



Welcome to [E-XFL.COM](https://www.e-xfl.com)

What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "[Embedded - Microcontrollers](#)"

Details

| | |
|----------------------------|---|
| Product Status | Obsolete |
| Core Processor | HC08 |
| Core Size | 8-Bit |
| Speed | 8MHz |
| Connectivity | SCI, SPI |
| Peripherals | LVD, POR, PWM |
| Number of I/O | 13 |
| Program Memory Size | 4KB (4K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 128 x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.5V |
| Data Converters | A/D 10x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 16-TSSOP (0.173", 4.40mm Width) |
| Supplier Device Package | 16-TSSOP |
| Purchase URL | https://www.e-xfl.com/product-detail/nxp-semiconductors/mc908qb4cdte |

This page intentionally blank.

Enhanced Serial Communications Interface (ESCI) Module

| | | |
|----------|---------------------------------|-----|
| 13.1 | Introduction | 109 |
| 13.2 | Features | 109 |
| 13.3 | Functional Description | 111 |
| 13.3.1 | Data Format | 112 |
| 13.3.2 | Transmitter | 112 |
| 13.3.2.1 | Character Length | 113 |
| 13.3.2.2 | Character Transmission | 113 |
| 13.3.2.3 | Break Characters | 113 |
| 13.3.2.4 | Idle Characters | 114 |
| 13.3.2.5 | Inversion of Transmitted Output | 114 |
| 13.3.3 | Receiver | 114 |
| 13.3.3.1 | Character Length | 114 |
| 13.3.3.2 | Character Reception | 114 |
| 13.3.3.3 | Data Sampling | 116 |
| 13.3.3.4 | Framing Errors | 117 |
| 13.3.3.5 | Baud Rate Tolerance | 117 |
| 13.3.3.6 | Receiver Wakeup | 119 |
| 13.4 | Interrupts | 119 |
| 13.4.1 | Transmitter Interrupts | 120 |
| 13.4.2 | Receiver Interrupts | 120 |
| 13.4.3 | Error Interrupts | 120 |
| 13.5 | Low-Power Modes | 120 |
| 13.5.1 | Wait Mode | 120 |
| 13.5.2 | Stop Mode | 121 |
| 13.6 | ESCI During Break Interrupts | 121 |
| 13.7 | I/O Signals | 121 |
| 13.7.1 | ESCI Transmit Data (TxD) | 121 |
| 13.7.2 | ESCI Receive Data (RxD) | 121 |
| 13.8 | Registers | 121 |
| 13.8.1 | ESCI Control Register 1 | 122 |
| 13.8.2 | ESCI Control Register 2 | 123 |
| 13.8.3 | ESCI Control Register 3 | 125 |
| 13.8.4 | ESCI Status Register 1 | 126 |
| 13.8.5 | ESCI Status Register 2 | 129 |
| 13.8.6 | ESCI Data Register | 129 |
| 13.8.7 | ESCI Baud Rate Register | 130 |
| 13.8.8 | ESCI Prescaler Register | 131 |
| 13.9 | ESCI Arbiter | 135 |
| 13.9.1 | ESCI Arbiter Control Register | 135 |
| 13.9.2 | ESCI Arbiter Data Register | 136 |
| 13.9.3 | Bit Time Measurement | 136 |
| 13.9.4 | Arbitration Mode | 138 |

Chapter 14 System Integration Module (SIM)

Chapter 1

General Description

1.1 Introduction

The MC68HC908QB8 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

Table 1-1. Summary of Device Variations

| Device | FLASH/RAM Memory Size | ADC | 16-Bit Timer Channels | ESCI | SPI | Pin Count |
|--------------|-----------------------|--------------------|-----------------------|------|-----|-----------|
| MC68HC908QB8 | 8K/256 bytes | 10 channel, 10 bit | 4 | Yes | Yes | 16 pins |
| MC68HC908QB4 | 4K/128 bytes | 10 channel, 10 bit | 4 | Yes | Yes | 16 pins |
| MC68HC908QY8 | 8K/256 bytes | 4 channel, 10 bit | 2 | No | No | 16 pins |

1.2 Features

Features include:

- High-performance M68HC08 CPU core
- Fully upward-compatible object code with M68HC05 Family
- 5-V and 3-V operating voltages (V_{DD})
- 8-MHz internal bus operation at 5 V, 4-MHz at 3 V
- Trimmable internal oscillator
 - Software selectable 1 MHz, 2 MHz, or 3.2 MHz internal bus operation
 - 8-bit trim capability
 - $\pm 25\%$ untrimmed
 - Trimmable to approximately 0.4% ⁽¹⁾
- Software selectable crystal oscillator range, 32–100 kHz, 1–8 MHz, and 8–32 MHz
- Software configurable input clock from either internal or external source
- Auto wakeup from STOP capability using dedicated internal 32-kHz RC or bus clock source
- On-chip in-application programmable FLASH memory
 - Internal program/erase voltage generation
 - Monitor ROM containing user callable program/erase routines
 - FLASH security⁽²⁾

1. See [18.11 Oscillator Characteristics](#) for internal oscillator specifications

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

Memory

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|--|---|--------|---------------------|---------|---------|---------|------------------|----------|--------------|------------------|
| \$0014 | ESCI Status Register 2 (SCS2) See page 129. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | BKF | RPF |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$0015 | ESCI Data Register (SCDR) See page 129. | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| | | Reset: | Unaffected by reset | | | | | | | |
| \$0016 | ESCI Baud Rate Register (SCBR) See page 130. | Read: | LINT | LINR | SCP1 | SCP0 | R | SCR2 | SCR1 | SCR0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$0017 | ESCI Prescaler Register (SCPSC) See page 131. | Read: | PDS2 | PDS1 | PDS0 | PSSB4 | PSSB3 | PSSB2 | PSSB1 | PSSB0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$0018 | ESCI Arbiter Control Register (SCIACTL) See page 135. | Read: | AM1 | AHOST | AM0 | ACLK | AFIN | ARUN | AROVFL | ARD8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$0019 | ESCI Arbiter Data Register (SCIA DAT) See page 136. | Read: | ARD7 | ARD6 | ARD5 | ARD4 | ARD3 | ARD2 | ARD1 | ARD0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$001A | Keyboard Status and Control Register (KBSCR) See page 87. | Read: | 0 | 0 | 0 | 0 | KEYF | 0 | IMASKK | MODEK |
| | | Write: | | | | | | ACKK | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$001B | Keyboard Interrupt Enable Register (KBIER) See page 88. | Read: | 0 | AWUIE | KBIE5 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$001C | Keyboard Interrupt Polarity Register (KBIPR) See page 88. | Read: | 0 | 0 | KBIP5 | KBIP4 | KBIP3 | KBIP2 | KBIP1 | KBIP0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$001D | IRQ Status and Control Register (INTSCR) See page 81. | Read: | 0 | 0 | 0 | 0 | IRQF | 0 | IMASK | MODE |
| | | Write: | | | | | | ACK | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$001E | Configuration Register 2 (CONFIG2) ⁽¹⁾ See page 57. | Read: | IRQPUD | IRQEN | R | R | R | ESCBDSRC | OSCENIN-STOP | RSTEN |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 ⁽²⁾ |
| 1. One-time writable register after each reset. 2. RSTEN reset to 0 by a power-on reset (POR) only. | | | | | | | | | | |
| \$001F | Configuration Register 1 (CONFIG1) ⁽¹⁾ See page 58. | Read: | COPRS | LVISTOP | LVIRSTD | LVIPWRD | LVITRIP | SSREC | STOP | COPD |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 ⁽²⁾ | 0 | 0 | 0 |
| 1. One-time writable register after each reset. 2. LVITRIP reset to 0 by a power-on reset (POR) only. | | | | | | | | | | |
| \$0020 | TIM Status and Control Register (TSC) See page 183. | Read: | TOF | TOIE | TSTOP | 0 | 0 | PS2 | PS1 | PS0 |
| | | Write: | 0 | | | TRST | | | | |
| | | Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| \$0021 | TIM Counter Register High (TCNTH) See page 185. | Read: | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| = Unimplemented R = Reserved U = Unaffected | | | | | | | | | | |

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

Memory

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|-----------------------|---|--------|---------|---------|-------|--------|-------|--------|--------|--------|
| \$0036 | Oscillator Status and Control Register (OSCSR) See page 100. | Read: | OSCOPT1 | OSCOPT0 | ICFS1 | ICFS0 | ECFS1 | ECFS0 | ECGN | ECGST |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$0037 | Reserved | | | | | | | | | |
| \$0038 | Oscillator Trim Register (OSCTRIM) See page 101. | Read: | TRIM7 | TRIM6 | TRIM5 | TRIM4 | TRIM3 | TRIM2 | TRIM1 | TRIM0 |
| | | Write: | | | | | | | | |
| | | Reset: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$0039 ↓ \$003B | Reserved | | | | | | | | | |
| \$003C | ADC10 Status and Control Register (ADCSR) See page 46. | Read: | COCO | AIEN | ADCO | ADCH4 | ADCH3 | ADCH2 | ADCH1 | ADCH0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| \$003D | ADC10 Data Register High (ADRH) See page 48. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | AD9 | AD8 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$003E | ADC10 Data Register Low (ADRL) See page 48. | Read: | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$003F | ADC10 Clock Register (ADCLK) See page 49. | Read: | ADLPC | ADIV1 | ADIV0 | ADICLK | MODE1 | MODE0 | ADLSMP | ACLKEN |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$FE00 | Break Status Register (BSR) See page 195. | Read: | R | R | R | R | R | R | SBSW | R |
| | | Write: | | | | | | | 0 | |
| | | Reset: | | | | | | | 0 | |
| \$FE01 | SIM Reset Status Register (SRSR) See page 152. | Read: | POR | PIN | COP | ILOP | ILAD | MODRST | LVI | 0 |
| | | Write: | | | | | | | | |
| | | POR: | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$FE02 | Break Auxiliary Register (BRKAR) See page 195. | Read: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BDCOP |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$FE03 | Break Flag Control Register (BFCR) See page 195. | Read: | BCFE | R | R | R | R | R | R | R |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | | | | | | | |
| \$FE04 | Interrupt Status Register 1 (INT1) See page 149. | Read: | IF6 | IF5 | IF4 | IF3 | IF2 | IF1 | 0 | 0 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$FE05 | Interrupt Status Register 2 (INT2) See page 149. | Read: | IF14 | IF13 | IF12 | IF11 | IF10 | IF9 | IF8 | IF7 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| \$FE06 | Interrupt Status Register 3 (INT3) See page 149. | Read: | IF22 | IF21 | IF20 | IF19 | IF18 | IF17 | IF16 | IF15 |
| | | Write: | R | R | R | R | R | R | R | R |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

 = Unimplemented
 R = Reserved
 U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 4 of 5)

2.6 FLASH Memory (FLASH)

The FLASH memory is intended primarily for program storage. In-circuit programming allows the operating program to be loaded into the FLASH memory after final assembly of the application product. It is possible to program the entire array through the single-wire monitor mode interface. Because no special voltages are needed for FLASH erase and programming operations, in-application programming is also possible through other software-controlled communication paths.

This subsection describes the operation of the embedded FLASH memory. The FLASH memory can be read, programmed, and erased from the internal V_{DD} supply. The program and erase operations are enabled through the use of an internal charge pump.

The minimum size of FLASH memory that can be erased is 64 bytes; and the maximum size of FLASH memory that can be programmed in a program cycle is 32 bytes (a row). Program and erase operations are facilitated through control bits in the FLASH control register (FLCR). Details for these operations appear later in this section.

NOTE

An erased bit reads as a 1 and a programmed bit reads as a 0. A security feature prevents viewing of the FLASH contents.⁽¹⁾

2.6.1 FLASH Control Register

The FLASH control register (FLCR) controls FLASH program and erase operations.

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|--------|-------|---|---|---|------|------|-------|-------|
| Read: | 0 | 0 | 0 | 0 | HVEN | MASS | ERASE | PGM |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |


 = Unimplemented

Figure 2-3. FLASH Control Register (FLCR)

HVEN — High Voltage Enable Bit

This read/write bit enables high voltage from the charge pump to the memory for either program or erase operation. It can only be set if either PGM = 1 or ERASE = 1 and the proper sequence for program or erase is followed.

- 1 = High voltage enabled to array and charge pump on
- 0 = High voltage disabled to array and charge pump off

MASS — Mass Erase Control Bit

This read/write bit configures the memory for mass erase operation.

- 1 = Mass erase operation selected
- 0 = Mass erase operation unselected

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

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 equal to 1 or set to 1 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 equal to 1 or set to 1 at the same time.

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

2.6.2 FLASH Page Erase Operation

Use the following procedure to erase a page of FLASH memory. A page consists of 64 consecutive bytes starting from addresses \$XX00, \$XX40, \$XX80, or \$XXC0. The 48-byte user interrupt vectors area also forms a page. Any FLASH memory page can be erased alone.

1. Set the ERASE bit and clear the MASS bit in the FLASH control register.
2. Read the FLASH block protect register.
3. Write any data to any FLASH location within the address range of the block to be erased.
4. Wait for a time, t_{NVS} .
5. Set the HVEN bit.
6. Wait for a time, t_{Erase} .
7. Clear the ERASE bit.
8. Wait for a time, t_{NVH} .
9. Clear the HVEN bit.
10. After time, t_{RCV} , the memory can be accessed in read mode again.

NOTE

The COP register at location \$FFFF should not be written between steps 5-9, when the HVEN bit is set. Since this register is located at a valid FLASH address, unpredictable behavior may occur if this location is written while HVEN is set.

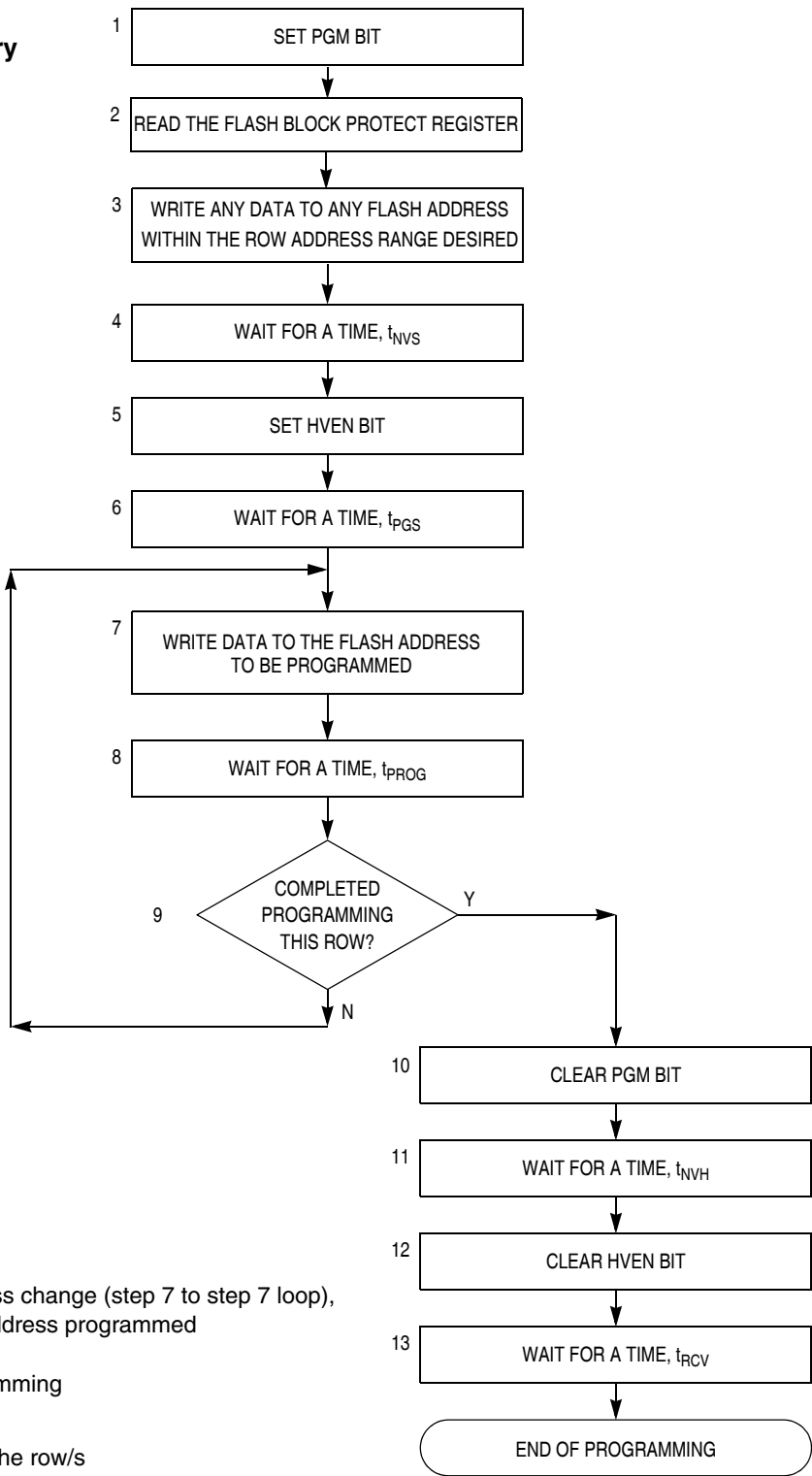
NOTE

Programming and erasing of FLASH locations cannot be performed by code being executed from the FLASH memory. While these operations must be performed in the order as shown, other unrelated operations may occur between the steps.

CAUTION

A page erase of the vector page will erase the internal oscillator trim value at \$FFC0.

**Algorithm for Programming
a Row (32 Bytes) of FLASH Memory**



NOTES:

The time between each FLASH address change (step 7 to step 7 loop), or the time between the last FLASH address programmed to clearing PGM bit (step 7 to step 10) must not exceed the maximum programming time, $t_{\text{PROG max}}$.

This row program algorithm assumes the row/s to be programmed are initially erased.

Figure 2-4. FLASH Programming Flowchart

charging. If externally available, connect the V_{REFL} pin to the same potential as V_{SSA} at the single point ground location.

3.7.5 ADC10 Channel Pins (ADn)

The ADC10 has multiple input channels. Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. 0.01 μ F capacitors with good high-frequency characteristics are sufficient. These capacitors are not necessary in all cases, but when used they must be placed as close as possible to the package pins and be referenced to V_{SSA} .

3.8 Registers

These registers control and monitor operation of the ADC10:

- ADC10 status and control register, ADSCR
- ADC10 data registers, ADRH and ADRL
- ADC10 clock register, ADCLK

3.8.1 ADC10 Status and Control Register

This section describes the function of the ADC10 status and control register (ADSCR). Writing ADSCR aborts the current conversion and initiates a new conversion (if the ADCH[4:0] bits are equal to a value other than all 1s).

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|--------|-------|------|------|-------|-------|-------|-------|-------|
| Read: | COCO | | | | | | | |
| Write: | | AIEN | ADCO | ADCH4 | ADCH3 | ADCH2 | ADCH1 | ADCH0 |
| Reset: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |


 = Unimplemented

Figure 3-3. ADC10 Status and Control Register (ADSCR)

COCO — Conversion Complete Bit

COCO is a read-only bit which is set each time a conversion is completed. This bit is cleared whenever the status and control register is written or whenever the data register (low) is read.

- 1 = Conversion completed
- 0 = Conversion not completed

AIEN — ADC10 Interrupt Enable Bit

When this bit is set, an interrupt is generated at the end of a conversion. The interrupt signal is cleared when the data register is read or the status/control register is written.

- 1 = ADC10 interrupt enabled
- 0 = ADC10 interrupt disabled

ADCO — ADC10 Continuous Conversion Bit

When this bit is set, the ADC10 will begin to convert samples continuously (continuous conversion mode) and update the result registers at the end of each conversion, provided the ADCH[4:0] bits do not decode to all 1s. The ADC10 will continue to convert until the MCU enters reset, the MCU enters stop mode (if ACLKEN is clear), ADCLK is written, or until ADSCR is written again. If stop is entered

3.8.4 ADC10 Clock Register (ADCLK)

This register selects the clock frequency for the ADC10 and the modes of operation.

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|--------|-------|-------|-------|--------|-------|-------|--------|--------|
| Read: | ADLPC | ADIV1 | ADIV0 | ADICLK | MODE1 | MODE0 | ADLSMP | ACLKEN |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 3-7. ADC10 Clock Register (ADCLK)

ADLPC — ADC10 Low-Power Configuration Bit

ADLPC controls the speed and power configuration of the successive approximation converter. This is used to optimize power consumption when higher sample rates are not required.

1 = Low-power configuration: The power is reduced at the expense of maximum clock speed.

0 = High-speed configuration

ADIV[1:0] — ADC10 Clock Divider Bits

ADIV1 and ADIV0 select the divide ratio used by the ADC10 to generate the internal clock ADCK.

Table 3-3 shows the available clock configurations.

Table 3-3. ADC10 Clock Divide Ratio

| ADIV1 | ADIV0 | Divide Ratio (ADIV) | Clock Rate |
|-------|-------|---------------------|-----------------|
| 0 | 0 | 1 | Input clock ÷ 1 |
| 0 | 1 | 2 | Input clock ÷ 2 |
| 1 | 0 | 4 | Input clock ÷ 4 |
| 1 | 1 | 8 | Input clock ÷ 8 |

ADICLK — Input Clock Select Bit

If ACLKEN is clear, ADICLK selects either the bus clock or an alternate clock source as the input clock source to generate the internal clock ADCK. If the alternate clock source is less than the minimum clock speed, use the internally-generated bus clock as the clock source. As long as the internal clock ADCK, which is equal to the selected input clock divided by ADIV, is at a frequency (f_{ADCK}) between the minimum and maximum clock speeds (considering ALPC), correct operation can be guaranteed.

1 = The internal bus clock is selected as the input clock source

0 = The alternate clock source IS SELECTED

MODE[1:0] — 10- or 8-Bit or Hardware Triggered Mode Selection

These bits select 10- or 8-bit operation. The successive approximation converter generates a result that is rounded to 8- or 10-bit value based on the mode selection. This rounding process sets the transfer function to transition at the midpoint between the ideal code voltages, causing a quantization error of $\pm 1/2\text{LSB}$.

Reset returns 8-bit mode.

00 = 8-bit, right-justified, ADSCR software triggered mode enabled

01 = 10-bit, right-justified, ADSCR software triggered mode enabled

10 = Reserved

11 = 10-bit, right-justified, hardware triggered mode enabled

4.3 Functional Description

The function of the auto wakeup logic is to generate periodic wakeup requests to bring the microcontroller unit (MCU) out of stop mode. The wakeup requests are treated as regular keyboard interrupt requests, with the difference that instead of a pin, the interrupt signal is generated by an internal logic.

Entering stop mode will enable the auto wakeup generation logic. Writing the AWUIE bit in the keyboard interrupt enable register enables or disables the auto wakeup interrupt input (see [Figure 4-1](#)). A 1 applied to the AWUIREQ input with auto wakeup interrupt request enabled, latches an auto wakeup interrupt request.

Auto wakeup latch, AWUL, can be read directly from the bit 6 position of port A data register (PTA). This is a read-only bit which is occupying an empty bit position on PTA. No PTA associated registers, such as PTA6 data direction or PTA6 pullup exist for this bit.

There are two clock sources for the AWU. An internal RC oscillator (INTRCOSC, exclusive for the auto wakeup feature) drives the wakeup request generator provided the OSCENINSTOP bit in the CONFIG2 register [Figure 4-1](#) is cleared. More accurate wakeup periods are possible using the BUSCLKX2 signal (from the oscillator module) which is selected by setting OSCENINSTOP.

Once the overflow count is reached in the generator counter, a wakeup request, AWUIREQ, is latched and sent to the KBI logic. See [Figure 4-1](#).

Wakeup interrupt requests will only be serviced if the associated interrupt enable bit, AWUIE, in KBIER is set. The AWU shares the keyboard interrupt vector.

The overflow count can be selected from two options defined by the COPRS bit in CONFIG1. This bit was “borrowed” from the computer operating properly (COP) using the fact that the COP feature is idle (no MCU clock available) in stop mode. COPRS = 1 selects the short wakeup period while COPRS = 0 selects the long wakeup period.

The auto wakeup RC oscillator is highly dependent on operating voltage and temperature. This feature is not recommended for use as a time-keeping function.

The wakeup request is latched to allow the interrupt source identification. The latched value, AWUL, can be read directly from the bit 6 position of PTA data register. This is a read-only bit which is occupying an empty bit position on PTA. No PTA associated registers, such as PTA6 data, PTA6 direction, and PTA6 pullup exist for this bit. The latch can be cleared by writing to the ACKK bit in the KBSCR register. Reset also clears the latch. AWUIE bit in KBI interrupt enable register (see [Figure 4-1](#)) has no effect on AWUL reading.

The AWU oscillator and counters are inactive in normal operating mode and become active only upon entering stop mode.

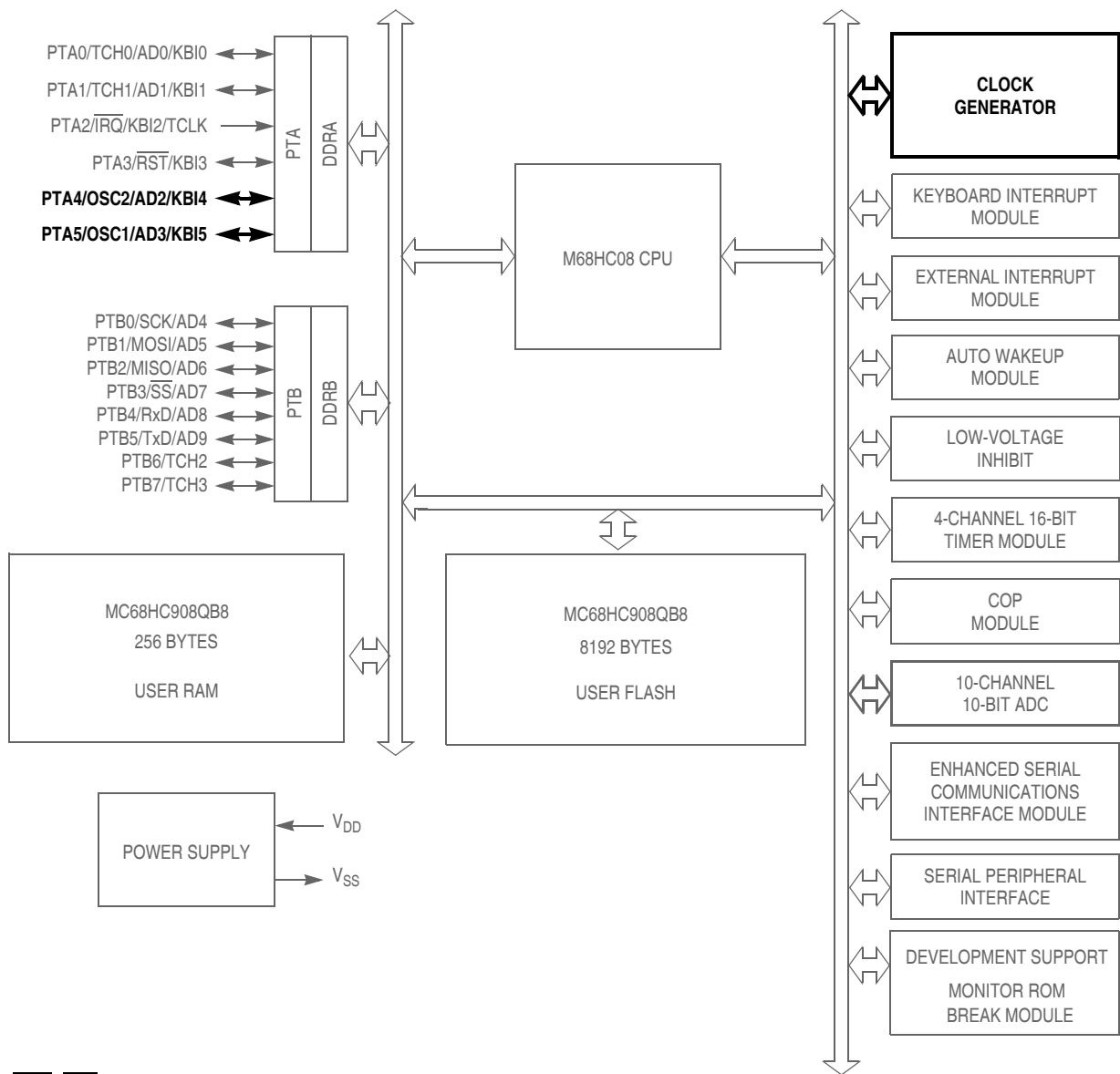
4.4 Interrupts

The AWU can generate an interrupt requests:.

AWU Latch (AWUL) — The AWUL bit is set when the AWU counter overflows. The auto wakeup interrupt mask bit, AWUIE, is used to enable or disable AWU interrupt requests.

The AWU shares its interrupt with the KBI vector.

Oscillator Module (OSC)



$\overline{\text{RST}}$, $\overline{\text{IRQ}}$: Pins have internal pull up device
All port pins have programmable pull up device
PTA[0:5]: Higher current sink and source capability

Figure 11-1. Block Diagram Highlighting OSC Block and Pins

12.2.4 Port A Summary Table

The following table summarizes the operation of the port A pins when used as a general-purpose input/output pins.

Table 12-1. Port A Pin Functions

| PTAPUE Bit | DDRA Bit | PTA Bit | I/O Pin Mode | Accesses to DDRA | Accesses to PTA | |
|------------|----------|------------------|---------------------------------------|------------------|-----------------|--------------------------|
| | | | | Read/Write | Read | Write |
| 1 | 0 | X ⁽¹⁾ | Input, V _{DD} ⁽²⁾ | DDRA5–DDRA0 | Pin | PTA5–PTA0 ⁽³⁾ |
| 0 | 0 | X | Input, Hi-Z ⁽⁴⁾ | DDRA5–DDRA0 | Pin | PTA5–PTA0 ⁽³⁾ |
| X | 1 | X | Output | DDRA5–DDRA0 | PTA5–PTA0 | PTA5–PTA0 ⁽⁵⁾ |

1. X = don't care
2. I/O pin pulled to V_{DD} by internal pullup.
3. Writing affects data register, but does not affect input.
4. Hi-Z = high impedance
5. Output does not apply to PTA2

12.3 Port B

Port B is an 8-bit special function port that shares its pins with the 4-channel timer interface module (TIM) (see [Chapter 16 Timer Interface Module \(TIM\)](#)), the 10-bit ADC (see [Chapter 3 Analog-to-Digital Converter \(ADC10\) Module](#)), the serial peripheral interface (SPI) module (see [Chapter 15 Serial Peripheral Interface \(SPI\) Module](#)) and the enhanced serial communications interface (ESCI) module (see [Chapter 13 Enhanced Serial Communications Interface \(ESCI\) Module](#)).

Each port B pin also has a software configurable pullup device if the corresponding port pin is configured as an input port.

12.3.1 Port B Data Register

The port B data register (PTB) contains a data latch for each of the port B pins.

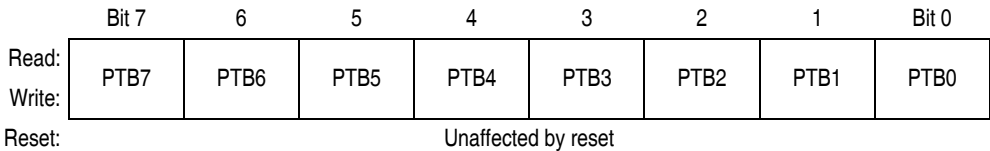


Figure 12-5. Port B Data Register (PTB)

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.

Chapter 13

Enhanced Serial Communications Interface (ESCI) Module

13.1 Introduction

The enhanced serial communications interface (ESCI) module allows asynchronous communications with peripheral devices and other microcontroller units (MCU).

The ESCI module shares its pins with general-purpose input/output (I/O) port pins. See [Figure 13-1](#) for port location of these shared pins. The ESCI baud rate clock source is controlled by a bit (ESCIBDSRC) located in the configuration register.

13.2 Features

Features include:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- Programmable baud rates
- Programmable 8-bit or 9-bit character length
- Separately enabled transmitter and receiver
- Separate receiver and transmitter interrupt requests
- Programmable transmitter output polarity
- Receiver wakeup methods
 - Idle line
 - Address mark
- 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

Enhanced Serial Communications Interface (ESCI) Module

- Enables the receiver
- Enables ESCI wakeup
- Transmits ESCI break characters

| | 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 13-10. ESCI Control Register 2 (SCC2)

SCTIE — ESCI Transmit Interrupt Enable Bit

This read/write bit enables the SCTE bit to generate ESCI transmitter interrupt requests. Setting the SCTIE bit in SCC2 enables the SCTE bit to generate interrupt requests.

- 1 = SCTE enabled to generate interrupt
- 0 = SCTE not enabled to generate interrupt

TCIE — Transmission Complete Interrupt Enable Bit

This read/write bit enables the TC bit to generate ESCI transmitter interrupt requests.

- 1 = TC enabled to generate interrupt requests
- 0 = TC not enabled to generate interrupt requests

SCRIE — ESCI Receive Interrupt Enable Bit

This read/write bit enables the SCRF bit to generate ESCI receiver interrupt requests. Setting the SCRIE bit in SCC2 enables the SCRF bit to generate interrupt requests.

- 1 = SCRF enabled to generate interrupt
- 0 = SCRF not enabled to generate interrupt

ILIE — Idle Line Interrupt Enable Bit

This read/write bit enables the IDLE bit to generate ESCI receiver interrupt requests.

- 1 = IDLE enabled to generate interrupt requests
- 0 = IDLE not enabled to generate interrupt requests

TE — Transmitter Enable Bit

Setting this read/write bit begins the transmission by sending a preamble of 10 or 11 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 (high). Clearing and then setting TE during a transmission queues an idle character to be sent after the character currently being transmitted.

- 1 = Transmitter enabled
- 0 = Transmitter disabled

NOTE

Writing to the TE bit is not allowed when the enable ESCI bit (ENSCI) is clear. ENSCI is in ESCI control register 1.

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.

- 1 = Receiver enabled
- 0 = Receiver disabled

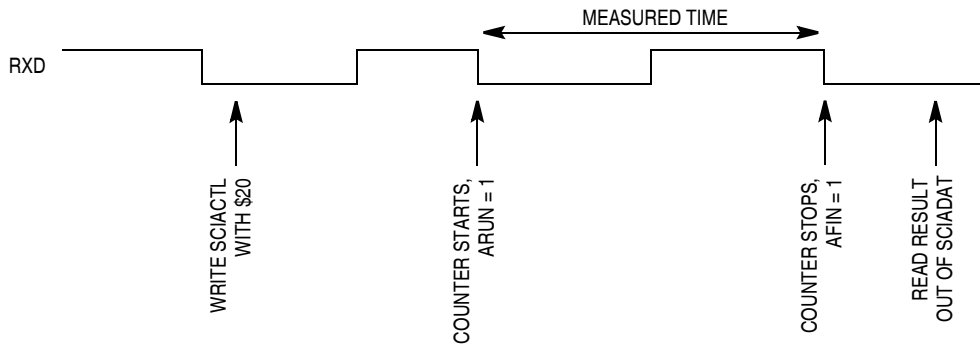


Figure 13-20. Bit Time Measurement with ACLK = 0

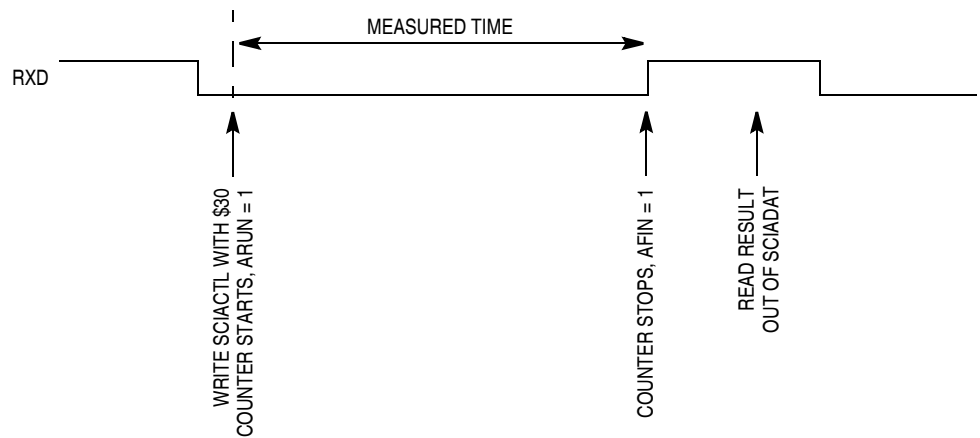


Figure 13-21. Bit Time Measurement with ACLK = 1, Scenario A

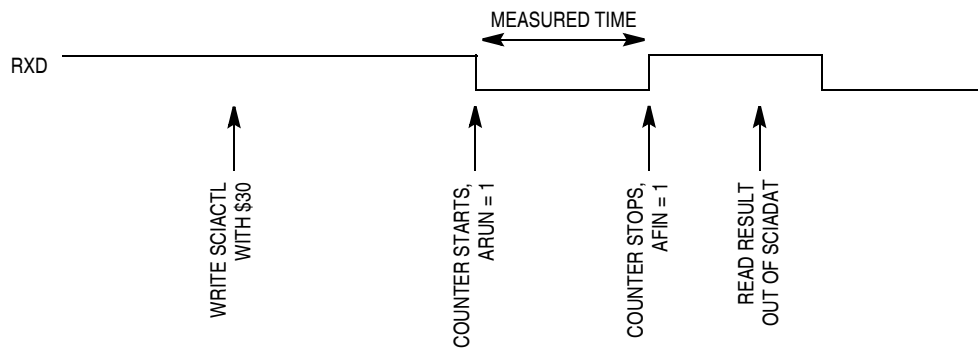


Figure 13-22. Bit Time Measurement with ACLK = 1, Scenario B

14.6 Exception Control

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

1. Interrupts
 - a. Maskable hardware CPU interrupts
 - b. Non-maskable software interrupt instruction (SWI)
2. Reset
3. Break interrupts

14.6.1 Interrupts

An interrupt temporarily changes the sequence of program execution to respond to a particular event. [Figure 14-7](#) flow charts the handling of system interrupts.

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).

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 14-8](#) shows interrupt entry timing. [Figure 14-9](#) shows interrupt recovery timing.

14.6.1.1 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 14-10](#) 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 return-from-interrupt (RTI) instructions. However, in the case of the INT1 RTI prefetch, this is a redundant operation.

NOTE

To maintain compatibility with the M6805 Family, 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.

14.6.1.2 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.*

14.6.2 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources. [Table 14-3](#) summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

Table 14-3. Interrupt Sources

| Priority | Source | Flag | Mask ⁽¹⁾ | INT Register Flag | Vector Address |
|--|-----------------------------------|------------------|------------------------|-------------------|----------------|
| <div style="display: flex; align-items: center;"> <div style="margin-right: 10px;">Highest</div> <div style="flex-grow: 1; border-left: 1px solid black; position: relative;"> <div style="position: absolute; top: -10px; left: 50%; transform: translateX(-50%);">↑</div> <div style="position: absolute; bottom: -10px; left: 50%; transform: translateX(-50%);">↓</div> </div> <div style="margin-left: 10px;">Lowest</div> </div> | Reset | — | — | — | \$FFFE–\$FFFF |
| | SWI instruction | — | — | — | \$FFFC–\$FFFD |
| | IRQ pin | IRQF | IMASK | IF1 | \$FFFA–\$FFFB |
| | Timer channel 0 interrupt | CH0F | CH0IE | IF3 | \$FFF6–\$FFF7 |
| | Timer channel 1 interrupt | CH1F | CH1IE | IF4 | \$FFF4–\$FFF5 |
| | Timer overflow interrupt | TOF | TOIE | IF5 | \$FFF2–\$FFF3 |
| | TIM channel 2 vector | CH2F | CH2IE | IF6 | \$FFF0–\$FFF1 |
| | TIM channel 3 vector | CH3F | CH3IE | IF7 | \$FFEE–\$FFEF |
| | ESCI error vector | OR, HF, FE, PE | ORIE, NEIE, FEIE, PEIE | IF9 | \$FFEA–\$FFEB |
| | ESCI receive vector | SCRF | SCRIE | IF10 | \$FFE8–\$FFE9 |
| | ESCI transmit vector | SCTE, TC | SCTIE, TCIE | IF11 | \$FFE6–\$FFE7 |
| | SPI receive | SPRF, OVRF, MODF | SPRIE, ERRIE | IF12 | \$FFE4–\$FFE5 |
| | SPI transmit | SPTF | SPTIE | IF13 | \$FFE2–\$FFE3 |
| | Keyboard interrupt | KEYF | IMASKK | IF14 | \$FFE0–\$FFE1 |
| | ADC conversion complete interrupt | COCO | AIEN | IF15 | \$FFDE–\$FFDF |

1. The I bit in the condition code register is a global mask for all interrupt sources except the SWI instruction.

15.7 I/O Signals

The SPI module can share its pins with the general-purpose I/O pins. See [Figure 15-1](#) for the port pins that are shared.

The SPI module has four I/O pins:

- MISO — Master input/slave output
- MOSI — Master output/slave input
- SPSCCK — Serial clock
- \overline{SS} — Slave select

15.7.1 MISO (Master In/Slave Out)

MISO is one of the two SPI module pins that transmits 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 0 and its \overline{SS} pin is low. To support a multiple-slave system, a high 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.

15.7.2 MOSI (Master Out/Slave In)

MOSI is one of the two SPI module pins that transmits 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.

15.7.3 SPSCCK (Serial Clock)

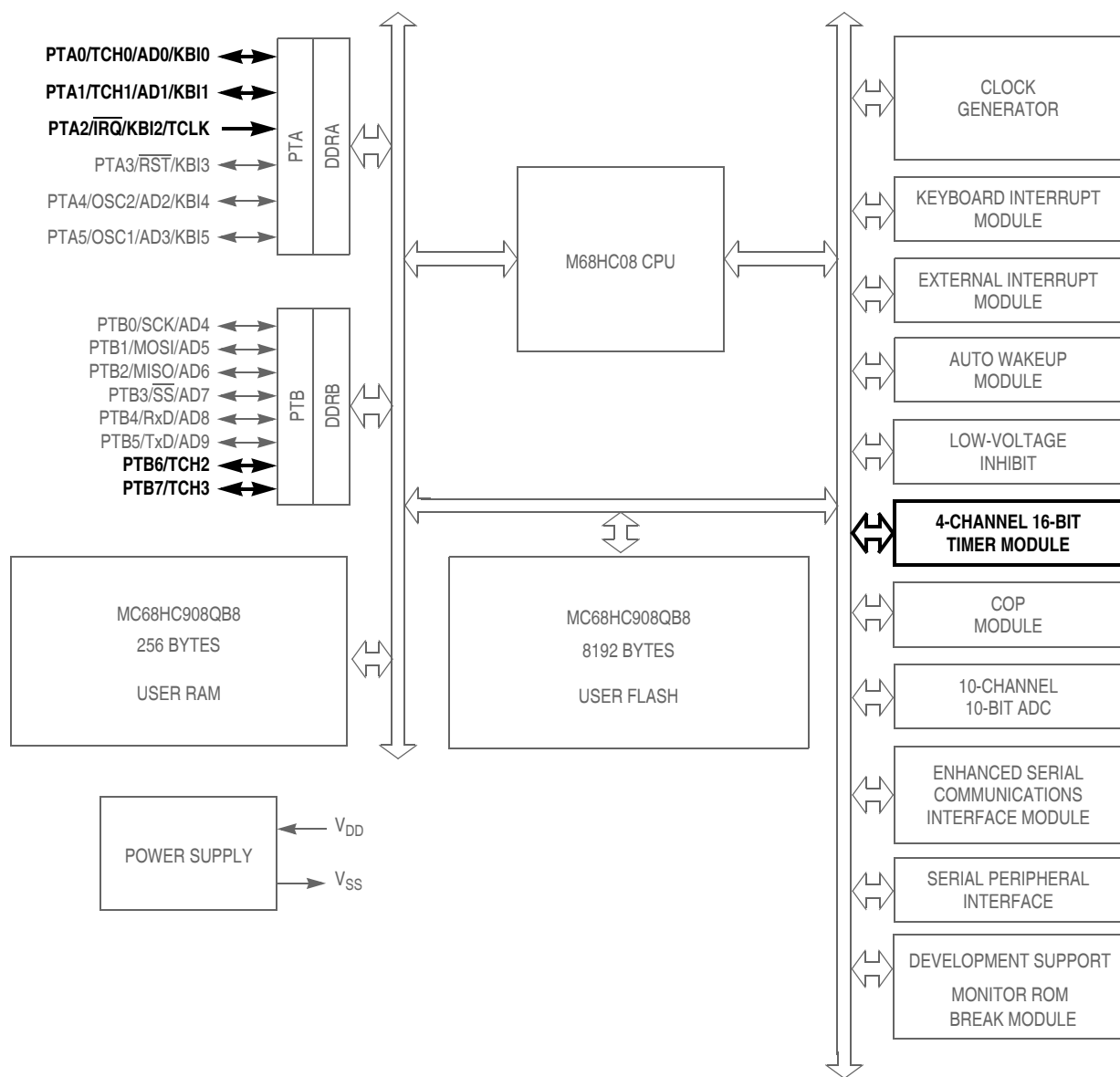
The serial clock synchronizes data transmission between master and slave devices. In a master MCU, the SPSCCK pin is the clock output. In a slave MCU, the SPSCCK pin is the clock input. In full-duplex operation, the master and slave MCUs exchange a byte of data in eight serial clock cycles.

When enabled, the SPI controls data direction of the SPSCCK pin regardless of the state of the data direction register of the shared I/O port.

15.7.4 \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, \overline{SS} is used to select a slave. For CPHA = 0, the \overline{SS} is used to define the start of a transmission. (See [15.3.3 Transmission Formats](#).) Because it is used to indicate the start of a transmission, \overline{SS} must be toggled high and low between each byte transmitted for the CPHA = 0 format. However, it can remain low between transmissions for the CPHA = 1 format. See [Figure 15-12](#).

Timer Interface Module (TIM)



\overline{RST} , \overline{IRQ} : Pins have internal pull up device
 All port pins have programmable pull up device
 PTA[0:5]: Higher current sink and source capability

Figure 16-1. Block Diagram Highlighting TIM Block and Pins

16.3.2 Input Capture

With the input capture function, the TIM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TIM latches the contents of the counter into the TIM channel registers, TCHxH:TCHxL. The polarity of the active edge is programmable. Input captures can be enabled to generate interrupt requests.