS}$ ) is inconsistent with the mode of the SPI. MODF is in the SPI status and control register.

### Overflow Error

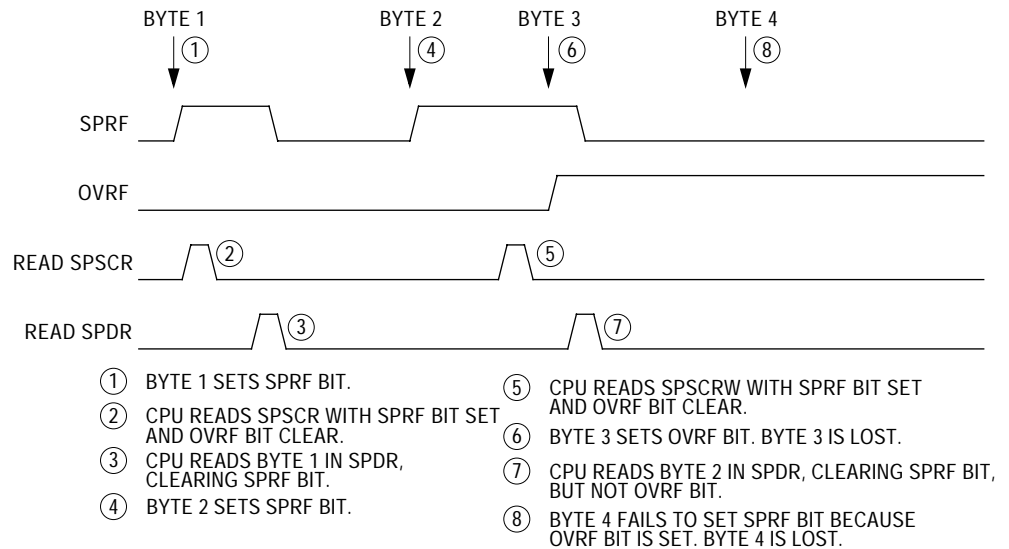
The overflow flag (OVRF in SPSCR) becomes set if the SPI receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. (See [Figure 3](#) and [Figure 4](#).) If an overflow occurs, the data being received is not transferred to the receive data register so that the unread data can still be read. Therefore, an overflow error always indicates the loss of data.

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE in SPSCR) is also set. MODF and OVRF can generate a receiver/error CPU interrupt request. (See [Figure 8](#)). It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

If an end-of-block transmission interrupt was meant to pull the MCU out of wait, having an overflow condition without overflow interrupts enabled causes the MCU to hang in wait mode. If the OVRF is enabled to generate an interrupt, it can pull the MCU out of wait mode instead.

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. [Figure 6](#) shows how it is possible to miss an overflow.

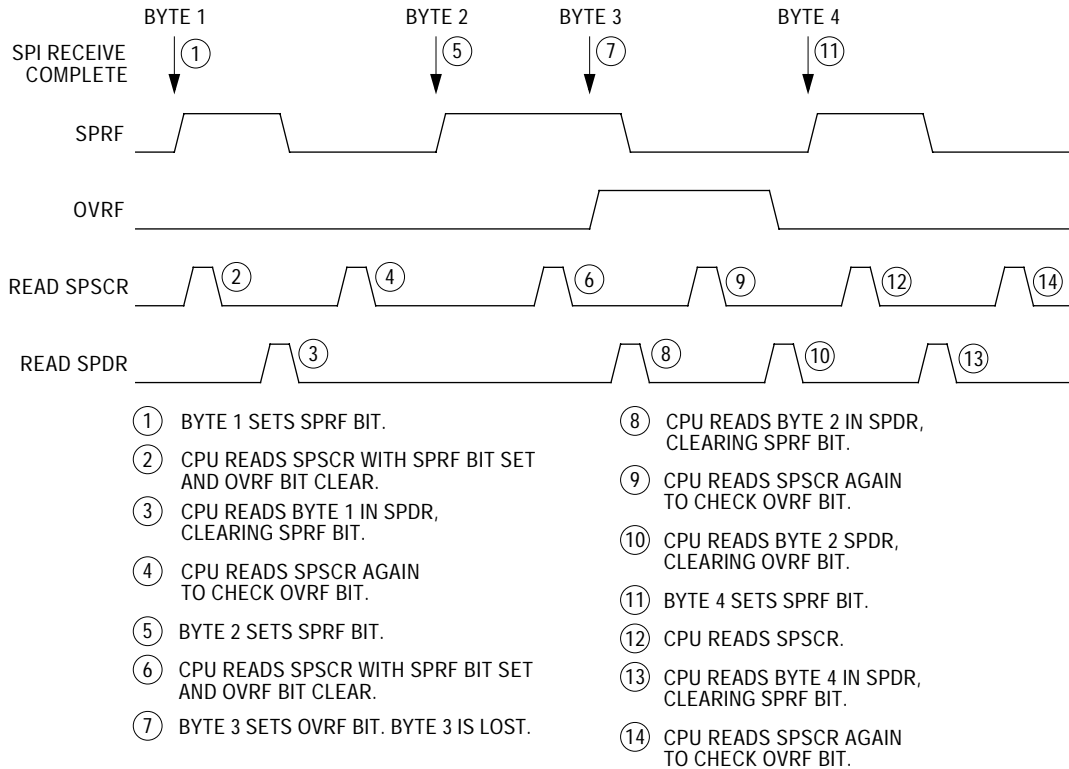
## Serial Peripheral Interface Module (SPI)



**Figure 6. Missed Read of Overflow Condition**

The first part of [Figure 6](#) shows how to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF flag can be set in between the time that SPSCR and SPDR are read.

In this case, an overflow can be easily missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it will not be obvious that bytes are being lost as more transmissions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPSCR after the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions will complete with an SPRF interrupt. [Figure 7](#) illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit (SPSCR).



**Figure 7. Clearing SPRF When OVRF Interrupt Is Not Enabled**

### Mode Fault Error

For the MODF flag (in SPSCR) to be set, the mode fault error enable bit (MODFEN in SPSCR) must be set. Clearing the MODFEN bit does not clear the MODF flag but does prevent MODF from being set again after MODF is cleared.

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE in SPSCR) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. MODF and OVRF can generate a receiver/error CPU interrupt request. (See [Figure 8](#)). It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

## Serial Peripheral Interface Module (SPI)

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if  $\overline{SS}$  goes to logic 0. A mode fault in a master SPI causes the following events to occur:

- If ERRIE = 1, the SPI generates an SPI receiver/error CPU interrupt request.
- The SPE bit is cleared.
- The SPTE bit is set.
- The SPI state counter is cleared.
- The data direction register of the shared I/O port regains control of port drivers.

**NOTE:** *To prevent bus contention with another master SPI after a mode fault error, clear all data direction register (DDR) bits associated with the SPI shared port pins.*

**NOTE:** *Setting the MODF flag (SPSCR) does not clear the SPMSTR bit. Reading SPMSTR when MODF = 1 will indicate a MODE fault error occurred in either master mode or slave mode.*

When configured as a slave (SPMSTR = 0), the MODF flag is set if  $\overline{SS}$  goes high during a transmission. When CPHA = 0, a transmission begins when  $\overline{SS}$  goes low and ends once the incoming SPSCCK returns to its idle level after the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCCK leaves its idle level and  $\overline{SS}$  is already low. The transmission continues until the SPSCCK returns to its IDLE level after the shift of the last data bit. (See [Transmission Formats](#) on page 244).

**NOTE:** *When CPHA = 0, a MODF occurs if a slave is selected ( $\overline{SS}$  is at logic 0) and later unselected ( $\overline{SS}$  is at logic 1) even if no SPSCCK is sent to that slave. This happens because  $\overline{SS}$  at logic 0 indicates the start of the transmission (MISO driven out with the value of MSB) for CPHA = 0. When CPHA = 1, a slave can be selected and then later unselected with no transmission occurring. Therefore, MODF does not occur since a transmission was never begun.*

In a slave SPI (MSTR = 0), the MODF bit generates an SPI receiver/error CPU interrupt request if the ERRIE bit is set. The MODF

bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transmission by toggling the SPE bit of the slave.

**NOTE:** *A logic 1 voltage on the  $\overline{SS}$  pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SPSCCK clocks, even if a transmission has begun.*

To clear the MODF flag, read the SPSCR and then write to the SPCR register. This entire clearing procedure must occur with no MODF condition existing or else the flag will not be cleared.

## Interrupts

Four SPI status flags can be enabled to generate CPU interrupt requests:

**Table 4. SPI Interrupts**

| Flag                     | Request   |
|--------------------------|---|
| SPTE (Transmitter Empty) | SPI Transmitter CPU Interrupt Request (SPTIE = 1)                       |
| SPRF (Receiver Full)     | SPI Receiver CPU Interrupt Request (SPRIE = 1)                          |
| OVRF (Overflow)          | SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1)             |
| MODF (Mode Fault)        | SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1, MODFEN = 1) |

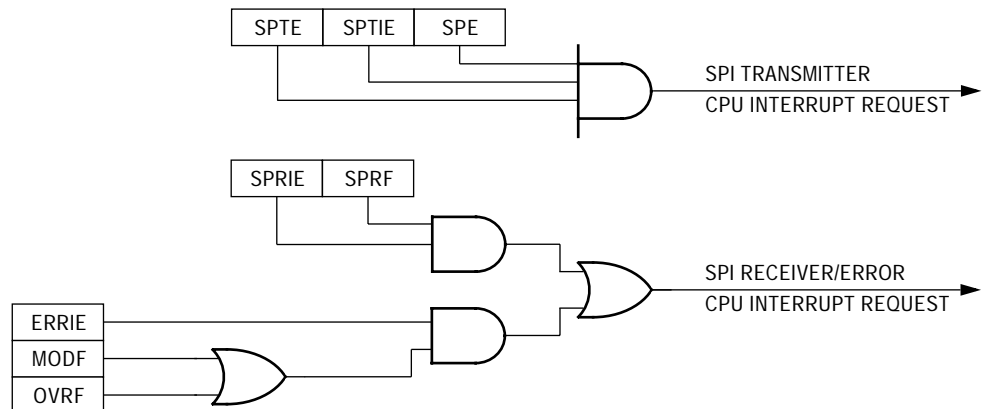
The SPI transmitter interrupt enable bit (SPTIE) enables the SPTE flag to generate transmitter CPU interrupt requests.

The SPI receiver interrupt enable bit (SPRIE) enables the SPRF bit to generate receiver CPU interrupt, provided that the SPI is enabled (SPE = 1).

The error interrupt enable bit (ERRIE) enables both the MODF and OVRF flags to generate a receiver/error CPU interrupt request.

## Serial Peripheral Interface Module (SPI)

The mode fault enable bit (MODFEN) can prevent the MODF flag from being set so that only the OVRF flag is enabled to generate receiver/error CPU interrupt requests.



**Figure 8. SPI Interrupt Request Generation**

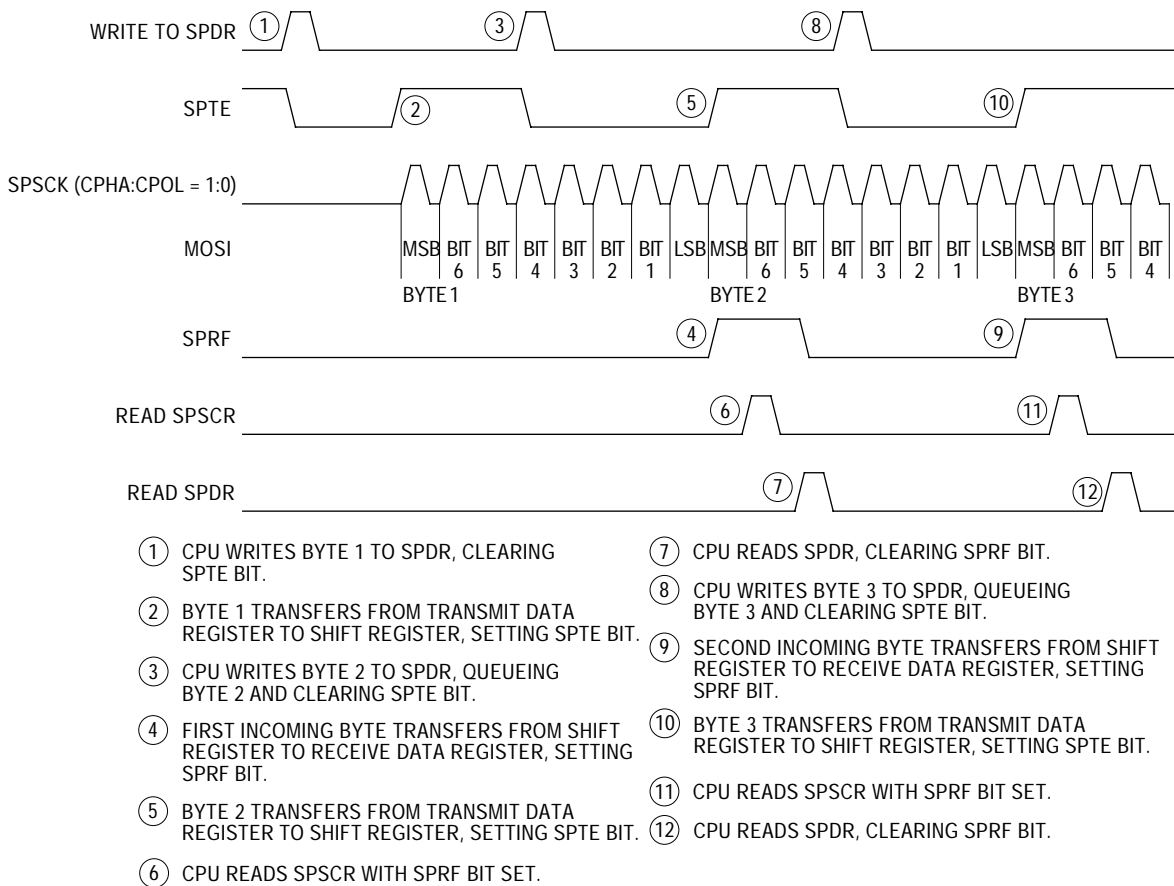
Two sources in the SPI status and control register can generate CPU interrupt requests:

1. SPI receiver full bit (SPRF) — The SPRF bit becomes set every time a byte transfers from the shift register to the receive data register. If the SPI receiver interrupt enable bit, SPRIE, is also set, SPRF can generate an SPI receiver/error CPU interrupt request.
2. SPI transmitter empty (SPTE) — The SPTE bit becomes set every time a byte transfers from the transmit data register to the shift register. If the SPI transmit interrupt enable bit, SPTIE, is also set, SPTE can generate an SPTE CPU interrupt request.



## Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE in SPSCR) indicates when the transmit data buffer is ready to accept new data. Write to the SPI data register only when the SPTE bit is high. **Figure 9** shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA:CPOL = 1:0).



**Figure 9. SPRF/SPTE CPU Interrupt Timing**

## Serial Peripheral Interface Module (SPI)

For a slave, the transmit data buffer allows back-to-back transmissions to occur without the slave having to time the write of its data between the transmissions. Also, if no new data is written to the data buffer, the last value contained in the shift register will be the next data word transmitted.

---

---

### Resetting the SPI

Any system reset completely resets the SPI. Partial reset occurs whenever the SPI enable bit (SPE) is low. Whenever SPE is low, the following occurs:

- The SPTE flag is set.
- Any transmission currently in progress is aborted.
- The shift register is cleared.
- The SPI state counter is cleared, making it ready for a new complete transmission.
- All the SPI port logic is defaulted back to being general-purpose I/O.

The following additional items are reset only by a system reset:

- All control bits in the SPCR register
- All control bits in the SPSCR register (MODFEN, ERRIE, SPR1, and SPR0)
- The status flags SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, the user can clear SPE between transmissions without having to reset all control bits when SPE is set back to high for the next transmission.

By not resetting the SPRF, OVRF, and MODF flags, the user can still service these interrupts after the SPI has been disabled. The user can disable the SPI by writing 0 to the SPE bit. The SPI also can be disabled by a mode fault occurring in an SPI that was configured as a master with the MODFEN bit set.

## Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

### Wait Mode

The SPI module remains active after the execution of a WAIT instruction. In wait mode, the SPI module registers are not accessible by the CPU. Any enabled CPU interrupt request from the SPI module can bring the MCU out of wait mode.

If SPI module functions are not required during wait mode, reduce power consumption by disabling the SPI module before executing the WAIT instruction.

To exit wait mode when an overflow condition occurs, enable the OVRF bit to generate CPU interrupt requests by setting the error interrupt enable bit (ERRIE). (See [Interrupts](#) on page 253).

### Stop Mode

The SPI module is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions. SPI operation resumes after the MCU exits stop mode. If stop mode is exited by reset, any transfer in progress is aborted and the SPI is reset.

---

---

### SPI During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR, \$FE03) enables software to clear status bits during the break state. (See [SIM Break Flag Control Register](#) on page 125).

To allow software to clear status bits during a break interrupt, write a logic 1 to the BCFE bit. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a two-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

Since the SPTE bit cannot be cleared during a break with the BCFE bit cleared, a write to the data register in break mode will not initiate a transmission nor will this data be transferred into the shift register. Therefore, a write to the SPDR in break mode with the BCFE bit cleared has no effect.

---

---

## I/O Signals

The SPI module has four I/O pins and shares three of them with a parallel I/O port.

- MISO — Data received
- MOSI — Data transmitted
- SPCK — Serial clock
- $\overline{SS}$  — Slave select
- $V_{SS}$  — Clock ground

The SPI has limited inter-integrated circuit (I<sup>2</sup>C) capability (requiring software support) as a master in a single-master environment. To communicate with I<sup>2</sup>C peripherals, MOSI becomes an open-drain output when the SPWOM bit in the SPI control register is set. In I<sup>2</sup>C communication, the MOSI and MISO pins are connected to a bidirectional pin from the I<sup>2</sup>C peripheral and through a pullup resistor to  $V_{DD}$ .

### MISO (Master In/Slave Out)

MISO is one of the two SPI module pins that transmit serial data. In full duplex operation, the MISO pin of the master SPI module is connected to the MISO pin of the slave SPI module. The master SPI simultaneously receives data on its MISO pin and transmits data from its MOSI pin.

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is logic 0 and its  $\overline{SS}$  pin is at logic 0. To support a multiple-slave system, a logic 1 on the  $\overline{SS}$  pin puts the MISO pin in a high-impedance state.

When enabled, the SPI controls data direction of the MISO pin regardless of the state of the data direction register of the shared I/O port.

## Serial Peripheral Interface Module (SPI)

### MOSI (Master Out/Slave In)

MOSI is one of the two SPI module pins that transmit serial data. In full duplex operation, the MOSI pin of the master SPI module is connected to the MOSI pin of the slave SPI module. The master SPI simultaneously transmits data from its MOSI pin and receives data on its MISO pin.

When enabled, the SPI controls data direction of the MOSI pin regardless of the state of the data direction register of the shared I/O port.

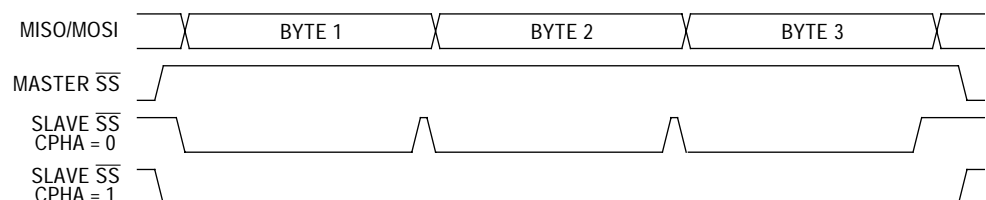
### SPSCK (Serial Clock)

The serial clock synchronizes data transmission between master and slave devices. In a master MCU, the SPSCK pin is the clock output. In a slave MCU, the SPSCK pin is the clock input. In full duplex operation, the master and slave MCUs exchange a byte of data in eight serial clock cycles.

When enabled, the SPI controls data direction of the SPSCK pin regardless of the state of the data direction register of the shared I/O port.

### $\overline{SS}$ (Slave Select)

The  $\overline{SS}$  pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the  $\overline{SS}$  is used to select a slave. For  $CPHA = 0$ , the  $\overline{SS}$  is used to define the start of a transmission. (See Transmission Formats.) Since it is used to indicate the start of a transmission, the  $\overline{SS}$  must be toggled high and low between each byte transmitted for the  $CPHA = 0$  format. However, it can remain low throughout the transmission for the  $CPHA = 1$  format. See [Figure 10](#).



**Figure 10. CPHA/ $\overline{SS}$  Timing**

When an SPI is configured as a slave, the  $\overline{SS}$  pin is always configured as an input. It cannot be used as a general-purpose I/O regardless of the

state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of the  $\overline{SS}$  from creating a MODF error. (See [SPI Status and Control Register](#) on page 264).

**NOTE:** *A logic 1 voltage on the  $\overline{SS}$  pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCCK clocks, even if a transmission already has begun.*

When an SPI is configured as a master, the  $\overline{SS}$  input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCCK. (See [Mode Fault Error](#) on page 251). For the state of the  $\overline{SS}$  pin to set the MODF flag, the MODFEN bit in the SPSCCK register must be set. If the MODFEN bit is low for an SPI master, the  $\overline{SS}$  pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. With MODFEN high, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the  $\overline{SS}$  pin by configuring the appropriate pin as an input and reading the data register. (See [Table 5](#)).

**Table 5. SPI Configuration**

| SPE | SPMSTR | MODFEN | SPI Configuration   | State of $\overline{SS}$ Logic                         |
|-----|--------|--------|---------------------|--|
| 0   | X      | X      | Not Enabled         | General-Purpose I/O;<br>$\overline{SS}$ Ignored by SPI |
| 1   | 0      | X      | Slave               | Input-Only to SPI                                      |
| 1   | 1      | 0      | Master without MODF | General-Purpose I/O;<br>$\overline{SS}$ Ignored by SPI |
| 1   | 1      | 1      | Master with MODF    | Input-Only to SPI                                      |

X = don't care

**V<sub>SS</sub> (Clock Ground)**

V<sub>SS</sub> is the ground return for the serial clock pin, SPSCCK, and the ground for the port output buffers. To reduce the ground return path loop and minimize radio frequency (RF) emissions, connect the ground pin of the slave to the V<sub>SS</sub> pin.

# Serial Peripheral Interface Module (SPI)

## I/O Registers

Three registers control and monitor SPI operation:

- SPI control register (SPCR \$0010)
- SPI status and control register (SPSCR \$0011)
- SPI data register (SPDR \$0012)

### SPI Control Register

The SPI control register:

- Enables SPI module interrupt requests
- Selects CPU interrupt requests
- Configures the SPI module as master or slave
- Selects serial clock polarity and phase
- Configures the SPSCCK, MOSI, and MISO pins as open-drain outputs
- Enables the SPI module

Address: \$0010

|        | Bit 7 | 6 | 5      | 4    | 3    | 2     | 1   | Bit 0 |
|--------|-------|---|--------|------|------|-------|-----|-------|
| Read:  | SPRIE | R | SPMSTR | CPOL | CPHA | SPWOM | SPE | SPTIE |
| Write: |       |   |        |      |      |       |     |       |
| Reset: | 0     | 0 | 1      | 0    | 1    | 0     | 0   | 0     |

|   |
|---|
| R |
|---|

 = Reserved

**Figure 11. SPI Control Register (SPCR)**

#### SPRIE — SPI Receiver Interrupt Enable Bit

This read/write bit enables CPU interrupt requests generated by the SPRF bit. The SPRF bit is set when a byte transfers from the shift register to the receive data register. Reset clears the SPRIE bit.

- 1 = SPRF CPU interrupt requests enabled
- 0 = SPRF CPU interrupt requests disabled



### SPMSTR — SPI Master Bit

This read/write bit selects master mode operation or slave mode operation. Reset sets the SPMSTR bit.

- 1 = Master mode
- 0 = Slave mode

### CPOL — Clock Polarity Bit

This read/write bit determines the logic state of the SPSCCK pin between transmissions. (See [Figure 3](#) and [Figure 4](#).) To transmit data between SPI modules, the SPI modules must have identical CPOL bits. Reset clears the CPOL bit.

### CPHA — Clock Phase Bit

This read/write bit controls the timing relationship between the serial clock and SPI data. (See [Figure 3](#) and [Figure 4](#).) To transmit data between SPI modules, the SPI modules must have identical CPHA bits. When CPHA = 0, the  $\overline{SS}$  pin of the slave SPI module must be set to logic 1 between bytes. (See [Figure 10](#)). Reset sets the CPHA bit.

When CPHA = 0 for a slave, the falling edge of  $\overline{SS}$  indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the MSB of its data. Once the transmission begins, no new data is allowed into the shift register from the data register. Therefore, the slave data register must be loaded with the desired transmit data before the falling edge of  $\overline{SS}$ . Any data written after the falling edge is stored in the data register and transferred to the shift register at the current transmission.

When CPHA = 1 for a slave, the first edge of the SPSCCK indicates the beginning of the transmission. The same applies when  $\overline{SS}$  is high for a slave. The MISO pin is held in a high-impedance state, and the incoming SPSCCK is ignored. In certain cases, it may also cause the MODF flag to be set. (See [Mode Fault Error](#) on page 251). A logic 1 on the  $\overline{SS}$  pin does not in any way affect the state of the SPI state machine.

## Serial Peripheral Interface Module (SPI)

### SPWOM — SPI Wired-OR Mode Bit

This read/write bit disables the pullup devices on pins SPSCCK, MOSI, and MISO so that those pins become open-drain outputs.

1 = Wired-OR SPSCCK, MOSI, and MISO pins

0 = Normal push-pull SPSCCK, MOSI, and MISO pins

### SPE — SPI Enable Bit

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI (see [Resetting the SPI](#) on page 256). Reset clears the SPE bit.

1 = SPI module enabled

0 = SPI module disabled

### SPTIE — SPI Transmit Interrupt Enable Bit

This read/write bit enables CPU interrupt requests generated by the SPTE bit. SPTE is set when a byte transfers from the transmit data register to the shift register. Reset clears the SPTIE bit.

1 = SPTE CPU interrupt requests enabled

0 = SPTE CPU interrupt requests disabled

## SPI Status and Control Register

The SPI status and control register contains flags to signal the following conditions:

- Receive data register full
- Failure to clear SPRF bit before next byte is received (overflow error)
- Inconsistent logic level on  $\overline{SS}$  pin (mode fault error)
- Transmit data register empty

The SPI status and control register also contains bits that perform these functions:

- Enable error interrupts
- Enable mode fault error detection
- Select master SPI baud rate

Address: \$0011

|        | Bit 7 | 6     | 5    | 4    | 3    | 2      | 1    | Bit 0 |
|--------|-------|-------|------|------|------|--------|------|-------|
| Read:  | SPRF  | ERRIE | OVRF | MODF | SPTE | MODFEN | SPR1 | SPR0  |
| Write: | R     |       | R    | R    | R    |        |      |       |
| Reset: | 0     | 0     | 0    | 0    | 1    | 0      | 0    | 0     |

R = Reserved

**Figure 12. SPI Status and Control Register (SPSCR)**

**SPRF — SPI Receiver Full Bit**

This clearable, read-only flag is set each time a byte transfers from the shift register to the receive data register. SPRF generates a CPU interrupt request if the SPRIE bit in the SPI control register is set also. During an SPRF CPU interrupt, the CPU clears SPRF by reading the SPI status and control register with SPRF set and then reading the SPI data register. Any read of the SPI data register clears the SPRF bit.

Reset clears the SPRF bit.

- 1 = Receive data register full
- 0 = Receive data register not full

**ERRIE — Error Interrupt Enable Bit**

This read-only bit enables the MODF and OVRF flags to generate CPU interrupt requests. Reset clears the ERRIE bit.

- 1 = MODF and OVRF can generate CPU interrupt requests
- 0 = MODF and OVRF cannot generate CPU interrupt requests

**OVRF — Overflow Bit**

This clearable, read-only flag is set if software does not read the byte in the receive data register before the next byte enters the shift register. In an overflow condition, the byte already in the receive data register is unaffected, and the byte that shifted in last is lost. Clear the OVRF bit by reading the SPI status and control register with OVRF set and then reading the SPI data register. Reset clears the OVRF flag.

- 1 = Overflow
- 0 = No overflow

## Serial Peripheral Interface Module (SPI)

### MODF — Mode Fault Bit

This clearable, read-only flag is set in a slave SPI if the  $\overline{SS}$  pin goes high during a transmission. In a master SPI, the MODF flag is set if the  $\overline{SS}$  pin goes low at any time. Clear the MODF bit by reading the SPI status and control register with MODF set and then writing to the SPI data register. Reset clears the MODF bit.

1 =  $\overline{SS}$  pin at inappropriate logic level

0 =  $\overline{SS}$  pin at appropriate logic level

### SPTE — SPI Transmitter Empty Bit

This clearable, read-only flag is set each time the transmit data register transfers a byte into the shift register. SPTE generates an SPTIE CPU interrupt request if the SPTIE bit in the SPI control register is set also.

**NOTE:** *Do not write to the SPI data register unless the SPTE bit is high.*

For an idle master or idle slave that has no data loaded into its transmit buffer, the SPTE will be set again within two bus cycles since the transmit buffer empties into the shift register. This allows the user to queue up a 16-bit value to send. For an already active slave, the load of the shift register cannot occur until the transmission is completed. This implies that a back-to-back write to the transmit data register is not possible. The SPTE indicates when the next write can occur.

Reset sets the SPTE bit.

1 = Transmit data register empty

0 = Transmit data register not empty

### MODFEN — Mode Fault Enable Bit

This read/write bit, when set to 1, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag. If the SPI is enabled as a master and the MODFEN bit is low, then the  $\overline{SS}$  pin is available as a general-purpose I/O.

If the MODFEN bit is set, then this pin is not available as a general purpose I/O. When the SPI is enabled as a slave, the  $\overline{SS}$  pin is not available as a general-purpose I/O regardless of the value of MODFEN. (See [SS \(Slave Select\)](#) on page 260).

If the MODFEN bit is low, the level of the  $\overline{SS}$  pin does not affect the operation of an enabled SPI configured as a master. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation. (See [Mode Fault Error](#) on page 251).

#### SPR1 and SPR0 — SPI Baud Rate Select Bits

In master mode, these read/write bits select one of four baud rates as shown in [Table 6](#). SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

**Table 6. SPI Master Baud Rate Selection**

| SPR1:SPR0 | Baud Rate Divisor (BD) |
|-----------|------------------------|
| 00        | 2                      |
| 01        | 8                      |
| 10        | 32                     |
| 11        | 128                    |

Use this formula to calculate the SPI baud rate:

$$\text{Baud rate} = \frac{\text{CGMOUT}}{2 \times \text{BD}}$$

where:

CGMOUT = base clock output of the clock generator module (CGM), see [Clock Generator Module \(CGM\)](#) on page 127.

BD = baud rate divisor

#### SPI Data Register

The SPI data register is the read/write buffer for the receive data register and the transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate buffers that can contain different values. See [Figure 1](#)

## Serial Peripheral Interface Module (SPI)

Address: \$0012

|        | Bit 7 | 6  | 5  | 4  | 3  | 2  | 1  | Bit 0 |
|--------|-------|----|----|----|----|----|----|-------|
| Read:  | R7    | R6 | R5 | R4 | R3 | R2 | R1 | R0    |
| Write: | T7    | T6 | T5 | T4 | T3 | T2 | T1 | T0    |

Reset: Indeterminate after Reset

**Figure 13. SPI Data Register (SPDR)**

R7–R0/T7–T0 — Receive/Transmit Data Bits

**NOTE:** Do not use read-modify-write instructions on the SPI data register since the buffer read is not the same as the buffer written.

# Timer Interface Module B (TIMB)

---

---

## Contents

|  |     |
|--|-----|
| Introduction . . . . .                                 | 270 |
| Features . . . . .                                     | 270 |
| Functional Description. . . . .                        | 273 |
| TIMB Counter Prescaler . . . . .                       | 273 |
| Input Capture . . . . .                                | 273 |
| Output Compare . . . . .                               | 275 |
| Unbuffered Output Compare . . . . .                    | 275 |
| Buffered Output Compare . . . . .                      | 276 |
| Pulse Width Modulation (PWM) . . . . .                 | 276 |
| Unbuffered PWM Signal Generation . . . . .             | 277 |
| Buffered PWM Signal Generation . . . . .               | 278 |
| PWM Initialization. . . . .                            | 279 |
| Interrupts . . . . .                                   | 280 |
| Low-Power Modes . . . . .                              | 281 |
| Wait Mode. . . . .                                     | 281 |
| Stop Mode. . . . .                                     | 281 |
| TIMB During Break Interrupts . . . . .                 | 282 |
| I/O Signals. . . . .                                   | 283 |
| TIMB Clock Pin (PTD4/TBLCK) . . . . .                  | 283 |
| TIMB Channel I/O Pins (PTF5/TBCH1–PTF4/TBCH0). . . . . | 283 |
| I/O Registers . . . . .                                | 284 |
| TIMB Status and Control Register . . . . .             | 284 |
| TIMB Counter Registers . . . . .                       | 286 |
| TIMB Counter Modulo Registers. . . . .                 | 287 |
| TIMB Channel Status and Control Registers . . . . .    | 288 |
| TIMB Channel Registers. . . . .                        | 292 |

## Timer Interface Module B (TIMB)

---

---

### Introduction

This section describes the timer interface module (TIMB). The TIMB is a 2-channel timer that provides a timing reference with input capture, output compare, and pulse width modulation functions. [Figure 1](#) is a block diagram of the TIMB.

---

---

### Features

Features of the TIMB include:

- Two Input Capture/Output Compare Channels
  - Rising-Edge, Falling-Edge, or Any-Edge Input Capture Trigger
  - Set, Clear, or Toggle Output Compare Action
- Buffered and Unbuffered Pulse Width Modulation (PWM) Signal Generation
- Programmable TIMB Clock Input
  - 7-Frequency Internal Bus Clock Prescaler Selection
  - External TIMB Clock Input (4-MHz Maximum Frequency)
- Free-Running or Modulo Up-Count Operation
- Toggle Any Channel Pin on Overflow
- TIMB Counter Stop and Reset Bits



Timer Interface Module B (TIMB)  
Features

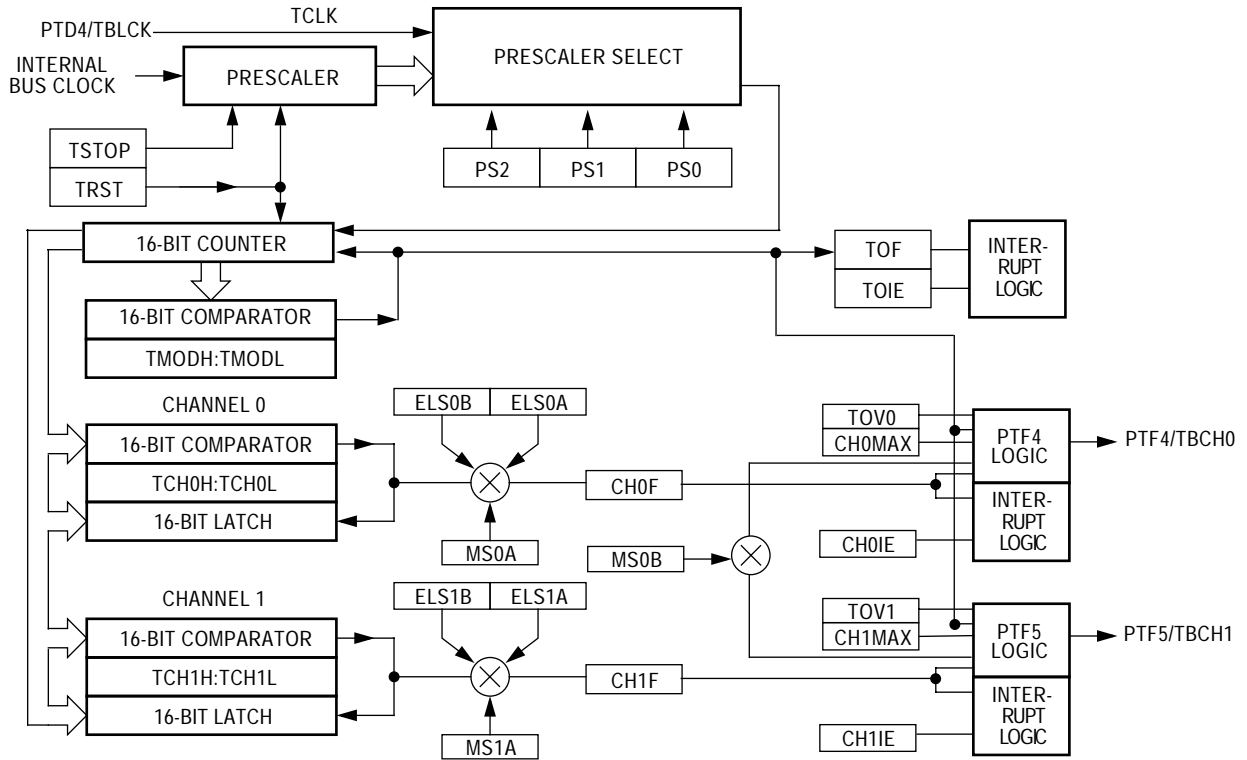


Figure 1. TIMB Block Diagram

## Timer Interface Module B (TIMB)

| Addr.  | Register Name                              | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
|--------|--|--------|-------|-------|------|-------|-------|------|--------|
| \$0040 | TIMB Status/Control Register (TBSC)        | TOF    | TOIE  | TSTOP | TRST | 0     | PS2   | PS1  | PS0    |
| \$0041 | TIMB Counter Register High (TBCNTH)        | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$0042 | TIMB Counter Register Low (TBCNTL)         | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
| \$0043 | TIMB Counter Modulo Reg. High (TBMODH)     | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$0044 | TIMB Counter Modulo Reg. Low (TBMODL)      | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
| \$0045 | TIMB Ch. 0 Status/Control Register (TBSC0) | CH0F   | CH0IE | MS0B  | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| \$0046 | TIMB Ch. 0 Register High (TBCH0H)          | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$0047 | TIMB Ch. 0 Register Low (TBCH0L)           | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
| \$0048 | TIMB Ch. 1 Status/Control Register (TBSC1) | CH1F   | CH1IE | 0     | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| \$0049 | TIMB Ch. 1 Register High (TBCH1H)          | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$004A | TIMB Ch. 1 Register Low (TBCH1L)           | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |

R = Reserved

**Figure 2. TIMB I/O Register Summary**

---



---

## Functional Description

**Figure 1** shows the TIMB structure. The central component of the TIMB is the 16-bit TIMB counter that can operate as a free-running counter or a modulo up-counter. The TIMB counter provides the timing reference for the input capture and output compare functions. The TIMB counter modulo registers, TBMODH–TBMODL, control the modulo value of the TIMB counter. Software can read the TIMB counter value at any time without affecting the counting sequence.

The two TIMB channels are programmable independently as input capture or output compare channels.

### TIMB Counter Prescaler

The TIMB clock source can be one of the seven prescaler outputs or the TIMB clock pin, PTD4/TBLCK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMB status and control register select the TIMB clock source.

### Input Capture

An input capture function has three basic parts: edge select logic, an input capture latch, and a 16-bit counter. Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TBSC0 through TBSC1 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMB latches the contents of the TIMB counter into the TIMB channel registers, TCHxH–TCHxL. Input captures can generate TIMB CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The result obtained by an input capture will be two more than the value of the free-running counter on the rising edge of the internal bus clock preceding the external transition. This delay is required for internal synchronization.

## Timer Interface Module B (TIMB)

The free-running counter contents are transferred to the TIMB channel status and control register (TBCHxH–TBCHxL, see [TIMB Channel Registers](#) on page 292) on each proper signal transition regardless of whether the TIMB channel flag (CH0F–CH1F in TBSC0–TBSC1 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or “captured” is the time of the event. Because this value is stored in the input capture register 2 bus cycles after the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see [TIMB Channel Registers](#) on page 292). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the input capture channel register (TBCHxH–TBCHxL).

## Output Compare

With the output compare function, the TIMB can generate a periodic pulse with a programmable polarity, duration, and frequency. When the counter reaches the value in the registers of an output compare channel, the TIMB can set, clear, or toggle the channel pin. Output compares can generate TIMB CPU interrupt requests.

### *Unbuffered Output Compare*

Any output compare channel can generate unbuffered output compare pulses as described in [Output Compare](#) on page 275. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMB channel registers.

An unsynchronized write to the TIMB channel registers to change an output compare value could cause incorrect operation for up to two counter overflow periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that counter overflow period. Also, using a TIMB overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIMB may pass the new value before it is written.

Use the following methods to synchronize unbuffered changes in the output compare value on channel x:

- When changing to a smaller value, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current output compare pulse. The interrupt routine has until the end of the counter overflow period to write the new value.
- When changing to a larger output compare value, enable channel x TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

## Timer Interface Module B (TIMB)

### Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTF5/TBCH1 pin. The TIMB channel registers of the linked pair alternately control the output.

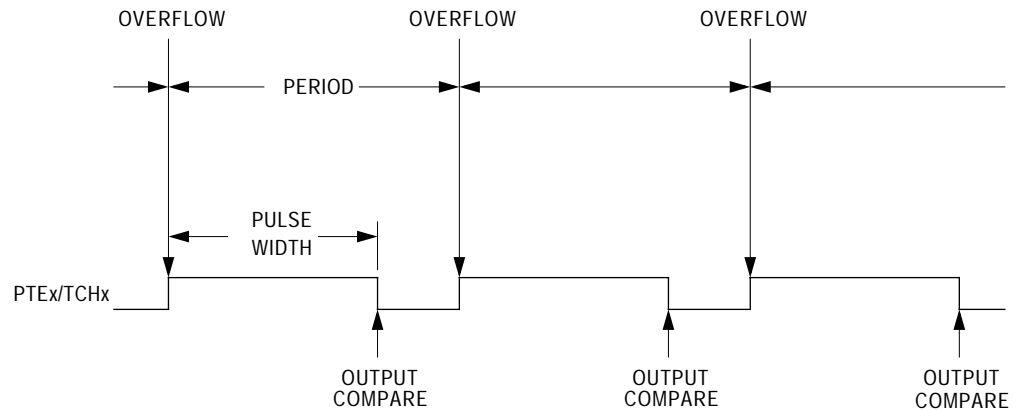
Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The output compare value in the TIMB channel 0 registers initially controls the output on the PTE2/TACH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the output after the TIMB overflows. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTF4/TBCH0, is available as a general-purpose I/O pin.

**NOTE:** *In buffered output compare operation, do not write new output compare values to the currently active channel registers. Writing to the active channel registers is the same as generating unbuffered output compares.*

### Pulse Width Modulation (PWM)

By using the toggle-on-overflow feature with an output compare channel, the TIMB can generate a PWM signal. The value in the TIMB counter modulo registers determines the period of the PWM signal. The channel pin toggles when the counter reaches the value in the TIMB counter modulo registers. The time between overflows is the period of the PWM signal.

As [Figure 3](#) shows, the output compare value in the TIMB channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMB to clear the channel pin on output compare if the state of the PWM pulse is logic 1. Program the TIMB to set the pin if the state of the PWM pulse is logic 0.



**Figure 3. PWM Period and Pulse Width**

The value in the TIMB counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing \$00FF (255) to the TIMB counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is \$000 (see TIMB Status and Control Register).

The value in the TIMB channel registers determines the pulse width of the PWM output. The pulse width of an 8-bit PWM signal is variable in 256 increments. Writing \$0080 (128) to the TIMB channel registers produces a duty cycle of 128/256 or 50%.

### *Unbuffered PWM Signal Generation*

Any output compare channel can generate unbuffered PWM pulses as described in [Pulse Width Modulation \(PWM\)](#) on page 276. The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the value currently in the TIMB channel registers.

An unsynchronized write to the TIMB channel registers to change a pulse width value could cause incorrect operation for up to two PWM periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that PWM period. Also, using a TIMB overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIMB may pass the new value before it is written to the TIMB channel registers.

## Timer Interface Module B (TIMB)

Use the following methods to synchronize unbuffered changes in the PWM pulse width on channel x:

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable channel x TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

**NOTE:** *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare also can cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

### Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTF4/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The TIMB channel 0 registers initially control the pulse width on the PTF4/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the pulse width are the ones written to last. TBSC0 controls and monitors the buffered PWM function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTF5/TBCH1, is available as a general-purpose I/O pin.



**NOTE:** *In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. Writing to the active channel registers is the same as generating unbuffered PWM signals.*

### PWM Initialization

To ensure correct operation when generating unbuffered or buffered PWM signals, use the following initialization procedure:

1. In the TIMB status and control register (TBSC):
  - a. Stop the TIMB counter by setting the TIMB stop bit, TSTOP.
  - b. Reset the TIMB counter by setting the TIMB reset bit, TRST.
2. In the TIMB counter modulo registers (TBMODH–TBMODL), write the value for the required PWM period.
3. In the TIMB channel x registers (TBCHxH–TBCHxL), write the value for the required pulse width.
4. In TIMB channel x status and control register (TBSCx):
  - a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB–MSxA. (See [Table 2](#)).
  - b. Write 1 to the toggle-on-overflow bit, TOVx.
  - c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB–ELSxA. The output action on compare must force the output to the complement of the pulse width level. (See [Table 2](#).)

**NOTE:** *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare can also cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

5. In the TIMB status control register (TBSC), clear the TIMB stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMB channel 0 registers (TBCH0H–TBCH0L) initially control the buffered PWM output. TIMB status control register 0

## Timer Interface Module B (TIMB)

(TBSC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on TIMB overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

Setting the channel x maximum duty cycle bit (CHxMAX) and clearing the TOVx bit generates a 100% duty cycle output. (See [TIMB Channel Status and Control Registers](#) on page 288.)

---

---

## Interrupts

The following TIMB sources can generate interrupt requests:

- TIMB overflow flag (TOF) — The TOF bit is set when the TIMB counter value rolls over to \$0000 after matching the value in the TIMB counter modulo registers. The TIMB overflow interrupt enable bit, TOIE, enables TIMB overflow CPU interrupt requests. TOF and TOIE are in the TIMB status and control register.
- TIMB channel flags (CH1F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIMB CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

## Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

### Wait Mode

The TIMB remains active after the execution of a WAIT instruction. In wait mode, the TIMB registers are not accessible by the CPU. Any enabled CPU interrupt request from the TIMB can bring the MCU out of wait mode.

If TIMB functions are not required during wait mode, reduce power consumption by stopping the TIMB before executing the WAIT instruction.

### Stop Mode

The TIMB is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions or the state of the TIMB counter. TIMB operation resumes when the MCU exits stop mode.

---

---

### TIMB During Break Interrupts

A break interrupt stops the TIMB counter and inhibits input captures.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See [SIM Break Flag Control Register](#) on page 125).

To allow software to clear status bits during a break interrupt, write a logic 1 to the BCFE bit. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

---



---

## I/O Signals

Port D shares one of its pins with the TIMB. Port F shares two of its pins with the TIMB. PTD4/TBLCK is an external clock input to the TIMB prescaler. The two TIMB channel I/O pins are PTF4/TBCH0 and PTF5/TBCH1.

### TIMB Clock Pin (PTD4/TBLCK)

PTD4/TBLCK is an external clock input that can be the clock source for the TIMB counter instead of the prescaled internal bus clock. Select the PTD4/TBLCK input by writing logic 1s to the three prescaler select bits, PS[2:0]. (See TIMB Status and Control Register.) The minimum TCLK pulse width,  $TCLK_{L\text{MIN}}$  or  $TCLK_{H\text{MIN}}$ , is:

$$\frac{1}{\text{bus frequency}} + t_{\text{SU}}$$

The maximum TCLK frequency is the least: 4 MHz or bus frequency  $\div$  2.

PTD4/TBLCK is available as a general-purpose I/O pin or ADC channel when not used as the TIMB clock input. When the PTD6/TACLK pin is the TIMB clock input, it is an input regardless of the state of the DDRD6 bit in data direction register D.

### TIMB Channel I/O Pins (PTF5/TBCH1–PTF4/ TBCH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTF4/TBCH0 and PTF5/TBCH1 can be configured as buffered output compare or buffered PWM pins.

## Timer Interface Module B (TIMB)

### I/O Registers

These I/O registers control and monitor TIMB operation:

- TIMB status and control register (TBSC)
- TIMB control registers (TBCNTH–TBCNTL)
- TIMB counter modulo registers (TBMODH–TBMODL)
- TIMB channel status and control registers (TBSC0 and TBSC1)
- TIMB channel registers (TBCH0H–TBCH0L, TBCH1H–TBCH1L)

### TIMB Status and Control Register

The TIMB status and control register:

- Enables TIMB overflow interrupts
- Flags TIMB overflows
- Stops the TIMB counter
- Resets the TIMB counter
- Prescales the TIMB counter clock

Address: \$0040

|        | Bit 7 | 6    | 5     | 4    | 3 | 2   | 1   | Bit 0 |
|--------|-------|------|-------|------|---|-----|-----|-------|
| Read:  | TOF   | TOIE | TSTOP | 0    | 0 | PS2 | PS1 | PS0   |
| Write: | 0     |      |       | TRST | R |     |     |       |
| Reset: | 0     | 0    | 1     | 0    | 0 | 0   | 0   | 0     |

R = Reserved

**Figure 4. TIMB Status and Control Register (TBSC)**

#### TOF — TIMB Overflow Flag Bit

This read/write flag is set when the TIMB counter resets to \$0000 after reaching the modulo value programmed in the TIMB counter modulo registers. Clear TOF by reading the TIMB status and control register when TOF is set and then writing a logic 0 to TOF. If another TIMB overflow occurs before the clearing sequence is complete, then

writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

1 = TIMB counter has reached modulo value

0 = TIMB counter has not reached modulo value

#### TOIE — TIMB Overflow Interrupt Enable Bit

This read/write bit enables TIMB overflow interrupts when the TOF bit becomes set. Reset clears the TOIE bit.

1 = TIMB overflow interrupts enabled

0 = TIMB overflow interrupts disabled

#### TSTOP — TIMB Stop Bit

This read/write bit stops the TIMB counter. Counting resumes when TSTOP is cleared. Reset sets the TSTOP bit, stopping the TIMB counter until software clears the TSTOP bit.

1 = TIMB counter stopped

0 = TIMB counter active

**NOTE:** *Do not set the TSTOP bit before entering wait mode if the TIMB is required to exit wait mode. Also, when the TSTOP bit is set and the timer is configured for input capture operation, input captures are inhibited until TSTOP is cleared.*

#### TRST — TIMB Reset Bit

Setting this write-only bit resets the TIMB counter and the TIMB prescaler. Setting TRST has no effect on any other registers. Counting resumes from \$0000. TRST is cleared automatically after the TIMB counter is reset and always reads as logic 0. Reset clears the TRST bit.

1 = Prescaler and TIMB counter cleared

0 = No effect

**NOTE:** *Setting the TSTOP and TRST bits simultaneously stops the TIMB counter at a value of \$0000.*

## Timer Interface Module B (TIMB)

### PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTD4/TBLCK pin or one of the seven prescaler outputs as the input to the TIMB counter as [Table 1](#) shows. Reset clears the PS[2:0] bits.

**Table 1. Prescaler Selection**

| PS[2:0] | TIMB Clock Source       |
|---------|-------------------------|
| 000     | Internal Bus Clock ÷ 1  |
| 001     | Internal Bus Clock ÷ 2  |
| 010     | Internal Bus Clock ÷ 4  |
| 011     | Internal Bus Clock ÷ 8  |
| 100     | Internal Bus Clock ÷ 16 |
| 101     | Internal Bus Clock ÷ 32 |
| 110     | Internal Bus Clock ÷ 64 |
| 111     | PTD4/TBLCK              |

### TIMB Counter Registers

The two read-only TIMB counter registers contain the high and low bytes of the value in the TIMB counter. Reading the high byte (TBCNTH) latches the contents of the low byte (TBCNTL) into a buffer. Subsequent reads of TBCNTH do not affect the latched TBCNTL value until TBCNTL is read. Reset clears the TIMB counter registers. Setting the TIMB reset bit (TRST) also clears the TIMB counter registers.

**NOTE:** *If TBCNTH is read during a break interrupt, be sure to unlatch TBCNTL by reading TBCNTL before exiting the break interrupt. Otherwise, TBCNTL retains the value latched during the break.*

Register Name and Address TBCNTH — \$0041

|        | Bit 7  | 6      | 5      | 4      | 3      | 2      | 1     | Bit 0 |
|--------|--------|--------|--------|--------|--------|--------|-------|-------|
| Read:  | BIT 15 | BIT 14 | BIT 13 | BIT 12 | BIT 11 | BIT 10 | BIT 9 | BIT 8 |
| Write: | R      | R      | R      | R      | R      | R      | R     | R     |
| Reset: | 0      | 0      | 0      | 0      | 0      | 0      | 0     | 0     |

**Figure 5. TIMB Counter Registers (TBCNTH and TBCNTL)**



Register Name and Address TBCNTL — \$0042

|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
| Read:  | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| Write: | R     | R     | R     | R     | R     | R     | R     | R     |
| Reset: | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

R = Reserved

**Figure 5. TIMB Counter Registers (TBCNTH and TBCNTL)**

### TIMB Counter Modulo Registers

The read/write TIMB modulo registers contain the modulo value for the TIMB counter. When the TIMB counter reaches the modulo value, the overflow flag (TOF) becomes set, and the TIMB counter resumes counting from \$0000 at the next clock. Writing to the high byte (TMODH) inhibits the TOF bit and overflow interrupts until the low byte (TMODL) is written. Reset sets the TIMB counter modulo registers.

Register Name and Address TBMODH — \$0043

|        | Bit 7  | 6      | 5      | 4      | 3      | 2      | 1     | Bit 0 |
|--------|--------|--------|--------|--------|--------|--------|-------|-------|
| Read:  | BIT 15 | BIT 14 | BIT 13 | BIT 12 | BIT 11 | BIT 10 | BIT 9 | BIT 8 |
| Write: |        |        |        |        |        |        |       |       |
| Reset: | 1      | 1      | 1      | 1      | 1      | 1      | 1     | 1     |

Register Name and Address TBMODL — \$0044

|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
| Read:  | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| Write: |       |       |       |       |       |       |       |       |
| Reset: | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     |

**Figure 6. TIMB Counter Modulo Registers (TMODH and TMODL)**

**NOTE:** Reset the TIMB counter before writing to the TIMB counter modulo registers.

## Timer Interface Module B (TIMB)

### TIMB Channel Status and Control Registers

Each of the TIMB channel status and control registers:

- Flags input captures and output compares
- Enables input capture and output compare interrupts
- Selects input capture, output compare, or PWM operation
- Selects high, low, or toggling output on output compare
- Selects rising edge, falling edge, or any edge as the active input capture trigger
- Selects output toggling on TIMB overflow
- Selects 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Register Name and Address TBSC0 — \$0045

|        | Bit 7 | 6     | 5    | 4    | 3     | 2     | 1    | Bit 0  |
|--------|-------|-------|------|------|-------|-------|------|--------|
| Read:  | CH0F  | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| Write: | 0     |       |      |      |       |       |      |        |
| Reset: | 0     | 0     | 0    | 0    | 0     | 0     | 0    | 0      |

Register Name and Address TBSC1 — \$0048

|        | Bit 7 | 6     | 5 | 4    | 3     | 2     | 1    | Bit 0  |
|--------|-------|-------|---|------|-------|-------|------|--------|
| Read:  | CH1F  | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| Write: | 0     |       | R |      |       |       |      |        |
| Reset: | 0     | 0     | 0 | 0    | 0     | 0     | 0    | 0      |

|   |              |
|---|--------------|
| R | R = Reserved |
|---|--------------|

**Figure 7. TIMB Channel Status and Control Registers (TBSC0–TBSC1)**

#### CHxF — Channel x Flag Bit

When channel x is an input capture channel, this read/write bit is set when an active edge occurs on the channel x pin. When channel x is an output compare channel, CHxF is set when the value in the TIMB counter registers matches the value in the TIMB channel x registers. When CHxIE = 0, clear CHxF by reading TIMB channel x status and control register with CHxF set, and then writing a logic 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

Reset clears the CHxF bit. Writing a logic 1 to CHxF has no effect.

1 = Input capture or output compare on channel x

0 = No input capture or output compare on channel x

#### CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIMB CPU interrupts on channel x.

Reset clears the CHxIE bit.

1 = Channel x CPU interrupt requests enabled

0 = Channel x CPU interrupt requests disabled

#### MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIMB channel 0.

Setting MS0B disables the channel 1 status and control register and reverts TBCH1 to general-purpose I/O.

Reset clears the MSxB bit.

1 = Buffered output compare/PWM operation enabled

0 = Buffered output compare/PWM operation disabled

## Timer Interface Module B (TIMB)

### MSxA — Mode Select Bit A

When ELSxB:A  $\neq$  00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. (See [Table 2](#)).

1 = Unbuffered output compare/PWM operation

0 = Input capture operation

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin once PWM, input capture, or output compare operation is enabled. (See [Table 2](#)). Reset clears the MSxA bit.

1 = Initial output level low

0 = Initial output level high

**NOTE:** *Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIMB status and control register (TBSC).*

### ELSxB and ELSxA — Edge/Level Select Bits

When channel x is an input capture channel, these read/write bits control the active edge-sensing logic on channel x.

When channel x is an output compare channel, ELSxB and ELSxA control the channel x output behavior when an output compare occurs.

When ELSxB and ELSxA are both clear, channel x is not connected to port E or port F, and pin PTE<sub>x</sub>/TBCH<sub>x</sub> or pin PTF<sub>x</sub>/TBCH<sub>x</sub> is available as a general-purpose I/O pin. However, channel x is at a state determined by these bits and becomes transparent to the respective pin when PWM, input capture, or output compare mode is enabled. [Table 2](#) shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

**Table 2. Mode, Edge, and Level Selection**

| MSxB:MSxA | ELSxB:ELSxA | Mode  | Configuration  |
|-----------|-------------|---|--|
| X0        | 00          | Output<br>Preset                                    | Pin under Port Control;<br>Initialize Timer<br>Output Level High |
| X1        | 00          |   | Pin under Port Control;<br>Initialize Timer<br>Output Level Low  |
| 00        | 01          | Input<br>Capture                                    | Capture on Rising Edge Only                                      |
| 00        | 10          |   | Capture on Falling Edge Only                                     |
| 00        | 11          |   | Capture on Rising or Falling Edge                                |
| 01        | 01          | Output<br>Compare<br>or PWM                         | Toggle Output on Compare   |
| 01        | 10          |   | Clear Output on Compare  |
| 01        | 11          |   | Set Output on Compare  |
| 1X        | 01          | Buffered<br>Output<br>Compare<br>or Buffered<br>PWM | Toggle Output on Compare   |
| 1X        | 10          |   | Clear Output on Compare  |
| 1X        | 11          |   | Set Output on Compare  |

**NOTE:** Before enabling a TIMB channel register for input capture operation, make sure that the PTFx/TBCHx pin is stable for at least two bus clocks.

#### TOVx — Toggle-On-Overflow Bit

When channel x is an output compare channel, this read/write bit controls the behavior of the channel x output when the TIMB counter overflows. When channel x is an input capture channel, TOVx has no effect. Reset clears the TOVx bit.

1 = Channel x pin toggles on TIMB counter overflow.

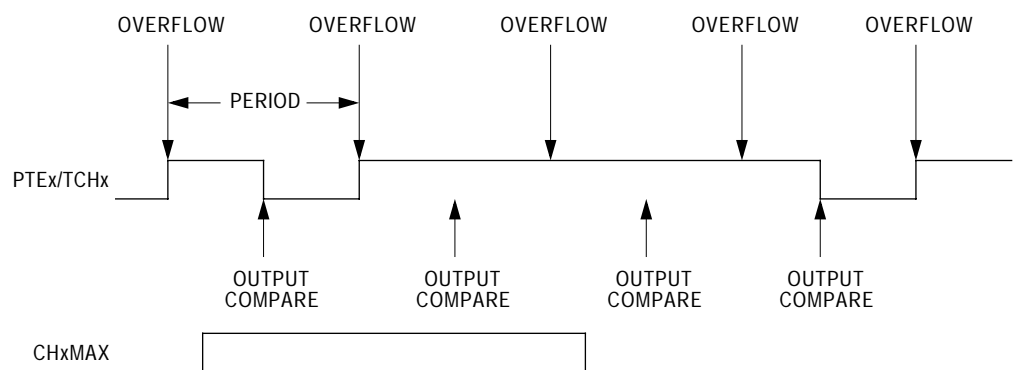
0 = Channel x pin does not toggle on TIMB counter overflow.

**NOTE:** When TOVx is set, a TIMB counter overflow takes precedence over a channel x output compare if both occur at the same time.

## Timer Interface Module B (TIMB)

### CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic 0, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As **Figure 8** shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.



**Figure 8. CHxMAX Latency**

### TIMB Channel Registers

These read/write registers contain the captured TIMB counter value of the input capture function or the output compare value of the output compare function. The state of the TIMB channel registers after reset is unknown.

In input capture mode ( $MSxB-MSxA = 0:0$ ), reading the high byte of the TIMB channel x registers (TBCHxH) inhibits input captures until the low byte (TBCHxL) is read.

In output compare mode ( $MSxB-MSxA \neq 0:0$ ), writing to the high byte of the TIMB channel x registers (TBCHxH) inhibits output compares and the CHxF bit until the low byte (TBCHxL) is written.

Register Name and Address TBCH0H — \$0046

|        |                           |        |        |        |        |        |       |       |
|--------|---------------------------|--------|--------|--------|--------|--------|-------|-------|
|        | Bit 7                     | 6      | 5      | 4      | 3      | 2      | 1     | Bit 0 |
| Read:  | Bit 15                    | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Write: |                           |        |        |        |        |        |       |       |
| Reset: | Indeterminate after Reset |        |        |        |        |        |       |       |

Register Name and Address TBCH0L — \$0047

|        |                           |       |       |       |       |       |       |       |
|--------|---------------------------|-------|-------|-------|-------|-------|-------|-------|
|        | Bit 7                     | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
| Read:  | Bit 7                     | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Write: |                           |       |       |       |       |       |       |       |
| Reset: | Indeterminate after Reset |       |       |       |       |       |       |       |

Register Name and Address TBCH1H — \$0049

|        |                           |        |        |        |        |        |       |       |
|--------|---------------------------|--------|--------|--------|--------|--------|-------|-------|
|        | Bit 7                     | 6      | 5      | 4      | 3      | 2      | 1     | Bit 0 |
| Read:  | Bit 15                    | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Write: |                           |        |        |        |        |        |       |       |
| Reset: | Indeterminate after Reset |        |        |        |        |        |       |       |

Register Name and Address TBCH1L — \$004A

|        |                           |       |       |       |       |       |       |       |
|--------|---------------------------|-------|-------|-------|-------|-------|-------|-------|
|        | Bit 7                     | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
| Read:  | Bit 7                     | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Write: |                           |       |       |       |       |       |       |       |
| Reset: | Indeterminate after Reset |       |       |       |       |       |       |       |

**Figure 9. TIMB Channel Registers  
(TBCH0H/L–TBCH1H/L)**

## Timer Interface Module B (TIMB)



# Programmable Interrupt Timer (PIT)

---

---

## Contents

|   |     |
|---|-----|
| Introduction . . . . .                    | 295 |
| Features . . . . .                        | 296 |
| Functional Description . . . . .          | 296 |
| PIT Counter Prescaler . . . . .           | 298 |
| Low-Power Modes . . . . .                 | 298 |
| Wait Mode . . . . .                       | 298 |
| Stop Mode . . . . .                       | 298 |
| PIT During Break Interrupts . . . . .     | 299 |
| I/O Registers . . . . .                   | 300 |
| PIT Status and Control Register . . . . . | 300 |
| PIT Counter Registers . . . . .           | 302 |
| PIT Counter Modulo Registers . . . . .    | 303 |

---

---

## Introduction

This section describes the programmable interrupt timer which is a timer whose counter is clocked internally via software programmable options. **Figure 1** is a block diagram of the PIT.

# Programmable Interrupt Timer

---

---

## Features

Features of the PIT include:

- Programmable PIT Clock Input
- Free-Running or Modulo Up-Count Operation
- PIT Counter Stop and Reset Bits

---

---

## Functional Description

**Figure 1** shows the structure of the PIT. The central component of the PIT is the 16-bit PIT counter that can operate as a free-running counter or a modulo up-counter. The counter provides the timing reference for the interrupt. The PIT counter modulo registers, PMODH–PMODL, control the modulo value of the counter. Software can read the counter value at any time without affecting the counting sequence.

Programmable Interrupt Timer (PIT)  
Functional Description

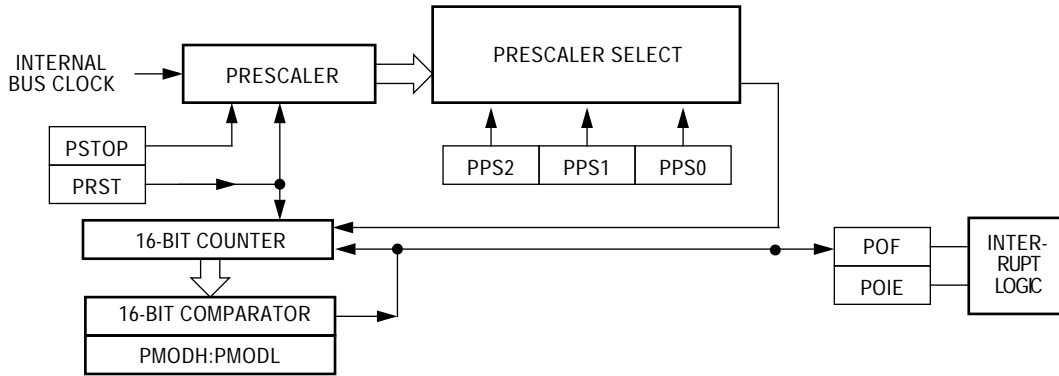


Figure 1. PIT Block Diagram

| Register Name                           | Bit 7        | 6    | 5     | 4    | 3  | 2    | 1    | Bit 0 |
|---|--------------|------|-------|------|----|------|------|-------|
| PIT Status and Control Register (PSC)   | Read: TOF    | POIE | PSTOP | 0    | 0  | PPS2 | PPS1 | PPS0  |
|   | Write: 0     |      |       | PRST |    |      |      |       |
|   | Reset:       | 0    | 0     | 1    | 0  | 0    | 0    | 0     |
| PIT Counter Register High (PCNTH)       | Read: Bit 15 | 14   | 13    | 12   | 11 | 10   | 9    | Bit 8 |
|   | Write:       |      |       |      |    |      |      |       |
|   | Reset:       | 0    | 0     | 0    | 0  | 0    | 0    | 0     |
| PIT Counter Register Low (PCNTL)        | Read: Bit 7  | 6    | 5     | 4    | 3  | 2    | 1    | Bit 0 |
|   | Write:       |      |       |      |    |      |      |       |
|   | Reset:       | 0    | 0     | 0    | 0  | 0    | 0    | 0     |
| PIT Counter Modulo Register High (PMDH) | Read: Bit 15 | 14   | 13    | 12   | 11 | 10   | 9    | Bit 8 |
|   | Write:       |      |       |      |    |      |      |       |
|   | Reset:       | 1    | 1     | 1    | 1  | 1    | 1    | 1     |
| PIT Counter Modulo Register Low (PMDL)  | Read: Bit 7  | 6    | 5     | 4    | 3  | 2    | 1    | Bit 0 |
|   | Write:       |      |       |      |    |      |      |       |
|   | Reset:       | 1    | 1     | 1    | 1  | 1    | 1    | 1     |

=Unimplemented

Figure 2. PIT I/O Register Summary

Table 1. PIT I/O Register Address Summary

| Register | PSC    | PCNTH  | PCNTL  | PMDH   | PMDL   |
|----------|--------|--------|--------|--------|--------|
| Address  | \$004B | \$004C | \$004D | \$004E | \$004F |

---

---

### PIT Counter Prescaler

The clock source can be one of the seven prescaler outputs. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PPS[2:0], in the status and control register select the PIT clock source.

The value in the PIT counter modulo registers and the selected prescaler output determines the frequency of the periodic interrupt. The PIT overflow flag (POF) is set when the PIT counter value rolls over to \$0000 after matching the value in the PIT counter modulo registers. The PIT interrupt enable bit, PIE, enables PIT overflow CPU interrupt requests. POF and PIE are in the PIT status and control register.

---

---

### Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

#### Wait Mode

The PIT remains active after the execution of a WAIT instruction. In wait mode the PIT registers are not accessible by the CPU. Any enabled CPU interrupt request from the PIT can bring the MCU out of wait mode.

If PIT functions are not required during wait mode, reduce power consumption by stopping the PIT before executing the WAIT instruction.

#### Stop Mode

The PIT is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions or the state of the PIT counter. PIT operation resumes when the MCU exits stop mode after an external interrupt.

## PIT During Break Interrupts

A break interrupt stops the PIT counter.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See [SIM Break Flag Control Register](#) on page 125).

To allow software to clear status bits during a break interrupt, write a logic 1 to the BCFE bit. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

# Programmable Interrupt Timer

## I/O Registers

The following I/O registers control and monitor operation of the PIT:

- PIT status and control register (PSC)
- PIT counter registers (PCNTH–PCNTL)
- PIT counter modulo registers (PMODH–PMODL)


### PIT Status and Control Register

The PIT status and control register:

- Enables PIT interrupt
- Flags PIT overflows
- Stops the PIT counter
- Resets the PIT counter
- Prescales the PIT counter clock

Address: \$004B

|        | Bit 7 | 6    | 5     | 4    | 3 | 2    | 1    | Bit 0 |
|--------|-------|------|-------|------|---|------|------|-------|
| Read:  | POF   | POIE | PSTOP | 0    | 0 | PPS2 | PPS1 | PPS0  |
| Write: | 0     |      |       | PRST |   |      |      |       |
| Reset: | 0     | 0    | 1     | 0    | 0 | 0    | 0    | 0     |

 = Unimplemented

**Figure 3. PIT Status and Control Register (PSC)**

### POF — PIT Overflow Flag Bit

This read/write flag is set when the PIT counter resets to \$0000 after reaching the modulo value programmed in the PIT counter modulo registers. Clear POF by reading the PIT status and control register when POF is set and then writing a logic 0 to POF. If another PIT overflow occurs before the clearing sequence is complete, then writing logic 0 to POF has no effect. Therefore, a POF interrupt request cannot be lost due to inadvertent clearing of POF. Reset clears the POF bit. Writing a logic 1 to POF has no effect.

1 = PIT counter has reached modulo value  
0 = PIT counter has not reached modulo value

**PIE** — PIT Overflow Interrupt Enable Bit

This read/write bit enables PIT overflow interrupts when the POF bit becomes set. Reset clears the POIE bit.

1 = PIT overflow interrupts enabled  
0 = PIT overflow interrupts disabled

**PSTOP** — PIT Stop Bit

This read/write bit stops the PIT counter. Counting resumes when PSTOP is cleared. Reset sets the PSTOP bit, stopping the PIT counter until software clears the PSTOP bit.

1 = PIT counter stopped  
0 = PIT counter active

**NOTE:** *Do not set the PSTOP bit before entering wait mode if the PIT is required to exit wait mode.*

**PRST** — PIT Reset Bit

Setting this write-only bit resets the PIT counter and the PIT prescaler. Setting PRST has no effect on any other registers. Counting resumes from \$0000. PRST is cleared automatically after the PIT counter is reset and always reads as logic zero. Reset clears the PRST bit.

1 = Prescaler and PIT counter cleared  
0 = No effect

**NOTE:** *Setting the PSTOP and PRST bits simultaneously stops the PIT counter at a value of \$0000.*

## Programmable Interrupt Timer

### PPS[2:0] — Prescaler Select Bits

These read/write bits select one of the seven prescaler outputs as the input to the PIT counter as [Table 2](#) shows. Reset clears the PPS[2:0] bits.

**Table 2. Prescaler Selection**

| PPS[2:0] | PIT Clock Source        |
|----------|-------------------------|
| 000      | Internal Bus Clock ÷ 1  |
| 001      | Internal Bus Clock ÷ 2  |
| 010      | Internal Bus Clock ÷ 4  |
| 011      | Internal Bus Clock ÷ 8  |
| 100      | Internal Bus Clock ÷ 16 |
| 101      | Internal Bus Clock ÷ 32 |
| 110      | Internal Bus Clock ÷ 64 |
| 111      | Internal Bus Clock ÷ 64 |

### PIT Counter Registers

The two read-only PIT counter registers contain the high and low bytes of the value in the PIT counter. Reading the high byte (PCNTH) latches the contents of the low byte (PCNTL) into a buffer. Subsequent reads of PCNTH do not affect the latched PCNTL value until PCNTL is read. Reset clears the PIT counter registers. Setting the PIT reset bit (PRST) also clears the PIT counter registers.

**NOTE:** *If you read PCNTH during a break interrupt, be sure to unlatch PCNTL by reading PCNTL before exiting the break interrupt. Otherwise, PCNTL retains the value latched during the break.*

Address: \$004C


|        |        |    |    |    |    |    |   |       |
|--------|--------|----|----|----|----|----|---|-------|
|        | Bit 7  | 6  | 5  | 4  | 3  | 2  | 1 | Bit 0 |
| Read:  | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| Write: |        |    |    |    |    |    |   |       |
| Reset: | 0      | 0  | 0  | 0  | 0  | 0  | 0 | 0     |

**Figure 4. PIT Counter Registers (PCNTH–PCNTL)**



Address: \$004D

|        |        |    |    |    |    |    |   |       |
|--------|--------|----|----|----|----|----|---|-------|
|        | Bit 7  | 6  | 5  | 4  | 3  | 2  | 1 | Bit 0 |
| Read:  | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| Write: |        |    |    |    |    |    |   |       |
| Reset: | 0      | 0  | 0  | 0  | 0  | 0  | 0 | 0     |

 = Unimplemented

**Figure 4. PIT Counter Registers (PCNTH–PCNTL)**

**PIT Counter Modulo Registers**

The read/write PIT modulo registers contain the modulo value for the PIT counter. When the PIT counter reaches the modulo value, the overflow flag (POF) becomes set, and the PIT counter resumes counting from \$0000 at the next clock. Writing to the high byte (PMDH) inhibits the POF bit and overflow interrupts until the low byte (PMDL) is written. Reset sets the PIT counter modulo registers.

Address: \$004E:\$004F

|        |        |    |    |    |    |    |   |       |
|--------|--------|----|----|----|----|----|---|-------|
|        | Bit 7  | 6  | 5  | 4  | 3  | 2  | 1 | Bit 0 |
| Read:  | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| Write: |        |    |    |    |    |    |   |       |
| Reset: | 1      | 1  | 1  | 1  | 1  | 1  | 1 | 1     |

Address: \$004E:\$004F

|        |       |   |   |   |   |   |   |       |
|--------|-------|---|---|---|---|---|---|-------|
|        | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read:  | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Write: |       |   |   |   |   |   |   |       |
| Reset: | 1     | 1 | 1 | 1 | 1 | 1 | 1 | 1     |

**Figure 5. PIT Counter Modulo Registers (PMDH–PMDL)**

**NOTE:** Reset the PIT counter before writing to the PIT counter modulo registers.

## Programmable Interrupt Timer

---

---

## Contents

|                                 |     |
|---------------------------------|-----|
| Introduction .....              | 306 |
| Port A .....                    | 307 |
| Port A Data Register .....      | 307 |
| Data Direction Register A ..... | 307 |
| Port B .....                    | 309 |
| Port B Data Register .....      | 309 |
| Data Direction Register B ..... | 310 |
| Port C .....                    | 312 |
| Port C Data Register .....      | 312 |
| Data Direction Register C ..... | 313 |
| Port D .....                    | 315 |
| Port D Data Register .....      | 315 |
| Data Direction Register D ..... | 316 |
| Port E .....                    | 318 |
| Port E Data Register .....      | 318 |
| Data Direction Register E ..... | 320 |
| Port F .....                    | 322 |
| Port F Data Register .....      | 322 |
| Data Direction Register F ..... | 323 |
| Port G .....                    | 325 |
| Port G Data Register .....      | 325 |
| Data Direction Register G ..... | 326 |
| Port H .....                    | 327 |
| Port H Data Register .....      | 327 |
| Data Direction Register H ..... | 328 |

## Introduction

Fifty bidirectional input/output (I/O) form seven parallel ports. All I/O pins are programmable as inputs or outputs.

**NOTE:** *Connect any unused I/O pins to an appropriate logic level, either  $V_{DD}$  or  $V_{SS}$ . Although the I/O ports do not require termination for proper operation, termination reduces excess current consumption and the possibility of electrostatic damage.*

| Addr.  | Register Name                    | Bit 7  | 6     | 5     | 4     | 3     | 2     | 1      | Bit 0 |
|--------|----------------------------------|--------|-------|-------|-------|-------|-------|--------|-------|
| \$0000 | Port A Data Register (PTA)       | PTA7   | PTA6  | PTA5  | PTA4  | PTA3  | PTA2  | PTA1   | PTA0  |
| \$0001 | Port B Data Register (PTB)       | PTB7   | PTB6  | PTB5  | PTB4  | PTB3  | PTB2  | PTB1   | PTB0  |
| \$0002 | Port C Data Register (PTC)       | 0      | 0     | PTC5  | PTC4  | PTC3  | PTC2  | PTC1   | PTC0  |
| \$0003 | Port D Data Register (PTD)       | PTD7   | PTD6  | PTD5  | PTD4  | PTD3  | PTD2  | PTD1   | PTD0  |
| \$0004 | Data Direction Register A (DDRA) | DDRA7  | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1  | DDRA0 |
| \$0005 | Data Direction Register B (DDRB) | DDRB7  | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1  | DDRB0 |
| \$0006 | Data Direction Register C (DDRC) | MCLKEN | 0     | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1  | DDRC0 |
| \$0007 | Data Direction Register D (DDRD) | DDRD7  | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1  | DDRD0 |
| \$0008 | Port E Data Register (PTE)       | PTE7   | PTE6  | PTE5  | PTE4  | PTE3  | PTE2  | PTE1   | PTE0  |
| \$0009 | Port F Data Register (PTF)       | 0      | PTF6  | PTF5  | PTF4  | PTF3  | PTF2  | PTF1   | PTF0  |
| \$000A | Port G Data Register (PTG)       | 0      | 0     | 0     | 0     | 0     | PTG2  | PTG1   | PTG0  |
| \$000B | Port H Data Register (PTH)       | 0      | 0     | 0     | 0     | 0     | 0     | PTH1   | PTH0  |
| \$000C | Data Direction Register E (DDRE) | DDRE7  | DDRE6 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | DDRE1  | DDRE0 |
| \$000D | Data Direction Register F (DDRF) | 0      | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1  | DDRF0 |
| \$000E | Data Direction Register G (DDRG) | 0      | 0     | 0     | 0     | 0     | DDRG2 | DDRG1  | DDRG0 |
| \$000F | Data Direction Register H (DDRH) | 0      | 0     | 0     | 0     | 0     | 0     | DDRH1' | DDRH0 |

**Figure 1. CAN Protocol I/O Port Register Summary**

---



---

## Port A

Port A is an 8-bit general-purpose bidirectional I/O port.

### Port A Data Register

The port A data register contains a data latch for each of the eight port A pins.

Address: \$0000

|        | Bit 7               | 6    | 5    | 4    | 3    | 2    | 1    | Bit 0 |
|--------|---------------------|------|------|------|------|------|------|-------|
| Read:  | PTA7                | PTA6 | PTA5 | PTA4 | PTA3 | PTA2 | PTA1 | PTA0  |
| Write: |                     |      |      |      |      |      |      |       |
| Reset: | Unaffected by Reset |      |      |      |      |      |      |       |

**Figure 2. Port A Data Register (PTA)**

### PTA[7:0] — Port A Data Bits

These read/write bits are software programmable. Data direction of each port A pin is under the control of the corresponding bit in data direction register A. Reset has no effect on port A data.

### Data Direction Register A

Data direction register A determines whether each port A pin is an input or an output. Writing a logic 1 to a DDRA bit enables the output buffer for the corresponding port A pin; a logic 0 disables the output buffer.

Address: \$0004

|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
| Read:  | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| Write: |       |       |       |       |       |       |       |       |
| Reset: | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

**Figure 3. Data Direction Register A (DDRA)**

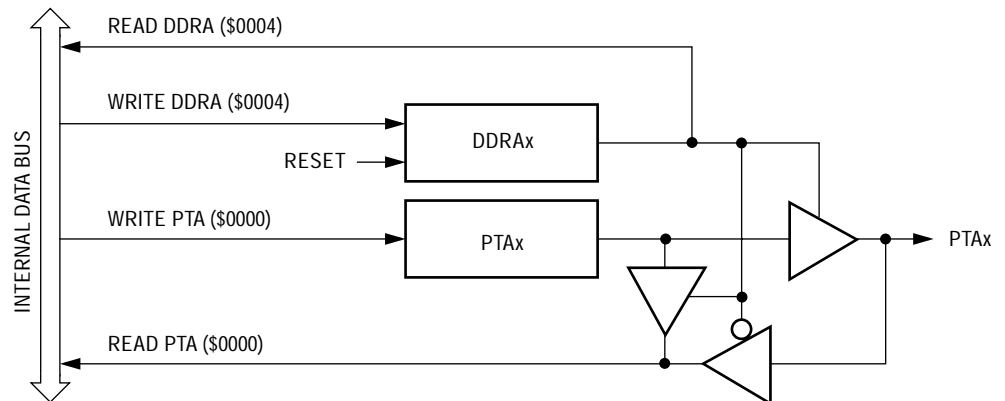
## DDRA[7:0] — Data Direction Register A Bits

These read/write bits control port A data direction. Reset clears DDRA[7:0], configuring all port A pins as inputs.

- 1 = Corresponding port A pin configured as output
- 0 = Corresponding port A pin configured as input

**NOTE:** Avoid glitches on port A pins by writing to the port A data register before changing data direction register A bits from 0 to 1.

Figure 4 shows the port A I/O logic.



**Figure 4. Port A I/O Circuit**

When bit DDRAx is a logic 1, reading address \$0000 reads the PTAx data latch. When bit DDRAx is a logic 0, reading address \$0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 1 summarizes the operation of the port A pins.

**Table 1. Port A Pin Functions**

| DDRA Bit | PTA Bit | I/O Pin Mode | Accesses to DDRA | Accesses to PTA |                         |
|----------|---------|--------------|------------------|-----------------|-------------------------|
|          |         |              | Read/Write       | Read            | Write                   |
| 0        | X       | Input, Hi-Z  | DDRA[7:0]        | Pin             | PTA[7:0] <sup>(1)</sup> |
| 1        | X       | Output       | DDRA[7:0]        | PTA[7:0]        | PTA[7:0]                |

X = don't care

Hi-Z = high impedance

1. Writing affects data register, but does not affect input.

---

---

## Port B

Port B is an 8-bit special function port that shares all of its pins with the analog-to-digital converter.

### Port B Data Register

The port B data register contains a data latch for each of the eight port B pins.

Address: \$0001

|                      | Bit 7               | 6    | 5    | 4    | 3    | 2    | 1    | Bit 0 |
|----------------------|---------------------|------|------|------|------|------|------|-------|
| Read:                | PTB7                | PTB6 | PTB5 | PTB4 | PTB3 | PTB2 | PTB1 | PTB0  |
| Write:               | PTB7                | PTB6 | PTB5 | PTB4 | PTB3 | PTB2 | PTB1 | PTB0  |
| Reset:               | Unaffected by Reset |      |      |      |      |      |      |       |
| Alternate Functions: | ATD7                | ATD6 | ATD5 | ATD4 | ATD3 | ATD2 | ATD1 | ATD0  |

**Figure 5. Port B Data Register (PTB)**

### PTB[7:0] — Port B Data Bits

These read/write bits are software programmable. Data direction of each port B pin is under the control of the corresponding bit in data direction register B. Reset has no effect on port B data.

## ATD[7:0] — ADC Channels

PTB7/ATD7–PTB0/ATD0 are eight of the analog-to-digital converter channels. The ADC channel select bits, CH[4:0], determine whether the PTB7/ATD7–PTB0/ATD0 pins are ADC channels or general-purpose I/O pins. If an ADC channel is selected and a read of this corresponding bit in the port B data register occurs, the data will be 0 if the data direction for this bit is programmed as an input. Otherwise, the data will reflect the value in the data latch. (See [Analog-to-Digital Converter \(ADC-15\)](#) on page 421). Data direction register B (DDRB) does not affect the data direction of port B pins that are being used by the ADC. However, the DDRB bits always determine whether reading port B returns to the states of the latches or logic 0.

## Data Direction Register B

Data direction register B determines whether each port B pin is an input or an output. Writing a logic 1 to a DDRB bit enables the output buffer for the corresponding port B pin; a logic 0 disables the output buffer.

Address: \$0005

|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
| Read:  | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| Write: |       |       |       |       |       |       |       |       |
| Reset: | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

**Figure 6. Data Direction Register B (DDRB)**

## DDRB[7:0] — Data Direction Register B Bits

These read/write bits control port B data direction. Reset clears DDRB[7:0], configuring all port B pins as inputs.

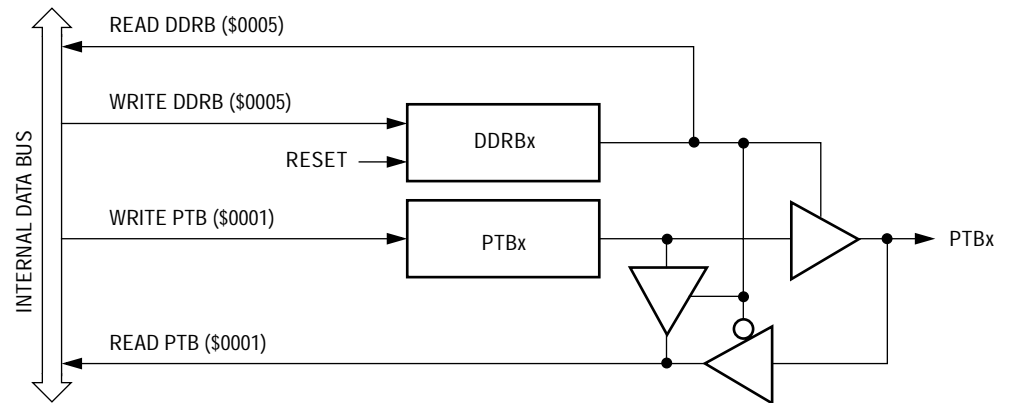
1 = Corresponding port B pin configured as output

0 = Corresponding port B pin configured as input

**NOTE:** *Avoid glitches on port B pins by writing to the port B data register before changing data direction register B bits from 0 to 1.*

**Figure 7** shows the port B I/O logic.





**Figure 7. Port B I/O Circuit**

When bit DDRBx is a logic 1, reading address \$0001 reads the PTBx data latch. When bit DDRBx is a logic 0, reading address \$0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. [Table 2](#) summarizes the operation of the port B pins.

**Table 2. Port B Pin Functions**

| DDRB Bit | PTB Bit | I/O Pin Mode | Accesses to DDRB | Accesses to PTB |                         |
|----------|---------|--------------|------------------|-----------------|-------------------------|
|          |         |              | Read/Write       | Read            | Write                   |
| 0        | X       | Input, Hi-Z  | DDRB[7:0]        | Pin             | PTB[7:0] <sup>(1)</sup> |
| 1        | X       | Output       | DDRB[7:0]        | PTB[7:0]        | PTB[7:0]                |

X = don't care

Hi-Z = high impedance

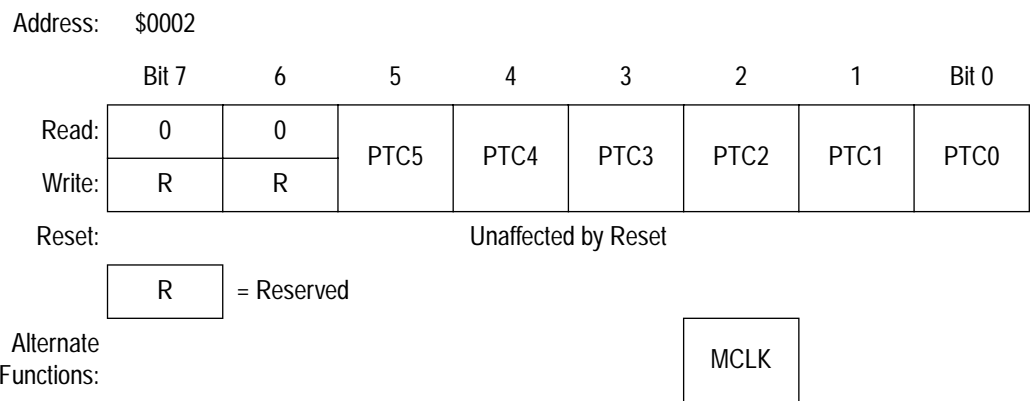
1. Writing affects data register, but does not affect input.

## Port C

Port C is an 6-bit general-purpose bidirectional I/O port.

### Port C Data Register

The port C data register contains a data latch for each of the six port C pins.



**Figure 8. Port C Data Register (PTC)**

#### PTC[5:0] — Port C Data Bits

These read/write bits are software-programmable. Data direction of each port C pin is under the control of the corresponding bit in data direction register C. Reset has no effect on port C data (5:0).

#### MCLK — T12 System Clock Bit

The system clock is driven out of PTC2 when enabled by MCLKEN bit in PTCDDR7.

## Data Direction Register C

Data direction register C determines whether each port C pin is an input or an output. Writing a logic 1 to a DDRC bit enables the output buffer for the corresponding port C pin; a logic 0 disables the output buffer.

Address: \$0006

|        | Bit 7  | 6 | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|--------|---|-------|-------|-------|-------|-------|-------|
| Read:  | MCLKEN | 0 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| Write: |        | R |       |       |       |       |       |       |
| Reset: | 0      | 0 | 0     | 0     | 0     | 0     | 0     | 0     |

R = Reserved

**Figure 9. Data Direction Register C (DDRC)**

### MCLKEN — MCLK Enable Bit

This read/write bit enables MCLK to be an output signal on PTC2. If MCLK is enabled, DDRC2 has no effect. Reset clears this bit.

1 = MCLK output enabled

0 = MCLK output disabled

### DDRC[5:0] — Data Direction Register C Bits

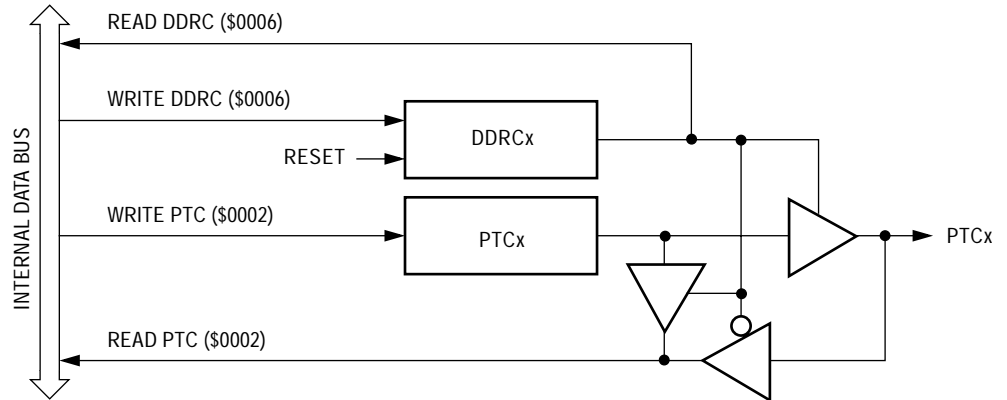
These read/write bits control port C data direction. Reset clears DDRC[7:0], configuring all port C pins as inputs.

1 = Corresponding port C pin configured as output

0 = Corresponding port C pin configured as input

**NOTE:** *Avoid glitches on port C pins by writing to the port C data register before changing data direction register C bits from 0 to 1.*

**Figure 10** shows the port C I/O logic.



**Figure 10. Port C I/O Circuit**

When bit DDRCx is a logic 1, reading address \$0002 reads the PTCx data latch. When bit DDRCx is a logic 0, reading address \$0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. [Table 3](#) summarizes the operation of the port C pins.

**Table 3. Port C Pin Functions**

| Bit Value | PTC Bit | I/O Pin Mode | Accesses to DDRC | Accesses to PTC |                         |
|-----------|---------|--------------|------------------|-----------------|-------------------------|
|           |         |              | Read/Write       | Read            | Write                   |
| 0         | 2       | Input, Hi-Z  | DDRC[2]          | Pin             | PTC2                    |
| 1         | 2       | Output       | DDRC[2]          | 0               | —                       |
| 0         | X       | Input, Hi-Z  | DDRC[5:0]        | Pin             | PTC[5:0] <sup>(1)</sup> |
| 1         | X       | Output       | DDRC[5:0]        | PTC[5:0]        | PTC[5:0]                |

X = don't care

Hi-Z = high impedance

1. Writing affects data register, but does not affect input.

---



---

## Port D

Port D is an 8-bit general-purpose I/O port.

### Port D Data Register

Port D is a 8-bit special function port that shares seven of its pins with the analog to digital converter and two with the timer interface modules.

|                      |                     |                 |       |                 |       |       |      |       |
|----------------------|---------------------|-----------------|-------|-----------------|-------|-------|------|-------|
| Address:             | \$0003              |                 |       |                 |       |       |      |       |
|                      | Bit 7               | 6               | 5     | 4               | 3     | 2     | 1    | Bit 0 |
| Read:                | PTD7                | PTD6            | PTD5  | PTD4            | PTD3  | PTD2  | PTD1 | PTD0  |
| Write:               | PTD7                | PTD6            | PTD5  | PTD4            | PTD3  | PTD2  | PTD1 | PTD0  |
| Reset:               | Unaffected by Reset |                 |       |                 |       |       |      |       |
| Alternate Functions: | R                   | ATD14/<br>TACLK | ATD13 | ATD12/<br>TBCLK | ATD11 | ATD10 | ATD9 | ATD8  |

**Figure 11. Port D Data Register (PTD)**

#### PTD[7:0] — Port D Data Bits

PTD[7:0] are read/write, software programmable bits. Data direction of PTD[7:0] pins are under the control of the corresponding bit in data direction register D.

#### ATD[14:8] — ADC Channel Status Bits

PTD6/ATD14/TACLK–PTD0/ATD8 are seven of the 15 analog-to-digital converter channels. The ADC channel select bits, CH[4:0], determine whether the PTD6/ATD14/TACLK–PTD0/ATD8 pins are ADC channels or general-purpose I/O pins. If an ADC channel is selected and a read of this corresponding bit in the port B data register occurs, the data will be 0 if the data direction for this bit is programmed as an input. Otherwise, the data will reflect the value in the data latch. (See [Analog-to-Digital Converter \(ADC-15\)](#) on page 421).

*Data direction register D (DDRD) does not affect the data direction of port D pins that are being used by the TIMA or TIMB. However, the*

DDRD bits always determine whether reading port D returns the states of the latches or logic 0.

## TACLK/TBCLK — Timer Clock Input Bit

The PTD6/TACLK pin is the external clock input for the TIMA. The PTD4/TBCLK pin is the external clock input for the TIMB. The prescaler select bits, PS[2:0], select PTD6/TACLK or PTD4/TBCLK as the TIM clock input. (See [TIMA Channel Status and Control Registers](#) on page 410 and [TIMB Channel Status and Control Registers](#) on page 288). When not selected as the TIM clock, PTD6/TACLK and PTD4/TBCLK are available for general-purpose I/O. While TACLK/TBCLK are selected corresponding DDRD bits have no effect.

## Data Direction Register D

Data direction register D determines whether each port D pin is an input or an output. Writing a logic 1 to a DDRD bit enables the output buffer for the corresponding port D pin; a logic 0 disables the output buffer.

Address: \$0007

|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
| Read:  | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
| Write: |       |       |       |       |       |       |       |       |
| Reset: | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

**Figure 12. Data Direction Register D (DDRD)**

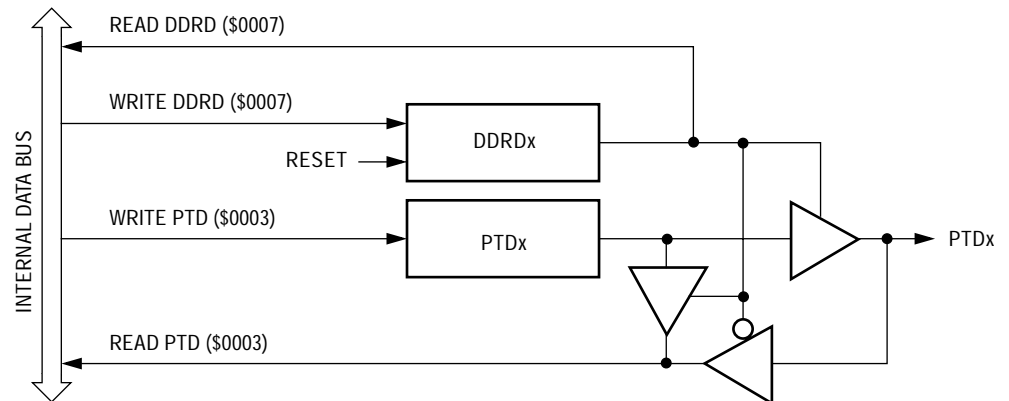
## DDRD[7:0] — Data Direction Register D Bits

These read/write bits control port D data direction. Reset clears DDRD[7:0], configuring all port D pins as inputs.

- 1 = Corresponding port D pin configured as output
- 0 = Corresponding port D pin configured as input

**NOTE:** Avoid glitches on port D pins by writing to the port D data register before changing data direction register D bits from 0 to 1.

[Figure 13](#) shows the port D I/O logic.



**Figure 13. Port D I/O Circuit**

When bit DDRDx is a logic 1, reading address \$0003 reads the PTDx data latch. When bit DDRDx is a logic 0, reading address \$0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. [Table 4](#) summarizes the operation of the port D pins.

**Table 4. Port D Pin Functions**

| DDRD Bit | PTD Bit | I/O Pin Mode | Accesses to DDRD | Accesses to PTD |                         |
|----------|---------|--------------|------------------|-----------------|-------------------------|
|          |         |              | Read/Write       | Read            | Write                   |
| 0        | X       | Input, Hi-Z  | DDRD[7:0]        | Pin             | PTD[7:0] <sup>(1)</sup> |
| 1        | X       | Output       | DDRD[7:0]        | PTD[7:0]        | PTD[7:0]                |

X = don't care

Hi-Z = high impedance

1. Writing affects data register, but does not affect input.

## Port E

Port E is an 8-bit special function port that shares two of its pins with the timer interface module (TIMA), two of its pins with the serial communications interface module (SCI), and four of its pins with the serial peripheral interface module (SPI).

### Port E Data Register

The port E data register contains a data latch for each of the eight port E pins.

Address: \$0008

|                     | Bit 7               | 6    | 5    | 4               | 3     | 2     | 1    | Bit 0 |
|---------------------|---------------------|------|------|-----------------|-------|-------|------|-------|
| Read:               | PTE7                | PTE6 | PTE5 | PTE4            | PTE3  | PTE2  | PTE1 | PTE0  |
| Write:              | PTE7                | PTE6 | PTE5 | PTE4            | PTE3  | PTE2  | PTE1 | PTE0  |
| Reset:              | Unaffected by Reset |      |      |                 |       |       |      |       |
| Alternate Function: | SPSCK               | MOSI | MISO | $\overline{SS}$ | TACH1 | TACH0 | RxD  | TxD   |

**Figure 14. Port E Data Register (PTE)**

#### PTE[7:0] — Port E Data Bits

PTE[7:0] are read/write, software programmable bits. Data direction of each port E pin is under the control of the corresponding bit in data direction register E.

#### SPSCK — SPI Serial Clock Bit

The PTE7/SPSCK pin is the serial clock input of an SPI slave module and serial clock output of an SPI master module. When the SPE bit is clear, the PTE7/SPSCK pin is available for general-purpose I/O. (See [SPI Control Register](#) on page 262).

#### MOSI — Master Out/Slave In Bit

The PTE6/MOSI pin is the master out/slave in terminal of the SPI module. When the SPE bit is clear, the PTE6/MOSI pin is available for general-purpose I/O.



**MISO** — Master In/Slave Out Bit

The PTE5/MISO pin is the master in/slave out terminal of the SPI module. When the SPI enable bit, SPE, is clear, the SPI module is disabled, and the PTE5/MISO pin is available for general-purpose I/O. (See [SPI Control Register](#) on page 262).

 **$\overline{SS}$**  — Slave Select Bit

The PTE4/ $\overline{SS}$  pin is the slave select input of the SPI module. When the SPE bit is clear, or when the SPI master bit, SPMSTR, is set and MODFEN bit is low, the PTE4/ $\overline{SS}$  pin is available for general-purpose I/O. (See [SS \(Slave Select\)](#) on page 260). When the SPI is enabled as a slave, the DDRF0 bit in data direction register E (DDRE) has no effect on the PTE4/ $\overline{SS}$  pin.

**NOTE:** *Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SPI module. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See [Table 5](#)).*

**TACH[1:0]** — Timer Channel I/O Bits

The PTE3/TACH1–PTE2/TACH0 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTE3/TACH1–PTE2/TACH0 pins are timer channel I/O pins or general-purpose I/O pins. (See [TIMA Channel Status and Control Registers](#) on page 410).

**NOTE:** *Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the TIM. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See [Table 5](#)).*

**RxD** — SCI Receive Data Input Bit

The PTE1/RxD pin is the receive data input for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE1/RxD pin is available for general-purpose I/O. (See [SCI Control Register 1](#) on page 220).

## TxD — SCI Transmit Data Output

The PTE0/TxD pin is the transmit data output for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE0/TxD pin is available for general-purpose I/O. (See [SCI Control Register 1](#) on page 220).

**NOTE:** *Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SCI module. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See [Table 5](#)).*

## Data Direction Register E

Data direction register E determines whether each port E pin is an input or an output. Writing a logic 1 to a DDRE bit enables the output buffer for the corresponding port E pin; a logic 0 disables the output buffer.

Address: \$000C

|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
| Read:  | DDRE7 | DDRE6 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | DDRE1 | DDRE0 |
| Write: | DDRE7 | DDRE6 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | DDRE1 | DDRE0 |
| Reset: | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

**Figure 15. Data Direction Register E (DDRE)**

## DDRE[7:0] — Data Direction Register E Bits

These read/write bits control port E data direction. Reset clears DDRE[7:0], configuring all port E pins as inputs.

1 = Corresponding port E pin configured as output

0 = Corresponding port E pin configured as input

**NOTE:** *Avoid glitches on port E pins by writing to the port E data register before changing data direction register E bits from 0 to 1.*

[Figure 16](#) shows the port E I/O logic.

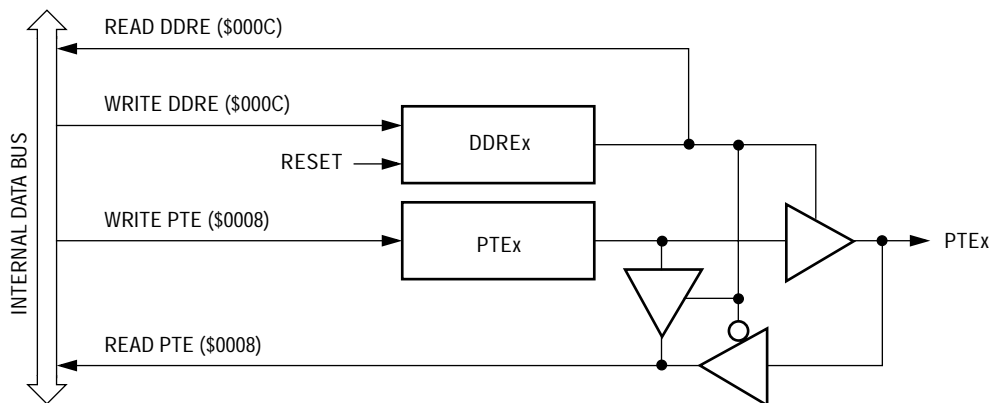


Figure 16. Port E I/O Circuit

When bit DDREx is a logic 1, reading address \$0008 reads the PTEx data latch. When bit DDREx is a logic 0, reading address \$0008 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. [Table 5](#) summarizes the operation of the port E pins.

Table 5. Port E Pin Functions

| DDRE Bit | PTE Bit | I/O Pin Mode | Accesses to DDRE | Accesses to PTE |                         |
|----------|---------|--------------|------------------|-----------------|-------------------------|
|          |         |              | Read/Write       | Read            | Write                   |
| 0        | X       | Input, Hi-Z  | DDRE[7:0]        | Pin             | PTE[7:0] <sup>(1)</sup> |
| 1        | X       | Output       | DDRE[7:0]        | PTE[7:0]        | PTE[7:0]                |

X = don't care

Hi-Z = high impedance

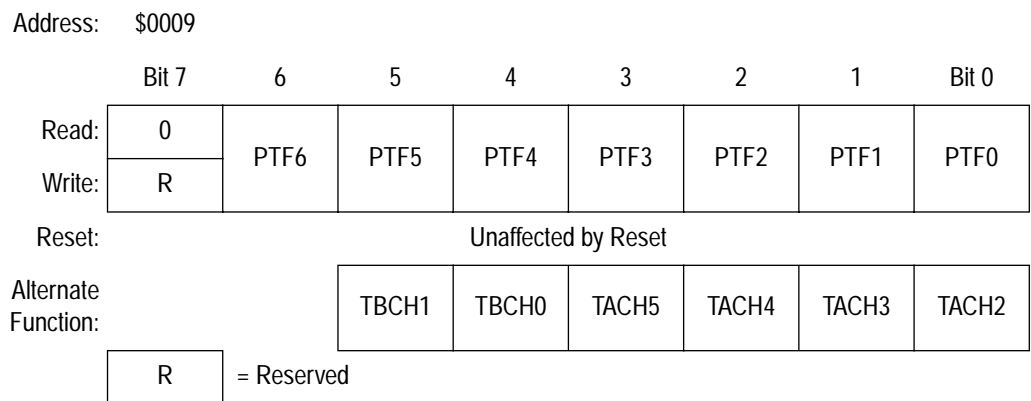
1. Writing affects data register, but does not affect input.

## Port F

Port F is a 7-bit special function port that shares four of its pins with the timer interface module (TIMA-6) and two of its pins with the timer interface module (TIMB).

### Port F Data Register

The port F data register contains a data latch for each of the seven port F pins.



**Figure 17. Port F Data Register (PTF)**

#### PTF[6:0] — Port F Data Bits

These read/write bits are software programmable. Data direction of each port F pin is under the control of the corresponding bit in data direction register F. Reset has no effect on PTF[6:0].

#### TACH[5:2] — Timer A Channel I/O Bits

The PTF3–PTF0/TACH2 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTF3–PTF0/TACH2 pins are timer channel I/O pins or general-purpose I/O pins. (See [TIMA Status and Control Register](#) on page 406).

**TBCH[1:0] — Timer B Channel I/O Bits**

The PTF5/TBCH1–PTF4/TBCH0 pins are the TIMB input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTF5/TBCH1–PTF4/TBCH0 pins are timer channel I/O pins or general-purpose I/O pins. (See [TIMB Status and Control Register](#) on page 284).

**NOTE:** *Data direction register F (DDRF) does not affect the data direction of port F pins that are being used by the TIM. However, the DDRF bits always determine whether reading port F returns the states of the latches or the states of the pins. (See [Table 6](#)).*

**Data Direction Register F**

Data direction register F determines whether each port F pin is an input or an output. Writing a logic 1 to a DDRF bit enables the output buffer for the corresponding port F pin; a logic 0 disables the output buffer.

Address: \$000D

|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
| Read:  | 0     | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| Write: | R     |       |       |       |       |       |       |       |
| Reset: | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

R = Reserved

**Figure 18. Data Direction Register F (DDRF)**

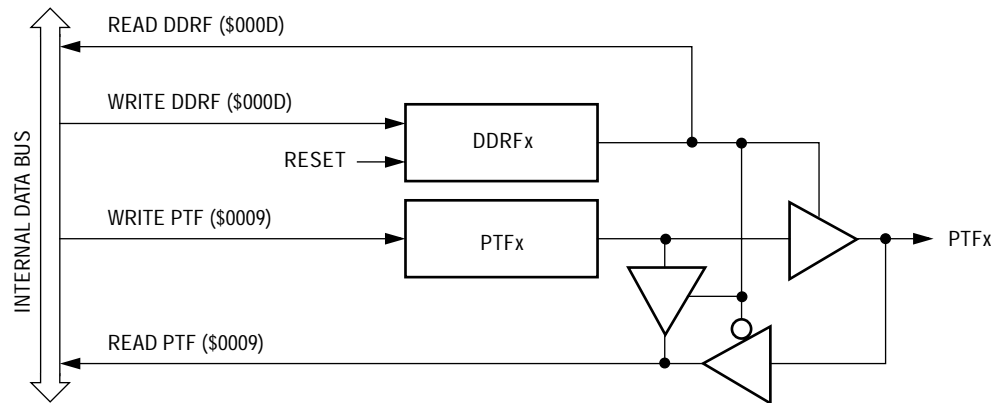
**DDRF[6:0] — Data Direction Register F Bits**

These read/write bits control port F data direction. Reset clears DDRF[6:0], configuring all port F pins as inputs.

- 1 = Corresponding port F pin configured as output
- 0 = Corresponding port F pin configured as input

**NOTE:** *Avoid glitches on port F pins by writing to the port F data register before changing data direction register F bits from 0 to 1.*

[Figure 19](#) shows the port F I/O logic.



**Figure 19. Port F I/O Circuit**

When bit DDRFx is a logic 1, reading address \$0009 reads the PTFx data latch. When bit DDRFx is a logic 0, reading address \$0009 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. [Table 6](#) summarizes the operation of the port F pins.

**Table 6. Port F Pin Functions**

| DDRF Bit | PTF Bit | I/O Pin Mode | Accesses to DDRF | Accesses to PTF |                         |
|----------|---------|--------------|------------------|-----------------|-------------------------|
|          |         |              | Read/Write       | Read            | Write                   |
| 0        | X       | Input, Hi-Z  | DDRF[6:0]        | Pin             | PTF[6:0] <sup>(1)</sup> |
| 1        | X       | Output       | DDRF[6:0]        | PTF[6:0]        | PTF[6:0]                |

X = don't care

Hi-Z = high impedance

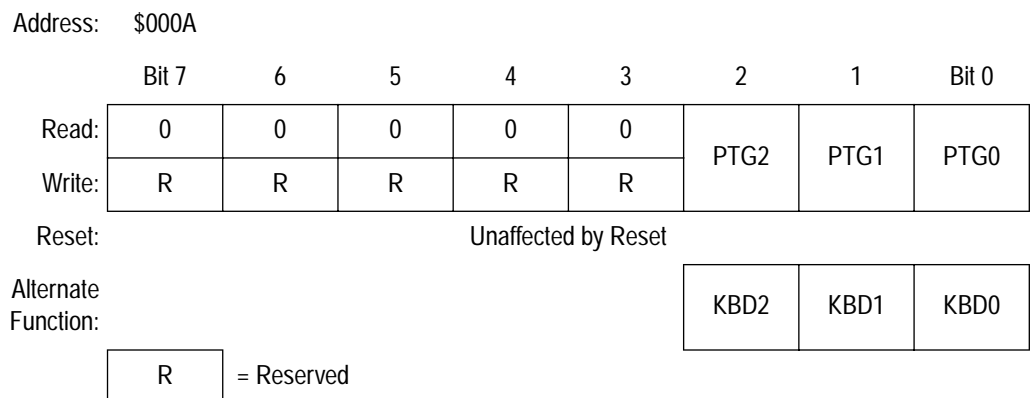
1. Writing affects data register, but does not affect input.

## Port G

Port G is a 3-bit special function port that shares all of its pins with the keyboard interrupt module (KBD).

### Port G Data Register

The port G data register contains a data latch for each of the three port G pins.



**Figure 20. Port G Data Register (PTG)**

#### PTG[2:0] — Port G Data Bits

These read/write bits are software programmable. Data direction of each port G pin is under the control of the corresponding bit in data direction register G. Reset has no effect on PTG[2:0].

#### KBD[2:0] — Keyboard Wakeup pins

The keyboard interrupt enable bits, KBIE[2:0], in the keyboard interrupt control register, enable the port G pins as external interrupt pins (See [Keyboard Module \(KBD\)](#) on page 381). Enabling an external interrupt pin will override the corresponding DDRGx.

## Data Direction Register G

Data direction register G determines whether each port G pin is an input or an output. Writing a logic 1 to a DDRG bit enables the output buffer for the corresponding port G pin; a logic 0 disables the output buffer.

Address: \$000E

|        | Bit 7 | 6 | 5 | 4 | 3 | 2     | 1     | Bit 0 |
|--------|-------|---|---|---|---|-------|-------|-------|
| Read:  | 0     | 0 | 0 | 0 | 0 | DDRG2 | DDRG1 | DDRG0 |
| Write: | R     | R | R | R | R |       |       |       |
| Reset: | 0     | 0 | 0 | 0 | 0 | 0     | 0     | 0     |

R = Reserved

**Figure 21. Data Direction Register G (DDRG)**

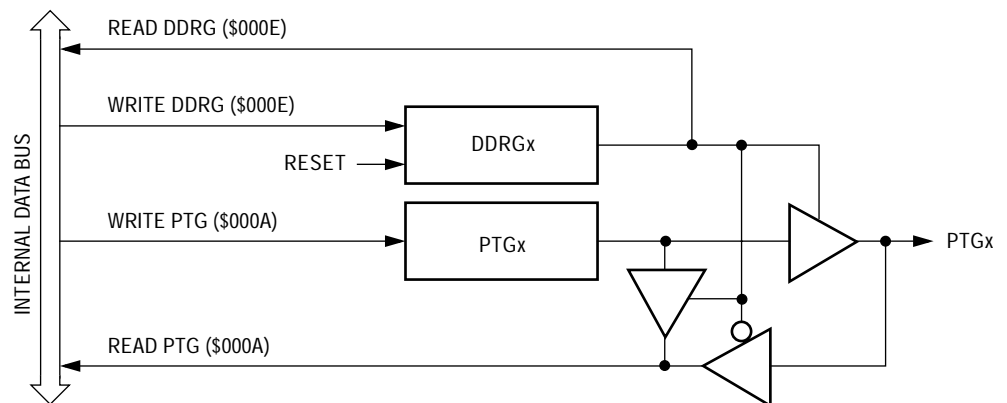
### DDRG[2:0] — Data Direction Register G Bits

These read/write bits control port G data direction. Reset clears DDRG[2:0], configuring all port G pins as inputs.

- 1 = Corresponding port G pin configured as output
- 0 = Corresponding port G pin configured as input

**NOTE:** Avoid glitches on port G pins by writing to the port G data register before changing data direction register G bits from 0 to 1.

Figure 22 shows the port G I/O logic.



**Figure 22. Port G I/O Circuit**



When bit DDRGx is a logic 1, reading address \$000A reads the PTGx data latch. When bit DDRGx is a logic 0, reading address \$000A reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. [Table 7](#) summarizes the operation of the port G pins.

**Table 7. Port G Pin Functions**

| DDRG Bit | PTG Bit | I/O Pin Mode | Accesses to DDRG | Accesses to PTG |                         |
|----------|---------|--------------|------------------|-----------------|-------------------------|
|          |         |              | Read/Write       | Read            | Write                   |
| 0        | X       | Input, Hi-Z  | DDRG[2:0]        | Pin             | PTG[2:0] <sup>(1)</sup> |
| 1        | X       | Output       | DDRG[2:0]        | PTG[2:0]        | PTG[2:0]                |

X = don't care

Hi-Z = high impedance

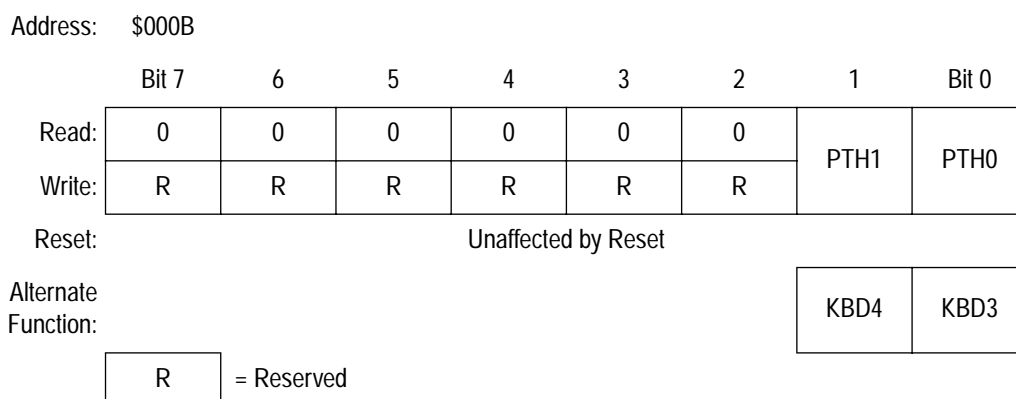
1. Writing affects data register, but does not affect input.

## Port H

Port H is a 2-bit special function port that shares all of its pins with the keyboard interrupt module (KBD).

### Port H Data Register

The port H data register contains a data latch for each of the two port H pins.



**Figure 23. Port H Data Register (PTH)**

## PTH[1:0] — Port H Data Bits

These read/write bits are software programmable. Data direction of each port H pin is under the control of the corresponding bit in data direction register H. Reset has no effect on PTH[1:0].

## KBD[4:3] — Keyboard Wake-up pins

The keyboard interrupt enable bits, KBIE[4:3], in the keyboard interrupt control register, enable the port H pins as external interrupt pins (See [Keyboard Module \(KBD\)](#) on page 381).

## Data Direction Register H

Data direction register H determines whether each port H pin is an input or an output. Writing a logic 1 to a DDRH bit enables the output buffer for the corresponding port H pin; a logic 0 disables the output buffer.

Address: \$000F

|        | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1     | Bit 0 |
|--------|-------|---|---|---|---|---|-------|-------|
| Read:  | 0     | 0 | 0 | 0 | 0 | 0 | DDRH1 | DDRH0 |
| Write: | R     | R | R | R | R | R |       |       |
| Reset: | 0     | 0 | 0 | 0 | 0 | 0 | 0     | 0     |

R = Reserved

**Figure 24. Data Direction Register H (DDRH)**

## DDRH[1:0] — Data Direction Register H Bits

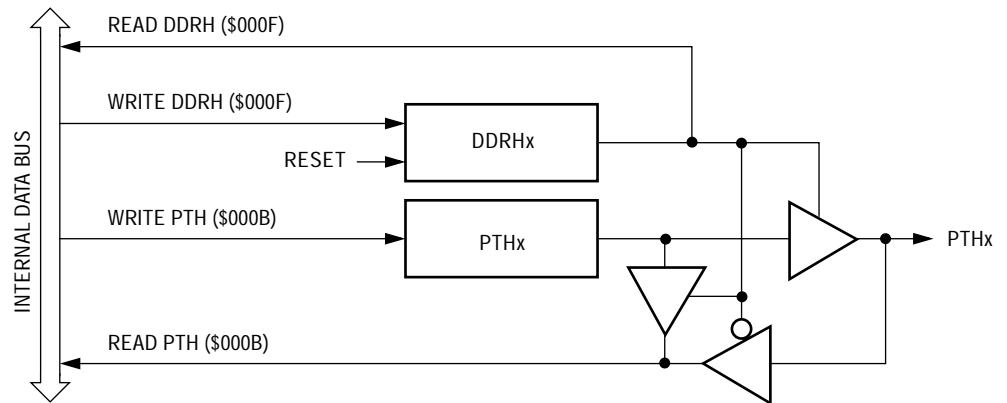
These read/write bits control port H data direction. Reset clears DDRG[1:0], configuring all port H pins as inputs.

1 = Corresponding port H pin configured as output

0 = Corresponding port H pin configured as input

**NOTE:** *Avoid glitches on port H pins by writing to the port H data register before changing data direction register H bits from 0 to 1.*

**Figure 25** shows the port H I/O logic.



**Figure 25. Port H I/O Circuit**

When bit DDRHx is a logic 1, reading address \$000B reads the PTHx data latch. When bit DDRHx is a logic 0, reading address \$000B reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. [Table 8](#) summarizes the operation of the port H pins.

**Table 8. Port H Pin Functions**

| DDRH Bit | PTH Bit | I/O Pin Mode | Accesses to DDRH | Accesses to PTH |                         |
|----------|---------|--------------|------------------|-----------------|-------------------------|
|          |         |              | Read/Write       | Read            | Write                   |
| 0        | X       | Input, Hi-Z  | DDRH[1:0]        | Pin             | PTH[1:0] <sup>(1)</sup> |
| 1        | X       | Output       | DDRH[1:0]        | PTH[1:0]        | PTH[1:0]                |

X = don't care

Hi-Z = high impedance

1. Writing affects data register, but does not affect input.



# MSCAN Controller (MSCAN08)

---

---

## Contents

|   |     |
|---|-----|
| Introduction . . . . .                            | 332 |
| Features . . . . .                                | 333 |
| External Pins . . . . .                           | 334 |
| Message Storage . . . . .                         | 335 |
| Background . . . . .                              | 335 |
| Receive Structures . . . . .                      | 336 |
| Transmit Structures . . . . .                     | 339 |
| Identifier Acceptance Filter . . . . .            | 340 |
| Interrupts . . . . .                              | 344 |
| Interrupt Acknowledge . . . . .                   | 345 |
| Interrupt Vectors . . . . .                       | 345 |
| Protocol Violation Protection . . . . .           | 346 |
| Low Power Modes . . . . .                         | 346 |
| MSCAN08 Sleep Mode . . . . .                      | 347 |
| MSCAN08 Soft Reset Mode . . . . .                 | 349 |
| MSCAN08 Power Down Mode . . . . .                 | 349 |
| CPU Wait Mode . . . . .                           | 350 |
| Programmable Wakeup Function . . . . .            | 350 |
| Timer Link . . . . .                              | 350 |
| Clock System . . . . .                            | 351 |
| Memory Map . . . . .                              | 355 |
| Programmer's Model of Message Storage . . . . .   | 355 |
| Message Buffer Outline . . . . .                  | 356 |
| Identifier Registers . . . . .                    | 356 |
| Data Length Register (DLR) . . . . .              | 359 |
| Data Segment Registers (DSRn) . . . . .           | 359 |
| Transmit Buffer Priority Registers . . . . .      | 360 |
| Programmer's Model of Control Registers . . . . . | 360 |
| MSCAN08 Module Control Register 0 . . . . .       | 363 |
| MSCAN08 Module Control Register 1 . . . . .       | 364 |
| MSCAN08 Bus Timing Register 0 . . . . .           | 366 |
| MSCAN08 Bus Timing Register 1 . . . . .           | 367 |

## MSCAN Controller (MSCAN08)

|  |     |
|--|-----|
| MSCAN08 Receiver Flag Register (CRFLG) . . . . .         | 368 |
| MSCAN08 Receiver Interrupt Enable Register . . . . .     | 371 |
| MSCAN08 Transmitter Flag Register . . . . .              | 372 |
| MSCAN08 Transmitter Control Register . . . . .           | 374 |
| MSCAN08 Identifier Acceptance Control Register . . . . . | 375 |
| MSCAN08 Receive Error Counter . . . . .                  | 376 |
| MSCAN08 Transmit Error Counter . . . . .                 | 376 |
| MSCAN08 Identifier Acceptance Registers . . . . .        | 377 |
| MSCAN08 Identifier Mask Registers (CIDMR0-3) . . . . .   | 378 |

---

---

### Introduction

The MSCAN08 is the specific implementation of the Motorola scalable controller area network (MSCAN) concept targeted for the Motorola M68HC08 Microcontroller Family.

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September 1991.

The CAN protocol was primarily, but not exclusively, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the electromagnetic interference (EMI) environment of a vehicle, cost-effectiveness and required bandwidth.

MSCAN08 utilizes an advanced buffer arrangement, resulting in a predictable real-time behavior, and simplifies the application software.

---

---

## Features

Basic features of the MSCAN08 are:

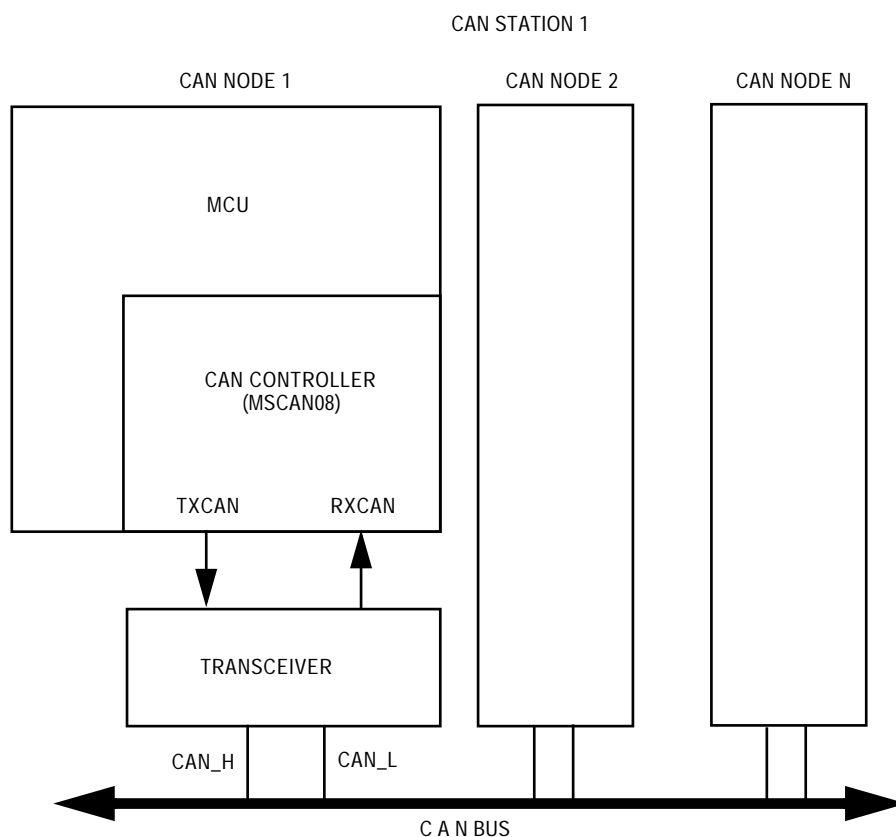
- Modular Architecture
- Implementation of the CAN Protocol — Version 2.0A/B
  - Standard and Extended Data Frames.
  - 0–8 Bytes Data Length.
  - Programmable Bit Rate up to 1 Mbps Depending on the Actual Bit Timing and the Clock Jitter of the PLL
- Support for Remote Frames
- Double-Buffered Receive Storage Scheme
- Triple-Buffered Transmit Storage Scheme with Internal Priorisation Using a “Local Priority” Concept
- Flexible Maskable Identifier Filter Supports Alternatively One Full Size Extended Identifier Filter or Two 16-Bit Filters or Four 8-Bit Filters
- Programmable Wakeup Functionality with Integrated Low-Pass Filter
- Programmable Loop-Back Mode Supports Self-Test Operation
- Separate Signalling and Interrupt Capabilities for All CAN Receiver and Transmitter Error States (Warning, Error Passive, Bus Off)
- Programmable MSCAN08 Clock Source Either CPU Bus Clock or Crystal Oscillator Output
- Programmable Link to On-Chip Timer Interface Module (TIMB) for Time-Stamping and Network Synchronization
- Low-Power Sleep Mode

## MSCAN Controller (MSCAN08)

### External Pins

The MSCAN08 uses two external pins, one input (RxCAN) and one output (TxCAN). The TxCAN output pin represents the logic level on the CAN: 0 is for a dominant state, and 1 is for a recessive state.

A typical CAN system with MSCAN08 is shown in [Figure 1](#).



**Figure 1. The CAN System**

Each CAN station is connected physically to the CAN bus lines through a transceiver chip. The transceiver is capable of driving the large current needed for the CAN and has current protection against defected CAN or defected stations.



---

---

## Message Storage

MSCAN08 facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

### Background

Modern application layer software is built under two fundamental assumptions:

1. Any CAN node is able to send out a stream of scheduled messages without releasing the bus between two messages. Such nodes will arbitrate for the bus right after sending the previous message and will only release the bus in case of lost arbitration.
2. The internal message queue within any CAN node is organized as such that the highest priority message will be sent out first if more than one message is ready to be sent.

Above behavior cannot be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message has been sent. This loading process lasts a definite amount of time and has to be completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme would de-couple the re-loading of the transmit buffers from the actual message being sent and as such reduces the reactivity requirements on the CPU. Problems may arise if the sending of a message would be finished just while the CPU re-loads the second buffer. In that case, no buffer would then be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN08 has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN08 implements with the “local priority” concept described in [Receive Structures](#) on page 336.

## MSCAN Controller (MSCAN08)

**Receive Structures** The received messages are stored in a 2-stage input first in first out (FIFO). The two message buffers are mapped using a Ping Pong arrangement into a single memory area (see [Figure 2](#)). While the background receive buffer (RxBG) is exclusively associated to the MSCAN08, the foreground receive buffer (RxFG) is addressable by the CPU08. This scheme simplifies the handler software, because only one address area is applicable for the receive process.

Both buffers have a size of 13 bytes to store the CAN control bits, the identifier (standard or extended), and the data content (for details, see [Programmer's Model of Message Storage](#) on page 355).

The receiver full flag (RXF) in the MSCAN08 receiver flag register (CRFLG) (see [MSCAN08 Receiver Flag Register \(CRFLG\)](#) on page 368), signals the status of the foreground receive buffer. When the buffer contains a correctly received message with matching identifier, this flag is set.

On reception, each message is checked to see if it passes the filter (for details see [Identifier Acceptance Filter](#) on page 340) and in parallel is written into RxBG. The MSCAN08 copies the content of RxBG into RxFG<sup>1</sup>, sets the RXF flag, and generates a receive interrupt to the CPU<sup>2</sup>. The user's receive handler has to read the received message from RxFG and to reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message which can follow immediately after the IFS field of the CAN frame, is received into RxBG. The overwriting of the background buffer is independent of the identifier filter function.

When the MSCAN08 module is transmitting, the MSCAN08 receives its own messages into the background receive buffer, RxBG. It does NOT overwrite RxFG, generate a receive interrupt or acknowledge its own messages on the CAN bus. The exception to this rule is in loop-back mode (see [MSCAN08 Module Control Register 1](#) on page 364), where the MSCAN08 treats its own messages exactly like all other incoming messages. The MSCAN08 receives its own transmitted messages in the

---

1. Only if the RXF flag is not set.

2. The receive interrupt will occur only if not masked. A polling scheme can be applied on RXF also.

event that it loses arbitration. If arbitration is lost, the MSCAN08 must be prepared to become receiver.

An overrun condition occurs when both the foreground and the background receive message buffers are filled with correctly received messages with accepted identifiers and another message is correctly received from the bus with an accepted identifier. The latter message will be discarded and an error interrupt with overrun indication will be generated if enabled. The MSCAN08 is still able to transmit messages with both receive message buffers filled, but all incoming messages are discarded.

# MSCAN Controller (MSCAN08)

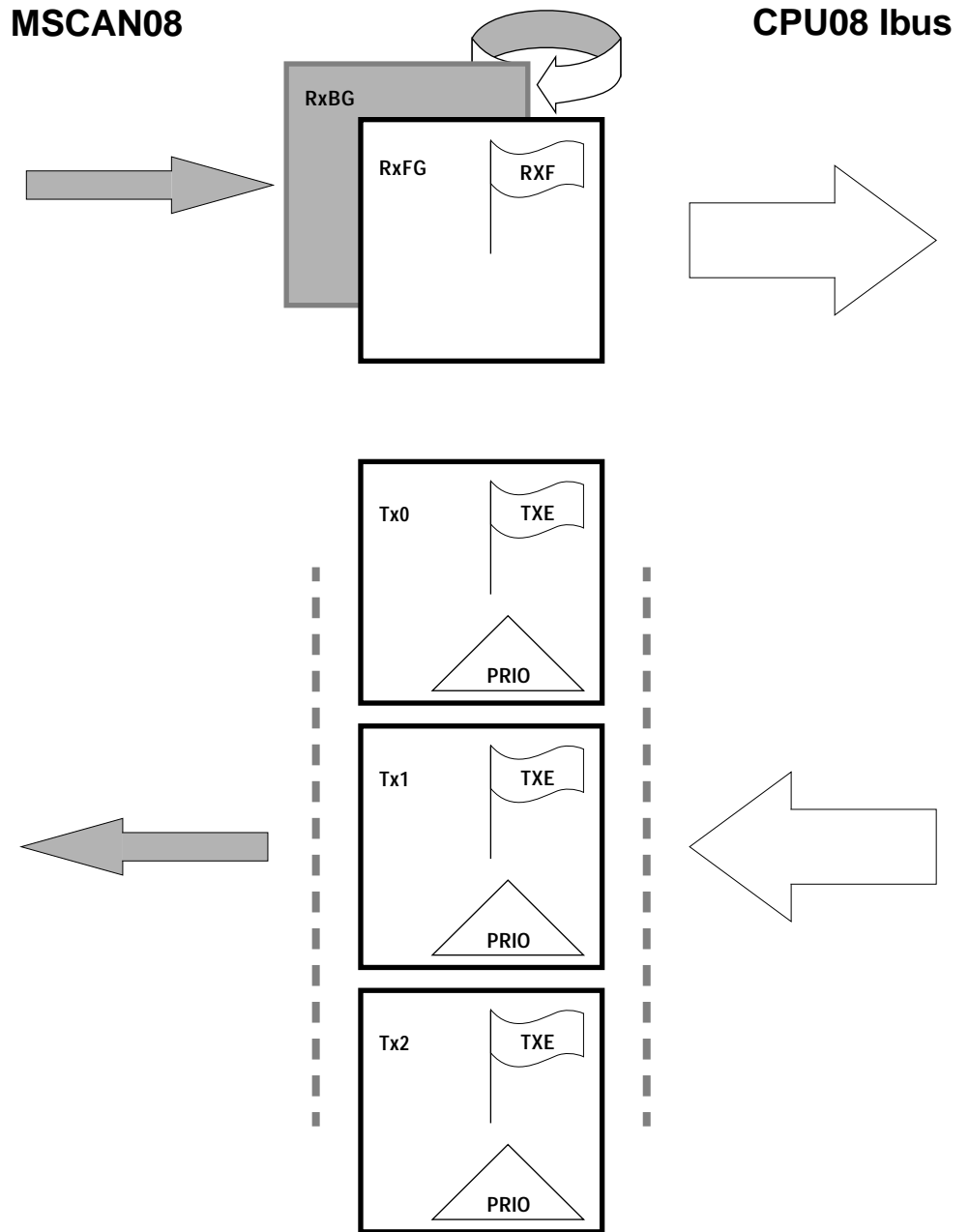


Figure 2. User Model for Message Buffer Organization

## Transmit Structures

The MSCAN08 has a triple transmit buffer scheme to allow multiple messages to be set up in advance and to achieve an optimized real-time performance. The three buffers are arranged as shown in [Figure 2](#).

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see [Programmer's Model of Message Storage](#) on page 355). An additional transmit buffer priority register (TBPR) contains an 8-bit “local priority” field (PRIO) (see [Transmit Buffer Priority Registers](#) on page 360).

To transmit a message, the CPU08 has to identify an available transmit buffer which is indicated by a set transmit buffer empty (TXE) flag in the MSCAN08 transmitter flag register (CTFLG) (see [MSCAN08 Transmitter Flag Register](#) on page 372).

The CPU08 then stores the identifier, the control bits and the data content into one of the transmit buffers. Finally, the buffer has to be flagged ready for transmission by clearing the TXE flag.

The MSCAN08 then will schedule the message for transmission and will signal the successful transmission of the buffer by setting the TXE flag. A transmit interrupt is generated<sup>1</sup> when TXE is set and can be used to drive the application software to re-load the buffer.

In case more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN08 uses the local priority setting of the three buffers for prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software sets this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being emitted from this node. The lowest binary value of the PRIO field is defined as the highest priority.

---

1. The transmit interrupt will occur only if not masked. A polling scheme can be applied on TXE also.

## MSCAN Controller (MSCAN08)

The internal scheduling process takes place whenever the MSCAN08 arbitrates for the bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message being set up in one of the three transmit buffers. As messages that are already under transmission cannot be aborted, the user has to request the abort by setting the corresponding abort request flag (ABTRQ) in the transmission control register (CTCR). The MSCAN08 will then grant the request, if possible, by setting the corresponding abort request acknowledge (ABTAK) and the TXE flag in order to release the buffer and by generating a transmit interrupt. The transmit interrupt handler software can tell from the setting of the ABTAK flag whether the message was actually aborted (ABTAK = 1) or sent (ABTAK = 0).

---

---

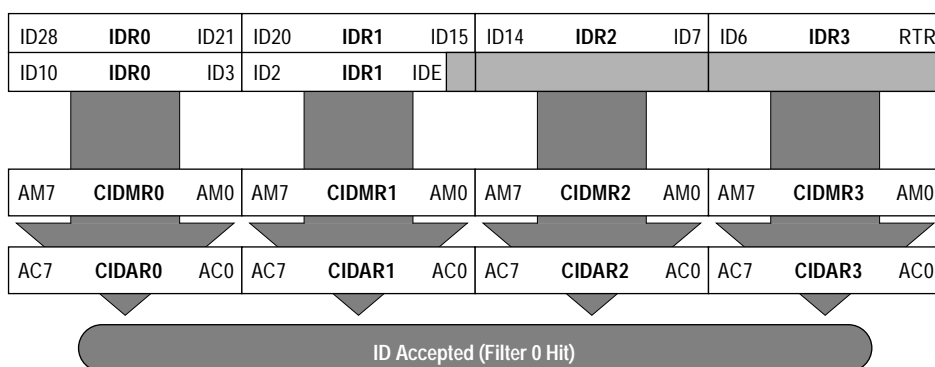
### Identifier Acceptance Filter

The Identifier Acceptance Registers (CIDAR0-3) define the acceptance patterns of the standard or extended identifier (ID10-ID0 or ID28-ID0). Any of these bits can be marked 'don't care' in the Identifier Mask Registers (CIDMR0-3).

A filter hit is indicated to the application on software by a set RXF (Receive Buffer Full Flag, see [MSCAN08 Receiver Flag Register \(CRFLG\)](#) on page 368) and two bits in the Identifier Acceptance Control Register (see [MSCAN08 Identifier Acceptance Control Register](#) on page 375). These Identifier Hit Flags (IDHIT1-0) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. In case that more than one hit occurs (two or more filters match) the lower hit has priority.

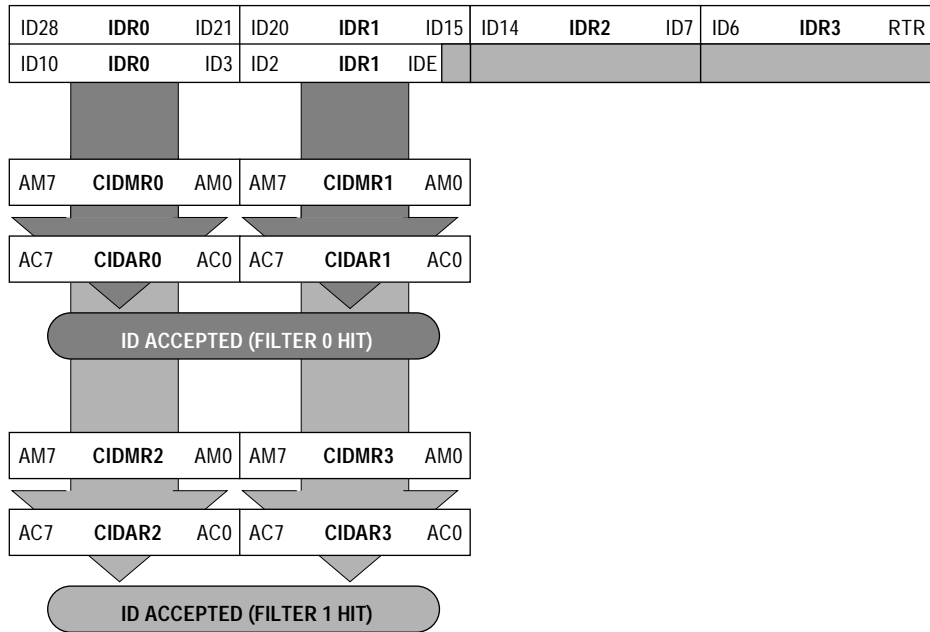
A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

- Single identifier acceptance filter, each to be applied to a) the full 29 bits of the extended identifier and to the following bits of the CAN frame: RTR, IDE, SRR or b) the 11 bits of the standard identifier plus the RTR and IDE bits of CAN 2.0A/B messages. This mode implements a single filter for a full length CAN 2.0B compliant extended identifier. **Figure 3** shows how the 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces a filter 0 hit.
- Two identifier acceptance filters, each to be applied to a) the 14 most significant bits of the extended identifier plus the SRR and the IDE bits of CAN2.0B messages, or b) the 11 bits of the identifier plus the RTR and IDE bits of CAN 2.0A/B messages. **Figure 4** shows how the 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces filter 0 and 1 hits.
- Four identifier acceptance filters, each to be applied to the first eight bits of the identifier. This mode implements four independent filters for the first eight bits of a CAN 2.0A/B compliant standard identifier. **Figure 5** shows how the 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces filter 0 to 3 hits.
- Closed filter. No CAN message will be copied into the foreground buffer RxFG, and the RXF flag will never be set.



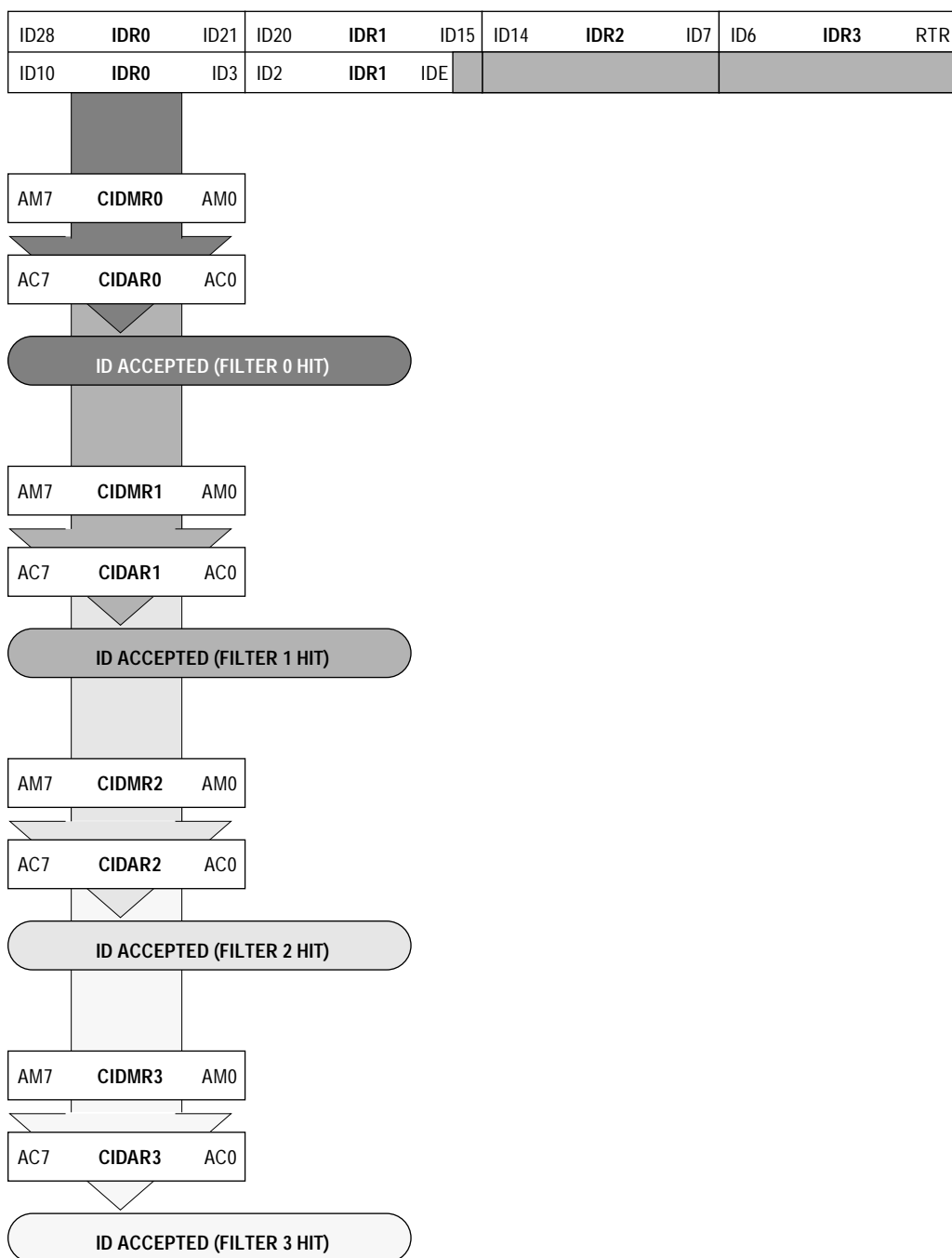
**Figure 3. Single 32-Bit Maskable Identifier Acceptance Filter**

# MSCAN Controller (MSCAN08)



**Figure 4. Dual 16-Bit Maskable Acceptance Filters**





**Figure 5. Quadruple 8-Bit Maskable Acceptance Filters**

---

---

### Interrupts

The MSCAN08 supports four interrupt vectors mapped onto eleven different interrupt sources, any of which can be individually masked (for details see [MSCAN08 Receiver Flag Register \(CRFLG\)](#) on page 368, to [MSCAN08 Transmitter Control Register](#) on page 374).

- *Transmit Interrupt:* At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXE flags of the empty message buffers are set.
- *Receive Interrupt:* A message has been received successfully and loaded into the foreground receive buffer. This interrupt will be emitted immediately after receiving the EOF symbol. The RXF flag is set.
- *Wakeup Interrupt:* An activity on the CAN bus occurred during MSCAN08 internal sleep mode or power-down mode (provided SLPACK=WUPIE=1).
- *Error Interrupt:* An overrun, error, or warning condition occurred. The receiver flag register (CRFLG) will indicate one of the following conditions:
  - *Overrun:* An overrun condition as described in [Receive Structures](#) on page 336, has occurred.
  - *Receiver Warning:* The receive error counter has reached the CPU Warning limit of 96.
  - *Transmitter Warning:* The transmit error counter has reached the CPU Warning limit of 96.
  - *Receiver Error Passive:* The receive error counter has exceeded the error passive limit of 127 and MSCAN08 has gone to error passive state.
  - *Transmitter Error Passive:* The transmit error counter has exceeded the error passive limit of 127 and MSCAN08 has gone to error passive state.

- *Bus Off*. The transmit error counter has exceeded 255 and MSCAN08 has gone to bus off state.

### Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the MSCAN08 receiver flag register (CRFLG) or the MSCAN08 transmitter flag register (CTFLG). Interrupts are pending as long as one of the corresponding flags is set. The flags in the above registers must be reset within the interrupt handler in order to handshake the interrupt. The flags are reset through writing a '1' to the corresponding bit position. A flag cannot be cleared if the respective condition still prevails.

**NOTE:** *Bit manipulation instructions (BSET) shall not be used to clear interrupt flags.*

### Interrupt Vectors

The MSCAN08 supports four interrupt vectors as shown in [Table 1](#). The vector addresses and the relative interrupt priority are dependent on the chip integration and to be defined.

**Table 1. MSCAN08 Interrupt Vector Addresses**

| Function         | Source | Local Mask | Global Mask |
|------------------|--------|------------|-------------|
| Wakeup           | WUPIF  | WUPIE      | I Bit       |
| Error Interrupts | RWRNIF | RWRNIE     |             |
|                  | TWRNIF | TWRNIE     |             |
|                  | RERRIF | RERRIE     |             |
|                  | TERRIF | TERRIE     |             |
|                  | BOFFIF | BOFFIE     |             |
|                  | OVRIF  | OVRIE      |             |
| Receive          | RXF    | RXFIE      |             |
| Transmit         | TXE0   | TXEIE0     |             |
|                  | TXE1   | TXEIE1     |             |
|                  | TXE2   | TXEIE2     |             |

---

---

### Protocol Violation Protection

The MSCAN08 will protect the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN08 can not be modified while the MSCAN08 is on-line. The SFTRES bit in the MSCAN08 module control register (see [MSCAN08 Module Control Register 0](#) on page 363) serves as a lock to protect the following registers:
  - MSCAN08 module control register 1 (CMCR1)
  - MSCAN08 bus timing register 0 and 1 (CBTR0 and CBTR1)
  - MSCAN08 identifier acceptance control register (CIDAC)
  - MSCAN08 identifier acceptance registers (CIDAR0–3)
  - MSCAN08 identifier mask registers (CIDMR0–3)
- The TxCAN pin is forced to recessive when the MSCAN08 is in any of the Low Power Modes.

---

---

### Low Power Modes

In addition to normal mode, the MSCAN08 has three modes with reduced power consumption: Sleep, Soft Reset and Power Down modes. In Sleep and Soft Reset mode, power consumption is reduced by stopping all clocks except those to access the registers. In Power Down mode, all clocks are stopped and no power is consumed.

The WAIT and STOP instructions put the MCU in low power consumption stand-by modes. [Table 2](#) summarizes the combinations of MSCAN08 and CPU modes. A particular combination of modes is entered for the given settings of the bits SLPK and SFTRES. For all modes, an MSCAN wake-up interrupt can occur only if SLPK=WUPIE=1.

**Table 2. MSCAN08 vs CPU operating modes**

| MSCAN Mode | CPU Mode                               |                         |
|------------|--|-------------------------|
|            | STOP                                   | WAIT or RUN             |
| Power Down | SLPAK = X <sup>(1)</sup><br>SFTRES = X |                         |
| Sleep      |  | SLPAK = 1<br>SFTRES = 0 |
| Soft Reset |  | SLPAK = 0<br>SFTRES = 1 |
| Normal     |  | SLPAK = 0<br>SFTRES = 0 |

1. 'X' means don't care.

### MSCAN08 Sleep Mode

The CPU can request the MSCAN08 to enter the low-power mode by asserting the SLPRQ bit in the module configuration register (see [Figure 6](#)). The time when the MSCAN08 enters Sleep mode depends on its activity:

- if it is transmitting, it continues to transmit until there is no more message to be transmitted, and then goes into Sleep mode
- if it is receiving, it waits for the end of this message and then goes into Sleep mode
- if it is neither transmitting or receiving, it will immediately go into Sleep mode

**NOTE:** *The application software must avoid setting up a transmission (by clearing or more TXE flags) and immediately request Sleep mode (by setting SLPRQ). It then depends on the exact sequence of operations whether MSCAN08 starts transmitting or goes into Sleep mode directly.*

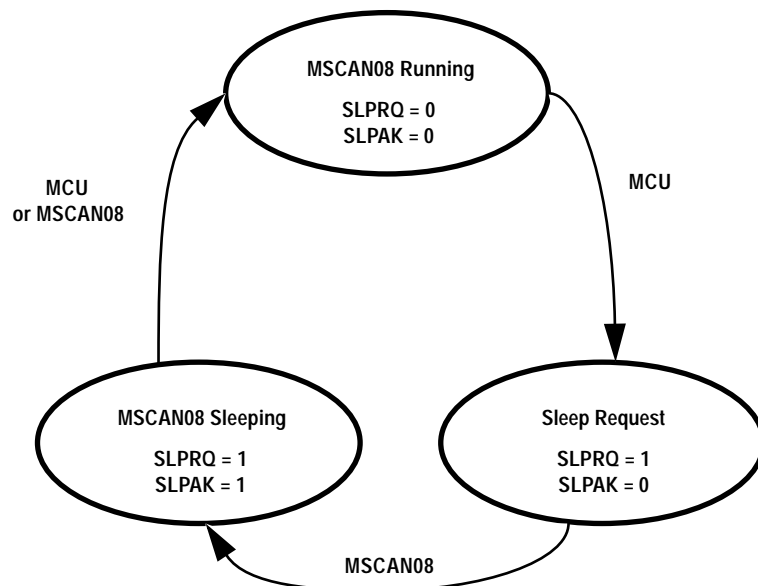
During Sleep mode, the SLPAK flag is set. The application software should use SLPAK as a handshake indication for the request (SLPRQ) to go into Sleep mode. When in Sleep mode, the MSCAN08 stops its internal clocks. However, clocks to allow register accesses still run. If the MSCAN08 is in buss-off state, it stops counting the 128\*11 consecutive recessive bits due to the stopped clocks. The TxCAN pin stays in

## MSCAN Controller (MSCAN08)

recessive state. If RXF=1, the message can be read and RXF can be cleared. Copying of RxGB into RxFG doesn't take place while in Sleep mode. It is possible to access the transmit buffers and to clear the TXE flags. No message abort takes place while in Sleep mode.

The MSCAN08 leaves Sleep mode (wake-up) when:

- bus activity occurs or
- the MCU clears the SLPRQ bit or
- the MCU sets the SFTRES bit



**Figure 6. Sleep Request/Acknowledge Cycle**

**NOTE:** The MCU cannot clear the SLPRQ bit before the MSCAN08 is in Sleep mode (SLPAK=1).

After wake-up, the MSCAN08 waits for 11 consecutive recessive bits to synchronize to the bus. As a consequence, if the MSCAN08 is woken-up by a CAN frame, this frame is not received. The receive message buffers (RxFG and RxBG) contain messages if they were received before Sleep mode was entered. All pending actions are executed upon wake-up: copying of RxBG into RxFG, message aborts and message

transmissions. If the MSCAN08 is still in bus-off state after Sleep mode was left, it continues counting the 128\*11 consecutive recessive bits.

### MSCAN08 Soft Reset Mode

In Soft Reset mode, the MSCAN08 is stopped. Registers can still be accessed. This mode is used to initialize the module configuration, bit timing and the CAN message filter. See [MSCAN08 Module Control Register 0](#) on page 363 for a complete description of the Soft Reset mode.

When setting the SFTRES bit, the MSCAN08 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations.

**NOTE:** *The user is responsible to take care that the MSCAN08 is not active when Soft Reset mode is entered. The recommended procedure is to bring the MSCAN08 into Sleep mode before the SFTRES bit is set.*

### MSCAN08 Power Down Mode

The MSCAN08 is in Power Down mode when the CPU is in Stop mode.

When entering the Power Down mode, the MSCAN08 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations.

**NOTE:** *The user is responsible to take care that the MSCAN08 is not active when Power Down mode is entered. The recommended procedure is to bring the MSCAN08 into Sleep mode before the STOP instruction is executed.*

To protect the CAN bus system from fatal consequences of violations to the above rule, the MSCAN08 drives the TxCAN pin into recessive state.

In Power Down mode, no registers can be accessed.

CAN bus activity can wakt the MCU from CPU STOP/CAN power-down mode. However, until the oscillator starts up and synchronisation is achieved, the CAN will not respond to incoming data.

## MSCAN Controller (MSCAN08)

|                                     |  |
|-------------------------------------|--|
| <b>CPU Wait Mode</b>                | The MSCAN08 module remains active during CPU wait mode. The MSCAN08 will stay synchronized to the CAN bus and generates transmit, receive, and error interrupts to the CPU, if enabled. Any such interrupt will bring the MCU out of wait mode.  |
| <b>Programmable Wakeup Function</b> | The MSCAN08 can be programmed to apply a low-pass filter function to the RxCAN input line while in internal sleep mode (see information on control bit WUPM in <a href="#">MSCAN08 Module Control Register 1</a> on page 364). This feature can be used to protect the MSCAN08 from wake-up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic inference within noisy environments. |

---

---

### Timer Link

The MSCAN08 will generate a timer signal whenever a valid frame has been received. Because the CAN specification defines a frame to be valid if no errors occurred before the EOF field has been transmitted successfully, the timer signal will be generated right after the EOF. A pulse of one bit time is generated. As the MSCAN08 receiver engine also receives the frames being sent by itself, a timer signal also will be generated after a successful transmission.

The previously described timer signal can be routed into the on-chip timer interface module (TIM). This signal is connected to the timer n channel m input<sup>1</sup> under the control of the timer link enable (TLNKEN) bit in the CMCR0.

After timer n has been programmed to capture rising edge events, it can be used under software control to generate 16-bit time stamps which can be stored with the received message.

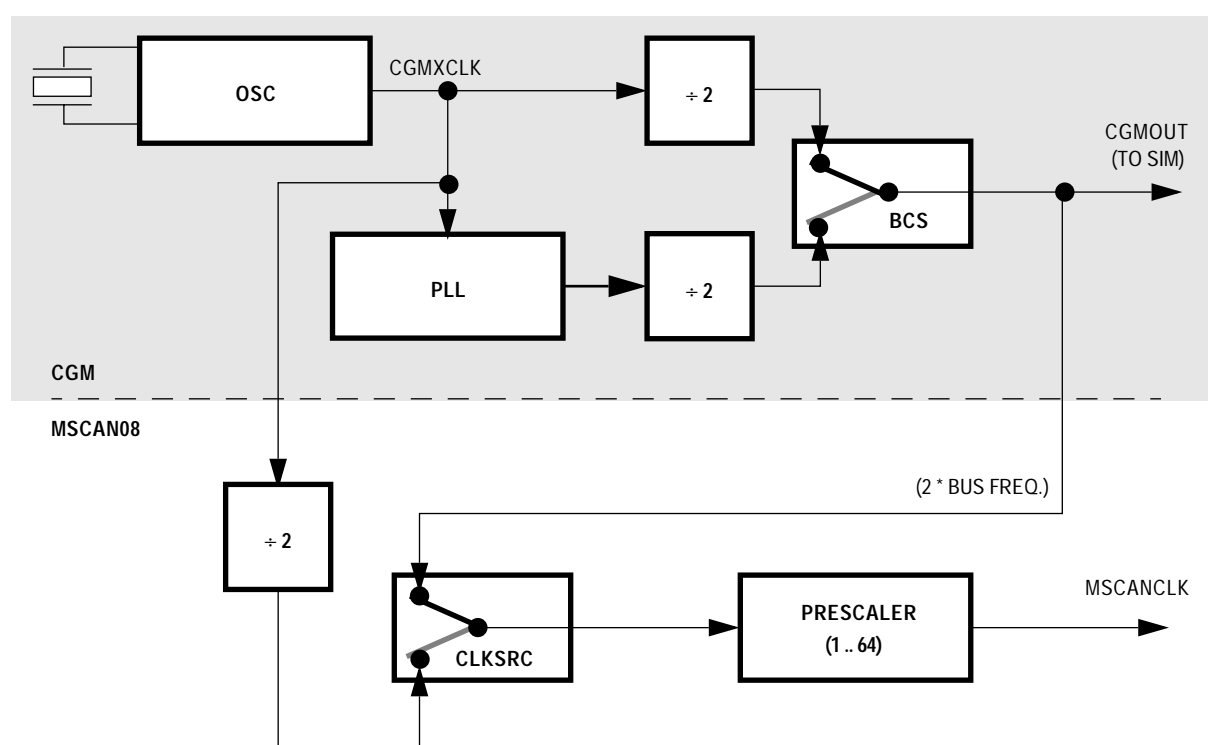
---

1. The timer channel being used for the timer link is integration dependent.



## Clock System

**Figure 7** shows the structure of the MSCAN08 clock generation circuitry and its interaction with the clock generation module (CGM). With this flexible clocking scheme the MSCAN08 is able to handle CAN bus rates ranging from 10 kbps up to 1 Mbps.



**Figure 7. Clocking Scheme**

The clock source bit (CLKSRC) in the MSCAN08 module control register (CMCR1) (see [MSCAN08 Module Control Register 0](#) on page 363) defines whether the MSCAN08 is connected to the output of the crystal oscillator or to the PLL output.

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met.

## MSCAN Controller (MSCAN08)

**NOTE:** *If the system clock is generated from a PLL, it is recommended to select the crystal clock source rather than the system clock source due to jitter considerations, especially at faster CAN bus rates.*

A programmable prescaler is used to generate out of the MSCAN08 clock the time quanta (T<sub>q</sub>) clock. A time quantum is the atomic unit of time handled by the MSCAN08.

$$f_{Tq} = \frac{f_{MSCANCLK}}{\text{Presc value}}$$

A bit time is subdivided into three segments<sup>1</sup>(see [Figure 8](#)).

- SYNC\_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time segment 1: This segment includes the PROP\_SEG and the PHASE\_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time segment 2: This segment represents PHASE\_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

$$\text{Bit rate} = \frac{f_{Tq}}{\text{No. of time quanta}}$$

The synchronization jump width (SJW) can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

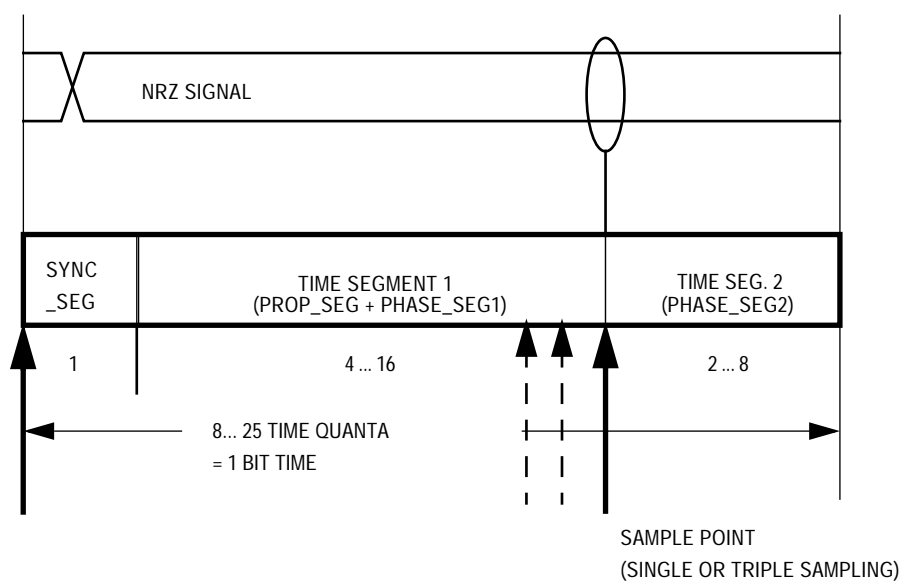
The above parameters can be set by programming the bus timing registers, CBTR0–CBTR1, see [MSCAN08 Bus Timing Register 0](#) on page 366 and [MSCAN08 Bus Timing Register 1](#) on page 367).

---

1. For further explanation of the underlying concepts please refer to ISO/DIS 11 519-1, Section 10.3.

**NOTE:** *It is the user's responsibility to make sure that the bit timing settings are in compliance with the CAN standard,*

**Table 8** gives an overview on the CAN conforming segment settings and the related parameter values.



**Figure 8. Segments within the Bit Time**

**Table 3. Time segment syntax**

|                |   |
|----------------|---|
| SYNC_SEG       | System expects transitions to occur on the bus during this period.  |
| Transmit point | A node in transmit mode will transfer a new value to the CAN bus at this point.   |
| Sample point   | A node in receive mode will sample the bus at this point. If the three samples per bit option is selected then this point marks the position of the third sample. |

**Table 4. CAN Standard Compliant Bit Time Segment Settings**

| Time Segment 1 | TSEG1   | Time Segment 2 | TSEG2 | Synchron. Jump Width | SJW    |
|----------------|---------|----------------|-------|----------------------|--------|
| 5 .. 10        | 4 .. 9  | 2              | 1     | 1 .. 2               | 0 .. 1 |
| 4 .. 11        | 3 .. 10 | 3              | 2     | 1 .. 3               | 0 .. 2 |
| 5 .. 12        | 4 .. 11 | 4              | 3     | 1 .. 4               | 0 .. 3 |
| 6 .. 13        | 5 .. 12 | 5              | 4     | 1 .. 4               | 0 .. 3 |
| 7 .. 14        | 6 .. 13 | 6              | 5     | 1 .. 4               | 0 .. 3 |
| 8 .. 15        | 7 .. 14 | 7              | 6     | 1 .. 4               | 0 .. 3 |
| 9 .. 16        | 8 .. 15 | 8              | 7     | 1 .. 4               | 0 .. 3 |

## Memory Map

The MSCAN08 occupies 128 bytes in the CPU08 memory space. The absolute mapping is implementation dependent with the base address being a multiple of 128.

|        |                   |
|--------|-------------------|
| \$xx00 | CONTROL REGISTERS |
| \$xx08 | 9 BYTES           |
| \$xx09 | RESERVED          |
| \$xx0D | 5 BYTES           |
| \$xx0E | ERROR COUNTERS    |
| \$xx0F | 2 BYTES           |
| \$xx10 | IDENTIFIER FILTER |
| \$xx17 | 8 BYTES           |
| \$xx18 | RESERVED          |
| \$xx3F | 40 BYTES          |
| \$xx40 | RECEIVE BUFFER    |
| \$xx4F |                   |
| \$xx50 | TRANSMIT BUFFER 0 |
| \$xx5F |                   |
| \$xx60 | TRANSMIT BUFFER 1 |
| \$xx6F |                   |
| \$xx70 | TRANSMIT BUFFER 2 |
| \$xx7F |                   |

**Figure 9. MSCAN08 Memory Map**

## Programmer's Model of Message Storage

This section details the organization of the receive and transmit message buffers and the associated control registers. For reasons of programmer interface simplification, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13-byte data structure. An additional transmit buffer priority register (TBPR) is defined for the transmit buffers.

## MSCAN Controller (MSCAN08)

| Addr   | Register Name                                    |
|--------|--|
| \$05b0 | IDENTIFIER REGISTER 0                            |
| \$05b1 | IDENTIFIER REGISTER 1                            |
| \$05b2 | IDENTIFIER REGISTER 2                            |
| \$05b3 | IDENTIFIER REGISTER 3                            |
| \$05b4 | DATA SEGMENT REGISTER 0                          |
| \$05b5 | DATA SEGMENT REGISTER 1                          |
| \$05b6 | DATA SEGMENT REGISTER 2                          |
| \$05b7 | DATA SEGMENT REGISTER 3                          |
| \$05b8 | DATA SEGMENT REGISTER 4                          |
| \$05b9 | DATA SEGMENT REGISTER 5                          |
| \$05bA | DATA SEGMENT REGISTER 6                          |
| \$05bB | DATA SEGMENT REGISTER 7                          |
| \$05bC | DATA LENGTH REGISTER                             |
| \$05bD | TRANSMIT BUFFER PRIORITY REGISTER <sup>(1)</sup> |
| \$05bE | UNUSED   |
| \$05bF | UNUSED   |

1. Not applicable for receive buffers

**Figure 10. Message Buffer Organization**

### Message Buffer Outline

**Figure 11** shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in **Figure 12**. All bits of the 13-byte data structure are undefined out of reset.

**NOTE:** *The foreground receive buffer can be read anytime but cannot be written. The transmit buffers can be read or written anytime.*


### Identifier Registers

The identifiers consist of either 11 bits (ID10–ID0) for the standard, or 29 bits (ID28–ID0) for the extended format. ID10/28 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

### SRR — Substitute Remote Request

This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and will be stored as received on the CAN bus for receive buffers.


| Addr   | Register |                 | Bit 7 | 6    | 5    | 4        | 3        | 2    | 1    | Bit 0 |
|--------|----------|-----------------|-------|------|------|----------|----------|------|------|-------|
| \$05b0 | IDR0     | Read:<br>Write: | ID28  | ID27 | ID26 | ID25     | ID24     | ID23 | ID22 | ID21  |
| \$05b1 | IDR1     | Read:<br>Write: | ID20  | ID19 | ID18 | SRR (=1) | IDE (=1) | ID17 | ID16 | ID15  |
| \$05b2 | IDR2     | Read:<br>Write: | ID14  | ID13 | ID12 | ID11     | ID10     | ID9  | ID8  | ID7   |
| \$05b3 | IDR3     | Read:<br>Write: | ID6   | ID5  | ID4  | ID3      | ID2      | ID1  | ID0  | RTR   |
| \$05b4 | DSR0     | Read:<br>Write: | DB7   | DB6  | DB5  | DB4      | DB3      | DB2  | DB1  | DB0   |
| \$05b5 | DSR1     | Read:<br>Write: | DB7   | DB6  | DB5  | DB4      | DB3      | DB2  | DB1  | DB0   |
| \$05b6 | DSR2     | Read:<br>Write: | DB7   | DB6  | DB5  | DB4      | DB3      | DB2  | DB1  | DB0   |
| \$05b7 | DSR3     | Read:<br>Write: | DB7   | DB6  | DB5  | DB4      | DB3      | DB2  | DB1  | DB0   |
| \$05b8 | DSR4     | Read:<br>Write: | DB7   | DB6  | DB5  | DB4      | DB3      | DB2  | DB1  | DB0   |
| \$05b9 | DSR5     | Read:<br>Write: | DB7   | DB6  | DB5  | DB4      | DB3      | DB2  | DB1  | DB0   |
| \$05bA | DSR6     | Read:<br>Write: | DB7   | DB6  | DB5  | DB4      | DB3      | DB2  | DB1  | DB0   |
| \$05bB | DSR7     | Read:<br>Write: | DB7   | DB6  | DB5  | DB4      | DB3      | DB2  | DB1  | DB0   |
| \$05bC | DLR      | Read:           |       |      |      |          | DLC3     | DLC2 | DLC1 | DLC0  |
|        |          | Write:          |       |      |      |          |          |      |      |       |

 = Unimplemented

**Figure 11. Receive/Transmit Message Buffer Extended Identifier (IDRn)**

## MSCAN Controller (MSCAN08)

| Addr   | Register |                 | Bit 7 | 6   | 5   | 4   | 3       | 2   | 1   | Bit 0 |
|--------|----------|-----------------|-------|-----|-----|-----|---------|-----|-----|-------|
| \$05b0 | IDR0     | Read:<br>Write: | ID10  | ID9 | ID8 | ID7 | ID6     | ID5 | ID4 | ID3   |
| \$05b1 | IDR1     | Read:<br>Write: | ID2   | ID1 | ID0 | RTR | IDE(=0) |     |     |       |
| \$05b2 | IDR2     | Read:<br>Write: |       |     |     |     |         |     |     |       |
| \$05b3 | IDR3     | Read:<br>Write: |       |     |     |     |         |     |     |       |

 = Unimplemented

**Figure 12. Standard Identifier Mapping**

### IDE — ID Extended

This flag indicates whether the extended or standard identifier format is applied in this buffer. In case of a receive buffer, the flag is set as being received and indicates to the CPU how to process the buffer identifier registers. In case of a transmit buffer, the flag indicates to the MSCAN08 what type of identifier to send.

1 = Extended format, 29 bits

0 = Standard format, 11 bits

### RTR — Remote Transmission Request

This flag reflects the status of the remote transmission request bit in the CAN frame. In case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.

1 = Remote frame

0 = Data frame



## Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

### DLC3–DLC0 — Data Length Code Bits

The data length code contains the number of bytes (data byte count) of the respective message. At transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. [Table 5](#) shows the effect of setting the DLC bits.

**Table 5. Data Length Codes**

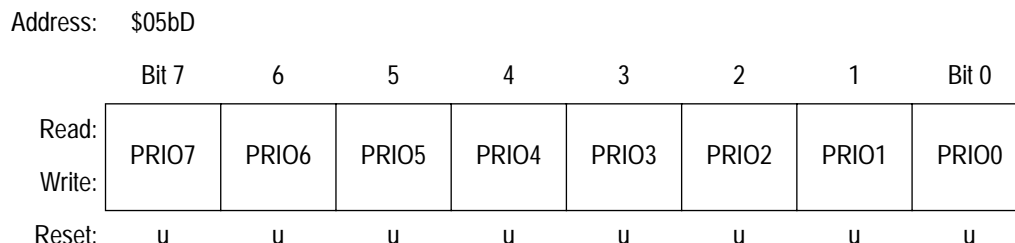
| Data Length Code |      |      |      | Data Byte Count |
|------------------|------|------|------|-----------------|
| DLC3             | DLC2 | DLC1 | DLC0 |                 |
| 0                | 0    | 0    | 0    | 0               |
| 0                | 0    | 0    | 1    | 1               |
| 0                | 0    | 1    | 0    | 2               |
| 0                | 0    | 1    | 1    | 3               |
| 0                | 1    | 0    | 0    | 4               |
| 0                | 1    | 0    | 1    | 5               |
| 0                | 1    | 1    | 0    | 6               |
| 0                | 1    | 1    | 1    | 7               |
| 1                | 0    | 0    | 0    | 8               |

## Data Segment Registers (DSRn)

The eight data segment registers contain the data to be transmitted or received. The number of bytes to be transmitted or being received is determined by the data length code in the corresponding DLR.

## MSCAN Controller (MSCAN08)

### Transmit Buffer Priority Registers



**Figure 13. Transmit Buffer Priority Register (TBPR)**

#### PRI07–PRI00 — Local Priority

This field defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the MSCAN08 and is defined to be highest for the smallest binary number. The MSCAN08 implements the following internal prioritization mechanism:

- All transmission buffers with a cleared TXE flag participate in the prioritization right before the SOF is sent.
- The transmission buffer with the lowest local priority field wins the prioritization.
- In case more than one buffer has the same lowest priority, the message buffer with the lower index number wins.

---


---

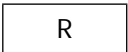
## Programmer's Model of Control Registers

The programmer's model has been laid out for maximum simplicity and efficiency. [Figure 14](#) gives an overview on the control register block of the MSCAN08.

MSCAN Controller (MSCAN08)  
Programmer's Model of Control Registers

| Addr   | Register |        | Bit 7 | 6      | 5      | 4      | 3      | 2      | 1      | Bit 0  |
|--------|----------|--------|-------|--------|--------|--------|--------|--------|--------|--------|
| \$0500 | CMCR0    | Read:  | 0     | 0      | 0      | SYNCH  | TLNKEN | SLPAK  | SLPRQ  | SFTRES |
|        |          | Write: |       |        |        |        |        |        |        |        |
| \$0501 | CMCR1    | Read:  | 0     | 0      | 0      | 0      | 0      | LOOPB  | WUPM   | CLKSRC |
|        |          | Write: |       |        |        |        |        |        |        |        |
| \$0502 | CBTR0    | Read:  | SJW1  | SJW0   | BRP5   | BRP4   | BRP3   | BRP2   | BRP1   | BRP0   |
|        |          | Write: |       |        |        |        |        |        |        |        |
| \$0503 | CBTR1    | Read:  | SAMP  | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
|        |          | Write: |       |        |        |        |        |        |        |        |
| \$0504 | CRFLG    | Read:  | WUPIF | RWRNIF | TWRNIF | RERRIF | TERRIF | BOFFIF | OVRIF  | RXF    |
|        |          | Write: |       |        |        |        |        |        |        |        |
| \$0505 | CRIER    | Read:  | WUPIE | RWRNIE | TWRNIE | RERRIE | TERRIE | BOFFIE | OVRIE  | RXFIE  |
|        |          | Write: |       |        |        |        |        |        |        |        |
| \$0506 | CTFLG    | Read:  | 0     | ABTAK2 | ABTAK1 | ABTAK0 | 0      | TXE2   | TXE1   | TXE0   |
|        |          | Write: |       |        |        |        |        |        |        |        |
| \$0507 | CTCR     | Read:  | 0     | ABTRQ2 | ABTRQ1 | ABTRQ0 | 0      | TXEIE2 | TXEIE1 | TXEIE0 |
|        |          | Write: |       |        |        |        |        |        |        |        |
| \$0508 | CIDAC    | Read:  | 0     | 0      | IDAM1  | IDAM0  | 0      | 0      | IDHIT1 | IDHIT0 |
|        |          | Write: |       |        |        |        |        |        |        |        |
| \$0509 | Reserved | Read:  | R     | R      | R      | R      | R      | R      | R      | R      |
|        |          | Write: |       |        |        |        |        |        |        |        |

 = Unimplemented

 = Reserved

**Figure 14. MSCAN08 Control Register Structure**

## MSCAN Controller (MSCAN08)

| Addr   | Register |        | Bit 7  | 6      | 5      | 4      | 3      | 2      | 1      | Bit 0  |
|--------|----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| \$050E | CRXERR   | Read:  | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
|        |          | Write: |        |        |        |        |        |        |        |        |
| \$050F | CTXERR   | Read:  | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
|        |          | Write: |        |        |        |        |        |        |        |        |
| \$0510 | CIDAR0   | Read:  | AC7    | AC6    | AC5    | AC4    | AC3    | AC2    | AC1    | AC0    |
|        |          | Write: |        |        |        |        |        |        |        |        |
| \$0511 | CIDAR1   | Read:  | AC7    | AC6    | AC5    | AC4    | AC3    | AC2    | AC1    | AC0    |
|        |          | Write: |        |        |        |        |        |        |        |        |
| \$0512 | CIDAR2   | Read:  | AC7    | AC6    | AC5    | AC4    | AC3    | AC2    | AC1    | AC0    |
|        |          | Write: |        |        |        |        |        |        |        |        |
| \$0513 | CIDAR3   | Read:  | AC7    | AC6    | AC5    | AC4    | AC3    | AC2    | AC1    | AC0    |
|        |          | Write: |        |        |        |        |        |        |        |        |
| \$0514 | CIDMR0   | Read:  | AM7    | AM6    | AM5    | AM4    | AM3    | AM2    | AM1    | AM0    |
|        |          | Write: |        |        |        |        |        |        |        |        |
| \$0515 | CIDMR1   | Read:  | AM7    | AM6    | AM5    | AM4    | AM3    | AM2    | AM1    | AM0    |
|        |          | Write: |        |        |        |        |        |        |        |        |
| \$0516 | CIDMR2   | Read:  | AM7    | AM6    | AM5    | AM4    | AM3    | AM2    | AM1    | AM0    |
|        |          | Write: |        |        |        |        |        |        |        |        |
| \$0517 | CIDMR3   | Read:  | AM7    | AM6    | AM5    | AM4    | AM3    | AM2    | AM1    | AM0    |
|        |          | Write: |        |        |        |        |        |        |        |        |

**Figure 14. MSCAN08 Control Register Structure (Continued)**

## MSCAN08 Module Control Register 0

Address: \$0500

|        | Bit 7 | 6 | 5 | 4     | 3      | 2     | 1     | Bit 0  |
|--------|-------|---|---|-------|--------|-------|-------|--------|
| Read:  | 0     | 0 | 0 | SYNCH | TLNKEN | SLPAK | SLPRQ | SFTRES |
| Write: |       |   |   |       |        |       |       |        |
| Reset: | 0     | 0 | 0 | 0     | 0      | 0     | 0     | 1      |

= Unimplemented

**Figure 15. Module Control Register 0 (CMCR0)**

### SYNCH — Synchronized Status

This bit indicates whether the MSCAN08 is synchronized to the CAN bus and as such can participate in the communication process.

1 = MSCAN08 synchronized to the CAN bus

0 = MSCAN08 not synchronized to the CAN bus

### TLNKEN — Timer Enable

This flag is used to establish a link between the MSCAN08 and the on-chip timer (see [Timer Link](#) on page 350).

1 = The MSCAN08 timer signal output is connected to the timer input.

0 = The port is connected to the timer input.

### SLPAK — Sleep Mode Acknowledge

This flag indicates whether the MSCAN08 is in module internal sleep mode. It shall be used as a handshake for the sleep mode request (see [MSCAN08 Sleep Mode](#) on page 347). If the MSCAN08 detects bus activity while in Sleep mode, it clears the flag.

1 = Sleep – MSCAN08 in internal sleep mode

0 = Wakeup – MSCAN08 is not in Sleep mode

## MSCAN Controller (MSCAN08)

### SLPRQ — Sleep Request, Go to Internal Sleep Mode

This flag requests the MSCAN08 to go into an internal power-saving mode (see [MSCAN08 Sleep Mode](#) on page 347).

- 1 = Sleep — The MSCAN08 will go into internal sleep mode.
- 0 = Wakeup — The MSCAN08 will function normally.

### SFTRES — Soft Reset

When this bit is set by the CPU, the MSCAN08 immediately enters the soft reset state. Any ongoing transmission or reception is aborted and synchronization to the bus is lost.

The following registers enter and stay in their hard reset state: CMCR0, CRFLG, CRIER, CTFLG, and CTCR.

The registers CMCR1, CBTR0, CBTR1, CIDAC, CIDAR0–3, and CIDMR0–3 can only be written by the CPU when the MSCAN08 is in soft reset state. The values of the error counters are not affected by soft reset.

When this bit is cleared by the CPU, the MSCAN08 tries to synchronize to the CAN bus. If the MSCAN08 is not in bus-off state, it will be synchronized after 11 recessive bits on the bus; if the MSCAN08 is in bus-off state, it continues to wait for 128 occurrences of 11 recessive bits.


Clearing SFTRES and writing to other bits in CMCR0 must be in separate instructions.

- 1 = MSCAN08 in soft reset state
- 0 = Normal operation

## MSCAN08 Module Control Register 1

Address: \$0501

|        | Bit 7 | 6 | 5 | 4 | 3 | 2     | 1    | Bit 0  |
|--------|-------|---|---|---|---|-------|------|--------|
| Read:  | 0     | 0 | 0 | 0 | 0 | LOOPB | WUPM | CLKSRC |
| Write: |       |   |   |   |   |       |      |        |
| Reset: | 0     | 0 | 0 | 0 | 0 | 0     | 0    | 0      |

 = Unimplemented

**Figure 16. Module Control Register (CMCR1)**

#### LOOPB — Loop Back Self-Test Mode

When this bit is set, the MSCAN08 performs an internal loop back which can be used for self-test operation: the bit stream output of the transmitter is fed back to the receiver internally. The RxCAN input pin is ignored and the TxCAN output goes to the recessive state (logic '1'). The MSCAN08 behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state the MSCAN08 ignores the bit sent during the ACK slot of the CAN frame Acknowledge field to insure proper reception of its own message. Both transmit and receive interrupt are generated.

- 1 = Activate loop back self-test mode
- 0 = Normal operation

#### WUPM — Wakeup Mode

This flag defines whether the integrated low-pass filter is applied to protect the MSCAN08 from spurious wakeups (see [Programmable Wakeup Function](#) on page 350).

- 1 = MSCAN08 will wake up the CPU only in cases of a dominant pulse on the bus which has a length of at least  $t_{wup}$ .
- 0 = MSCAN08 will wake up the CPU after any recessive to dominant edge on the CAN bus.

#### CLKSRC — Clock Source

This flag defines which clock source the MSCAN08 module is driven from (see [Clock System](#) on page 351).

- 1 = The MSCAN08 clock source is CGMOUT (see [Figure 7](#)).
- 0 = The MSCAN08 clock source is CGMXCLK/2 (see [Figure 7](#)).

**NOTE:** *The CMCR1 register can be written only if the SFTRES bit in the MSCAN08 module control register is set*

## MSCAN Controller (MSCAN08)

### MSCAN08 Bus Timing Register 0

Address: \$0502

|        | Bit 7 | 6    | 5    | 4    | 3    | 2    | 1    | Bit 0 |
|--------|-------|------|------|------|------|------|------|-------|
| Read:  | SJW1  | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0  |
| Write: |       |      |      |      |      |      |      |       |
| Reset: | 0     | 0    | 0    | 0    | 0    | 0    | 0    | 0     |

**Figure 17. Bus Timing Register 0 (CBTR0)**

#### SJW1 and SJW0 — Synchronization Jump Width

The synchronization jump width (SJW) defines the maximum number of time quanta ( $T_q$ ) clock cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see [Table 6](#)).

**Table 6. Synchronization Jump Width**

| SJW1 | SJW0 | Synchronization Jump Width |
|------|------|----------------------------|
| 0    | 0    | 1 $T_q$ cycle              |
| 0    | 1    | 2 $T_q$ cycle              |
| 1    | 0    | 3 $T_q$ cycle              |
| 1    | 1    | 4 $T_q$ cycle              |

#### BRP5–BRP0 — Baud Rate Prescaler

These bits determine the time quanta ( $T_q$ ) clock, which is used to build up the individual bit timing, according to [Table 7](#).

**Table 7. Baud Rate Prescaler**

| BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 | Prescaler Value (P) |
|------|------|------|------|------|------|---------------------|
| 0    | 0    | 0    | 0    | 0    | 0    | 1                   |
| 0    | 0    | 0    | 0    | 0    | 1    | 2                   |
| 0    | 0    | 0    | 0    | 1    | 0    | 3                   |
| 0    | 0    | 0    | 0    | 1    | 1    | 4                   |
| :    | :    | :    | :    | :    | :    | :                   |
| :    | :    | :    | :    | :    | :    | :                   |
| 1    | 1    | 1    | 1    | 1    | 1    | 64                  |

**NOTE:** The CBTR0 register can be written only if the SFTRES bit in the MSCAN08 module control register is set.



**MSCAN08 Bus  
 Timing Register 1**

Address: \$0503

|        |       |        |        |        |        |        |        |        |
|--------|-------|--------|--------|--------|--------|--------|--------|--------|
|        | Bit 7 | 6      | 5      | 4      | 3      | 2      | 1      | Bit 0  |
| Read:  | SAMP  | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| Write: |       |        |        |        |        |        |        |        |
| Reset: | 0     | 0      | 0      | 0      | 0      | 0      | 0      | 0      |

**Figure 18. Bus Timing Register 1 (CBTR1)**

**SAMP — Sampling**

This bit determines the number of serial bus samples to be taken per bit time. If set, three samples per bit are taken, the regular one (sample point) and two preceding samples, using a majority rule. For higher bit rates, SAMP should be cleared, which means that only one sample will be taken per bit.

- 1 = Three samples per bit<sup>1</sup>
- 0 = One sample per bit

**TSEG22–TSEG10 — Time Segment**

Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point.

Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in [Table 9](#).

**Table 8. Time Segment Values**

| TSEG13 | TSEG12 | TSEG11 | TSEG10 | Time Segment 1                         | TSEG22 | TSEG21 | TSEG20 | Time Segment 2                        |
|--------|--------|--------|--------|--|--------|--------|--------|---------------------------------------|
| 0      | 0      | 0      | 0      | 1 T <sub>q</sub> Cycle <sup>(1)</sup>  | 0      | 0      | 0      | 1 T <sub>q</sub> Cycle <sup>(1)</sup> |
| 0      | 0      | 0      | 1      | 2 T <sub>q</sub> Cycles <sup>(1)</sup> | 0      | 0      | 1      | 2 T <sub>q</sub> Cycles               |
| 0      | 0      | 1      | 0      | 3T <sub>q</sub> Cycles <sup>(1)</sup>  | .      | .      | .      | .                                     |
| 0      | 0      | 1      | 1      | 4 T <sub>q</sub> Cycles                | .      | .      | .      | .                                     |
| .      | .      | .      | .      | .                                      | 1      | 1      | 1      | 8T <sub>q</sub> Cycles                |
| .      | .      | .      | .      | .                                      |        |        |        |                                       |
| 1      | 1      | 1      | 1      | 16 T <sub>q</sub> Cycles               |        |        |        |                                       |

1. This setting is not valid. Please refer to [Table 4](#) for valid settings.

1. In this case PHASE\_SEG1 must be at least 2 time quanta.

## MSCAN Controller (MSCAN08)

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta ( $T_q$ ) clock cycles per bit as shown in [Table 9](#)).

$$\text{Bit time} = \frac{\text{Pres value}}{f_{\text{MSCANCLK}}} \cdot \text{number of Time Quanta}$$

**NOTE:** The *CBTR1* register can only be written if the *SFTRES* bit in the *MSCAN08* module control register is set.

### MSCAN08 Receiver Flag Register (CRFLG)

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. A flag can be cleared only when the condition which caused the setting is valid no more. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the *CRIER* register. A hard or soft reset will clear the register.

Address: \$0504

|        | Bit 7 | 6      | 5      | 4      | 3      | 2      | 1     | Bit 0 |
|--------|-------|--------|--------|--------|--------|--------|-------|-------|
| Read:  |       |        |        |        |        |        |       |       |
| Write: | WUPIF | RWRNIF | TWRNIF | RERRIF | TERRIF | BOFFIF | OVRIF | RXF   |
| Reset: | 0     | 0      | 0      | 0      | 0      | 0      | 0     | 0     |

**Figure 19. Receiver Flag Register (CRFLG)**

#### WUPIF — Wakeup Interrupt Flag

If the MSCAN08 detects bus activity while in Sleep mode, it sets the WUPIF flag. If not masked, a wake-up interrupt is pending while this flag is set.

1 = MSCAN08 has detected activity on the bus and requested wake-up.

0 = No wake-up interrupt has occurred.

#### RWRNIF — Receiver Warning Interrupt Flag

This flag is set when the MSCAN08 goes into warning status due to the receive error counter (REC) exceeding 96 and neither one of the Error Interrupt flags or the Bus-off Interrupt flag is set<sup>1</sup>. If not masked, an error interrupt is pending while this flag is set.

- 1 = MSCAN08 has gone into receiver warning status.
- 0 = No receiver warning status has been reached.

#### TWRNIF — Transmitter Warning Interrupt Flag

This flag is set when the MSCAN08 goes into warning status due to the transmit error counter (TEC) exceeding 96 and neither one of the error interrupt flags or the bus-off interrupt flag is set<sup>2</sup>. If not masked, an error interrupt is pending while this flag is set.

- 1 = MSCAN08 has gone into transmitter warning status.
- 0 = No transmitter warning status has been reached.

#### RERRIF — Receiver Error Passive Interrupt Flag

This flag is set when the MSCAN08 goes into error passive status due to the receive error counter exceeding 127 and the bus-off interrupt flag is not set<sup>3</sup>. If not masked, an Error interrupt is pending while this flag is set.

- 1 = MSCAN08 has gone into receiver error passive status.
- 0 = No receiver error passive status has been reached.

#### TERRIF — Transmitter Error Passive Interrupt Flag

This flag is set when the MSCAN08 goes into error passive status due to the Transmit Error counter exceeding 127 and the Bus-off interrupt flag is not set<sup>4</sup>. If not masked, an Error interrupt is pending while this flag is set.

- 1 = MSCAN08 went into transmit error passive status.
- 0 = No transmit error passive status has been reached.

- 
1. Condition to set the flaf:  $RWRNIF = (96 \leq REC) \& \overline{RERRIF} \& \overline{TERRIF} \& \overline{BOFFIF}$
  2. Condition to set the flaf:  $TWRNIF = (96 \leq TEC) \& \overline{RERRIF} \& \overline{TERRIF} \& \overline{BOFFIF}$
  3. Condition to set the flaf:  $RERRIF = (127 \leq REC \leq 255) \& \overline{BOFFIF}$
  4. Condition to set the flaf:  $TERRIF = (128 \leq TEC \leq 255) \& \overline{BOFFIF}$

## MSCAN Controller (MSCAN08)

### BOFFIF — Bus-Off Interrupt Flag

This flag is set when the MSCAN08 goes into bus-off status, due to the transmit error counter exceeding 255. It cannot be cleared before the MSCAN08 has monitored 128 times 11 consecutive 'recessive' bits on the bus. If not masked, an Error interrupt is pending while this flag is set.

1 = MSCAN08 has gone into bus-off status.

0 = No bus-off status has been reached.

### OVRIF — Overrun Interrupt Flag

This flag is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this flag is set.

1 = A data overrun has been detected since last clearing the flag.

0 = No data overrun has occurred.

### RXF — Receive Buffer Full

The RXF flag is set by the MSCAN08 when a new message is available in the foreground receive buffer. This flag indicates whether the buffer is loaded with a correctly received message. After the CPU has read that message from the receive buffer the RXF flag must be cleared to release the buffer. A set RXF flag prohibits the exchange of the background receive buffer into the foreground buffer. If not masked, a receive interrupt is pending while this flag is set.

1 = The receive buffer is full. A new message is available.

0 = The receive buffer is released (not full).

**NOTE:** *To ensure data integrity, no registers of the receive buffer shall be read while the RXF flag is cleared.*

**NOTE:** *The CRFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.*

**MSCAN08**  
**Receiver Interrupt**  
**Enable Register**

Address: \$0505

|        | Bit 7 | 6      | 5      | 4      | 3      | 2      | 1     | Bit 0 |
|--------|-------|--------|--------|--------|--------|--------|-------|-------|
| Read:  | WUPIE | RWRNIE | TWRNIE | RERRIE | TERRIE | BOFFIE | OVRIE | RXFIE |
| Write: |       |        |        |        |        |        |       |       |
| Reset: | 0     | 0      | 0      | 0      | 0      | 0      | 0     | 0     |

**Figure 20. Receiver Interrupt Enable Register (CRIER)**

**WUPIE** — Wakeup Interrupt Enable

1 = A wakeup event will result in a wakeup interrupt.

0 = No interrupt will be generated from this event.

**RWRNIE** — Receiver Warning Interrupt Enable

1 = A receiver warning status event will result in an error interrupt.

0 = No interrupt is generated from this event.

**TWRNIE** — Transmitter Warning Interrupt Enable

1 = A transmitter warning status event will result in an error interrupt.

0 = No interrupt is generated from this event.

**RERRIE** — Receiver Error Passive Interrupt Enable

1 = A receiver error passive status event will result in an error interrupt.

0 = No interrupt is generated from this event.

**TERRIE** — Transmitter Error Passive Interrupt Enable

1 = A transmitter error passive status event will result in an error interrupt.

0 = No interrupt is generated from this event.

**BOFFIE** — Bus-Off Interrupt Enable

1 = A bus-off event will result in an error interrupt.

0 = No interrupt is generated from this event.

**OVRIE** — Overrun Interrupt Enable

1 = An overrun event will result in an error interrupt.

0 = No interrupt is generated from this event.

## MSCAN Controller (MSCAN08)

RXFIE — Receiver Full Interrupt Enable

1 = A receive buffer full (successful message reception) event will result in a receive interrupt.

0 = No interrupt will be generated from this event.


**NOTE:** The CRIER register is held in the reset state when the SFTRES bit in CMCR0 is set.

### MSCAN08 Transmitter Flag Register

The Abort Acknowledge flags are read only. The Transmitter Buffer Empty flags are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag setting. The Transmitter Buffer Empty flags each have an associated interrupt enable bit in the CTCR register. A hard or soft reset will reset the register.

Address: \$0506 5

|        | Bit 7 | 6      | 5      | 4      | 3 | 2    | 1    | Bit 0 |
|--------|-------|--------|--------|--------|---|------|------|-------|
| Read:  | 0     | ABTAK2 | ABTAK1 | ABTAK0 | 0 | TXE2 | TXE1 | TXE0  |
| Write: |       |        |        |        |   |      |      |       |
| Reset: | 0     | 0      | 0      | 0      | 0 | 1    | 1    | 1     |

 = Unimplemented

**Figure 21. Transmitter Flag Register (CTFLG)**

ABTAK2–ABTAK0 — Abort Acknowledge

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. After a particular message buffer has been flagged empty, this flag can be used by the application software to identify whether the message has been aborted successfully or has been sent. The ABTAKx flag is cleared implicitly whenever the corresponding TXE flag is cleared.

1 = The message has been aborted.

0 = The message has not been aborted, thus has been sent out.

### TXE2–TXE0 — Transmitter Empty

This flag indicates that the associated transmit message buffer is empty, thus not scheduled for transmission. The CPU must handshake (clear) the flag after a message has been set up in the transmit buffer and is due for transmission. The MSCAN08 sets the flag after the message has been sent successfully. The flag is also set by the MSCAN08 when the transmission request was successfully aborted due to a pending abort request (see [Transmit Buffer Priority Registers](#) on page 360). If not masked, a receive interrupt is pending while this flag is set.

Clearing a TXEx flag also clears the corresponding ABTAKx flag (ABTAK, see above). When a TXEx flag is set, the corresponding ABTRQx bit (ABTRQ, see [MSCAN08 Transmitter Control Register](#) on page 374) is cleared.

1 = The associated message buffer is empty (not scheduled).

0 = The associated message buffer is full (loaded with a message due for transmission).

**NOTE:** *To ensure data integrity, no registers of the transmit buffers should be written to while the associated TXE flag is cleared.*


**NOTE:** *The CTFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.*

## MSCAN Controller (MSCAN08)

### MSCAN08 Transmitter Control Register

Address: \$0507

|        | Bit 7 | 6      | 5      | 4      | 3 | 2      | 1      | Bit 0  |
|--------|-------|--------|--------|--------|---|--------|--------|--------|
| Read:  | 0     | ABTRQ2 | ABTRQ1 | ABTRQ0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| Write: |       |        |        |        |   |        |        |        |
| Reset: | 0     | 0      | 0      | 0      | 0 | 0      | 0      | 0      |

 = Unimplemented

**Figure 22. Transmitter Control Register (CTCR)**

#### ABTRQ2–ABTRQ0 — Abort Request

The CPU sets an ABTRQx bit to request that an already scheduled message buffer (TXE = 0) be aborted. The MSCAN08 will grant the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted the associated TXE and the abort acknowledge flag (ABTAK) (see [MSCAN08 Transmitter Flag Register](#) on page 372) will be set and an TXE interrupt is generated if enabled. The CPU cannot reset ABTRQx. ABTRQx is cleared implicitly whenever the associated TXE flag is set.

1 = Abort request pending

0 = No abort request

**NOTE:** *The software must not clear one or more of the TXE flags in CTFLG and simultaneously set the respective ABTRQ bit(s).*

#### TXEIE2–TXEIE0 — Transmitter Empty Interrupt Enable

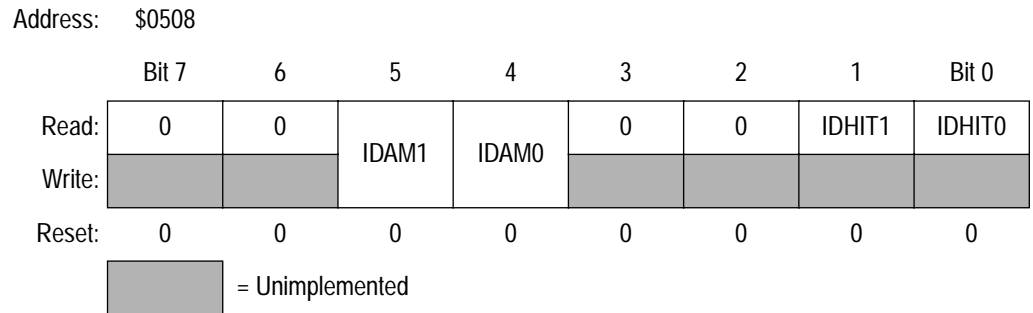
1 = A transmitter empty (transmit buffer available for transmission) event results in a transmitter empty interrupt.

0 = No interrupt is generated from this event.

**NOTE:** *The CTCR register is held in the reset state when the SFTRES bit in CMCR0 is set.*



**MSCAN08  
 Identifier  
 Acceptance  
 Control Register**



**Figure 23. Identifier Acceptance Control Register (CIDAC)**

**IDAM1–IDAM0— Identifier Acceptance Mode**

The CPU sets these flags to define the identifier acceptance filter organization (see [Identifier Acceptance Filter](#) on page 340). **Table 9** summarizes the different settings. In “filter closed” mode no messages will be accepted so that the foreground buffer will never be reloaded.

**Table 9. Identifier Acceptance Mode Settings**

| IDAM1 | IDAM0 | Identifier Acceptance Mode      |
|-------|-------|---------------------------------|
| 0     | 0     | Single 32-Bit Acceptance Filter |
| 0     | 1     | Two 16-Bit Acceptance Filter    |
| 1     | 0     | Four 8-Bit Acceptance Filters   |
| 1     | 1     | Filter Closed                   |

**IDHIT1–IDHIT0— Identifier Acceptance Hit Indicator**

The MSCAN08 sets these flags to indicate an identifier acceptance hit (see [Identifier Acceptance Filter](#) on page 340). **Table 9** summarizes the different settings.

**Table 10. Identifier Acceptance Hit Indication**

| IDHIT1 | IDHIT0 | Identifier Acceptance Hit |
|--------|--------|---------------------------|
| 0      | 0      | Filter 0 Hit              |
| 0      | 1      | Filter 1 Hit              |
| 1      | 0      | Filter 2 Hit              |
| 1      | 1      | Filter 3 Hit              |

## MSCAN Controller (MSCAN08)


The IDHIT indicators are always related to the message in the foreground buffer. When a message gets copied from the background to the foreground buffer, the indicators are updated as well.

**NOTE:** The CIDAC register can be written only if the SFTRES bit in the CMCR0 is set.

### MSCAN08 Receive Error Counter

Address: \$050E

|        | Bit 7  | 6      | 5      | 4      | 3      | 2      | 1      | Bit 0  |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Read:  | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| Write: |        |        |        |        |        |        |        |        |
| Reset: | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |

 = Unimplemented


**Figure 24. Receiver Error Counter (CRXERR)**

This register reflects the status of the MSCAN08 receive error counter. The register is read only.

### MSCAN08 Transmit Error Counter

Address: \$050F

|        | Bit 7  | 6      | 5      | 4      | 3      | 2      | 1      | Bit 0  |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Read:  | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| Write: |        |        |        |        |        |        |        |        |
| Reset: | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |

 = Unimplemented

**Figure 25. Transmit Error Counter (CTXERR)**

This register reflects the status of the MSCAN08 transmit error counter. The register is read only.

**NOTE:** Both error counters may only be read when in Sleep or Soft Reset mode.

**MSCAN08  
 Identifier  
 Acceptance  
 Registers**

On reception each message is written into the background receive buffer. The CPU is only signalled to read the message, however, if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message will be overwritten by the next message (dropped).

The acceptance registers of the MSCAN08 are applied on the IDR0 to IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers only the first two (CIDMR0/1 and CIDAR0/1) are applied.

**CIDAR0** Address: \$0510

|        | Bit 7               | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
|--------|---------------------|-----|-----|-----|-----|-----|-----|-------|
| Read:  |                     |     |     |     |     |     |     |       |
| Write: | AC7                 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0   |
| Reset: | Unaffected by Reset |     |     |     |     |     |     |       |

**CIDAR1** Address: \$050511

|        | Bit 7               | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
|--------|---------------------|-----|-----|-----|-----|-----|-----|-------|
| Read:  |                     |     |     |     |     |     |     |       |
| Write: | AC7                 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0   |
| Reset: | Unaffected by Reset |     |     |     |     |     |     |       |

**CIDAR2** Address: \$0512

|        | Bit 7               | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
|--------|---------------------|-----|-----|-----|-----|-----|-----|-------|
| Read:  |                     |     |     |     |     |     |     |       |
| Write: | AC7                 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0   |
| Reset: | Unaffected by Reset |     |     |     |     |     |     |       |

**CIDAR3** Address: \$0513

|        | Bit 7               | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
|--------|---------------------|-----|-----|-----|-----|-----|-----|-------|
| Read:  |                     |     |     |     |     |     |     |       |
| Write: | AC7                 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0   |
| Reset: | Unaffected by Reset |     |     |     |     |     |     |       |

**Figure 26. Identifier Acceptance Registers (CIDAR0–CIDAR3)**

## MSCAN Controller (MSCAN08)

### AC7–AC0 — Acceptance Code Bits

AC7–AC0 comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDR<sub>n</sub>) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

**NOTE:** The CIDAR0–3 registers can be written only if the SFTRES bit in CMCR0 is set

### MSCAN08 Identifier Mask Registers (CIDMR0-3)

The identifier mask registers specify which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. For standard identifiers it is required to program the last three bits (AM2–AM0) in the mask register CIDMR1 to ‘don’t care’.

|               |                     |     |     |     |     |     |     |       |
|---------------|---------------------|-----|-----|-----|-----|-----|-----|-------|
| <b>CIDMR0</b> | Address: \$0514     |     |     |     |     |     |     |       |
|               | Bit 7               | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
| Read:         | AM7                 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0   |
| Write:        |                     |     |     |     |     |     |     |       |
| Reset:        | Unaffected by Reset |     |     |     |     |     |     |       |
| <b>CIDMR1</b> | Address: \$0515     |     |     |     |     |     |     |       |
|               | Bit 7               | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
| Read:         | AM7                 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0   |
| Write:        |                     |     |     |     |     |     |     |       |
| Reset:        | Unaffected by Reset |     |     |     |     |     |     |       |
| <b>CIDMR2</b> | Address: \$0516     |     |     |     |     |     |     |       |
|               | Bit 7               | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
| Read:         | AM7                 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0   |
| Write:        |                     |     |     |     |     |     |     |       |
| Reset:        | Unaffected by Reset |     |     |     |     |     |     |       |
| <b>CIDMR3</b> | Address: \$0517     |     |     |     |     |     |     |       |
|               | Bit 7               | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
| Read:         | AM7                 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0   |
| Write:        |                     |     |     |     |     |     |     |       |
| Reset:        | Unaffected by Reset |     |     |     |     |     |     |       |

**Figure 27. Identifier Mask Registers (CIDMR0–CIDMR3)**

#### AM7–AM0 — Acceptance Mask Bits

If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match will be detected. The message will be accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register will not affect whether or not the message is accepted.

1 = Ignore corresponding acceptance code register bit.

0 = Match corresponding acceptance code register and identifier bits.

**NOTE:** *The CIDMR0-3 registers can be written only if the SFTRES bit in the CMCR0 is set*

## MSCAN Controller (MSCAN08)

# Keyboard Module (KBD)

---

---

## Contents

|   |     |
|---|-----|
| Introduction .....                            | 381 |
| Features .....                                | 382 |
| Functional Description .....                  | 382 |
| Keyboard Initialization .....                 | 385 |
| Low-Power Modes .....                         | 386 |
| Wait Mode .....                               | 386 |
| Stop Mode .....                               | 386 |
| Keyboard Module During Break Interrupts ..... | 386 |
| I/O Registers .....                           | 387 |
| Keyboard Status and Control Register .....    | 387 |
| Keyboard Interrupt Enable Register .....      | 388 |

---

---

## Introduction

The keyboard interrupt module (KBD) provides five independently maskable external interrupt pins.

---

---

### Features

KBD features include:

- Five Keyboard Interrupt Pins with Separate Keyboard Interrupt Enable Bits and One Keyboard Interrupt Mask
- Hysteresis Buffers
- Programmable Edge-Only or Edge- and Level- Interrupt Sensitivity
- Automatic Interrupt Acknowledge
- Exit from Low-Power Modes

---

---

### Functional Description

Writing to the KBIE4–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port G or port H pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin also enables its internal pullup device. A logic 0 applied to an enabled keyboard interrupt pin latches a keyboard interrupt request.

A keyboard interrupt is latched when one or more keyboard pins goes low after all were high. The MODEK bit in the keyboard status and control register controls the triggering mode of the keyboard interrupt.

- If the keyboard interrupt is edge-sensitive only, a falling edge on a keyboard pin does not latch an interrupt request if another keyboard pin is already low. To prevent losing an interrupt request on one pin because another pin is still low, software can disable the latter pin while it is low.
- If the keyboard interrupt is falling edge- and low level-sensitive, an interrupt request is present as long as any keyboard pin is low.



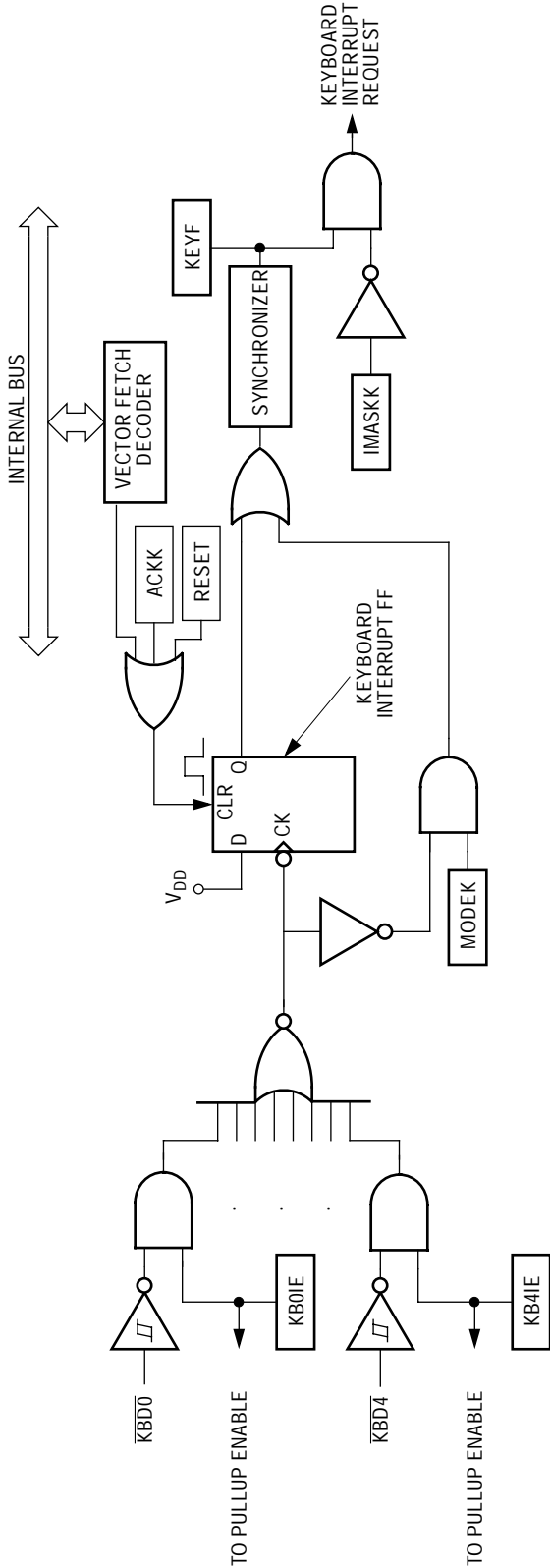


Figure 1. Keyboard Module Block Diagram

| Register Name                                | Bit 7  | 6 | 5 | 4 | 3     | 2     | 1      | Bit 0 |
|--|--------|---|---|---|-------|-------|--------|-------|
| Keyboard Status and Control Register (KBSCR) | Read:  | 0 | 0 | 0 | KEYF  | 0     | IMASKK | MODEK |
|  | Write: |   |   |   |       | ACKK  |        |       |
|  | Reset: | 0 | 0 | 0 | 0     | 0     | 0      | 0     |
| Keyboard Interrupt Enable Register (KBIER)   | Read:  | 0 | 0 | 0 | KBIE4 | KBIE3 | KBIE2  | KBIE1 |
|  | Write: |   |   |   |       |       |        | KBIE0 |
|  | Reset: | 0 | 0 | 0 | 0     | 0     | 0      | 0     |

■ = Unimplemented

Figure 2. I/O Register Summary

Table 1. I/O Register Address Summary

| Register | KBSCR  | KBIER  |
|----------|--------|--------|
| Address  | \$001A | \$001B |

## Keyboard Module (KBD)

If the MODEK bit is set, the keyboard interrupt pins are both falling edge- and low level-sensitive, and both of the following actions must occur to clear a keyboard interrupt request:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACKK bit in the keyboard status and control register (KBSCR). The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit prior to leaving an interrupt service routine also can prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt pins. A falling edge that occurs after writing to the ACKK bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the vector address at locations \$FFDE and \$FFDF.
- Return of all enabled keyboard interrupt pins to logic 1. As long as any enabled keyboard interrupt pin is at logic 0, the keyboard interrupt remains set.

The vector fetch or software clear and the return of all enabled keyboard interrupt pins to logic 1 may occur in any order.

If the MODEK bit is clear, the keyboard interrupt pin is falling edge-sensitive only. With MODEK clear, a vector fetch or software clear immediately clears the keyboard interrupt request.

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt pin stays at logic 0.

The keyboard flag bit (KEYF) in the keyboard status and control register can be used to see if a pending interrupt exists. The KEYF bit is not affected by the keyboard interrupt mask bit (IMASKK) which makes it useful in applications where polling is preferred.

To determine the logic level on a keyboard interrupt pin, use the data direction register to configure the pin as an input and read the data register.

**NOTE:** *Setting a keyboard interrupt enable bit (KBIE<sub>x</sub>) forces the corresponding keyboard interrupt pin to be an input, overriding the data direction register. However, the data direction register bit must be a logic 0 for software to read the pin.*

---

---

## Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pullup to reach a logic 1. Therefore, a false interrupt can occur as soon as the pin is enabled.

To prevent a false interrupt on keyboard initialization:

1. Mask keyboard interrupts by setting the IMASKK bit in the keyboard status and control register
2. Enable the KBI pins by setting the appropriate KBIE<sub>x</sub> bits in the keyboard interrupt enable register
3. Write to the ACKK bit in the keyboard status and control register to clear any false interrupts
4. Clear the IMASKK bit.

An interrupt signal on an edge-triggered pin can be acknowledged immediately after enabling the pin. An interrupt signal on an edge- and level-triggered interrupt pin must be acknowledged after a delay that depends on the external load.

Another way to avoid a false interrupt:

1. Configure the keyboard pins as outputs by setting the appropriate DDRG bits in data direction register G.
2. Configure the keyboard pins as outputs by setting the appropriate DDRH bits in data direction register H.
3. Write logic 1s to the appropriate port G and port H data register bits.
4. Enable the KBI pins by setting the appropriate KBIE<sub>x</sub> bits in the keyboard interrupt enable register.

---

---

### Low-Power Modes

The WAIT and STOP instructions put the MCU in low-power-consumption standby modes.

#### Wait Mode

The keyboard module remains active in wait mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of wait mode.

#### Stop Mode

The keyboard module remains active in stop mode. Clearing the IMASKK bit in the keyboard status and control register enables keyboard interrupt requests to bring the MCU out of stop mode.

---

---

### Keyboard Module During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See [Break Module](#) on page 161.

To allow software to clear the KEYF bit during a break interrupt, write a logic 1 to the BCFE bit. If KEYF is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect the KEYF bit during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0, writing to the keyboard acknowledge bit (ACKK) in the keyboard status and control register during the break state has no effect. See [Keyboard Status and Control Register](#) on page 387.

## I/O Registers

The following registers control and monitor operation of the keyboard module:

- Keyboard status and control register (KBSCR)
- Keyboard interrupt enable register (KBIER)

### Keyboard Status and Control Register

The keyboard status and control register:

- Flags keyboard interrupt requests
- Acknowledges keyboard interrupt requests
- Masks keyboard interrupt requests
- Controls keyboard interrupt triggering sensitivity

Address: \$001A

|        | Bit 7 | 6 | 5 | 4 | 3    | 2    | 1      | Bit 0 |
|--------|-------|---|---|---|------|------|--------|-------|
| Read:  | 0     | 0 | 0 | 0 | KEYF | 0    | IMASKK | MODEK |
| Write: |       |   |   |   |      | ACKK |        |       |
| Reset: | 0     | 0 | 0 | 0 | 0    | 0    | 0      | 0     |

= Unimplemented

**Figure 3. Keyboard Status and Control Register (KBSCR)**

Bits 7–4 — Not used

These read-only bits always read as logic 0s.

**KEYF** — Keyboard Flag Bit

This read-only bit is set when a keyboard interrupt is pending. Reset clears the KEYF bit.

1 = Keyboard interrupt pending

0 = No keyboard interrupt pending

## Keyboard Module (KBD)

### ACKK — Keyboard Acknowledge Bit

Writing a logic 1 to this write-only bit clears the keyboard interrupt request. ACKK always reads as logic 0. Reset clears ACKK.

### IMASKK — Keyboard Interrupt Mask Bit

Writing a logic 1 to this read/write bit prevents the output of the keyboard interrupt mask from generating interrupt requests. Reset clears the IMASKK bit.

- 1 = Keyboard interrupt requests masked
- 0 = Keyboard interrupt requests not masked

### MODEK — Keyboard Triggering Sensitivity Bit

This read/write bit controls the triggering sensitivity of the keyboard interrupt pins. Reset clears MODEK.

- 1 = Keyboard interrupt requests on falling edges and low levels
- 0 = Keyboard interrupt requests on falling edges only

## Keyboard Interrupt Enable Register

The keyboard interrupt enable register enables or disables each port G and each port H pin to operate as a keyboard interrupt pin.

Address: \$001B

|        | Bit 7 | 6 | 5 | 4     | 3     | 2     | 1     | Bit 0 |
|--------|-------|---|---|-------|-------|-------|-------|-------|
| Read:  | 0     | 0 | 0 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| Write: |       |   |   |       |       |       |       |       |
| Reset: | 0     | 0 | 0 | 0     | 0     | 0     | 0     | 0     |

= Unimplemented

**Figure 4. Keyboard Interrupt Enable Register (KBIER)**

### KBIE4–KBIE0 — Keyboard Interrupt Enable Bits

Each of these read/write bits enables the corresponding keyboard interrupt pin to latch interrupt requests. Reset clears the keyboard interrupt enable register.

- 1 = PDx pin enabled as keyboard interrupt pin
- 0 = PDx pin not enabled as keyboard interrupt pin

# Timer Interface Module A (TIMA-6)

---

---

## Contents

|  |     |
|--|-----|
| Introduction . . . . .   | 390 |
| Features . . . . .   | 390 |
| Functional Description. . . . .  | 393 |
| TIMA Counter Prescaler . . . . .   | 393 |
| Input Capture . . . . .  | 393 |
| Output Compare . . . . .   | 395 |
| Unbuffered Output Compare . . . . .  | 395 |
| Buffered Output Compare . . . . .  | 396 |
| Pulse Width Modulation (PWM) . . . . .   | 397 |
| Unbuffered PWM Signal Generation . . . . .   | 398 |
| Buffered PWM Signal Generation . . . . .   | 399 |
| PWM Initialization. . . . .  | 400 |
| Interrupts . . . . .   | 402 |
| Low-Power Modes . . . . .  | 403 |
| Wait Mode. . . . .   | 403 |
| Stop Mode. . . . .   | 403 |
| TIMA During Break Interrupts . . . . .   | 404 |
| I/O Signals. . . . .   | 405 |
| TIMA Clock Pin (PTD6/ATD14/ TCLK) . . . . .  | 405 |
| TIMA Channel I/O Pins (PTF3/TACH5–PTF0/TACH2 and<br>PTE3/TACH1–PTE2/TACH0) . . . . . | 405 |
| I/O Registers . . . . .  | 406 |
| TIMA Status and Control Register . . . . .   | 406 |
| TIMA Counter Registers . . . . .   | 408 |
| TIMA Counter Modulo Registers. . . . .   | 409 |
| TIMA Channel Status and Control Registers . . . . .                                  | 410 |
| TIMA Channel Registers. . . . .  | 416 |

## Timer Interface Module A (TIMA-6)

---

---

### Introduction

This section describes the timer interface module (TIMA). The TIMA is a 6-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. [Figure 1](#) is a block diagram of the TIMA.

---

---

### Features

Features of the TIMA include:

- Six Input Capture/Output Compare Channels
  - Rising-Edge, Falling-Edge, or Any-Edge Input Capture Trigger
  - Set, Clear, or Toggle Output Compare Action
- Buffered and Unbuffered Pulse Width Modulation (PWM) Signal Generation
- Programmable TIMA Clock Input
  - 7-Frequency Internal Bus Clock Prescaler Selection
  - External TIMA Clock Input (4-MHz Maximum Frequency)
- Free-Running or Modulo Up-Count Operation
- Toggle Any Channel Pin on Overflow
- TIMA Counter Stop and Reset Bits





## Timer Interface Module A (TIMA-6)

| Addr.  | Register Name                              | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
|--------|--|--------|-------|-------|------|-------|-------|------|--------|
| \$0020 | TIMA Status/Control Register (TASC)        | TOF    | TOIE  | TSTOP | TRST | 0     | PS2   | PS1  | PS0    |
| \$0021 | Reserved                                   | R      | R     | R     | R    | R     | R     | R    | R      |
| \$0022 | TIMA Counter Register High (TACNTH)        | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$0023 | TIMA Counter Register Low (TACNTL)         | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
| \$0024 | TIMA Counter Modulo Reg. High (TAMODH)     | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$0025 | TIMA Counter Modulo Reg. Low (TAMODL)      | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
| \$0026 | TIMA Ch. 0 Status/Control Register (TASC0) | CH0F   | CH0IE | MS0B  | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| \$0027 | TIMA Ch. 0 Register High (TACH0H)          | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$0028 | TIMA Ch. 0 Register Low (TACH0L)           | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
| \$0029 | TIMA Ch. 1 Status/Control Register (TASC1) | CH1F   | CH1IE | 0     | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| \$002A | TIMA Ch. 1 Register High (TACH1H)          | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$002B | TIMA Ch. 1 Register Low (TACH1L)           | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
| \$002C | TIMA Ch. 2 Status/Control Register (TASC2) | CH2F   | CH2IE | MS2B  | MS2A | ELS2B | ELS2A | TOV2 | CH2MAX |
| \$002D | TIMA Ch. 2 Register High (TACH2H)          | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$002E | TIMA Ch. 2 Register Low (TACH2L)           | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
| \$002F | TIMA Ch. 3 Status/Control Register (TASC3) | CH3F   | CH3IE | 0     | MS3A | ELS3B | ELS3A | TOV3 | CH3MAX |
| \$0030 | TIMA Ch. 3 Register High (TACH3H)          | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$0031 | TIMA Ch. 3 Register Low (TACH3L)           | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
| \$0032 | TIMA Ch. 4 Status/Control Register (TASC4) | CH4F   | CH4IE | MS4B  | MS4A | ELS4B | ELS4A | TOV4 | CH4MAX |
| \$0033 | TIMA Ch. 4 Register High (TACH4H)          | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$0034 | TIMA Ch. 4 Register Low (TACH4L)           | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |
| \$0035 | TIMA Ch. 5 Status/Control Register (TASC5) | CH5F   | CH5IE | 0     | MS5A | ELS5B | ELS5A | TOV5 | CH5MAX |
| \$0036 | TIMA Ch. 5 Register High (TACH5H)          | Bit 15 | 14    | 13    | 12   | 11    | 10    | 9    | Bit 8  |
| \$0037 | TIMA Ch. 5 Register Low (TACH5L)           | Bit 7  | 6     | 5     | 4    | 3     | 2     | 1    | Bit 0  |

R = Reserved

**Figure 2. TIMA I/O Register Summary**

---

---

## Functional Description

**Figure 1** shows the TIMA structure. The central component of the TIMA is the 16-bit TIMA counter that can operate as a free-running counter or a modulo up-counter. The TIMA counter provides the timing reference for the input capture and output compare functions. The TIMA counter modulo registers, TAMODH–TAMODL, control the modulo value of the TIMA counter. Software can read the TIMA counter value at any time without affecting the counting sequence.

The six TIMA channels are programmable independently as input capture or output compare channels.

### TIMA Counter Prescaler

The TIMA clock source can be one of the seven prescaler outputs or the TIMA clock pin, PTD6/TACLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMA status and control register select the TIMA clock source.

### Input Capture

An input capture function has three basic parts: edge select logic, an input capture latch, and a 16-bit counter. Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TASC0 through TASC5 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMA latches the contents of the TIMA counter into the TIMA channel registers, TACHxH–TACHxL. Input captures can generate TIMA CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The result obtained by an input capture will be two more than the value of the free-running counter on the rising edge of the internal bus clock preceding the external transition. This delay is required for internal synchronization.

## Timer Interface Module A (TIMA-6)

The free-running counter contents are transferred to the TIMA channel status and control register (TACHxH–TACHxL, see [TIMA Channel Registers](#) on page 416), on each proper signal transition regardless of whether the TIMA channel flag (CH0F–CH5F in TASC0–TASC5 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or “captured” is the time of the event. Because this value is stored in the input capture register 2 bus cycles after the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see [TIMA Channel Registers](#) on page 416). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the input capture channel register (TACHxH–TACHxL).

## Output Compare

With the output compare function, the TIMA can generate a periodic pulse with a programmable polarity, duration, and frequency. When the counter reaches the value in the registers of an output compare channel, the TIMA can set, clear, or toggle the channel pin. Output compares can generate TIMA CPU interrupt requests.

### *Unbuffered Output Compare*

Any output compare channel can generate unbuffered output compare pulses as described in [Output Compare](#) on page 395. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMA channel registers.

An unsynchronized write to the TIMA channel registers to change an output compare value could cause incorrect operation for up to two counter overflow periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that counter overflow period. Also, using a TIMA overflow interrupt routine to write a new, smaller output compare value may cause the compare to be missed. The TIMA may pass the new value before it is written.

Use the following methods to synchronize unbuffered changes in the output compare value on channel x:

- When changing to a smaller value, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current output compare pulse. The interrupt routine has until the end of the counter overflow period to write the new value.
- When changing to a larger output compare value, enable channel x TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

## Timer Interface Module A (TIMA-6)

### *Buffered Output Compare*

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTE2/TACH0 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The output compare value in the TIMA channel 0 registers initially controls the output on the PTE2/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the output are the ones written to last. TASC0 controls and monitors the buffered output compare function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TACH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered output compare channel whose output appears on the PTF0/TACH2 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS2B bit in TIMA channel 2 status and control register (TASC2) links channel 2 and channel 3. The output compare value in the TIMA channel 2 registers initially controls the output on the PTF0/TACH2 pin. Writing to the TIMA channel 3 registers enables the TIMA channel 3 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (2 or 3) that control the output are the ones written to last. TASC2 controls and monitors the buffered output compare function, and TIMA channel 3 status and control register (TASC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF1/TACH3, is available as a general-purpose I/O pin.

Channels 4 and 5 can be linked to form a buffered output compare channel whose output appears on the PTF2/TACH4 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS4B bit in TIMA channel 4 status and control register (TSC4) links channel 4 and channel 5. The output compare value in the TIMA channel 4 registers initially controls the output on the PTF2/TACH4 pin. Writing to the TIMA channel 5 registers enables the

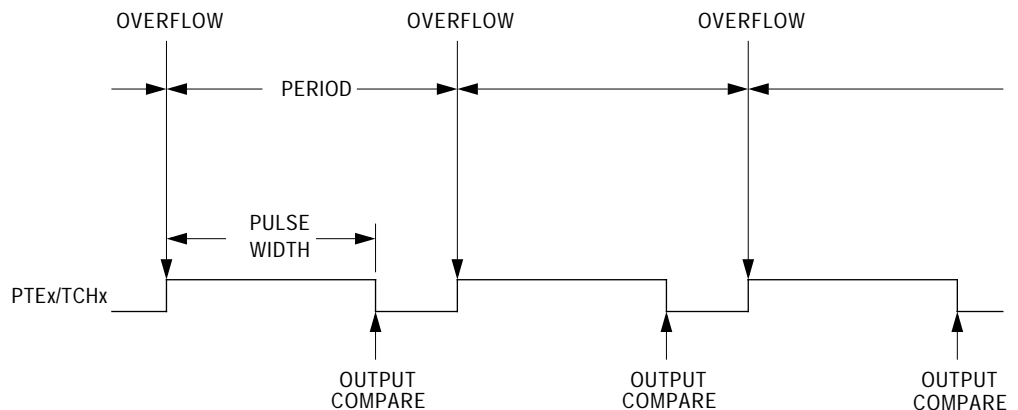
TIMA channel 5 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (4 or 5) that control the output are the ones written to last. TASC4 controls and monitors the buffered output compare function, and TIMA channel 5 status and control register (TASC5) is unused. While the MS4B bit is set, the channel 5 pin, PTF3/TACH5, is available as a general-purpose I/O pin.

**NOTE:** *In buffered output compare operation, do not write new output compare values to the currently active channel registers. Writing to the active channel registers is the same as generating unbuffered output compares.*

### Pulse Width Modulation (PWM)

By using the toggle-on-overflow feature with an output compare channel, the TIMA can generate a PWM signal. The value in the TIMA counter modulo registers determines the period of the PWM signal. The channel pin toggles when the counter reaches the value in the TIMA counter modulo registers. The time between overflows is the period of the PWM signal.

As **Figure 3** shows, the output compare value in the TIMA channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMA to clear the channel pin on output compare if the state of the PWM pulse is logic 1. Program the TIMA to set the pin if the state of the PWM pulse is logic 0.



**Figure 3. PWM Period and Pulse Width**

## Timer Interface Module A (TIMA-6)

The value in the TIMA counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing \$00FF (255) to the TIMA counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is \$000 (see [TIMA Status and Control Register](#) on page 406).

The value in the TIMA channel registers determines the pulse width of the PWM output. The pulse width of an 8-bit PWM signal is variable in 256 increments. Writing \$0080 (128) to the TIMA channel registers produces a duty cycle of 128/256 or 50%.

### *Unbuffered PWM Signal Generation*

Any output compare channel can generate unbuffered PWM pulses as described in [Pulse Width Modulation \(PWM\)](#) on page 397. The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the value currently in the TIMA channel registers.

An unsynchronized write to the TIMA channel registers to change a pulse width value could cause incorrect operation for up to two PWM periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that PWM period. Also, using a TIMA overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIMA may pass the new value before it is written to the TIMA channel registers.

Use the following methods to synchronize unbuffered changes in the PWM pulse width on channel x:

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable channel x TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA overflow interrupt occurs at the end of



the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

**NOTE:** *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare also can cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

#### *Buffered PWM Signal Generation*

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTE2/TACH0 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The TIMA channel 0 registers initially control the pulse width on the PTE2/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the pulse width are the ones written to last. TASC0 controls and monitors the buffered PWM function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TACH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered PWM channel whose output appears on the PTF0/TACH2 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS2B bit in TIMA channel 2 status and control register (TASC2) links channel 2 and channel 3. The TIMA channel 2 registers initially control the pulse width on the PTF0/TACH2 pin. Writing to the TIMA channel 3 registers enables the TIMA channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (2 or 3) that control the pulse width are the ones written to last. TASC2 controls and monitors the buffered PWM function, and TIMA channel 3

## Timer Interface Module A (TIMA-6)

status and control register (TASC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF1/TACH3, is available as a general-purpose I/O pin.

Channels 4 and 5 can be linked to form a buffered PWM channel whose output appears on the PTF2/TACH4 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS4B bit in TIMA channel 4 status and control register (TASC4) links channel 4 and channel 5. The TIMA channel 4 registers initially control the pulse width on the PTF2/TACH4 pin. Writing to the TIMA channel 5 registers enables the TIMA channel 5 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (4 or 5) that control the pulse width are the ones written to last. TASC4 controls and monitors the buffered PWM function, and TIMA channel 5 status and control register (TASC5) is unused. While the MS4B bit is set, the channel 5 pin, PTF3/TACH5, is available as a general-purpose I/O pin.

**NOTE:** *In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. Writing to the active channel registers is the same as generating unbuffered PWM signals.*

### PWM Initialization

To ensure correct operation when generating unbuffered or buffered PWM signals, use the following initialization procedure:

1. In the TIMA status and control register (TASC):
  - a. Stop the TIMA counter by setting the TIMA stop bit, TSTOP.
  - b. Reset the TIMA counter by setting the TIMA reset bit, TRST.
2. In the TIMA counter modulo registers (TAMODH–TAMODL), write the value for the required PWM period.
3. In the TIMA channel x registers (TACHxH–TACHxL), write the value for the required pulse width.
4. In TIMA channel x status and control register (TSCx):
  - a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB–MSxA. (See [Table 2](#)).

- b. Write 1 to the toggle-on-overflow bit, TOVx.
- c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB–ELSxA. The output action on compare must force the output to the complement of the pulse width level. (See [Table 2.](#))

**NOTE:** *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% duty cycle generation and removes the ability of the channel to self-correct in the event of software error or noise. Toggling on output compare can also cause incorrect PWM signal generation when changing the PWM pulse width to a new, much larger value.*

5. In the TIMA status control register (TASC), clear the TIMA stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMA channel 0 registers (TACH0H–TACH0L) initially control the buffered PWM output. TIMA status control register 0 (TASC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Setting MS2B links channels 2 and 3 and configures them for buffered PWM operation. The TIMA channel 2 registers (TACH2H–TACH2L) initially control the PWM output. TIMA status control register 2 (TASC2) controls and monitors the PWM signal from the linked channels. MS2B takes priority over MS2A.

Setting MS4B links channels 4 and 5 and configures them for buffered PWM operation. The TIMA channel 4 registers (TACH4H–TACH4L) initially control the PWM output. TIMA status control register 4 (TASC4) controls and monitors the PWM signal from the linked channels. MS4B takes priority over MS4A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on TIMA overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

## Timer Interface Module A (TIMA-6)

Setting the channel x maximum duty cycle bit (CHxMAX) and clearing the TOVx bit generates a 100% duty cycle output. (See [TIMA Channel Status and Control Registers](#) on page 410).

---

---

### Interrupts

The following TIMA sources can generate interrupt requests:

- TIMA overflow flag (TOF) — The TOF bit is set when the TIMA counter value rolls over to \$0000 after matching the value in the TIMA counter modulo registers. The TIMA overflow interrupt enable bit, TOIE, enables TIMA overflow CPU interrupt requests. TOF and TOIE are in the TIMA status and control register.
- TIMA channel flags (CH5F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIMA CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

## Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes.

### Wait Mode

The TIMA remains active after the execution of a WAIT instruction. In wait mode, the TIMA registers are not accessible by the CPU. Any enabled CPU interrupt request from the TIMA can bring the MCU out of wait mode.

If TIMA functions are not required during wait mode, reduce power consumption by stopping the TIMA before executing the WAIT instruction.

### Stop Mode

The TIMA is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions or the state of the TIMA counter. TIMA operation resumes when the MCU exits stop mode.

---

---

### TIMA During Break Interrupts

A break interrupt stops the TIMA counter and inhibits input captures.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See [SIM Break Flag Control Register](#) on page 125).

To allow software to clear status bits during a break interrupt, write a logic 1 to the BCFE bit. If a status bit is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

---



---

## I/O Signals

Port D shares one of its pins with the TIMA. Port E shares two of its pins with the TIMA and port F shares four of its pins with the TIMA.

PTD6/TACLK is an external clock input to the TIMA prescaler. The six TIMA channel I/O pins are PTE2/TACH0, PTE3/TACH1, PTF0/TACH2, PTF1/TACH3, PTF2/TACH4, and PTF3/TACH5.

### TIMA Clock Pin (PTD6/ATD14/ TCLK)

PTD6/TACLK is an external clock input that can be the clock source for the TIMA counter instead of the prescaled internal bus clock. Select the PTD6/TACLK input by writing logic 1s to the three prescaler select bits, PS[2:0]. (See TIMA Status and Control Register.) The minimum TCLK pulse width,  $TCLK_{LMIN}$  or  $TCLK_{HMIN}$ , is:

$$\frac{1}{\text{bus frequency}} + t_{su}$$

The maximum TCLK frequency is the least: 4 MHz or bus frequency  $\div$  2.

PTD6/TACLK is available as a general-purpose I/O pin or ADC channel when not used as the TIMA clock input. When the PTD6/TACLK pin is the TIMA clock input, it is an input regardless of the state of the DDRD6 bit in data direction register D.

### TIMA Channel I/O Pins (PTF3/TACH5–PTF0/ TACH2 and PTE3/TACH1–PTE2/ TACH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTE2/TACH0, PTE6/TACH2, and PTF2/TACH4 can be configured as buffered output compare or buffered PWM pins.

## Timer Interface Module A (TIMA-6)

### I/O Registers

These I/O registers control and monitor TIMA operation:

- TIMA status and control register (TASC)
- TIMA control registers (TACNTH–TACNTL)
- TIMA counter modulo registers (TAMODH–TAMODL)
- TIMA channel status and control registers (TASC0, TASC1, TASC2, TASC3, TASC4, and TASC5)
- TIMA channel registers (TACH0H–TACH0L, TACH1H–TACH1L, TACH2H–TACH2L, TACH3H–TACH3L, TACH4H–TACH4L, and TACH5H–TACH5L)

### TIMA Status and Control Register

The TIMA status and control register:

- Enables TIMA overflow interrupts
- Flags TIMA overflows
- Stops the TIMA counter
- Resets the TIMA counter
- Prescales the TIMA counter clock

Address: \$0020

|        | Bit 7 | 6    | 5     | 4    | 3 | 2   | 1   | Bit 0 |
|--------|-------|------|-------|------|---|-----|-----|-------|
| Read:  | TOF   | TOIE | TSTOP | 0    | 0 | PS2 | PS1 | PS0   |
| Write: | 0     |      |       | TRST | R |     |     |       |
| Reset: | 0     | 0    | 1     | 0    | 0 | 0   | 0   | 0     |

R = Reserved

**Figure 4. TIMA Status and Control Register (TASC)**



#### TOF — TIMA Overflow Flag Bit

This read/write flag is set when the TIMA counter resets to \$0000 after reaching the modulo value programmed in the TIMA counter modulo registers. Clear TOF by reading the TIMA status and control register when TOF is set and then writing a logic 0 to TOF. If another TIMA overflow occurs before the clearing sequence is complete, then writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

- 1 = TIMA counter has reached modulo value.
- 0 = TIMA counter has not reached modulo value.

#### TOIE — TIMA Overflow Interrupt Enable Bit

This read/write bit enables TIMA overflow interrupts when the TOF bit becomes set. Reset clears the TOIE bit.

- 1 = TIMA overflow interrupts enabled
- 0 = TIMA overflow interrupts disabled

#### TSTOP — TIMA Stop Bit

This read/write bit stops the TIMA counter. Counting resumes when TSTOP is cleared. Reset sets the TSTOP bit, stopping the TIMA counter until software clears the TSTOP bit.

- 1 = TIMA counter stopped
- 0 = TIMA counter active

**NOTE:** *Do not set the TSTOP bit before entering wait mode if the TIMA is required to exit wait mode. Also, when the TSTOP bit is set and input capture mode is enabled, input captures are inhibited until TSTOP is cleared.*

#### TRST — TIMA Reset Bit

Setting this write-only bit resets the TIMA counter and the TIMA prescaler. Setting TRST has no effect on any other registers. Counting resumes from \$0000. TRST is cleared automatically after the TIMA counter is reset and always reads as logic 0. Reset clears the TRST bit.

- 1 = Prescaler and TIMA counter cleared
- 0 = No effect

## Timer Interface Module A (TIMA-6)

**NOTE:** Setting the *TSTOP* and *TRST* bits simultaneously stops the TIMA counter at a value of \$0000.

### PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTD6/TACLK pin or one of the seven prescaler outputs as the input to the TIMA counter as [Table 1](#) shows. Reset clears the PS[2:0] bits.

**Table 1. Prescaler Selection**

| PS[2:0] | TIMA Clock Source       |
|---------|-------------------------|
| 000     | Internal Bus Clock ÷ 1  |
| 001     | Internal Bus Clock ÷ 2  |
| 010     | Internal Bus Clock ÷ 4  |
| 011     | Internal Bus Clock ÷ 8  |
| 100     | Internal Bus Clock ÷ 16 |
| 101     | Internal Bus Clock ÷ 32 |
| 110     | Internal Bus Clock ÷ 64 |
| 111     | PTD6/TACLK              |

### TIMA Counter Registers

The two read-only TIMA counter registers contain the high and low bytes of the value in the TIMA counter. Reading the high byte (*TACNTH*) latches the contents of the low byte (*TACNTL*) into a buffer. Subsequent reads of *TACNTH* do not affect the latched *TACNTL* value until *TACNTL* is read. Reset clears the TIMA counter registers. Setting the TIMA reset bit (*TRST*) also clears the TIMA counter registers.

**NOTE:** If *TACNTH* is read during a break interrupt, be sure to unlatch *TACNTL* by reading *TACNTL* before exiting the break interrupt. Otherwise, *TACNTL* retains the value latched during the break.

Register Name and Address TCNTH — \$0022

|        | Bit 7  | 6      | 5      | 4      | 3      | 2      | 1     | Bit 0 |
|--------|--------|--------|--------|--------|--------|--------|-------|-------|
| Read:  | BIT 15 | BIT 14 | BIT 13 | BIT 12 | BIT 11 | BIT 10 | BIT 9 | BIT 8 |
| Write: | R      | R      | R      | R      | R      | R      | R     | R     |
| Reset: | 0      | 0      | 0      | 0      | 0      | 0      | 0     | 0     |

Register Name and Address TCNTL — \$0023

|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
| Read:  | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| Write: | R     | R     | R     | R     | R     | R     | R     | R     |
| Reset: | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

|   |
|---|
| R |
|---|

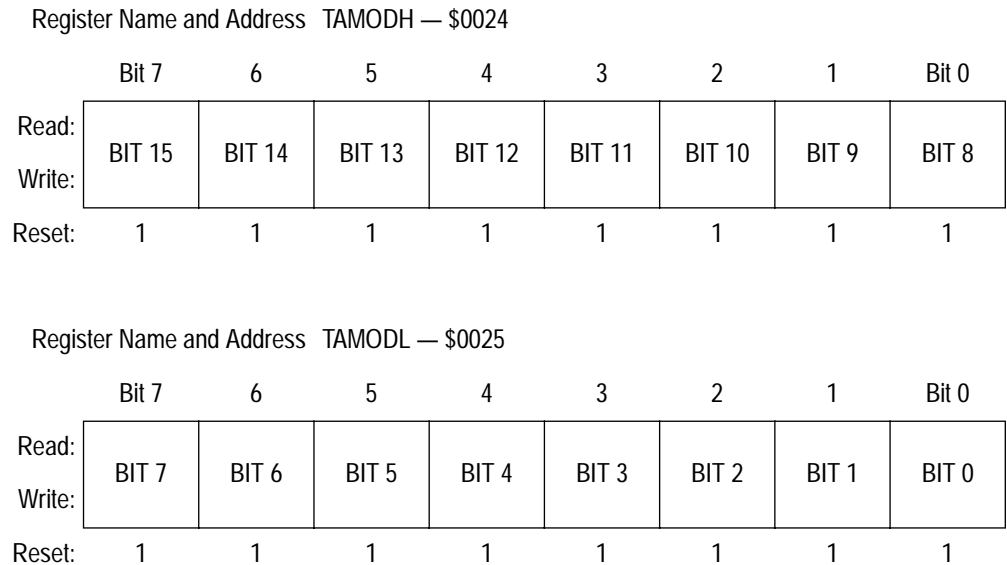
 = Reserved

**Figure 5. TIMA Counter Registers (TCNTH and TCNTL)**

### TIMA Counter Modulo Registers

The read/write TIMA modulo registers contain the modulo value for the TIMA counter. When the TIMA counter reaches the modulo value, the overflow flag (TOF) becomes set, and the TIMA counter resumes counting from \$0000 at the next clock. Writing to the high byte (TAMODH) inhibits the TOF bit and overflow interrupts until the low byte (TAMODL) is written. Reset sets the TIMA counter modulo registers.

## Timer Interface Module A (TIMA-6)



**Figure 6. TIMA Counter Modulo Registers (TAMODH and TAMODL)**

**NOTE:** Reset the TIMA counter before writing to the TIMA counter modulo registers.

### TIMA Channel Status and Control Registers

Each of the TIMA channel status and control registers:

- Flags input captures and output compares
- Enables input capture and output compare interrupts
- Selects input capture, output compare, or PWM operation
- Selects high, low, or toggling output on output compare
- Selects rising edge, falling edge, or any edge as the active input capture trigger
- Selects output toggling on TIMA overflow
- Selects 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Register Name and Address TASC0 — \$0026

|        | Bit 7 | 6     | 5    | 4    | 3     | 2     | 1    | Bit 0  |
|--------|-------|-------|------|------|-------|-------|------|--------|
| Read:  | CH0F  | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| Write: | 0     |       |      |      |       |       |      |        |
| Reset: | 0     | 0     | 0    | 0    | 0     | 0     | 0    | 0      |

Register Name and Address TASC1 — \$0029

|        | Bit 7 | 6     | 5 | 4    | 3     | 2     | 1    | Bit 0  |
|--------|-------|-------|---|------|-------|-------|------|--------|
| Read:  | CH1F  | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| Write: | 0     |       | R |      |       |       |      |        |
| Reset: | 0     | 0     | 0 | 0    | 0     | 0     | 0    | 0      |

|   |
|---|
| R |
|---|

 = Reserved

**Figure 7. TIMA Channel Status  
and Control Registers (TACC0–TASC5)**

## Timer Interface Module A (TIMA-6)

Register Name and Address TASC2 — \$002C

|        | Bit 7 | 6     | 5    | 4    | 3     | 2     | 1    | Bit 0  |
|--------|-------|-------|------|------|-------|-------|------|--------|
| Read:  | CH2F  | CH2IE | MS2B | MS2A | ELS2B | ELS2A | TOV2 | CH2MAX |
| Write: | 0     |       |      |      |       |       |      |        |
| Reset: | 0     | 0     | 0    | 0    | 0     | 0     | 0    | 0      |

Register Name and Address TASC3 — \$002F

|        | Bit 7 | 6     | 5 | 4    | 3     | 2     | 1    | Bit 0  |
|--------|-------|-------|---|------|-------|-------|------|--------|
| Read:  | CH3F  | CH3IE | 0 | MS3A | ELS3B | ELS3A | TOV3 | CH3MAX |
| Write: | 0     |       | R |      |       |       |      |        |
| Reset: | 0     | 0     | 0 | 0    | 0     | 0     | 0    | 0      |

Register Name and Address TASC4 — \$0032

|        | Bit 7 | 6     | 5    | 4    | 3     | 2     | 1    | Bit 0  |
|--------|-------|-------|------|------|-------|-------|------|--------|
| Read:  | CH4F  | CH4IE | MS4B | MS4A | ELS4B | ELS4A | TOV4 | CH4MAX |
| Write: | 0     |       |      |      |       |       |      |        |
| Reset: | 0     | 0     | 0    | 0    | 0     | 0     | 0    | 0      |

Register Name and Address TASC5 — \$0035

|        | Bit 7 | 6     | 5 | 4    | 3     | 2     | 1    | Bit 0  |
|--------|-------|-------|---|------|-------|-------|------|--------|
| Read:  | CH5F  | CH5IE | 0 | MS5A | ELS5B | ELS5A | TOV5 | CH5MAX |
| Write: | 0     |       | R |      |       |       |      |        |
| Reset: | 0     | 0     | 0 | 0    | 0     | 0     | 0    | 0      |

R = Reserved

**Figure 7. TIMA Channel Status and Control Registers (TACC0–TASC5) (Continued)**

#### CHxF — Channel x Flag Bit

When channel x is an input capture channel, this read/write bit is set when an active edge occurs on the channel x pin. When channel x is an output compare channel, CHxF is set when the value in the TIMA counter registers matches the value in the TIMA channel x registers. When CHxIE = 0, clear CHxF by reading TIMA channel x status and control register with CHxF set, and then writing a logic 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

Reset clears the CHxF bit. Writing a logic 1 to CHxF has no effect.

1 = Input capture or output compare on channel x

0 = No input capture or output compare on channel x

#### CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIMA CPU interrupts on channel x.

Reset clears the CHxIE bit.

1 = Channel x CPU interrupt requests enabled

0 = Channel x CPU interrupt requests disabled

#### MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIMA channel 0, TIMA channel 2, and TIMA channel 4 status and control registers.

Setting MS0B disables the channel 1 status and control register and reverts TACH1 pin to general-purpose I/O.

Setting MS2B disables the channel 3 status and control register and reverts TACH3 pin to general-purpose I/O.

Setting MS4B disables the channel 5 status and control register and reverts TACH5 pin to general-purpose I/O.

Reset clears the MSxB bit.

1 = Buffered output compare/PWM operation enabled

0 = Buffered output compare/PWM operation disabled

## Timer Interface Module A (TIMA-6)

### MSxA — Mode Select Bit A

When ELSxB:A  $\neq$  00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. See [Table 2](#).

- 1 = Unbuffered output compare/PWM operation
- 0 = Input capture operation

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin once PWM, output compare mode, or input capture mode is enabled. See [Table 2](#). Reset clears the MSxA bit.

- 1 = Initial output level low
- 0 = Initial output level high

**NOTE:** *Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIMA status and control register (TSC).*

### ELSxB and ELSxA — Edge/Level Select Bits

When channel x is an input capture channel, these read/write bits control the active edge-sensing logic on channel x.

When channel x is an output compare channel, ELSxB and ELSxA control the channel x output behavior when an output compare occurs.

When ELSxB and ELSxA are both clear, channel x is not connected to port E or port F, and pin PTE<sub>x</sub>/TACH<sub>x</sub> or pin PTF<sub>x</sub>/TACH<sub>x</sub> is available as a general-purpose I/O pin. However, channel x is at a state determined by these bits and becomes transparent to the respective pin when PWM, input capture mode, or output compare operation mode is enabled. [Table 2](#) shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.



**Table 2. Mode, Edge, and Level Selection**

| MSxB:MSxA | ELSxB:ELSxA | Mode  | Configuration  |
|-----------|-------------|---|--|
| X0        | 00          | Output<br>Preset                                    | Pin under Port Control;<br>Initialize Timer<br>Output Level High |
| X1        | 00          |   | Pin under Port Control;<br>Initialize Timer<br>Output Level Low  |
| 00        | 01          | Input<br>Capture                                    | Capture on Rising Edge Only                                      |
| 00        | 10          |   | Capture on Falling Edge Only                                     |
| 00        | 11          |   | Capture on Rising or Falling Edge                                |
| 01        | 01          | Output<br>Compare<br>or PWM                         | Toggle Output on Compare   |
| 01        | 10          |   | Clear Output on Compare  |
| 01        | 11          |   | Set Output on Compare  |
| 1X        | 01          | Buffered<br>Output<br>Compare<br>or Buffered<br>PWM | Toggle Output on Compare   |
| 1X        | 10          |   | Clear Output on Compare  |
| 1X        | 11          |   | Set Output on Compare  |

**NOTE:** Before enabling a TIMA channel register for input capture operation, make sure that the PTE<sub>x</sub>/TACH<sub>x</sub> pin or PTF<sub>x</sub>/TACH<sub>x</sub> pin is stable for at least two bus clocks.

#### TOV<sub>x</sub> — Toggle-On-Overflow Bit

When channel x is an output compare channel, this read/write bit controls the behavior of the channel x output when the TIMA counter overflows. When channel x is an input capture channel, TOV<sub>x</sub> has no effect. Reset clears the TOV<sub>x</sub> bit.

1 = Channel x pin toggles on TIMA counter overflow.

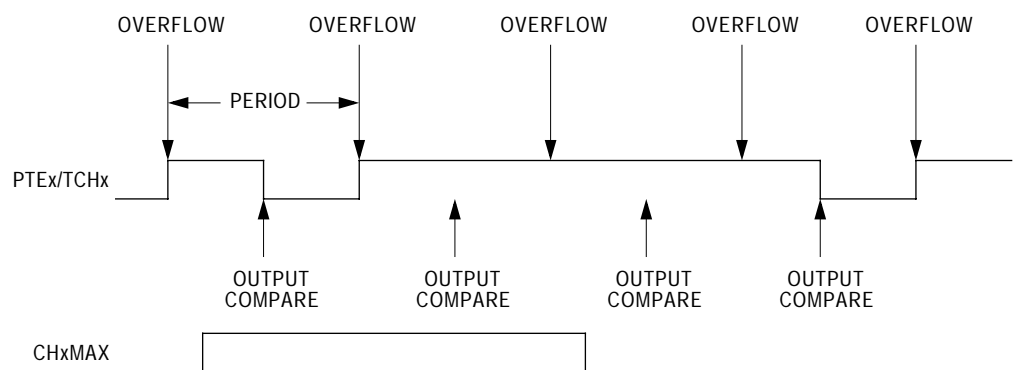
0 = Channel x pin does not toggle on TIMA counter overflow.

**NOTE:** When TOV<sub>x</sub> is set, a TIMA counter overflow takes precedence over a channel x output compare if both occur at the same time.

## Timer Interface Module A (TIMA-6)

### CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic 0, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As **Figure 8** shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.



**Figure 8. CHxMAX Latency**

### TIMA Channel Registers

These read/write registers contain the captured TIMA counter value of the input capture function or the output compare value of the output compare function. The state of the TIMA channel registers after reset is unknown.

In input capture mode ( $MSxB-MSxA = 0:0$ ), reading the high byte of the TIMA channel x registers (TCHxH) inhibits input captures until the low byte (TCHxL) is read.

In output compare mode ( $MSxB-MSxA \neq 0:0$ ), writing to the high byte of the TIMA channel x registers (TCHxH) inhibits output compares and the CHxF bit until the low byte (TCHxL) is written.

Register Name and Address TACH0H — \$0027

|        |        |        |        |        |        |        |       |       |
|--------|--------|--------|--------|--------|--------|--------|-------|-------|
|        | Bit 7  | 6      | 5      | 4      | 3      | 2      | 1     | Bit 0 |
| Read:  | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Write: |        |        |        |        |        |        |       |       |

Reset: Indeterminate after Reset

Register Name and Address TACH0L — \$0028

|        |       |       |       |       |       |       |       |       |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
| Read:  | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Write: |       |       |       |       |       |       |       |       |

Reset: Indeterminate after Reset

Register Name and Address TACH1H — \$002A

|        |        |        |        |        |        |        |       |       |
|--------|--------|--------|--------|--------|--------|--------|-------|-------|
|        | Bit 7  | 6      | 5      | 4      | 3      | 2      | 1     | Bit 0 |
| Read:  | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Write: |        |        |        |        |        |        |       |       |

Reset: Indeterminate after Reset

Register Name and Address TACH1L — \$002B

|        |       |       |       |       |       |       |       |       |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
| Read:  | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Write: |       |       |       |       |       |       |       |       |

Reset: Indeterminate after Reset

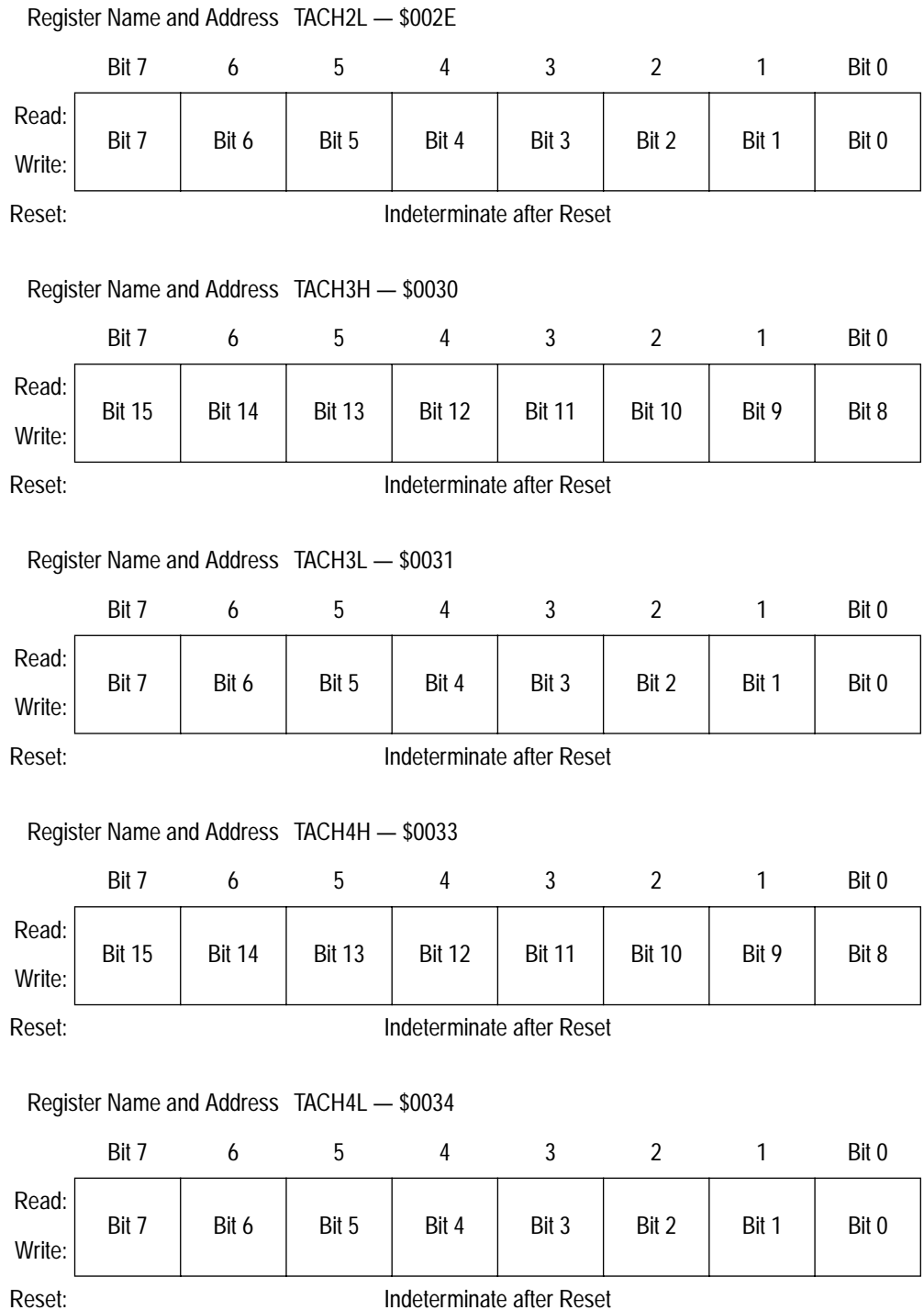
Register Name and Address TACH2H — \$002D

|        |        |        |        |        |        |        |       |       |
|--------|--------|--------|--------|--------|--------|--------|-------|-------|
|        | Bit 7  | 6      | 5      | 4      | 3      | 2      | 1     | Bit 0 |
| Read:  | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Write: |        |        |        |        |        |        |       |       |

Reset: Indeterminate after Reset

**Figure 9. TIMA Channel Registers  
(TACH0H/L–TACH3H/L) (Sheet 1 of 3)**

## Timer Interface Module A (TIMA-6)



**Figure 9. TIMA Channel Registers  
(TACH0H/L–TACH3H/L) (Sheet 2 of 3)**

Register Name and Address TACH5H — \$0036

|        | Bit 7                     | 6      | 5      | 4      | 3      | 2      | 1     | Bit 0 |
|--------|---------------------------|--------|--------|--------|--------|--------|-------|-------|
| Read:  | Bit 15                    | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| Write: |                           |        |        |        |        |        |       |       |
| Reset: | Indeterminate after Reset |        |        |        |        |        |       |       |

Register Name and Address TACH5L — \$0037

|        | Bit 7                     | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|---------------------------|-------|-------|-------|-------|-------|-------|-------|
| Read:  | Bit 7                     | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Write: |                           |       |       |       |       |       |       |       |
| Reset: | Indeterminate after Reset |       |       |       |       |       |       |       |

**Figure 9. TIMA Channel Registers  
(TACH0H/L–TACH3H/L) (Sheet 3 of 3)**

## Timer Interface Module A (TIMA-6)

# Analog-to-Digital Converter (ADC-15)

---

---

## Contents

|   |     |
|---|-----|
| Introduction . . . . .  | 421 |
| Features . . . . .  | 422 |
| Functional Description . . . . .  | 422 |
| ADC Port I/O Pins . . . . .   | 423 |
| Voltage Conversion . . . . .  | 424 |
| Conversion Time . . . . .   | 424 |
| Continuous Conversion . . . . .   | 424 |
| Accuracy and Precision . . . . .  | 425 |
| Interrupts . . . . .  | 425 |
| Low-Power Modes . . . . .   | 425 |
| Wait Mode . . . . .   | 425 |
| Stop Mode . . . . .   | 425 |
| I/O Signals . . . . .   | 426 |
| ADC Analog Power Pin (VDDAREF)/ADC Voltage Reference Pin<br>(VREFH) . . . . .   | 426 |
| ADC Analog Ground Pin (VSSA)/ADC Voltage Reference Low Pin<br>(VREFL) . . . . . | 426 |
| ADC Voltage In (ADCVIN) . . . . .   | 426 |
| I/O Registers . . . . .   | 427 |
| ADC Status and Control Register . . . . .                                       | 427 |
| ADC Data Register . . . . .   | 430 |
| ADC Input Clock Register . . . . .  | 430 |

---

---

## Introduction

This section describes the analog-to-digital converter (ADC-15). The ADC is an 8-bit analog-to-digital converter.

## Analog-to-Digital Converter (ADC-15)

---

---

### Features

Features of the ADC module include:

- 15 Channels with Multiplexed Input
- Linear Successive Approximation
- 8-Bit Resolution
- Single or Continuous Conversion
- Conversion Complete Flag or Conversion Complete Interrupt
- Selectable ADC Clock

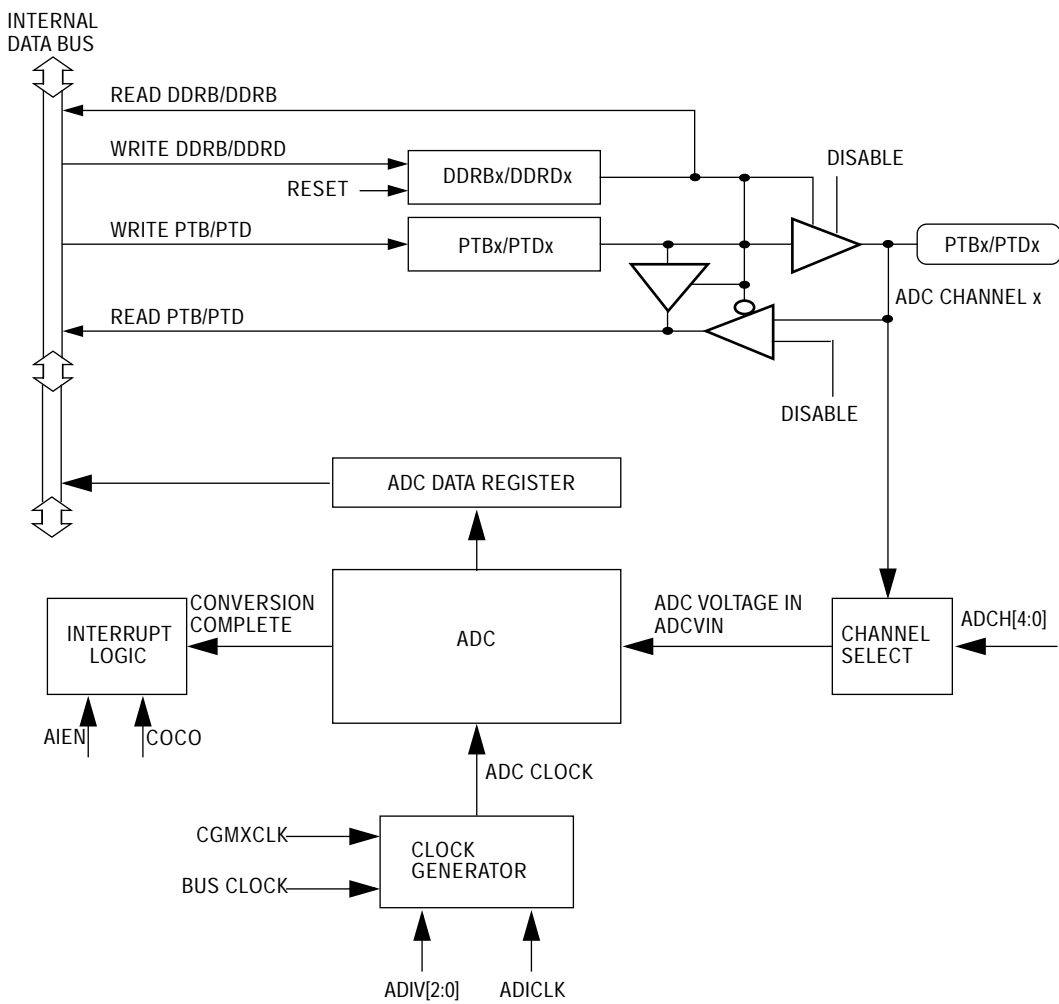
---

---

### Functional Description

Fifteen ADC channels are available for sampling external sources at pins PTD6/TACLK–PTD0 and PTB7/ATD7–PTB0/ATD0. An analog multiplexer allows the single ADC converter to select one of 15 ADC channels as ADC voltage in (ADCVIN). ADCVIN is converted by the successive approximation register-based counters. When the conversion is completed, ADC places the result in the ADC data register and sets a flag or generates an interrupt. See [Figure 1](#).





**Figure 1. ADC Block Diagram**

**ADC Port I/O Pins**

PTD6/TACLK-PTD0 and PTB7/ATD7-PTB0/ATD0 are general-purpose I/O pins that share with the ADC channels.

The channel select bits define which ADC channel/port pin will be used as the input signal. The ADC overrides the port I/O logic by forcing that pin as input to the ADC. The remaining ADC channels/port pins are controlled by the port I/O logic and can be used as general-purpose I/O. Writes to the port register or DDR will not have any affect on the port pin that is selected by the ADC. Read of a port pin which is in use by the

## Analog-to-Digital Converter (ADC-15)

ADC will return a logic 0 if the corresponding DDR bit is at logic 0. If the DDR bit is at logic 1, the value in the port data latch is read.

**NOTE:** Do not use ADC channels ATD14 or ATD12 when using the PTD6/TACLK or PTD4/TBLCK pins as the clock inputs for the 16-bit Timers.

### Voltage Conversion

When the input voltage to the ADC equals  $V_{REFH}$  (see [ADC Characteristics](#) on page 439), the ADC converts the signal to \$FF (full scale). If the input voltage equals  $V_{SSA}$ , the ADC converts it to \$00. Input voltages between  $V_{REFH}$  and  $V_{SSA}$  are a straight-line linear conversion. Conversion accuracy of all other input voltages is not guaranteed. Current injection on unused ADC pins can also cause conversion inaccuracies.

**NOTE:** Input voltage should not exceed the analog supply voltages.

### Conversion Time

Conversion starts after a write to the ADSCR (ADC status control register, \$0038), and requires between 16 and 17 ADC clock cycles to complete. Conversion time in terms of the number of bus cycles is a function of ADICLK select, CGMXCLK frequency, bus frequency, and ADIV prescaler bits. For example, with a CGMXCLK frequency of 4 MHz, bus frequency of 8 MHz, and fixed ADC clock frequency of 1 MHz, one conversion will take between 16 and 17  $\mu$ s and there will be between 128 bus cycles between each conversion. Sample rate is approximately 60 kHz.

Refer to [ADC Characteristics](#) on page 439.

$$\text{Conversion Time} = \frac{16 \text{ to } 17 \text{ ADC Clock Cycles}}{\text{ADC Clock Frequency}}$$

$$\text{Number of Bus Cycles} = \text{Conversion Time} \times \text{Bus Frequency}$$

### Continuous Conversion

In the continuous conversion mode, the ADC data register will be filled with new data after each conversion. Data from the previous conversion will be overwritten whether that data has been read or not. Conversions

will continue until the ADCO bit (ADC status control register, \$0038) is cleared. The COCO bit is set after the first conversion and will stay set for the next several conversions until the next write of the ADC status and control register or the next read of the ADC data register.

### Accuracy and Precision

The conversion process is monotonic and has no missing codes. See [ADC Characteristics](#) on page 439 for accuracy information.

---

---

## Interrupts

When the AIEN bit is set, the ADC module is capable of generating a CPU interrupt after each ADC conversion. A CPU interrupt is generated if the COCO bit (ADC status control register, \$0038) is at logic 0. If the COCO bit is set, an interrupt is generated. The COCO bit is not used as a conversion complete flag when interrupts are enabled.

---

---

## Low-Power Modes

The following subsections describe the low-power modes.

### Wait Mode

The ADC continues normal operation during wait mode. Any enabled CPU interrupt request from the ADC can bring the MCU out of wait mode. If the ADC is not required to bring the MCU out of wait mode, power down the ADC by setting the ADCH[4:0] bits in the ADC status and control register before executing the WAIT instruction.

### Stop Mode

The ADC module is inactive after the execution of a STOP instruction. Any pending conversion is aborted. ADC conversions resume when the MCU exits stop mode. Allow one conversion cycle to stabilize the analog circuitry before attempting a new ADC conversion after exiting stop mode.

## Analog-to-Digital Converter (ADC-15)

---

---

### I/O Signals

The ADC module has 15 channels that are shared with I/O ports B and D. Refer to [ADC Characteristics](#) on page 439 for voltages referenced below.

#### ADC Analog Power Pin ( $V_{DDAREF}$ )/ADC Voltage Reference Pin ( $V_{REFH}$ )

The ADC analog portion uses  $V_{DDAREF}$  as its power pin. Connect the  $V_{DDA}/V_{DDAREF}$  pin to the same voltage potential as  $V_{DD}$ . External filtering may be necessary to ensure clean  $V_{DDAREF}$  for good results.

$V_{REFH}$  is the high reference voltage for all analog-to-digital conversions.

**NOTE:** Route  $V_{DDAREF}$  carefully for maximum noise immunity and place bypass capacitors as close as possible to the package.  $V_{DDAREF}$  must be present for operation of the ADC.

#### ADC Analog Ground Pin ( $V_{SSA}$ )/ADC Voltage Reference Low Pin ( $V_{REFL}$ )

The ADC analog portion uses  $V_{SSA}$  as its ground pin. Connect the  $V_{SSA}$  pin to the same voltage potential as  $V_{SS}$ .

$V_{REFL}$  is the lower reference supply for the ADC.

#### ADC Voltage In (ADCVIN)

ADCVIN is the input voltage signal from one of the 15 ADC channels to the ADC module.

## I/O Registers

These I/O registers control and monitor ADC operation:

- ADC status and control register (ADSCR)
- ADC data register (ADR)
- ADC clock register (ADICLK)

### ADC Status and Control Register

The following paragraphs describe the function of the ADC status and control register.

Address: \$0038

|        | Bit 7 | 6    | 5    | 4   | 3   | 2   | 1   | Bit 0 |
|--------|-------|------|------|-----|-----|-----|-----|-------|
| Read:  | COCO  | AIEN | ADCO | CH4 | CH3 | CH2 | CH1 | CH0   |
| Write: | R     |      |      |     |     |     |     |       |
| Reset: | 0     | 0    | 0    | 1   | 1   | 1   | 1   | 1     |

R = Reserved

**Figure 2. ADC Status and Control Register (ADSCR)**

#### COCO — Conversions Complete Bit

When the AIEN bit is a logic 0, the COCO is a read-only bit which is set each time a conversion is completed. This bit is cleared whenever the ADC status and control register is written or whenever the ADC data register is read.

If the AIEN bit is a logic 1, the COCO is a read/write bit which selects the CPU to service the ADC interrupt request. Reset clears this bit.

1 = conversion completed (AIEN = 0)

0 = conversion not completed (AIEN = 0)

or

CPU interrupt enabled (AIEN = 1)

## Analog-to-Digital Converter (ADC-15)

### AIEN — ADC Interrupt Enable Bit

When this bit is set, an interrupt is generated at the end of an ADC conversion. The interrupt signal is cleared when the data register is read or the status/control register is written. Reset clears the AIEN bit.

- 1 = ADC interrupt enabled
- 0 = ADC interrupt disabled

### ADCO — ADC Continuous Conversion Bit

When set, the ADC will convert samples continuously and update the ADR register at the end of each conversion. Only one conversion is allowed when this bit is cleared. Reset clears the ADCO bit.

- 1 = Continuous ADC conversion
- 0 = One ADC conversion

### ADCH[4:0] — ADC Channel Select Bits

ADCH4, ADCH3, ADCH2, ADCH1, and ADCH0 form a 5-bit field which is used to select one of 15 ADC channels. Channel selection is detailed in the following table. Care should be taken when using a port pin as both an analog and a digital input simultaneously to prevent switching noise from corrupting the analog signal. See [Table 1](#).

The ADC subsystem is turned off when the channel select bits are all set to one. This feature allows for reduced power consumption for the MCU when the ADC is not used. Reset sets these bits.

**NOTE:** *Recovery from the disabled state requires one conversion cycle to stabilize.*

**Table 1. Mux Channel Select**

| ADCH4                              | ADCH3 | ADCH2 | ADCH1 | ADCH0 | Input Select                    |
|------------------------------------|-------|-------|-------|-------|---------------------------------|
| 0                                  | 0     | 0     | 0     | 0     | PTB0/ATD0                       |
| 0                                  | 0     | 0     | 0     | 1     | PTB1/ATD1                       |
| 0                                  | 0     | 0     | 1     | 0     | PTB2/ATD2                       |
| 0                                  | 0     | 0     | 1     | 1     | PTB3/ATD3                       |
| 0                                  | 0     | 1     | 0     | 0     | PTB4/ATD4                       |
| 0                                  | 0     | 1     | 0     | 1     | PTB5/ATD5                       |
| 0                                  | 0     | 1     | 1     | 0     | PTB6/ATD6                       |
| 0                                  | 0     | 1     | 1     | 1     | PTB7/ATD7                       |
| 0                                  | 1     | 0     | 0     | 0     | PTD0/ATD8                       |
| 0                                  | 1     | 0     | 0     | 1     | PTD1/ATD9                       |
| 0                                  | 1     | 0     | 1     | 0     | PTD2/ATD10                      |
| 0                                  | 1     | 0     | 1     | 1     | PTD3/ATD11                      |
| 0                                  | 1     | 1     | 0     | 0     | PTD4/TBLCK/ATD12                |
| 0                                  | 1     | 1     | 0     | 1     | PTD5/ATD13                      |
| 0                                  | 1     | 1     | 1     | 0     | PTD6/TACLK/ATD14                |
| Range 01111 (\$0F) to 11010 (\$1A) |       |       |       |       | Unused (see Note 1)             |
|                                    |       |       |       |       | Unused (see Note 1)             |
| 1                                  | 1     | 0     | 1     | 1     | Reserved                        |
| 1                                  | 1     | 1     | 0     | 0     | Unused(see Note 1)              |
| 1                                  | 1     | 1     | 0     | 1     | $V_{REFH}$<br>(see Note 2)      |
| 1                                  | 1     | 1     | 1     | 0     | $V_{SSA}/V_{REFL}$ (see Note 2) |
| 1                                  | 1     | 1     | 1     | 1     | [ADC power off]                 |

**NOTES:**

1. If any unused channels are selected, the resulting ADC conversion will be unknown.
2. The voltage levels supplied from internal reference nodes as specified in the table are used to verify the operation of the ADC converter both in production test and for user applications.

## Analog-to-Digital Converter (ADC-15)

**ADC Data Register** One 8-bit result register is provided. This register is updated each time an ADC conversion completes.

Address: \$0039

|        | Bit 7                     | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
|--------|---------------------------|-----|-----|-----|-----|-----|-----|-------|
| Read:  | AD7                       | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0   |
| Write: | R                         | R   | R   | R   | R   | R   | R   | R     |
| Reset: | Indeterminate after Reset |     |     |     |     |     |     |       |

R = Reserved

**Figure 3. ADC Data Register (ADR)**

**ADC Input Clock Register** This register selects the clock frequency for the ADC.

Address: \$003A

|        | Bit 7 | 6     | 5     | 4      | 3 | 2 | 1 | Bit 0 |
|--------|-------|-------|-------|--------|---|---|---|-------|
| Read:  | ADIV2 | ADIV1 | ADIV0 | ADICLK | 0 | 0 | 0 | 0     |
| Write: |       |       |       |        | R | R | R | R     |
| Reset: | 0     | 0     | 0     | 0      | 0 | 0 | 0 | 0     |

R = Reserved

**Figure 4. ADC Input Clock Register (ADICLK)**

### ADIV2–ADIV0 — ADC Clock Prescaler Bits

ADIV2, ADIV1, and ADIV0 form a 3-bit field which selects the divide ratio used by the ADC to generate the internal ADC clock. [Table 2](#) shows the available clock configurations. The ADC clock should be set to approximately 1 MHz.



**Table 2. ADC Clock Divide Ratio**

| ADIV2 | ADIV1 | ADIV0 | ADC Clock Rate       |
|-------|-------|-------|----------------------|
| 0     | 0     | 0     | ADC Input Clock / 1  |
| 0     | 0     | 1     | ADC Input Clock / 2  |
| 0     | 1     | 0     | ADC Input Clock / 4  |
| 0     | 1     | 1     | ADC Input Clock / 8  |
| 1     | X     | X     | ADC Input Clock / 16 |

X = don't care

#### ADICLK — ADC Input Clock Register Bit

ADICLK selects either bus clock or CGMXCLK as the input clock source to generate the internal ADC clock. Reset selects CGMXCLK as the ADC clock source.

If the external clock (CGMXCLK) is equal to or greater than 1 MHz, CGMXCLK can be used as the clock source for the ADC. If CGMXCLK is less than 1 MHz, use the PLL-generated bus clock as the clock source. As long as the internal ADC clock is at approximately 1 MHz, correct operation can be guaranteed. See [ADC Characteristics](#) on page 439.

1 = Internal bus clock

0 = External clock (CGMXCLK)

$$1 \text{ MHz} = \frac{f_{\text{XCLK or Bus Frequency}}}{\text{ADIV}[2:0]}$$

**NOTE:** *During the conversion process, changing the ADC clock will result in an incorrect conversion.*

## Analog-to-Digital Converter (ADC-15)

# Specifications

---

---

## Contents

|  |     |
|--|-----|
| Electrical Specifications .....                                | 434 |
| Maximum Ratings .....  | 434 |
| Functional Operating Range .....                               | 435 |
| Thermal Characteristics .....                                  | 435 |
| 5.0 Volt DC Electrical Characteristics .....                   | 436 |
| ADC Characteristics .....                                      | 439 |
| 5.0 Vdc ± 0.5 V Serial Peripheral Interface (SPI) Timing ..... | 440 |
| CGM Operating Conditions .....                                 | 443 |
| CGM Component Information .....                                | 443 |
| CGM Acquisition/Lock Time Information .....                    | 444 |
| Timer Module Characteristics .....                             | 444 |
| Memory Characteristics .....                                   | 445 |
| Mechanical Specifications .....                                | 447 |
| 64-Pin Quad Flat Pack (QFP) .....                              | 447 |

## Specifications

### Electrical Specifications

**Maximum Ratings** Maximum ratings are the extreme limits to which the MCU can be exposed without permanently damaging it.

**NOTE:** *This device is not guaranteed to operate properly at the maximum ratings. Refer to [5.0 Volt DC Electrical Characteristics](#) on page 436 for guaranteed operating conditions.*

| Rating   | Symbol     | Value                            | Unit |
|--|------------|----------------------------------|------|
| Supply Voltage   | $V_{DD}$   | -0.3 to +6.0                     | V    |
| Input Voltage  | $V_{IN}$   | $V_{SS} - 0.3$ to $V_{DD} + 0.3$ | V    |
| Maximum Current Per Pin<br>Excluding $V_{DD}$ and $V_{SS}$ | I          | $\pm 25$                         | mA   |
| Storage Temperature  | $T_{STG}$  | -55 to +150                      | °C   |
| Maximum Current out of $V_{SS}$                            | $I_{MVSS}$ | 100                              | mA   |
| Maximum Current into $V_{DD}$                              | $I_{MVDD}$ | 100                              | mA   |
| Reset $\overline{IRQ}$ Input Voltage                       | $V_{HI}$   | $V_{DD} + 4$                     | V    |

NOTE: Voltages are referenced to  $V_{SS}$ .

**NOTE:** *This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. For proper operation, it is recommended that  $V_{IN}$  and  $V_{OUT}$  be constrained to the range  $V_{SS} \leq (V_{IN} \text{ or } V_{OUT}) \leq V_{DD}$ . Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either  $V_{SS}$  or  $V_{DD}$ ).*

## Functional Operating Range

| Rating                                  | Symbol   | Value               | Unit |
|---|----------|---------------------|------|
| Operating Temperature Range<br>(Note 1) | $T_A$    | -40 to $T_{A(MAX)}$ | °C   |
| Operating Voltage Range                 | $V_{DD}$ | $5.0 \pm 0.5v$      | V    |

Note 1:  $T_{A (MAX)} = 125^{\circ}C$  for part suffix MFU  
 $T_{A (MAX)} = 105^{\circ}C$  for part suffix VFU  
 $T_{A (MAX)} = 85^{\circ}C$  for part suffix CFU

**NOTE:** For applications which use the LVI, Motorola guarantee the functionality of the device down to the LVI trip point ( $V_{LVI}$ ) within the constraints outlined in [Low-Voltage Inhibit \(LVI\)](#).

## Thermal Characteristics

| Characteristic                      | Symbol        | Value  | Unit |
|-------------------------------------|---------------|--|------|
| Thermal Resistance<br>QFP (64 Pins) | $\theta_{JA}$ | 70   | °C/W |
| I/O Pin Power Dissipation           | $P_{I/O}$     | User Determined  | W    |
| Power Dissipation (see Note 1)      | $P_D$         | $P_D = (I_{DD} \times V_{DD}) + P_{I/O}$<br>$= K / (T_J + 273^{\circ}C)$ | W    |
| Constant (see Note 2)               | K             | $P_D \times (T_A + 273^{\circ}C)$<br>$+ (P_D^2 \times \theta_{JA})$      | W/°C |
| Average Junction Temperature        | $T_J$         | $T_A = P_D \times \theta_{JA}$   | °C   |

**NOTES:**

- Power dissipation is a function of temperature.
- K is a constant unique to the device. K can be determined from a known  $T_A$  and measured  $P_D$ . With this value of K,  $P_D$  and  $T_J$  can be determined for any value of  $T_A$ .

## Specifications

### 5.0 Volt DC Electrical Characteristics

| Characteristic  | Symbol                | Min                              | Max                                 | Unit   |
|---|-----------------------|----------------------------------|-------------------------------------|--|
| Output High Voltage<br>( $I_{LOAD} = -2.0$ mA) All Ports<br>( $I_{LOAD} = -5.0$ mA) All Ports   | $V_{OH}$              | $V_{DD} - 0.8$<br>$V_{DD} - 1.5$ | —<br>—                              | V<br>V   |
| Total Source Current  | $IOH_{tot}$           | —                                | 15                                  | mA   |
| Output Low Voltage<br>( $I_{LOAD} = 1.6$ mA) All Ports<br>( $I_{LOAD} = 10.0$ mA) All Ports   | $V_{OL}$              | —<br>—                           | 0.4<br>1.5                          | V<br>V   |
| Total Sink Current  | $IOL_{tot}$           | —                                | 10                                  | mA   |
| Input High Voltage<br>All Ports, $\overline{IRQS}$ , RESET, OSC1  | $V_{IH}$              | $0.7 \times V_{DD}$              | $V_{DD}$                            | V  |
| Input Low Voltage<br>All Ports, $\overline{IRQS}$ , RESET, OSC1   | $V_{IL}$              | $V_{SS}$                         | $0.3 \times V_{DD}$                 | V  |
| $V_{DD}$ Supply Current<br>Run (see Note 2, 9)<br>Wait (see Note 3, 9)<br>Stop (see Note 4)<br>LVI enabled, $T_A = 25$ °C<br>LVI disabled, $T_A = 25$ °C<br>LVI enabled, $-40$ °C to $+125$ °C<br>LVI disabled, $-40$ °C to $+125$ °C | $I_{DD}$              | —<br>—<br>—<br>—<br>—<br>—       | 35<br>20<br>400<br>50<br>500<br>100 | mA<br>mA<br>$\mu$ A<br>$\mu$ A<br>$\mu$ A<br>$\mu$ A |
| I/O Ports Hi-Z Leakage Current  | $I_L$                 | -1                               | +1                                  | $\mu$ A  |
| Input Current   | $I_{IN}$              | -1                               | +1                                  | $\mu$ A  |
| Capacitance<br>Ports (As Input or Output)   | $C_{OUT}$<br>$C_{IN}$ | —<br>—                           | 12<br>8                             | pF   |
| Low-Voltage Reset Inhibit (trip)<br>(recover)   | $V_{LVI}$             | 4.0                              | 4.4                                 | V  |
| POR ReArm Voltage (see Note 5)  | $V_{POR}$             | 0                                | 200                                 | mV   |
| POR Reset Voltage (see Note 6)  | $V_{PORRST}$          | 0                                | 800                                 | mV   |
| POR Rise Time Ramp Rate (see Note 7)  | $R_{POR}$             | 0.02                             | —                                   | V/ms   |
| High COP Disable Voltage (see Note 8)   | $V_{HI}$              | $V_{DD} + 2$                     | $V_{DD} + 4$                        | V  |
| Monitor Mode Entry Voltage on $\overline{IRQ}$ (note 10)  | $V_{HI}$              | $V_{DD} + 2$                     | $V_{DD} + 4$                        | V  |

---

NOTES:

1.  $V_{DD} = 5.0 \text{ Vdc} \pm 10\%$ ,  $V_{SS} = 0 \text{ Vdc}$ ,  $T_A = -40 \text{ }^\circ\text{C}$  to  $+T_{A(\text{MAX})}$ , unless otherwise noted.
2. Run (Operating)  $I_{DD}$  measured using external square wave clock source ( $f_{BUS} = 8.4 \text{ MHz}$ ). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs.  $C_L = 20 \text{ pF}$  on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run  $I_{DD}$ . Measured with all modules enabled.
3. Wait  $I_{DD}$  measured using external square wave clock source ( $f_{BUS} = 8.4 \text{ MHz}$ ). All inputs 0.2 Vdc from rail. No dc loads. Less than 100 pF on all outputs,  $C_L = 20 \text{ pF}$  on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects wait  $I_{DD}$ . Measured with all modules enabled.
4. Stop  $I_{DD}$  measured with  $\text{OSC1} = V_{SS}$ .
5. Maximum is highest voltage that POR is guaranteed.
6. Maximum is highest voltage that POR is possible.
7. If minimum  $V_{DD}$  is not reached before the internal POR reset is released, RST must be driven low externally until minimum  $V_{DD}$  is reached.
8. See [COP Module During Break Interrupts on page 184](#).
9. Although  $I_{DD}$  is proportional to bus frequency, a current of several mA is present even at very low frequencies.
10. See Monitor Mode description within COP section.  $V_{HI}$  applied to  $\overline{\text{IRQ}}$  or  $\overline{\text{RST}}$ .

## Specifications

### Control Timing

| Characteristic  | Symbol                            | Min         | Max    | Unit      |
|---|-----------------------------------|-------------|--------|-----------|
| Bus Operating Frequency (4.5–5.5 V — $V_{DD}$ Only)   | $f_{BUS}$                         | —           | 8.4    | MHz       |
| $\overline{RESET}$ Pulse Width Low  | $t_{RL}$                          | 1.5         | —      | $t_{cyc}$ |
| $\overline{IRQ}$ Interrupt Pulse Width Low (Edge-Triggered)                                 | $t_{LHI}$                         | 1.5         | —      | $t_{cyc}$ |
| $\overline{IRQ}$ Interrupt Pulse Period   | $t_{LIL}$                         | Note 4      | —      | $t_{cyc}$ |
| 16-Bit Timer (see Note 2)<br>Input Capture Pulse Width (see Note 3)<br>Input Capture Period | $t_{TH}$ , $t_{TL}$<br>$t_{TLTL}$ | 2<br>Note 4 | —<br>— | $t_{cyc}$ |
| MSCAN Wake-up Filter Pulse Width (see Note 5)   | $t_{WUP}$                         | 2           | 5      | $\mu s$   |

**NOTES:**

- $V_{DD} = 5.0 \text{ Vdc} \pm 0.5\text{v}$ ,  $V_{SS} = 0 \text{ Vdc}$ ,  $T_A = -40 \text{ }^\circ\text{C}$  to  $T_{A(MAX)}$ , unless otherwise noted.
- The 2-bit timer prescaler is the limiting factor in determining timer resolution.
- Refer to [Table 2 . Mode, Edge, and Level Selection on page 415](#), and supporting note.
- The minimum period  $t_{TLTL}$  or  $t_{LIL}$  should not be less than the number of cycles it takes to execute the capture interrupt service routine plus TBD  $t_{cyc}$ .
- The minimum pulse width to wake up the MSCAN module is guaranteed by design but not tested.



## ADC Characteristics

| Characteristic  | Min                         | Max        | Unit             | Comments               |
|---|-----------------------------|------------|------------------|------------------------|
| Resolution  | 8                           | 8          | Bits             |                        |
| Absolute Accuracy<br>( $V_{REFL} = 0\text{ V}$ , $V_{DDA}/V_{DDAREF} = V_{REFH} = 5\text{ V} \pm 0.5\text{v}$ ) | -1                          | +1         | LSB              | Includes Quantization  |
| Conversion Range (see Note 1)   | $V_{REFL}$                  | $V_{REFH}$ | V                | $V_{REFL} = V_{SSA}$   |
| Power-Up Time   | 16                          | 17         | $\mu\text{s}$    | Conversion Time Period |
| Input Leakage (see Note 3)<br>Ports B and D   | -1                          | +1         | $\mu\text{A}$    |                        |
| Conversion Time   | 16                          | 17         | ADC Clock Cycles | Includes Sampling Time |
| Monotonicity  | Inherent within Total Error |            |                  |                        |
| Zero Input Reading  | 00                          | 01         | Hex              | $V_{IN} = V_{REFL}$    |
| Full-Scale Reading  | FE                          | FF         | Hex              | $V_{IN} = V_{REFH}$    |
| Sample Time (see Note 2)  | 5                           | —          | ADC Clock Cycles |                        |
| Input Capacitance   | —                           | 8          | pF               | Not Tested             |
| ADC Internal Clock  | 500 k                       | 1.048 M    | Hz               | Tested Only at 1 MHz   |
| Analog Input Voltage  | $V_{REFL}$                  | $V_{REFH}$ | V                |                        |

**NOTES:**

1.  $V_{DD} = 5.0\text{ Vdc} \pm 0.5\text{v}$ ,  $V_{SS} = 0\text{ Vdc}$ ,  $V_{DDA}/V_{DDAREF} = 5.0\text{ Vdc} \pm 0.5\text{v}$ ,  $V_{SSA} = 0\text{ Vdc}$ ,  $V_{REFH} = 5.0\text{ Vdc} \pm 0.5\text{v}$
2. Source impedances greater than 10 k $\Omega$  adversely affect internal RC charging time during input sampling.
3. The external system error caused by input leakage current is approximately equal to the product of R source and input current.

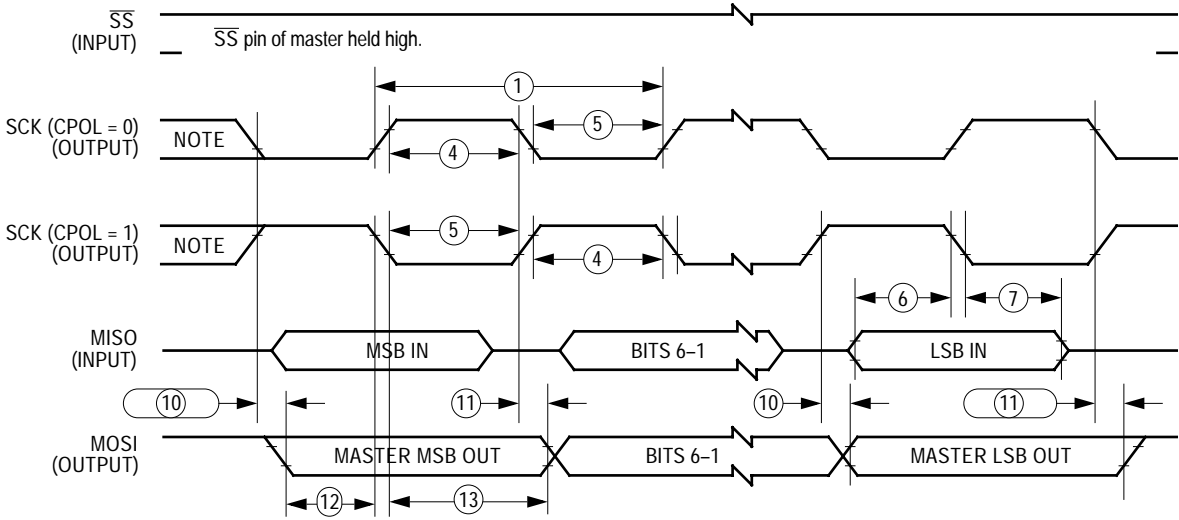
## Specifications

### 5.0 V<sub>dc</sub> ± 0.5 V Serial Peripheral Interface (SPI) Timing

| Num | Characteristic  | Symbol                           | Min                 | Max                      | Unit      |
|-----|---|----------------------------------|---------------------|--------------------------|-----------|
|     | Operating Frequency (see Note 3)<br>Master<br>Slave                 | $f_{BUS(M)}$<br>$f_{BUS(S)}$     | $f_{BUS}/128$<br>dc | $f_{BUS}/2$<br>$f_{BUS}$ | MHz       |
| 1   | Cycle Time<br>Master<br>Slave                                       | $t_{cyc(M)}$<br>$t_{cyc(S)}$     | 2<br>1              | 128<br>—                 | $t_{cyc}$ |
| 2   | Enable Lead Time  | $t_{Lead}$                       | 15                  | —                        | ns        |
| 3   | Enable Lag Time   | $t_{Lag}$                        | 15                  | —                        | ns        |
| 4   | Clock (SCK) High Time<br>Master<br>Slave                            | $t_{W(SCKH)M}$<br>$t_{W(SCKH)S}$ | 100<br>50           | —<br>—                   | ns        |
| 5   | Clock (SCK) Low Time<br>Master<br>Slave                             | $t_{W(SCKL)M}$<br>$t_{W(SCKL)S}$ | 100<br>50           | —<br>—                   | ns        |
| 6   | Data Setup Time (Inputs)<br>Master<br>Slave                         | $t_{SU(M)}$<br>$t_{SU(S)}$       | 45<br>5             | —<br>—                   | ns        |
| 7   | Data Hold Time (Inputs)<br>Master<br>Slave                          | $t_{H(M)}$<br>$t_{H(S)}$         | 0<br>15             | —<br>—                   | ns        |
| 8   | Access Time, Slave (see Note 4)<br>CPHA = 0<br>CPHA = 1             | $t_{A(CP0)}$<br>$t_{A(CP1)}$     | 0<br>0              | 40<br>20                 | ns        |
| 9   | Slave Disable Time (Hold Time to High-Impedance State)              | $t_{DIS}$                        | —                   | 25                       | ns        |
| 10  | Enable Edge Lead Time to Data Valid (see Note 6)<br>Master<br>Slave | $t_{EV(M)}$<br>$t_{EV(S)}$       | —<br>—              | 10<br>40                 | ns        |
| 11  | Data Hold Time (Outputs, after Enable Edge)<br>Master<br>Slave      | $t_{HO(M)}$<br>$t_{HO(S)}$       | 0<br>5              | —<br>—                   | ns        |
| 12  | Data Valid<br>Master (Before Capture Edge)                          | $t_{V(M)}$                       | 90                  | —                        | ns        |
| 13  | Data Hold Time (Outputs)<br>Master (Before Capture Edge)            | $t_{HO(M)}$                      | 100                 | —                        | ns        |

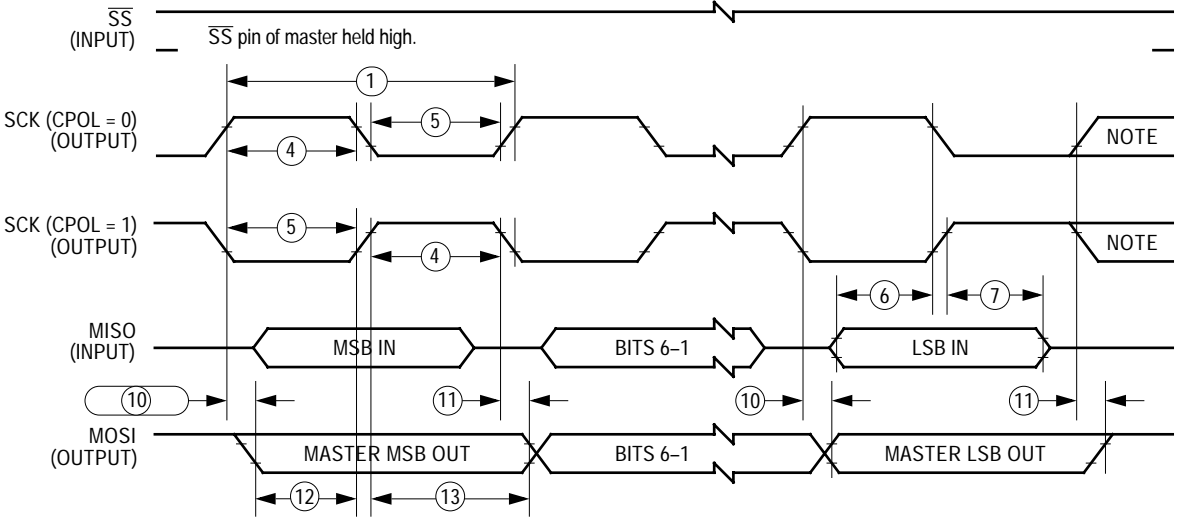
#### NOTES:

- All timing is shown with respect to 30% V<sub>DD</sub> and 70% V<sub>DD</sub>, unless otherwise noted; assumes 100 pF load on all SPI pins.
- Item numbers refer to dimensions in [Figure 1](#) and [Figure 2](#).
- $f_{BUS}$  = the currently active bus frequency for the microcontroller.
- Time to data active from high-impedance state.
- With 100 pF on all SPI pins



NOTE: This first clock edge is generated internally, but is not seen at the SCK pin.

**a) SPI Master Timing (CPHA = 0)**

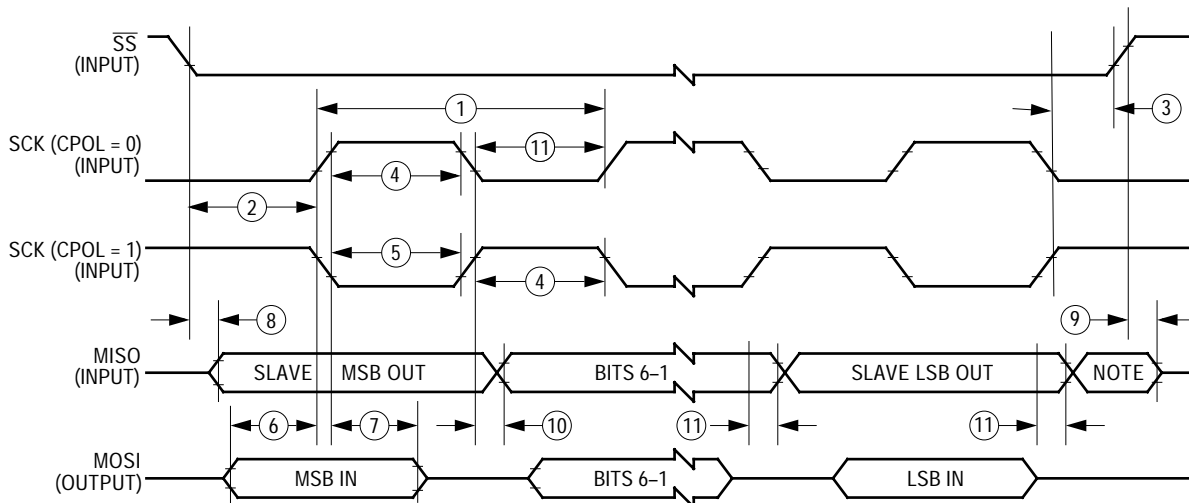


NOTE: This last clock edge is generated internally, but is not seen at the SCK pin.

**b) SPI Master Timing (CPHA = 1)**

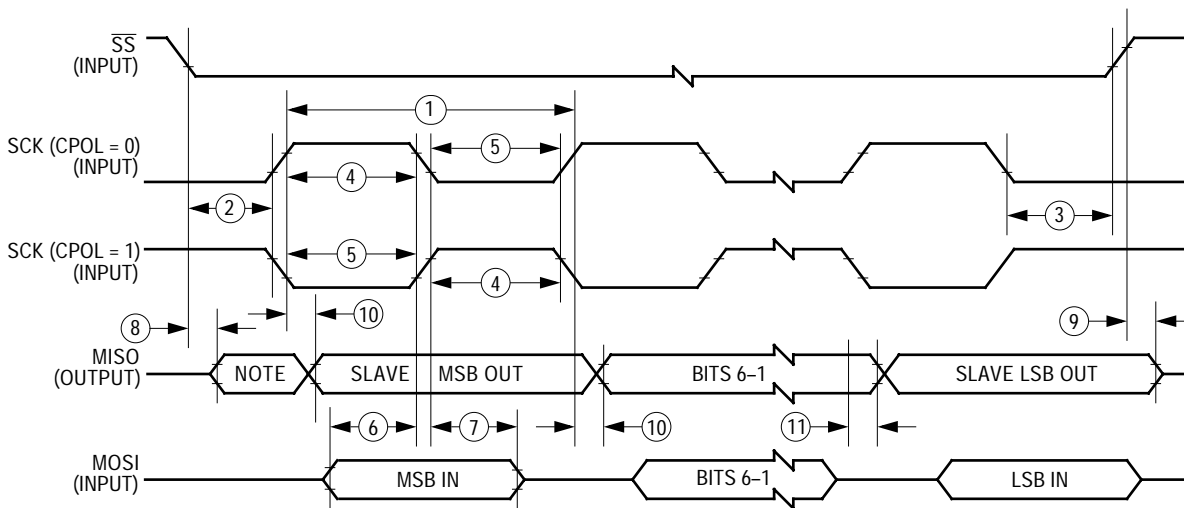
**Figure 1. SPI Master Timing Diagram**

# Specifications



NOTE: Not defined but normally MSB of character just received

## a) SPI Slave Timing (CPHA = 0)



NOTE: Not defined but normally LSB of character previously transmitted

## b) SPI Slave Timing (CPHA = 1)

Figure 2. SPI Slave Timing Diagram

## CGM Operating Conditions

| Characteristic                      | Symbol     | Min    | Typ        | Max   | Comments                     |
|-------------------------------------|------------|--------|------------|-------|------------------------------|
| Operating Voltage                   | $V_{DD}$   | 4.5 V  | —          | 5.5 V |                              |
| Crystal Reference Frequency         | $f_{RCLK}$ | 1      | 4.9152 MHz | 16    |                              |
| Module Crystal Reference Frequency  | $f_{XCLK}$ | —      | 4.9152 MHz | —     | Same Frequency as $f_{RCLK}$ |
| Range Nom. Multiplier (MHZ)         | $f_{NOM}$  | —      | 4.9152     | —     | 4.5–5.5 V, $V_{DD}$ only     |
| VCO Center-of-Range Frequency (MHZ) | $f_{VRS}$  | 4.9152 | —          | (1)   | 4.5–5.5 V, $V_{DD}$ only     |
| VCO Operating Frequency (MHZ)       | $f_{VCLK}$ | 4.9152 | —          | 32.0  |                              |

1.  $f_{VRS}$  is a nominal value described and mandated as an example in the CGM module section for the desired VCO operating frequency,  $f_{VCLK}$ .

## CGM Component Information

| Description                      | Symbol     | Min | Typ                                    | Max | Comments   |
|----------------------------------|------------|-----|--|-----|--|
| Crystal Load Capacitance         | $C_L$      | —   | —                                      | —   | Consult Crystal Manufacturer's Data  |
| Crystal Fixed Capacitance        | C1         | —   | 2 x $C_L$                              | —   | Consult Crystal Manufacturer's Data  |
| Crystal Tuning Capacitance       | C2         | —   | 2 x $C_L$                              | —   | Consult Crystal Manufacturer's Data  |
| Filter Capacitor Multiply Factor | $C_{FACT}$ |     | 0.0154                                 |     | F/s V  |
| Filter Capacitor                 | $C_F$      | —   | $C_{FACT} \times (V_{DDA} / f_{XCLK})$ | —   | See <a href="#">External Filter Capacitor Pin (CGMXFC)</a> on page 139   |
| Bypass Capacitor                 | $C_{BYP}$  | —   | 0.1 $\mu$ F                            | —   | CBYP must provide low AC impedance from $f = f_{XCLK}/100$ to $100 \times f_{VCLK}$ , so series resistance must be considered. |

## Specifications

### CGM Acquisition/Lock Time Information

| Description  | Symbol     | Min                  | Typ  | Max   | Notes                      |
|--|------------|----------------------|--|---|----------------------------|
| Manual Mode Time to Stable                               | $t_{ACQ}$  | —                    | $(8 \times V_{DDA}) / (f_{XCLK} \times K_{ACQ})$ | —   | If $C_F$ Chosen Correctly  |
| Manual Stable to Lock Time                               | $t_{AL}$   | —                    | $(4 \times V_{DDA}) / (f_{XCLK} \times K_{TRK})$ | —   | If $C_F$ Chosen Correctly  |
| Manual Acquisition Time                                  | $t_{LOCK}$ | —                    | $t_{ACQ} + t_{AL}$                               | —   |                            |
| Tracking Mode Entry Frequency Tolerance                  | $D_{TRK}$  | 0                    | —  | $\pm 3.6\%$                                   |                            |
| Acquisition Mode Entry Frequency Tolerance               | $D_{UNT}$  | $\pm 6.3\%$          | —  | $\pm 7.2\%$                                   |                            |
| LOCK Entry Freq. Tolerance                               | $D_{LOCK}$ | 0                    | —  | $\pm 0.9\%$                                   |                            |
| LOCK Exit Freq. Tolerance                                | $D_{UNL}$  | $\pm 0.9\%$          | —  | $\pm 1.8\%$                                   |                            |
| Reference Cycles per Acquisition Mode Measurement        | $n_{ACQ}$  | —                    | 32   | —   |                            |
| Reference Cycles per Tracking Mode Measurement           | $n_{TRK}$  | —                    | 128  | —   |                            |
| Automatic Mode Time to Stable                            | $t_{ACQ}$  | $n_{ACQ} / f_{XCLK}$ | $(8 \times V_{DDA}) / (f_{XCLK} \times K_{ACQ})$ | —   | If $C_F$ Chosen Correctly  |
| Automatic Stable to Lock Time                            | $t_{AL}$   | $n_{TRK} / f_{XCLK}$ | $(4 \times V_{DDA}) / (f_{XCLK} \times K_{TRK})$ | —   | If $C_F$ Chosen Correctly  |
| Automatic Lock Time                                      | $t_{LOCK}$ | —                    | $t_{ACQ} + t_{AL}$                               | —   |                            |
| PLL Jitter, Deviation of Average Bus Frequency over 2 ms |            | 0                    | —  | $\pm (f_{CRYS}) \times (.025\%) \times (N/4)$ | N = VCO Freq. Mult. (GBNT) |

#### NOTES:

1. GBNT guaranteed but not tested
2.  $V_{DD} = 5.0 \text{ Vdc} \pm 0.5\text{v}$ ,  $V_{SS} = 0 \text{ Vdc}$ ,  $T_A = -40 \text{ }^\circ\text{C}$  to  $T_{A(\text{MAX})}$ , unless otherwise noted.

### Timer Module Characteristics

| Characteristic            | Symbol                | Min              | Max | Unit |
|---------------------------|-----------------------|------------------|-----|------|
| Input Capture Pulse Width | $t_{TIH}$ , $t_{TIL}$ | 125              | —   | ns   |
| Input Clock Pulse Width   | $t_{TCH}$ , $t_{TCL}$ | $(1/f_{OP}) + 5$ | —   | ns   |

## Memory Characteristics

| Characteristic  | Symbol               | Min    | Max  | Unit              |
|---|----------------------|--------|------|-------------------|
| RAM Data Retention Voltage  | $V_{RDR}$            | 0.7    | —    | V                 |
| EEPROM Programming Time per Byte  | $t_{EEPGM}$          | 10     | —    | ms                |
| EEPROM Erasing Time per Byte  | $t_{EEBYTE}$         | 10     | —    | ms                |
| EEPROM Erasing Time per Block   | $t_{EEBLOCK}$        | 10     | —    | ms                |
| EEPROM Erasing Time per Bulk  | $t_{EEBULK}$         | 10     | —    | ms                |
| EEPROM Programming Voltage Discharge Period   | $t_{EEFPV}$          | 100    | —    | $\mu$ s           |
| EEPROM Enable Recovery Time   | $t_{EEOFF}$          | 600    | —    | $\mu$ s           |
| EEPROM Stop Recovery Time   | $t_{EESTOP}$         | 600    | —    | $\mu$ s           |
| EEPROM Write/Erase Cycles<br>@ 10 ms Write Time +125 °C   |                      | 10,000 | —    | Cycles            |
| EEPROM Data Retention<br>After 10,000 Write/Erase Cycles  |                      | 10     | —    | Years             |
| Flash Pages per Row   |                      | 8      | 8    | pages             |
| Flash Bytes per Page  |                      | 8      | 8    | bytes             |
| Flash Read Bus Clock Frequency  | $f_{READ}^{(1)}$     | 32K    | 8.4M | Hz                |
| FLASH Charge Pump Clock Frequency<br>(see <a href="#">FLASH Charge Pump Frequency Control</a> on page 41) | $f_{PUMP}^{(2)}$     | 1.8    | 2.3  | MHz               |
| FLASH Block/Bulk Erase Time   | $t_{ERASE}$          | 100    | 110  | ms                |
| FLASH High Voltage Kill Time  | $t_{KILL}$           | 200    | —    | $\mu$ s           |
| FLASH Return to Read Time   | $t_{HVD}$            | 50     | —    | $\mu$ s           |
| FLASH Page Program Pulses <sup>(10)</sup>   | $fls_{PULSES}^{(3)}$ | —      | 84   | pulses            |
| FLASH Page Program Step Size <sup>(10)</sup>  | $t_{STEP}^{(4)}$     | 0.8    | 1.2  | ms                |
| FLASH Cumulative Program Time per Row<br>Between Erase Cycles   | $t_{ROW}^{(5)}$      | —      | 8    | pages             |
| FLASH HVEN Low to MARGIN High Time  | $t_{HVTV}$           | 50     | —    | $\mu$ s           |
| FLASH MARGIN High to PGM Low Time   | $t_{VTP}$            | 150    | —    | $\mu$ s           |
| FLASH Return to Read after Margin Read Time   | $t_{RECOVERY}^{(6)}$ | 500    | —    | dummy read cycles |
| FLASH Row Erase Endurance <sup>(7)</sup> @ +125 °C  |                      | 100    | —    | cycles            |
| FLASH Row Program Endurance <sup>(8)</sup> @ +125 °C  |                      | 100    | —    | cycles            |
| FLASH Data Retention Time <sup>(9)</sup>  |                      | 10     | —    | years             |

1.  $f_{READ}$  is defined as the frequency range for which the FLASH memory can be read.

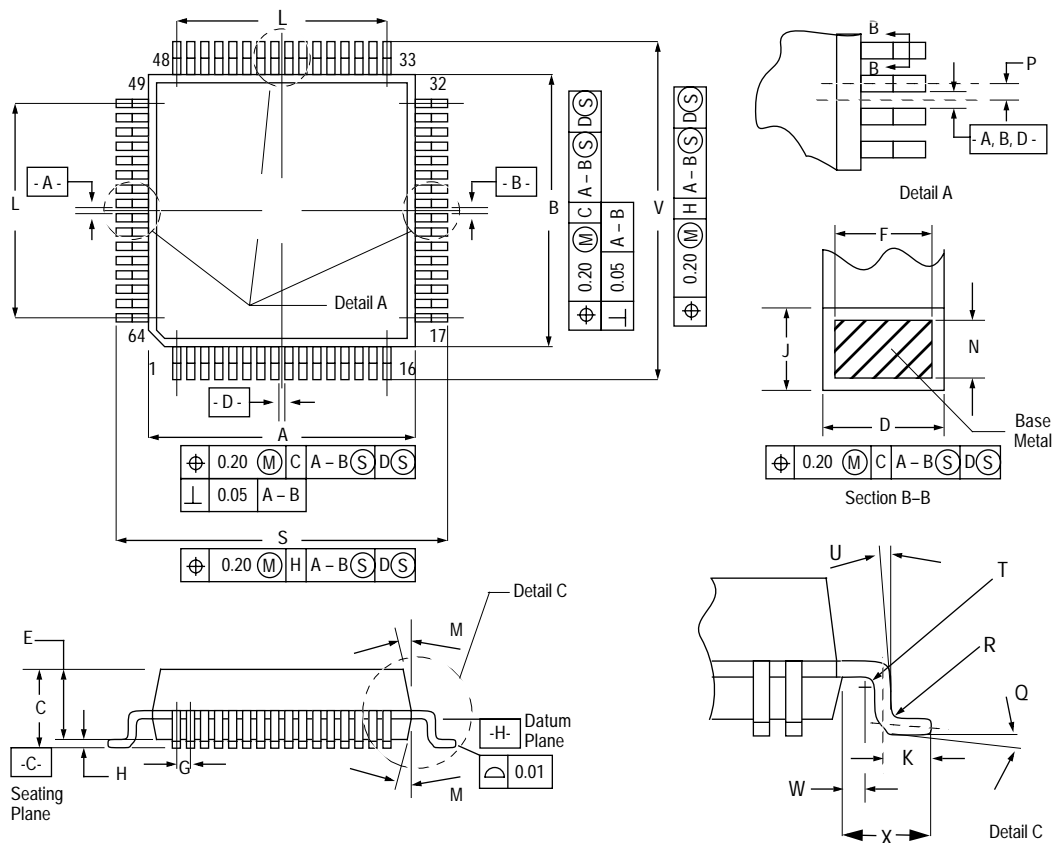
## Specifications

2.  $f_{\text{PUMP}}$  is defined as the charge pump clock frequency required for Program, Erase, and Margin Read operations.
3.  $\text{flS}_{\text{PULSES}}$  is defined as the number of pulses used during FLASH programming required by the Smart Program Algorithm.
4.  $t_{\text{STEP}}$  is defined as the amount of time during one page program cycle that HVEN is held high.
5.  $t_{\text{ROW}}$  is defined as the cumulative time a row can see the program voltage before the row must be erased before further programming.
6.  $t_{\text{RECOVERY}}$  is defined as the minimum number of dummy reads of any FLASH location before accurate data is read from the memory after Margin Read mode is used.
7. The minimum row erase endurance value specifies each row of the FLASH memory is guaranteed to work for at least this many erase cycles.
8. The minimum row program endurance value specifies each row of the FLASH memory is guaranteed to work for at least this many program cycles.
9. The flash is guaranteed to retain data over the entire specified temperature range for at least the minimum time specified.
10. Maximum program time must not exceed 100ms as the product of pulses and step size. Always use the programming algorithm in the flash sections to ensure fastest flash programming.



## Mechanical Specifications

### 64-Pin Quad Flat Pack (QFP)



| Dim. | Min.      | Max.  | Notes   | Dim. | Min.     | Max.  |
|------|-----------|-------|---|------|----------|-------|
| A    | 13.90     | 14.10 | 1. Dimensions and tolerancing per ANSI Y 14.5M, 1982.<br>2. All dimensions in mm.<br>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.<br>4. Datums A-B and -D to be determined at Datum Plane -H-.<br>5. Dimensions S and V to be determined at seating plane -C-.<br>6. Dimensions A and B do not include mould protrusion. Allowable mould protrusion is 0.25mm per side. Dimensions A and B do include mould mismatch and are determined at Datum Plane -H-.<br>7. Dimension D does not include dambar protrusion. Allowable dambar protrusion shall be 0.08 total in excess of the D dimension at maximum material condition. Dambar cannot be located on the lower radius or the foot. | M    | 5°       | 10°   |
| B    | 13.90     | 14.10 |   | N    | 0.13     | 0.17  |
| C    | 2.15      | 2.45  |   | P    | 0.40 BSC |       |
| D    | 0.30      | 0.45  |   | Q    | 0°       | 7°    |
| E    | 2.00      | 2.40  |   | R    | 0.13     | 0.30  |
| F    | 0.30      | 0.40  |   | S    | 16.95    | 17.45 |
| G    | 0.80 BSC  |       |   | T    | 0.13     | —     |
| H    | —         | 0.25  |   | U    | 0°       | —     |
| J    | 0.13      | 0.23  |   | V    | 16.95    | 17.45 |
| K    | 0.65      | 0.95  |   | W    | 0.35     | 0.45  |
| L    | 12.00 REF |       |   | X    | 1.6 REF  |       |

## Specifications

## Appendix A: Future EEPROM Registers

**NOTE:** *The following are proposed register addresses. Writing to them in current software will have no effect.*

---

---

### EEPROM Timebase Divider Control Registers

To program or erase the EEPROM content, the EEPROM control hardware requires a constant timebase of 35 $\mu$ s to drive its internal timer. EEPROM Timebase Divider EEDIV is a clock-divider which divides the selected reference clock source to generate this constant timebase. The reference clock input of the EEDIV is driven by either the CGMXCLK or the system bus clock. The selection of this reference clock is defined by the EEDIVCLK bit in the Configuration Register.

EEPROM Timebase Divider EEDIV are defined by two registers (EEDIVH and EEDIVL) and must be programmed with a proper value before starting any EEPROM erase/program steps. EEDIV registers must be re-programmed when ever its reference clock is changed. The EEDIV value can be either pre-programmed in the EEDIVHNVR and EEDIVLNVR non-volatile memory registers, (which upon reset will load their contents into the EEDIVH and EEDIVL registers,) or programmed directly by software into the EEDIVH and EEDIVL registers at system initialization . The function of the divider is to provide a constant clock source with a period of 35 $\mu$ s (better be within  $\pm$  2ms) to the internal timer and related EEPROM circuits for proper program/erase operations. The recommended frequency range of the reference clock is 250KHz to 16MHz.

## Appendix A: Future EEPROM Registers

### EEDIVH and EEDIVL Registers

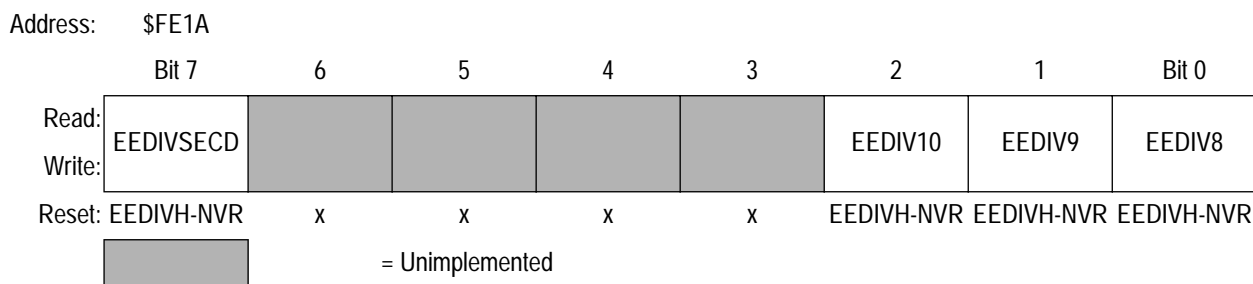
EEDIVH and EEDIVL are used to store the 11-bit EEDIV value which can be programmed by software at system initialization or during runtime if the EEDIVSECD bit in the EEDIVH is not cleared. The EEDIV value is calculated by the following formula:

$$EEDIV = \text{INT}[\text{Reference Frequency (Hz)} \times 35 \times 10^{-6} + 0.5]$$

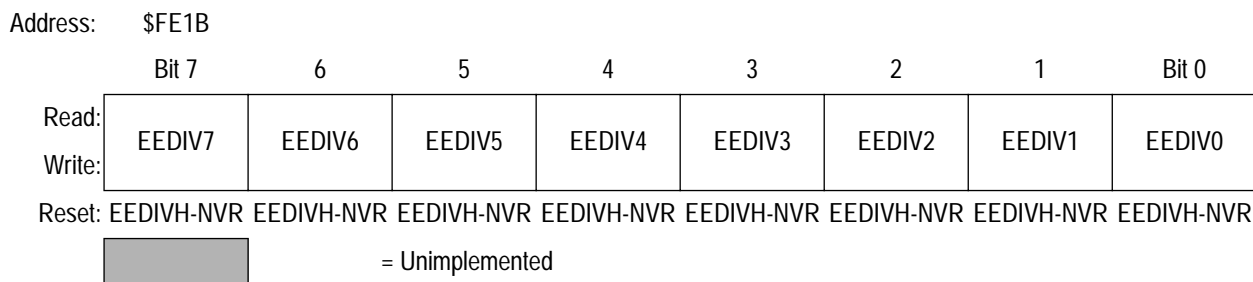
Where the result inside the bracket [ ] is rounded down to the nearest integer value.

For example, if the Reference Frequency is 4.9152MHz, the EEDIV value in the above formula will be 172. To examine the timebase output of the divider, the Reference Frequency is divided by the calculated EEDIV value (172), which equals to 28.577KHz in frequency or 34.99 $\mu$ s in period.

Programming/erasing the EEPROM with an improper EEDIV value may result in data lost and reduce endurance of the EEPROM device.



**Figure 3. EEPROM-2 Divider High Register (EEDIVH)**



**Figure 4. EEPROM-2 Divider Low Register (EEDIVL)**

### EEDIVSECD — EEPROM Divider Security Disable

This bit enables/disables the security feature of the EEDIV registers. When EEDIV security feature is enabled, the state of the registers EEDIVH and EEDIVL are locked (including this EEDIVSECD bit). Also the EEDIVHNVR and EEDIVLNVR non-volatile memory registers are protected from being erased/programmed.

- 1 = EEDIV security feature disabled
- 0 = EEDIV security feature enabled

### EEDIV10–EEDIV0 — EEPROM timebase prescalar.

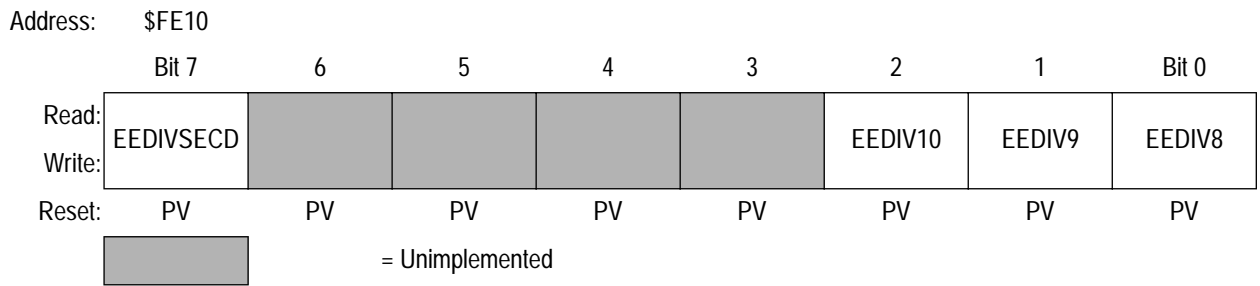
These prescalar bits store the value of EEDIV which is used as the divisor to derive a timebase of 35μs from the selected reference clock source for the EEPROM related internal timer and circuits. EEDIV0–10 are readable at any time. They are writable when EELAT is not set and EEDIVSECD is not cleared.

---

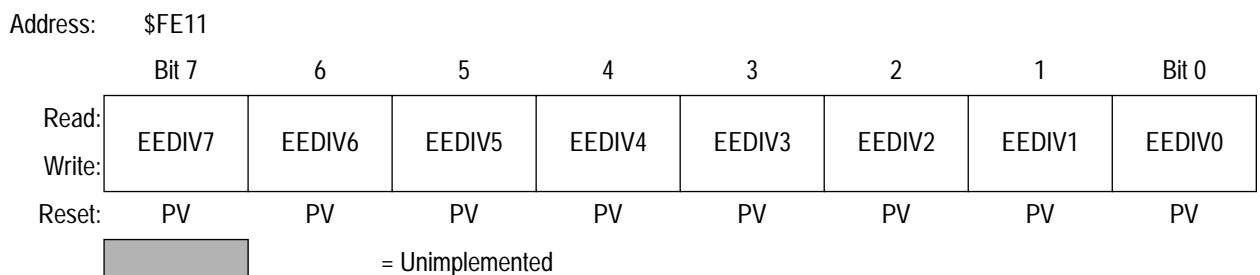


---

## EEDIV Non-volatile Registers



**Figure 5. EEPROM-2 Divider High Non-volatile Register (EEDIVHNVR)**



**Figure 6. EEPROM-2 Divider Low Non-volatile Register (EEDIVLNVR)**

## Appendix A: Future EEPROM Registers

PV = Programmed Value or '1' in the erased state.

The EEPROM Divider non-volatile registers (EEDIVHNVR and EEDIVLNVR) store the reset values of the EEDIV0–10 and EEDIVSECD bits which are non-volatile and are not modified by reset. On reset, these two special registers load the EEDIV0–10 and EEDIVSECD bits into the corresponding volatile EEDIV registers (EEDIVH and EEDIVL).

The EEDIVHNVR and EEDIVLNVR can be programmed/erased like normal EEPROM bytes if the Divider Security Disable bit (EEDIVSECD) in the EEDIVH is not cleared. The new 11-bit EEDIV value in the non-volatile registers (EEDIVHNVR and EEDIVLNVR) together with the EEPROM Divider Security Disable bit (EEDIVSECD) will only be loaded into the EEDIVH & EEDIVL registers with a system reset.

**NOTE:** *Once EEDIVSECD in the EEDIVHNVR is programmed to '0' and after a system reset, the EEDIV security feature is permanently enabled because the EEDIVSECD bit in the EEDIVH is always loaded with a '0' thereafter. Once this security feature is armed, erase and program modes are disabled for EEDIVHNVR and EEDIVLNVR. Modifications to the EEDIVH and EEDIVL registers are also disabled. Therefore, great care should be taken before programming a value into the EEDIVHNVR.*

**A** — See “accumulator (A).”

**accumulator (A)** — An 8-bit general-purpose register in the CPU08. The CPU08 uses the accumulator to hold operands and results of arithmetic and logic operations.

**acquisition mode** — A mode of PLL operation during startup before the PLL locks on a frequency. Also see “tracking mode.”

**address bus** — The set of wires that the CPU or DMA uses to read and write memory locations.

**addressing mode** — The way that the CPU determines the operand address for an instruction. The M68HC08 CPU has 16 addressing modes.

**ALU** — See “arithmetic logic unit (ALU).”

**arithmetic logic unit (ALU)** — The portion of the CPU that contains the logic circuitry to perform arithmetic, logic, and manipulation operations on operands.

**asynchronous** — Refers to logic circuits and operations that are not synchronized by a common reference signal.

**baud rate** — The total number of bits transmitted per unit of time.

**BCD** — See “binary-coded decimal (BCD).”

**binary** — Relating to the base 2 number system.

**binary number system** — The base 2 number system, having two digits, 0 and 1. Binary arithmetic is convenient in digital circuit design because digital circuits have two permissible voltage levels, low and high. The binary digits 0 and 1 can be interpreted to correspond to the two digital voltage levels.

**binary-coded decimal (BCD)** — A notation that uses 4-bit binary numbers to represent the 10 decimal digits and that retains the same positional structure of a decimal number. For example,

234 (decimal) = 0010 0011 0100 (BCD)

## Glossary

**bit** — A binary digit. A bit has a value of either logic 0 or logic 1.

**branch instruction** — An instruction that causes the CPU to continue processing at a memory location other than the next sequential address.

**break module** — A module in the M68HC08 Family. The break module allows software to halt program execution at a programmable point in order to enter a background routine.

**breakpoint** — A number written into the break address registers of the break module. When a number appears on the internal address bus that is the same as the number in the break address registers, the CPU executes the software interrupt instruction (SWI).

**break interrupt** — A software interrupt caused by the appearance on the internal address bus of the same value that is written in the break address registers.

**bus** — A set of wires that transfers logic signals.

**bus clock** — The bus clock is derived from the CGMOUT output from the CGM. The bus clock frequency,  $f_{op}$ , is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

**byte** — A set of eight bits.

**C** — The carry/borrow bit in the condition code register. The CPU08 sets the carry/borrow bit when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some logical operations and data manipulation instructions also clear or set the carry/borrow bit (as in bit test and branch instructions and shifts and rotates).

**CCR** — See “condition code register.”

**central processor unit (CPU)** — The primary functioning unit of any computer system. The CPU controls the execution of instructions.

**CGM** — See “clock generator module (CGM).”

**clear** — To change a bit from logic 1 to logic 0; the opposite of set.

**clock** — A square wave signal used to synchronize events in a computer.



- clock generator module (CGM)** — A module in the M68HC08 Family. The CGM generates a base clock signal from which the system clocks are derived. The CGM may include a crystal oscillator circuit and or phase-locked loop (PLL) circuit.
- comparator** — A device that compares the magnitude of two inputs. A digital comparator defines the equality or relative differences between two binary numbers.
- computer operating properly module (COP)** — A counter module in the M68HC08 Family that resets the MCU if allowed to overflow.
- condition code register (CCR)** — An 8-bit register in the CPU08 that contains the interrupt mask bit and five bits that indicate the results of the instruction just executed.
- control bit** — One bit of a register manipulated by software to control the operation of the module.
- control unit** — One of two major units of the CPU. The control unit contains logic functions that synchronize the machine and direct various operations. The control unit decodes instructions and generates the internal control signals that perform the requested operations. The outputs of the control unit drive the execution unit, which contains the arithmetic logic unit (ALU), CPU registers, and bus interface.
- COP** — See "computer operating properly module (COP)."
- counter clock** — The input clock to the TIM counter. This clock is the output of the TIM prescaler.
- CPU** — See "central processor unit (CPU)."
- CPU08** — The central processor unit of the M68HC08 Family.
- CPU clock** — The CPU clock is derived from the CGMOUT output from the CGM. The CPU clock frequency is equal to the frequency of the oscillator output, CGMXCLK, divided by four.
- CPU cycles** — A CPU cycle is one period of the internal bus clock, normally derived by dividing a crystal oscillator source by two or more so the high and low times will be equal. The length of time required to execute an instruction is measured in CPU clock cycles.

## Glossary

**CPU registers** — Memory locations that are wired directly into the CPU logic instead of being part of the addressable memory map. The CPU always has direct access to the information in these registers. The CPU registers in an M68HC08 are:

- A (8-bit accumulator)
- H:X (16-bit index register)
- SP (16-bit stack pointer)
- PC (16-bit program counter)
- CCR (condition code register containing the V, H, I, N, Z, and C bits)

**CSIC** — customer-specified integrated circuit

**cycle time** — The period of the operating frequency:  $t_{CYC} = 1/f_{OP}$ .

**decimal number system** — Base 10 numbering system that uses the digits zero through nine.

**direct memory access module (DMA)** — A M68HC08 Family module that can perform data transfers between any two CPU-addressable locations without CPU intervention. For transmitting or receiving blocks of data to or from peripherals, DMA transfers are faster and more code-efficient than CPU interrupts.

**DMA** — See "direct memory access module (DMA)."

**DMA service request** — A signal from a peripheral to the DMA module that enables the DMA module to transfer data.

**duty cycle** — A ratio of the amount of time the signal is on versus the time it is off. Duty cycle is usually represented by a percentage.

**EEPROM** — Electrically erasable, programmable, read-only memory. A nonvolatile type of memory that can be electrically reprogrammed.

**EPROM** — Erasable, programmable, read-only memory. A nonvolatile type of memory that can be erased by exposure to an ultraviolet light source and then reprogrammed.

**exception** — An event such as an interrupt or a reset that stops the sequential execution of the instructions in the main program.

**external interrupt module (IRQ)** — A module in the M68HC08 Family with both dedicated external interrupt pins and port pins that can be enabled as interrupt pins.

**fetch** — To copy data from a memory location into the accumulator.

**firmware** — Instructions and data programmed into nonvolatile memory.

**free-running counter** — A device that counts from zero to a predetermined number, then rolls over to zero and begins counting again.

**full-duplex transmission** — Communication on a channel in which data can be sent and received simultaneously.

**H** — The upper byte of the 16-bit index register (H:X) in the CPU08.

**H** — The half-carry bit in the condition code register of the CPU08. This bit indicates a carry from the low-order four bits of the accumulator value to the high-order four bits. The half-carry bit is required for binary-coded decimal arithmetic operations. The decimal adjust accumulator (DAA) instruction uses the state of the H and C bits to determine the appropriate correction factor.

**hexadecimal** — Base 16 numbering system that uses the digits 0 through 9 and the letters A through F.

**high byte** — The most significant eight bits of a word.

**illegal address** — An address not within the memory map

**illegal opcode** — A nonexistent opcode.

**I** — The interrupt mask bit in the condition code register of the CPU08. When I is set, all interrupts are disabled.

**index register (H:X)** — A 16-bit register in the CPU08. The upper byte of H:X is called H. The lower byte is called X. In the indexed addressing modes, the CPU uses the contents of H:X to determine the effective address of the operand. H:X can also serve as a temporary data storage location.

**input/output (I/O)** — Input/output interfaces between a computer system and the external world. A CPU reads an input to sense the level of an external signal and writes to an output to change the level on an external signal.

## Glossary

**instructions** — Operations that a CPU can perform. Instructions are expressed by programmers as assembly language mnemonics. A CPU interprets an opcode and its associated operand(s) and instruction.

**interrupt** — A temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine.

**interrupt request** — A signal from a peripheral to the CPU intended to cause the CPU to execute a subroutine.

**I/O** — See "input/output (I/O)."

**IRQ** — See "external interrupt module (IRQ)."

**jitter** — Short-term signal instability.

**latch** — A circuit that retains the voltage level (logic 1 or logic 0) written to it for as long as power is applied to the circuit.

**latency** — The time lag between instruction completion and data movement.

**least significant bit (LSB)** — The rightmost digit of a binary number.

**logic 1** — A voltage level approximately equal to the input power voltage ( $V_{DD}$ ).

**logic 0** — A voltage level approximately equal to the ground voltage ( $V_{SS}$ ).

**low byte** — The least significant eight bits of a word.

**low voltage inhibit module (LVI)** — A module in the M68HC08 Family that monitors power supply voltage.

**LVI** — See "low voltage inhibit module (LVI)."

**M68HC08** — A Motorola family of 8-bit MCUs.

**mark/space** — The logic 1/logic 0 convention used in formatting data in serial communication.

**mask** — 1. A logic circuit that forces a bit or group of bits to a desired state. 2. A photomask used in integrated circuit fabrication to transfer an image onto silicon.

**mask option** — A optional microcontroller feature that the customer chooses to enable or disable.

**mask option register (MOR)** — An EPROM location containing bits that enable or disable certain MCU features.

**MCU** — Microcontroller unit. See “microcontroller.”

**memory location** — Each M68HC08 memory location holds one byte of data and has a unique address. To store information in a memory location, the CPU places the address of the location on the address bus, the data information on the data bus, and asserts the write signal. To read information from a memory location, the CPU places the address of the location on the address bus and asserts the read signal. In response to the read signal, the selected memory location places its data onto the data bus.

**memory map** — A pictorial representation of all memory locations in a computer system.

**microcontroller** — Microcontroller unit (MCU). A complete computer system, including a CPU, memory, a clock oscillator, and input/output (I/O) on a single integrated circuit.

**modulo counter** — A counter that can be programmed to count to any number from zero to its maximum possible modulus.

**monitor ROM** — A section of ROM that can execute commands from a host computer for testing purposes.

**MOR** — See "mask option register (MOR)."

**most significant bit (MSB)** — The leftmost digit of a binary number.

**multiplexer** — A device that can select one of a number of inputs and pass the logic level of that input on to the output.

**N** — The negative bit in the condition code register of the CPU08. The CPU sets the negative bit when an arithmetic operation, logical operation, or data manipulation produces a negative result.

**nibble** — A set of four bits (half of a byte).

**object code** — The output from an assembler or compiler that is itself executable machine code, or is suitable for processing to produce executable machine code.

## Glossary

**opcode** — A binary code that instructs the CPU to perform an operation.

**open-drain** — An output that has no pullup transistor. An external pullup device can be connected to the power supply to provide the logic 1 output voltage.

**operand** — Data on which an operation is performed. Usually a statement consists of an operator and an operand. For example, the operator may be an add instruction, and the operand may be the quantity to be added.

**oscillator** — A circuit that produces a constant frequency square wave that is used by the computer as a timing and sequencing reference.

**OTPROM** — One-time programmable read-only memory. A nonvolatile type of memory that cannot be reprogrammed.

**overflow** — A quantity that is too large to be contained in one byte or one word.

**page zero** — The first 256 bytes of memory (addresses \$0000–\$00FF).

**parity** — An error-checking scheme that counts the number of logic 1s in each byte transmitted. In a system that uses odd parity, every byte is expected to have an odd number of logic 1s. In an even parity system, every byte should have an even number of logic 1s. In the transmitter, a parity generator appends an extra bit to each byte to make the number of logic 1s odd for odd parity or even for even parity. A parity checker in the receiver counts the number of logic 1s in each byte. The parity checker generates an error signal if it finds a byte with an incorrect number of logic 1s.

**PC** — See “program counter (PC).”

**peripheral** — A circuit not under direct CPU control.

**phase-locked loop (PLL)** — A oscillator circuit in which the frequency of the oscillator is synchronized to a reference signal.

**PLL** — See “phase-locked loop (PLL).”

**pointer** — Pointer register. An index register is sometimes called a pointer register because its contents are used in the calculation of the address of an operand, and therefore points to the operand.

**polarity** — The two opposite logic levels, logic 1 and logic 0, which correspond to two different voltage levels,  $V_{DD}$  and  $V_{SS}$ .

**polling** — Periodically reading a status bit to monitor the condition of a peripheral device.

**port** — A set of wires for communicating with off-chip devices.

**prescaler** — A circuit that generates an output signal related to the input signal by a fractional scale factor such as 1/2, 1/8, 1/10 etc.

**program** — A set of computer instructions that cause a computer to perform a desired operation or operations.

**program counter (PC)** — A 16-bit register in the CPU08. The PC register holds the address of the next instruction or operand that the CPU will use.

**pull** — An instruction that copies into the accumulator the contents of a stack RAM location. The stack RAM address is in the stack pointer.

**pullup** — A transistor in the output of a logic gate that connects the output to the logic 1 voltage of the power supply.

**pulse-width** — The amount of time a signal is on as opposed to being in its off state.

**pulse-width modulation (PWM)** — Controlled variation (modulation) of the pulse width of a signal with a constant frequency.

**push** — An instruction that copies the contents of the accumulator to the stack RAM. The stack RAM address is in the stack pointer.

**PWM period** — The time required for one complete cycle of a PWM waveform.

**RAM** — Random access memory. All RAM locations can be read or written by the CPU. The contents of a RAM memory location remain valid until the CPU writes a different value or until power is turned off.

**RC circuit** — A circuit consisting of capacitors and resistors having a defined time constant.

**read** — To copy the contents of a memory location to the accumulator.

**register** — A circuit that stores a group of bits.

**reserved memory location** — A memory location that is used only in special factory test modes. Writing to a reserved location has no effect. Reading a reserved location returns an unpredictable value.

**reset** — To force a device to a known condition.

## Glossary

**ROM** — Read-only memory. A type of memory that can be read but cannot be changed (written). The contents of ROM must be specified before manufacturing the MCU.

**SCI** — See "serial communication interface module (SCI)."

**serial** — Pertaining to sequential transmission over a single line.

**serial communications interface module (SCI)** — A module in the M68HC08 Family that supports asynchronous communication.

**serial peripheral interface module (SPI)** — A module in the M68HC08 Family that supports synchronous communication.

**set** — To change a bit from logic 0 to logic 1; opposite of clear.

**shift register** — A chain of circuits that can retain the logic levels (logic 1 or logic 0) written to them and that can shift the logic levels to the right or left through adjacent circuits in the chain.

**signed** — A binary number notation that accommodates both positive and negative numbers. The most significant bit is used to indicate whether the number is positive or negative, normally logic 0 for positive and logic 1 for negative. The other seven bits indicate the magnitude of the number.

**software** — Instructions and data that control the operation of a microcontroller.

**software interrupt (SWI)** — An instruction that causes an interrupt and its associated vector fetch.

**SPI** — See "serial peripheral interface module (SPI)."

**stack** — A portion of RAM reserved for storage of CPU register contents and subroutine return addresses.

**stack pointer (SP)** — A 16-bit register in the CPU08 containing the address of the next available storage location on the stack.

**start bit** — A bit that signals the beginning of an asynchronous serial transmission.

**status bit** — A register bit that indicates the condition of a device.

**stop bit** — A bit that signals the end of an asynchronous serial transmission.



**subroutine** — A sequence of instructions to be used more than once in the course of a program. The last instruction in a subroutine is a return from subroutine (RTS) instruction. At each place in the main program where the subroutine instructions are needed, a jump or branch to subroutine (JSR or BSR) instruction is used to call the subroutine. The CPU leaves the flow of the main program to execute the instructions in the subroutine. When the RTS instruction is executed, the CPU returns to the main program where it left off.

**synchronous** — Refers to logic circuits and operations that are synchronized by a common reference signal.

**TIM** — See "timer interface module (TIM)."

**timer interface module (TIM)** — A module used to relate events in a system to a point in time.

**timer** — A module used to relate events in a system to a point in time.

**toggle** — To change the state of an output from a logic 0 to a logic 1 or from a logic 1 to a logic 0.

**tracking mode** — Mode of low-jitter PLL operation during which the PLL is locked on a frequency. Also see "acquisition mode."

**two's complement** — A means of performing binary subtraction using addition techniques. The most significant bit of a two's complement number indicates the sign of the number (1 indicates negative). The two's complement negative of a number is obtained by inverting each bit in the number and then adding 1 to the result.

**unbuffered** — Utilizes only one register for data; new data overwrites current data.

**unimplemented memory location** — A memory location that is not used. Writing to an unimplemented location has no effect. Reading an unimplemented location returns an unpredictable value. Executing an opcode at an unimplemented location causes an illegal address reset.

**V** — The overflow bit in the condition code register of the CPU08. The CPU08 sets the V bit when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow bit.

**variable** — A value that changes during the course of program execution.

**VCO** — See "voltage-controlled oscillator."

## Glossary

**vector** — A memory location that contains the address of the beginning of a subroutine written to service an interrupt or reset.

**voltage-controlled oscillator (VCO)** — A circuit that produces an oscillating output signal of a frequency that is controlled by a dc voltage applied to a control input.

**waveform** — A graphical representation in which the amplitude of a wave is plotted against time.

**wired-OR** — Connection of circuit outputs so that if any output is high, the connection point is high.

**word** — A set of two bytes (16 bits).

**write** — The transfer of a byte of data from the CPU to a memory location.

**X** — The lower byte of the index register (H:X) in the CPU08.

**Z** — The zero bit in the condition code register of the CPU08. The CPU08 sets the zero bit when an arithmetic operation, logical operation, or data manipulation produces a result of \$00.

# Literature Updates

This document contains the latest data available at publication time. For updates, contact one of the centers listed below:

---

---

## Literature Distribution Centers

Order literature by mail or phone.

### **USA/Europe**

Motorola Literature Distribution  
P.O. Box 5405  
Denver, Colorado, 80217  
Phone 1-303-675-2140

### **Japan**

Nippon Motorola Ltd.  
Tatsumi-SPD-JLDC  
Toshikatsu Otsuki  
6F Seibu-Butsuryu Center  
3-14-2 Tatsumi Koto-Ku  
Tokyo 135, Japan  
Phone 03-3521-8315

## Literature Updates

### Hong Kong

Motorola Semiconductors H.K. Ltd.  
8B Tai Ping Industrial Park  
51 Ting Kok Road  
Tai Po, N.T., Hong Kong  
Phone 852-26629298

---

---

### Customer Focus Center

1-800-521-6274

---

---

### Microcontroller Division's Web Site

Directly access the Microcontroller Division's web site with the following URL:

<http://mcu.motsps.com/>

# Revision History

## Major Changes Between Revision 2.0 and Revision 1.0

The following table lists the major changes between the current revision of the MC68HC908AZ60 Technical Data Book, Rev 2.0, and the previous revision, Rev 1.0.

| Section affected               | Description of change   |
|--------------------------------|---|
| General Description            | Added note about Hysteresis to the pin summary  |
| Memory Map                     | Corrected RAM-1 size from 512 to 1024 bytes<br>Named TIM to PIT and SBSW to BW<br>Added note to recommend all vector addresses are filled   |
| Flash EEPROM 1 and 2           | Noted that future device will have an 'A' suffix<br>Corrected type errors<br>Added note about use of interrupts during programming<br>Corrected Flash-2 array size, location and architecture |
| EEPROM 1 and 2                 | Corrected type errors<br>Added note about array condition as shipped<br>Removed references to redundant mode  |
| CPU                            | Removed reference to addressing beyond 64K bytes  |
| SIM                            | Corrected type error<br>Corrected error: break does not bring MCU out of Stop mode  |
| CGM                            | Corrected introduction to reflect specification   |
| Configuration Register 1 and 2 | Corrected type error  |
| Break Module                   | Corrected error: break does not bring MCU out of Stop mode  |
| Monitor ROM                    | Corrected type errors<br>Added note in monitor mode circuit about crystal value   |
| COP                            | Corrected type errors   |
| LVI                            | Notes added to warn about noisy $V_{DD}$ and using the LVI for anything other than safe shutdown  |

## Revision History

| Section affected          | Description of change  |
|---------------------------|--|
| IRQ                       | IRQ1 references changed to IRQ   |
| SCI                       | Added note to Stop mode section about CPU interrupts   |
| TIMB                      | Corrected type error   |
| PIT                       | Changed name from TIM to PIT (Programmable Interrupt Timer)  |
| Ports                     | Changed name from CAN I/O Ports to I/O Ports   |
| MSCAN                     | Corrected type errors<br>Added comments about CAN activity bringing MCU out of CPU STOP/CAN power-down modes   |
| TIMA-6                    | Corrected type errors  |
| ADC                       | Corrected type errors<br>Added notes about current injection causing conversion errors, and $V_{DDAREF}$ being present for operation<br>Corrected error implying Port D not to be fully I/O<br>Removed note implying $V_{REFH} \neq V_{DD}$ is legal condition   |
| Electrical Specifications | Corrected type errors<br>Removed reference to 52 PLCC package<br>Added spec for ports total sink and source current<br>Increased crystal reference frequency to 16 MHz<br>Added $t_{EEOFF}$ and $t_{EESTOP}$ specs<br>Increased Flash endurance to 1000 cycles<br>Added note about $I_{DD}$ at very low frequencies<br>Corrected operating temperature range to reflect different grades of temperature<br>Added note that $f_{VRS}$ is a calculated value based on desired VCO frequency. |
| Future EEPROM             | Corrected maximum recommended clock frequency from 32MHz to 16MHz  |

## Revision History

## Revision History



## Revision History

## Revision History


## Revision History

## Revision History

## Revision History

## Revision History



Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola 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 Motorola 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. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

**How to reach us:**

**USA/EUROPE:** Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado 80217. 1-303-675-2140

**HOME PAGE:** <http://mcu.motsps.com/documentation/index.html>

**JAPAN:** Motorola Japan Ltd.; Tatsumi-SPD-JLDC, 6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 81-3-3521-8315

**HONG KONG:** Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

**CUSTOMER FOCUS CENTER:** 1-800-521-6274

© Motorola, Inc., 2000



**MOTOROLA**

**MC68HC908AZ60/D**