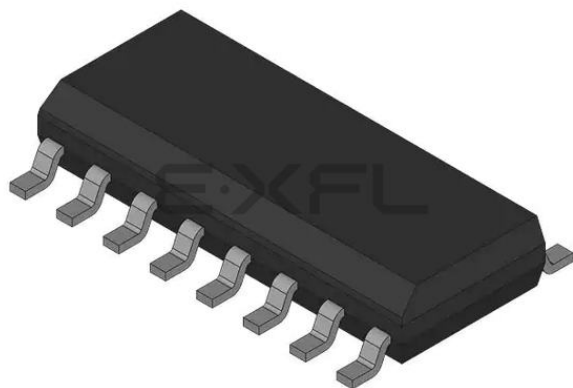




Welcome to [E-XFL.COM](https://www.e-xfl.com)

### 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        | 8KB (8K x 8)                                                                                                                              |
| Program Memory Type        | FLASH                                                                                                                                     |
| EEPROM Size                | -                                                                                                                                         |
| RAM Size                   | 256 x 8                                                                                                                                   |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.5V                                                                                                                               |
| Data Converters            | A/D 10x10b                                                                                                                                |
| Oscillator Type            | Internal                                                                                                                                  |
| Operating Temperature      | -40°C ~ 125°C (TA)                                                                                                                        |
| Mounting Type              | Surface Mount                                                                                                                             |
| Package / Case             | 16-SOIC (0.295", 7.50mm Width)                                                                                                            |
| Supplier Device Package    | 16-SOIC                                                                                                                                   |
| Purchase URL               | <a href="https://www.e-xfl.com/pro/item?MUrl=&amp;PartUrl=mc908qb8mdwe">https://www.e-xfl.com/pro/item?MUrl=&amp;PartUrl=mc908qb8mdwe</a> |

## Chapter 2

# Memory

### 2.1 Introduction

The central processor unit (CPU08) can address 64 Kbytes of memory space. The memory map, shown in [Figure 2-1](#).

### 2.2 Unimplemented Memory Locations

Executing code from an unimplemented location will cause an illegal address reset. In [Figure 2-1](#), unimplemented locations are shaded.

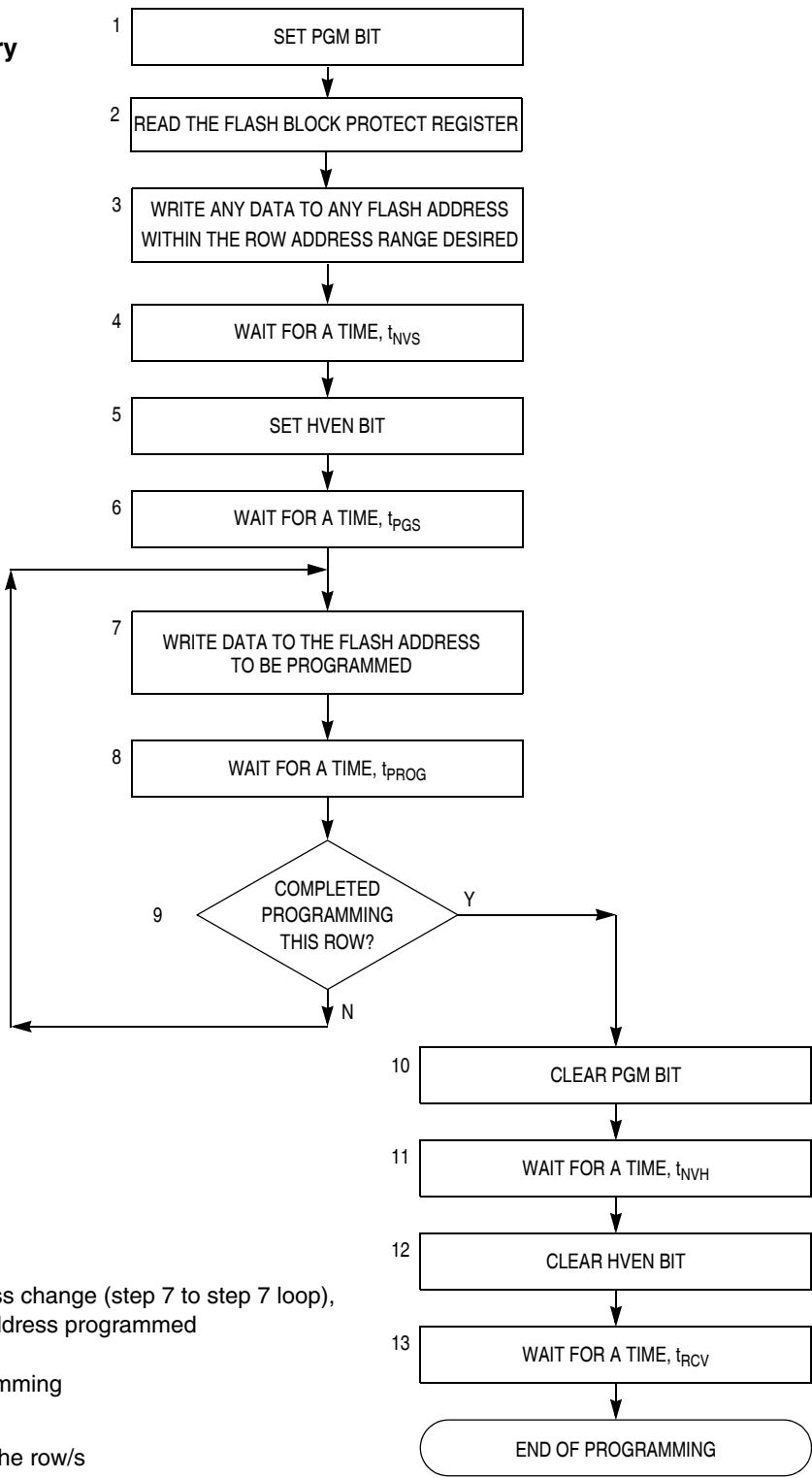
### 2.3 Reserved Memory Locations

Accessing a reserved location can have unpredictable effects on MCU operation. In [Figure 2-1](#), register locations are marked with the word Reserved or with the letter R.

### 2.4 Direct Page Registers

[Figure 2-2](#) shows the memory mapped registers of the MC68HC908QB8. Registers with addresses between \$0000 and \$00FF are considered direct page registers and all instructions including those with direct page addressing modes can access them. Registers between \$0100 and \$FFFF require non-direct page addressing modes. See [Chapter 7 Central Processor Unit \(CPU\)](#) for more information on addressing modes.

**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,  $t_{PROG}$  max.

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

**Figure 2-4. FLASH Programming Flowchart**

## 4.6.2 Keyboard Status and Control Register

The keyboard status and control register (KBSCR):

- Flags keyboard/auto wakeup interrupt requests
- Acknowledges keyboard/auto wakeup interrupt requests
- Masks keyboard/auto wakeup interrupt requests

|        | 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 4-3. Keyboard Status and Control Register (KBSCR)**

### Bits 7–4 — Not used

These read-only bits always read as 0s.

### KEYF — Keyboard Flag Bit

This read-only bit is set when a keyboard interrupt is pending on port A or auto wakeup. Reset clears the KEYF bit.

- 1 = Keyboard/auto wakeup interrupt pending
- 0 = No keyboard/auto wakeup interrupt pending

### ACKK — Keyboard Acknowledge Bit

Writing a 1 to this write-only bit clears the keyboard/auto wakeup interrupt request on port A and auto wakeup logic. ACKK always reads as 0. Reset clears ACKK.

### IMASKK— Keyboard Interrupt Mask Bit

Writing a 1 to this read/write bit prevents the output of the keyboard interrupt mask from generating interrupt requests on port A or auto wakeup. Reset clears the IMASKK bit.

- 1 = Keyboard/auto wakeup interrupt requests masked
- 0 = Keyboard/auto wakeup interrupt requests not masked

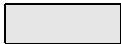
### NOTE

*MODEK is not used in conjunction with the auto wakeup feature. To see a description of this bit, see [9.8.1 Keyboard Status and Control Register \(KBSCR\)](#).*

## 4.6.3 Keyboard Interrupt Enable Register

The keyboard interrupt enable register (KBIER) enables or disables the auto wakeup to operate as a keyboard/auto wakeup interrupt input.

|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
| Read:  | 0     | AWUIE | KBIE5 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| Write: |       |       |       |       |       |       |       |       |
| Reset: | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

 = Unimplemented

**Figure 4-4. Keyboard Interrupt Enable Register (KBIER)**

### LVIRSTD — LVI Reset Disable Bit

LVIRSTD disables the reset signal from the LVI module.

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

### LVIPWRD — LVI Power Disable Bit

LVIPWRD disables the LVI module.

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

### LVITRIP — LVI Trip Point Selection Bit

LVITRIP selects the voltage operating mode of the LVI module. The voltage mode selected for the LVI should match the operating  $V_{DD}$  for the LVI's voltage trip points for each of the modes.

- 1 = LVI operates for a 5-V protection
- 0 = LVI operates for a 3-V protection

#### NOTE

*The LVITRIP bit is cleared by a power-on reset (POR) only. Other resets will leave this bit unaffected.*

### SSREC — Short Stop Recovery Bit

SSREC enables the CPU to exit stop mode with a delay of 32 BUSCLKX4 cycles instead of a 4096 BUSCLKX4 cycle delay.

- 1 = Stop mode recovery after 32 BUSCLKX4 cycles
- 0 = Stop mode recovery after 4096 BUSCLKX4 cycles

#### NOTE

*Exiting stop mode by an LVI reset will result in the long stop recovery.*

When using the LVI during normal operation but disabling during stop mode, the LVI will have an enable time of  $t_{EN}$ . The system stabilization time for power-on reset and long stop recovery (both 4096 BUSCLKX4 cycles) gives a delay longer than the LVI enable time for these startup scenarios. There is no period where the MCU is not protected from a low-power condition. However, when using the short stop recovery configuration option, the 32 BUSCLKX4 delay must be greater than the LVI's turn on time to avoid a period in startup where the LVI is not protecting the MCU.

### 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.

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

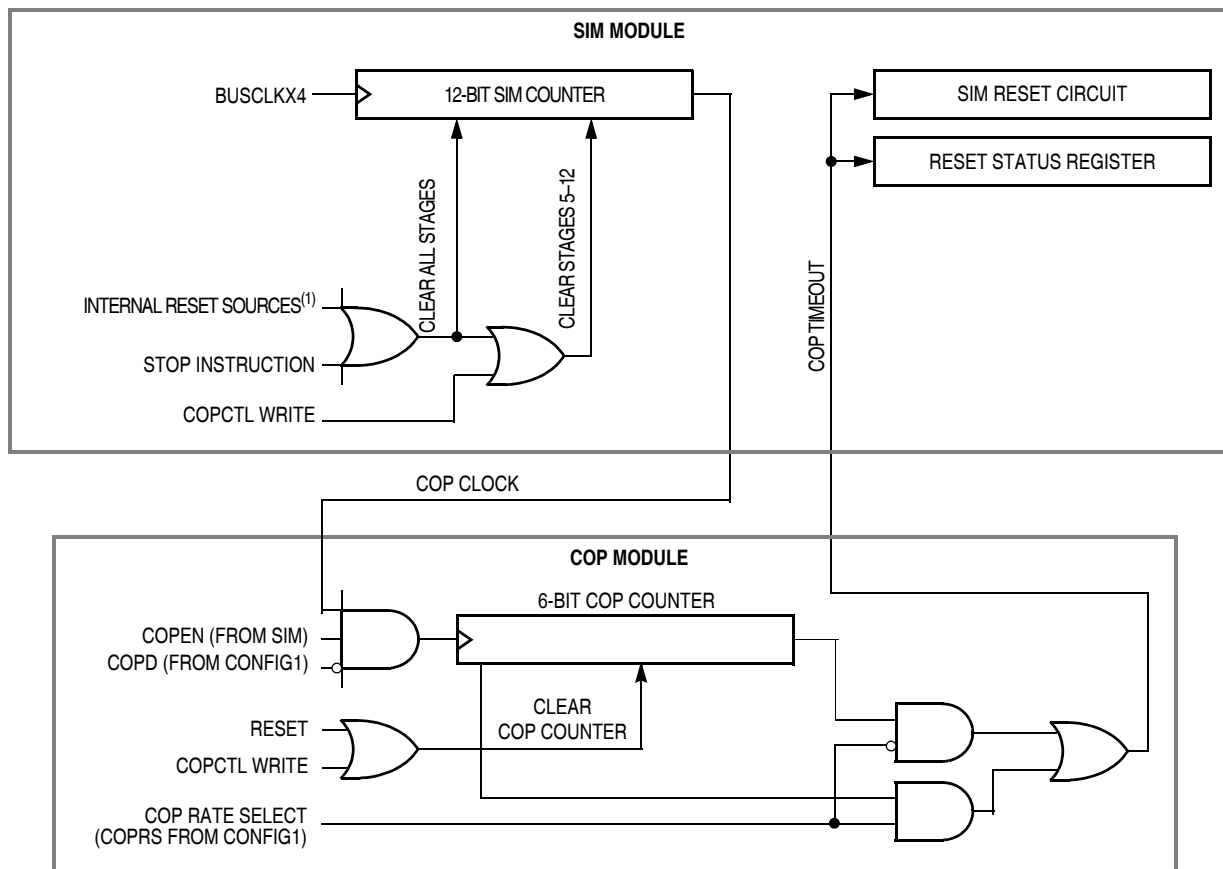
# Chapter 6

## Computer Operating Properly (COP)

### 6.1 Introduction

The computer operating properly (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 clearing the COP counter periodically. The COP module can be disabled through the COPD bit in the configuration 1 (CONFIG1) register.

### 6.2 Functional Description



1. See [Chapter 14 System Integration Module \(SIM\)](#) for more details.

**Figure 6-1. COP Block Diagram**

### 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

## 7.4 Arithmetic/Logic Unit (ALU)

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

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

## 7.5 Low-Power Modes

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

### 7.5.1 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

### 7.5.2 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.

## 7.6 CPU During Break Interrupts

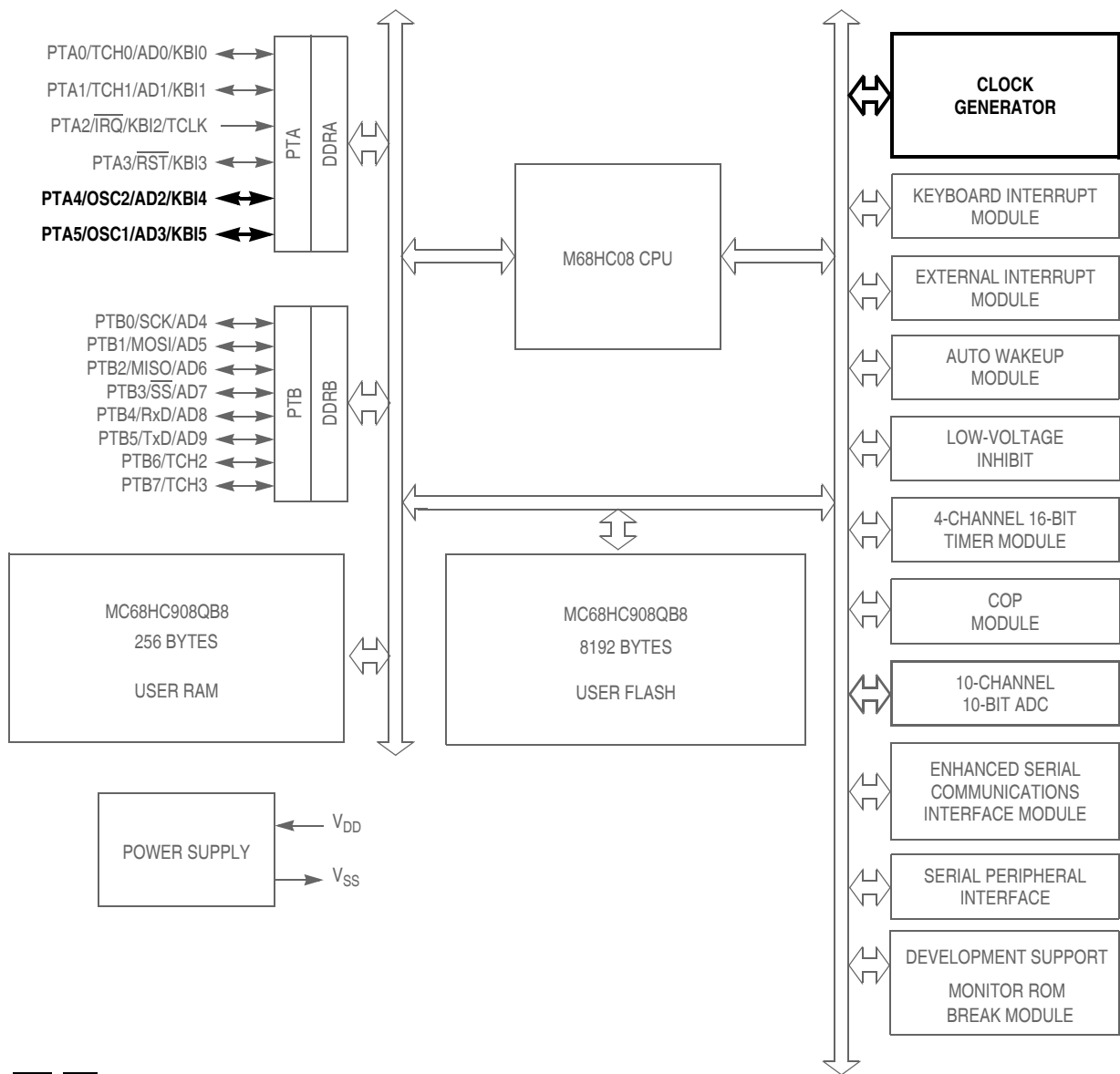
If a break module is present on the MCU, the CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with \$FFFC:\$FFFD or with \$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.

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.

# 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**

## 13.3 Functional Description

Figure 13-2 shows the structure of the ESCI module. The ESCI allows full-duplex, asynchronous, NRZ serial communication between the MCU and remote devices, including other MCUs. The transmitter and receiver of the ESCI operate independently, although they use the same baud rate generator.

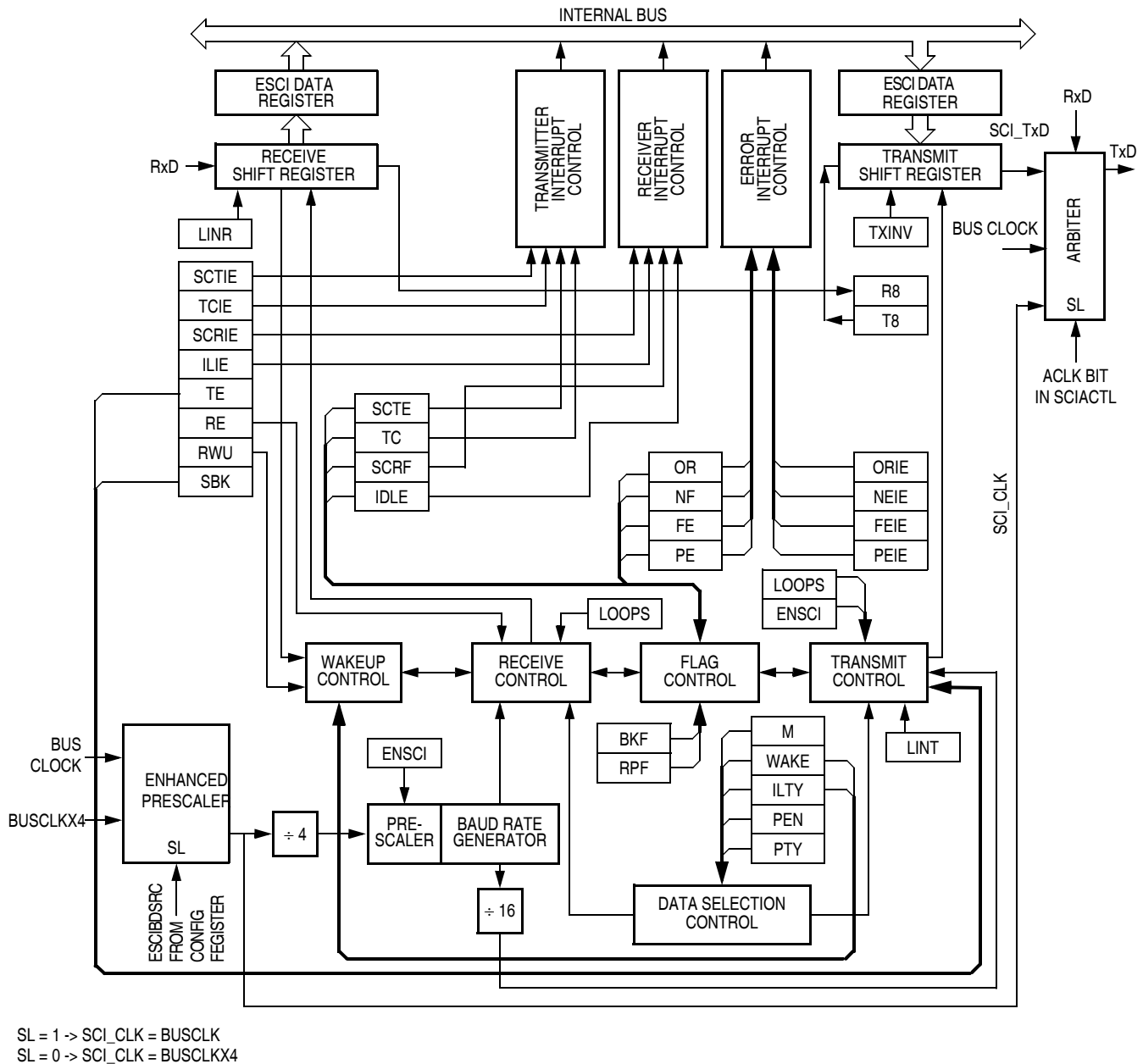


Figure 13-2. ESCI Module Block Diagram

### 13.3.2.1 Character Length

The transmitter can accommodate either 8-bit or 9-bit data. The state of the M bit in ESCI control register 1 (SCC1) determines character length. When transmitting 9-bit data, bit T8 in ESCI control register 3 (SCC3) is the ninth bit (bit 8).

### 13.3.2.2 Character Transmission

During an ESCI transmission, the transmit shift register shifts a character out to the TxD pin. The ESCI data register (SCDR) is the write-only buffer between the internal data bus and the transmit shift register.

To initiate an ESCI transmission:

1. Enable the ESCI by writing a 1 to the enable ESCI bit (ENSCI) in ESCI control register 1 (SCC1).
2. Enable the transmitter by writing a 1 to the transmitter enable bit (TE) in ESCI control register 2 (SCC2).
3. Clear the ESCI transmitter empty bit (SCTE) by first reading ESCI status register 1 (SCS1) and then writing to the SCDR. For 9-bit data, also write the T8 bit in SCC3.
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 1s. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A 0 start bit automatically goes into the least significant bit (LSB) position of the transmit shift register. A 1 stop bit goes into the most significant bit (MSB) position.

The ESCI 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 ESCI transmit interrupt enable bit, SCTIE, in SCC2 is also set, the SCTE bit generates a transmitter interrupt request.

When the transmit shift register is not transmitting a character, the TxD pin goes to the idle condition, high. If at any time software clears the ENSCI bit in ESCI control register 1 (SCC1), the transmitter and receiver relinquish control of the port pins.

### 13.3.2.3 Break Characters

Writing a 1 to the send break bit, SBK, in SCC2 loads the transmit shift register with a break character. For TXINV = 0 (output not inverted), a transmitted break character contains all 0s and has no start, stop, or parity bit. Break character length depends on the M bit in SCC1 and the LINR bits in SCBR. As long as SBK is set, 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 1. The automatic 1 at the end of a break character guarantees the recognition of the start bit of the next character.

When LINR is cleared in SCBR, the ESCI recognizes a break character when a start bit is followed by eight or nine 0 data bits and a 0 where the stop bit should be, resulting in a total of 10 or 11 consecutive 0 data bits. When LINR is set in SCBR, the ESCI recognizes a break character when a start bit is followed by 9 or 10 0 data bits and a 0 where the stop bit should be, resulting in a total of 11 or 12 consecutive 0 data bits.

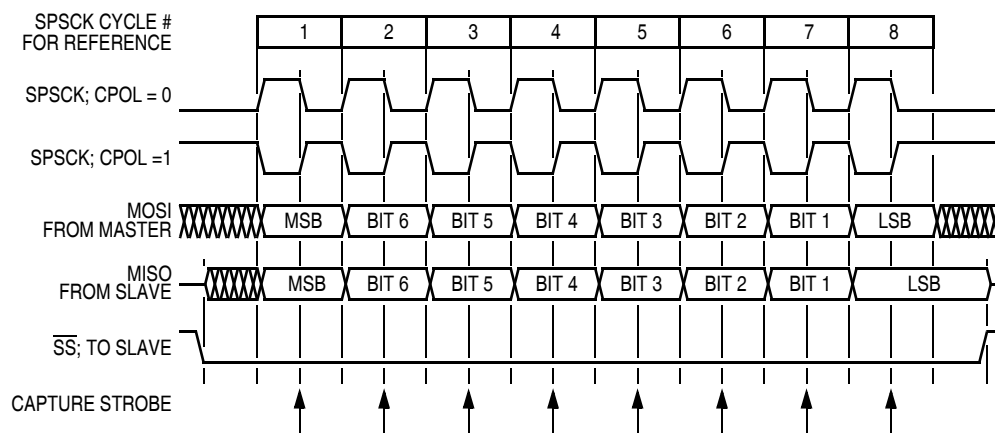
## 13.9.4 Arbitration Mode

If AM[1:0] is set to 10, the arbiter module operates in arbitration mode. On every rising edge of SCI\_TxD (output of the transmit shift register, see [Figure 13-2](#)), the counter is started. When the counter reaches \$38 (ACLK = 0) or \$08 (ACLK = 1), RxD is statically sensed. If in this case, RxD is sensed low (for example, another bus is driving the bus dominant) ALOST is set. As long as ALOST is set, the TxD pin is forced to 1, resulting in a seized transmission.

If SCI\_TxD senses 0 without having sensed a 0 before on RxD, the counter will be reset, arbitration operation will be restarted after the next rising edge of SCI\_TxD.



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 15-6. Transmission Format (CPHA = 1)**

When CPHA = 1 for a slave, the first edge of the SPSCCK 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. After the transmission begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the first edge of SPSCCK. Any data written after the first edge is stored in the transmit data register and transferred to the shift register after the current transmission.

#### 15.3.3.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), writing to the SPDR starts a transmission. CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SPSCCK signal. When CPHA = 0, the SPSCCK signal remains inactive for the first half of the first SPSCCK cycle. When CPHA = 1, the first SPSCCK cycle begins with an edge on the SPSCCK 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 15-7](#).) The internal SPI clock in the master is a free-running derivative of the internal MCU clock. To conserve power, it is enabled only when both the SPE and SPMSTR bits are set. Because the SPI clock is free-running, it is uncertain where the write to the SPDR occurs relative to the slower SPSCCK. This uncertainty causes the variation in the initiation delay shown in [Figure 15-7](#). This delay is no longer than a single SPI bit time. That is, the maximum delay 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.

## 15.3.6.2 Mode Fault Error

Setting SPMSTR selects master mode and configures the SPSCCK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode and configures the SPSCCK and MOSI pins as inputs and the MISO pin as an output. The mode fault bit, MODF, becomes set any time the state of the slave select pin,  $\overline{SS}$ , is inconsistent with the mode selected by SPMSTR.

To prevent SPI pin contention and damage to the MCU, a mode fault error occurs if:

- The  $\overline{SS}$  pin of a slave SPI goes high during a transmission
- The  $\overline{SS}$  pin of a master SPI goes low at any time

For the MODF flag to be set, the mode fault error enable bit (MODFEN) 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 interrupt request if the error interrupt enable bit (ERRIE) is also set. The SPRF, MODF, and OVRF interrupts share the same interrupt vector. (See [Figure 15-11](#).) It is not possible to enable MODF or OVRF individually to generate a receiver/error interrupt request. However, leaving MODFEN low prevents MODF from being set.

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if  $\overline{SS}$  goes low. A mode fault in a master SPI causes the following events to occur:

- If ERRIE = 1, the SPI generates an SPI receiver/error 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 SPI bits of the data direction register of the shared I/O port before enabling the SPI.*

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 after the incoming SPSCCK goes to its idle level following 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 following the shift of the last data bit. See

[15.3.3 Transmission Formats](#).

### NOTE

*Setting the MODF flag does not clear the SPMSTR bit. SPMSTR has no function when SPE = 0. Reading SPMSTR when MODF = 1 shows the difference between a MODF occurring when the SPI is a master and when it is a slave.*

*When CPHA = 0, a MODF occurs if a slave is selected ( $\overline{SS}$  is low) and later unselected ( $\overline{SS}$  is high) even if no SPSCCK is sent to that slave. This happens because  $\overline{SS}$  low 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 because a transmission was never begun.*

### 15.8.3 SPI Data Register

The SPI data register consists of the read-only receive data register and the write-only 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 registers that can contain different values. See [Figure 15-2](#).

|        | 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: | Unaffected by reset |    |    |    |    |    |    |       |

**Figure 15-15. 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 because the register read is not the same as the register written.*

## 16.7 I/O Signals

The TIM module can share its pins with the general-purpose I/O pins. See [Figure 16-1](#) for the port pins that are shared.

### 16.7.1 TIM Channel I/O Pins (TCH3:TCH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. TCH0 and TCH2 can be configured as buffered output compare or buffered PWM pins.

### 16.7.2 TIM Clock Pin (TCLK)

TCLK is an external clock input that can be the clock source for the counter instead of the prescaled internal bus clock. Select the TCLK input by writing 1s to the three prescaler select bits, PS[2:0]. (See [16.8.1 TIM Status and Control Register](#).) The minimum TCLK pulse width is specified in [18.16 Timer Interface Module Characteristics](#). The maximum TCLK frequency is the least of 4 MHz or bus frequency  $\div 2$ .

## 16.8 Registers

The following registers control and monitor operation of the TIM:

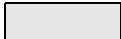
- TIM status and control register (TSC)
- TIM control registers (TCNTH:TCNTL)
- TIM counter modulo registers (TMODH:TMODL)
- TIM channel status and control registers (TSC0 through TSC3)
- TIM channel registers (TCH0H:TCH0L through TCH3H:TCH3L)

### 16.8.1 TIM Status and Control Register

The TIM status and control register (TSC) does the following:

- Enables TIM overflow interrupts
- Flags TIM overflows
- Stops the counter
- Resets the counter
- Prescales the counter clock

|        | Bit 7 | 6    | 5     | 4    | 3 | 2   | 1   | Bit 0 |
|--------|-------|------|-------|------|---|-----|-----|-------|
| Read:  | TOF   | TOIE | TSTOP | 0    | 0 | PS2 | PS1 | PS0   |
| Write: | 0     |      |       | TRST |   |     |     |       |
| Reset: | 0     | 0    | 1     | 0    | 0 | 0   | 0   | 0     |

 = Unimplemented

**Figure 16-4. TIM Status and Control Register (TSC)**

#### TOF — TIM Overflow Flag Bit

This read/write flag is set when the counter reaches the modulo value programmed in the TIM counter modulo registers. Clear TOF by reading the TSC register when TOF is set and then writing a 0 to TOF. If another TIM overflow occurs before the clearing sequence is complete, then writing 0 to TOF has no

Writing a 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 TIM interrupt service requests on channel x.

- 1 = Channel x interrupt requests enabled
- 0 = Channel x interrupt requests disabled

#### MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TSC0 and TSC2 registers.

Setting MS0B causes the contents of TSC1 to be ignored by the TIM and reverts TCH1 to general-purpose I/O.

Setting MS2B causes the contents of TSC3 to be ignored by the TIM and reverts TCH3 to general-purpose I/O.

- 1 = Buffered output compare/PWM operation enabled
- 0 = Buffered output compare/PWM operation disabled

#### MSxA — Mode Select Bit A

When ELSxB:A ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. See [Table 16-2](#).

- 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 (see [Table 16-2](#)).

- 1 = Initial output level low
- 0 = Initial output level high

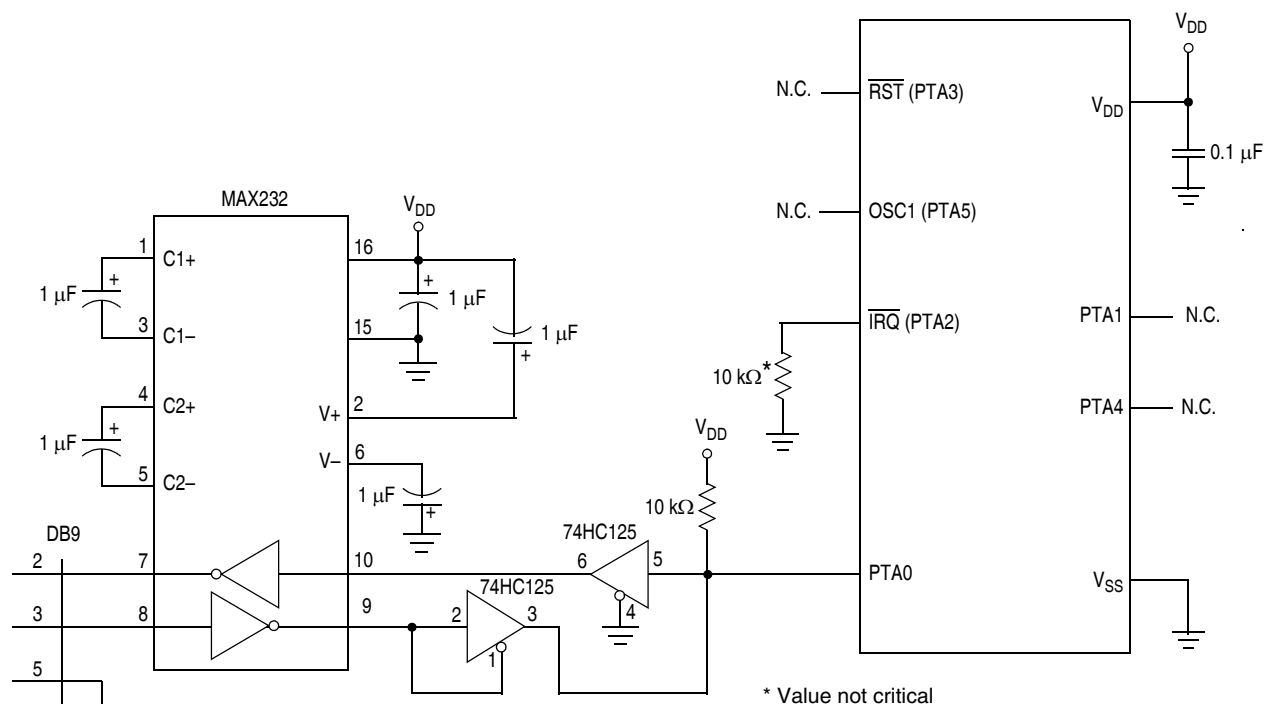
#### NOTE

*Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIM status and control register (TSC).*

**Table 16-2. Mode, Edge, and Level Selection**

| MSxB | MSxA | ELSxB | ELSxA | Mode                                    | Configuration                                     |
|------|------|-------|-------|-----------------------------------------|---------------------------------------------------|
| X    | 0    | 0     | 0     | Output preset                           | Pin under port control; initial output level high |
| X    | 1    | 0     | 0     |                                         | Pin under port control; initial output level low  |
| 0    | 0    | 0     | 1     | Input capture                           | Capture on rising edge only                       |
| 0    | 0    | 1     | 0     |                                         | Capture on falling edge only                      |
| 0    | 0    | 1     | 1     |                                         | Capture on rising or falling edge                 |
| 0    | 1    | 0     | 0     | Output compare or PWM                   | Software compare only                             |
| 0    | 1    | 0     | 1     |                                         | Toggle output on compare                          |
| 0    | 1    | 1     | 0     |                                         | Clear output on compare                           |
| 0    | 1    | 1     | 1     |                                         | Set output on compare                             |
| 1    | X    | 0     | 1     | Buffered output compare or buffered PWM | Toggle output on compare                          |
| 1    | X    | 1     | 0     |                                         | Clear output on compare                           |
| 1    | X    | 1     | 1     |                                         | Set output on compare                             |





**Figure 17-12. Monitor Mode Circuit (Internal Clock, No High Voltage)**

The monitor code has been updated from previous versions of the monitor code to allow enabling the internal oscillator to generate the internal clock. This addition, which is enabled when  $\overline{\text{IRQ}}$  is held low out of reset, is intended to support serial communication/programming at 9600 baud in monitor mode by using the internal oscillator, and the internal oscillator user trim value  $\text{OSCTRIM}$  (FLASH location \$FFC0, if programmed) to generate the desired internal frequency (3.2 MHz). Since this feature is enabled only when  $\overline{\text{IRQ}}$  is held low out of reset, it cannot be used when the reset vector is programmed (i.e., the value is not \$FFFF) because entry into monitor mode in this case requires  $V_{\text{TST}}$  on  $\overline{\text{IRQ}}$ . The  $\overline{\text{IRQ}}$  pin must remain low during this monitor session in order to maintain communication.

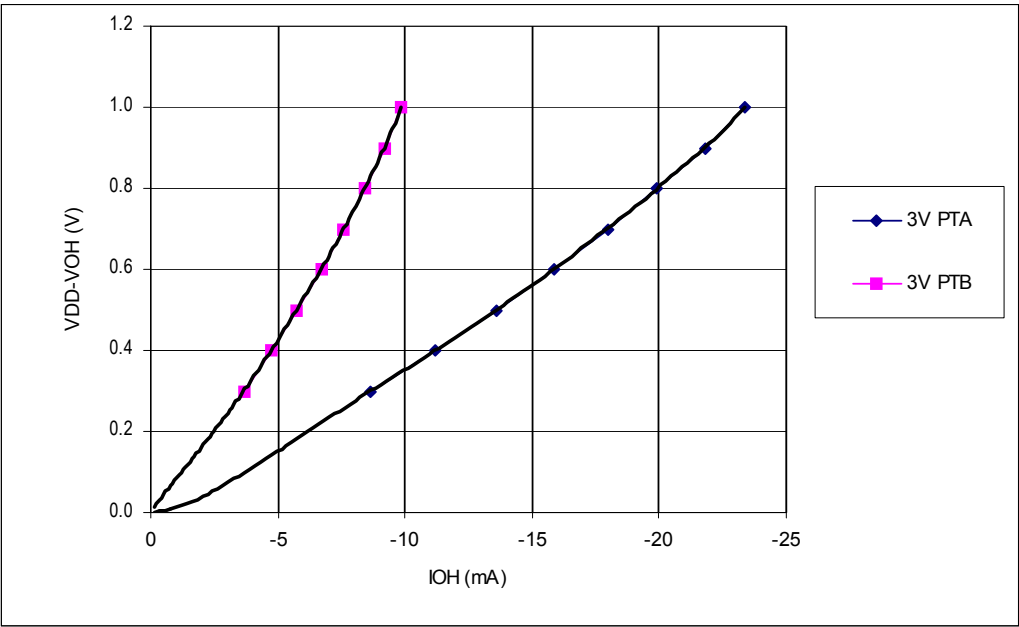
Table 17-1 shows the pin conditions for entering monitor mode. As specified in the table, monitor mode may be entered after a power-on reset (POR) and will allow communication at 9600 baud provided one of the following sets of conditions is met:

- If \$FFE and \$FFF do not contain \$FF (programmed state):
  - The external clock is 9.8304 MHz
  - $\overline{\text{IRQ}} = V_{\text{TST}}$
- If \$FFE and \$FFF contain \$FF (erased state):
  - The external clock is 9.8304 MHz
  - $\overline{\text{IRQ}} = V_{\text{DD}}$  (this can be implemented through the internal  $\overline{\text{IRQ}}$  pullup)
- If \$FFE and \$FFF contain \$FF (erased state):
  - $\overline{\text{IRQ}} = V_{\text{SS}}$  (internal oscillator is selected, no external clock required)

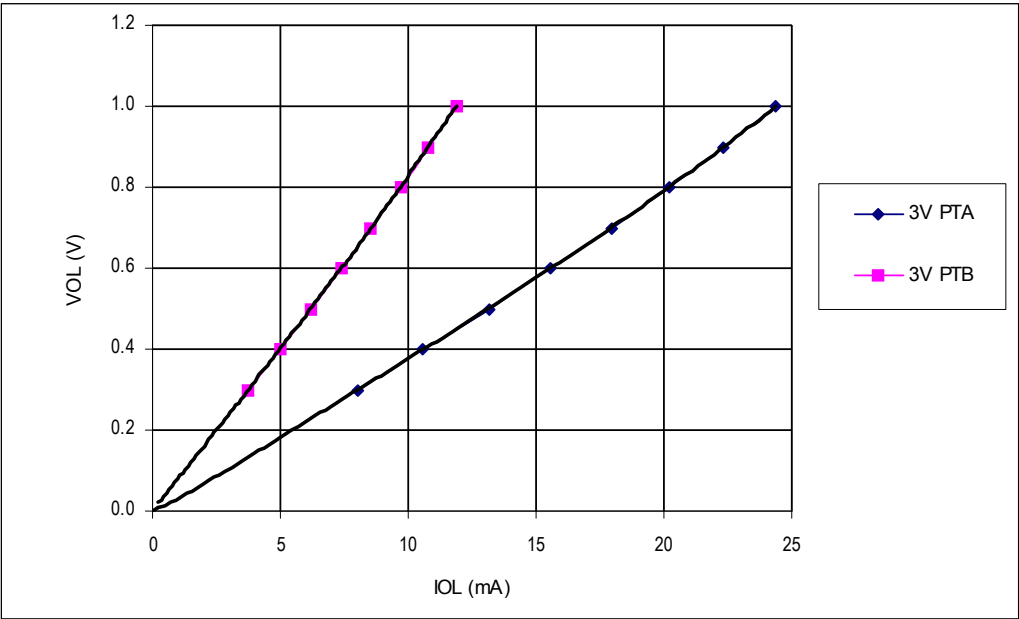
The rising edge of the internal  $\overline{\text{RST}}$  signal latches the monitor mode. Once monitor mode is latched, the values on PTA1 and PTA4 pins can be changed.

Once out of reset, the MCU waits for the host to send eight security bytes (see [17.3.2 Security](#)). After the security bytes, the MCU sends a break signal (10 consecutive 0s) to the host, indicating that it is ready to receive a command.

# 18.9 Typical 3-V Output Drive Characteristics



**Figure 18-4. Typical 3-Volt Output High Voltage versus Output High Current (25°C)**



**Figure 18-5. Typical 3-Volt Output Low Voltage versus Output Low Current (25°C)**

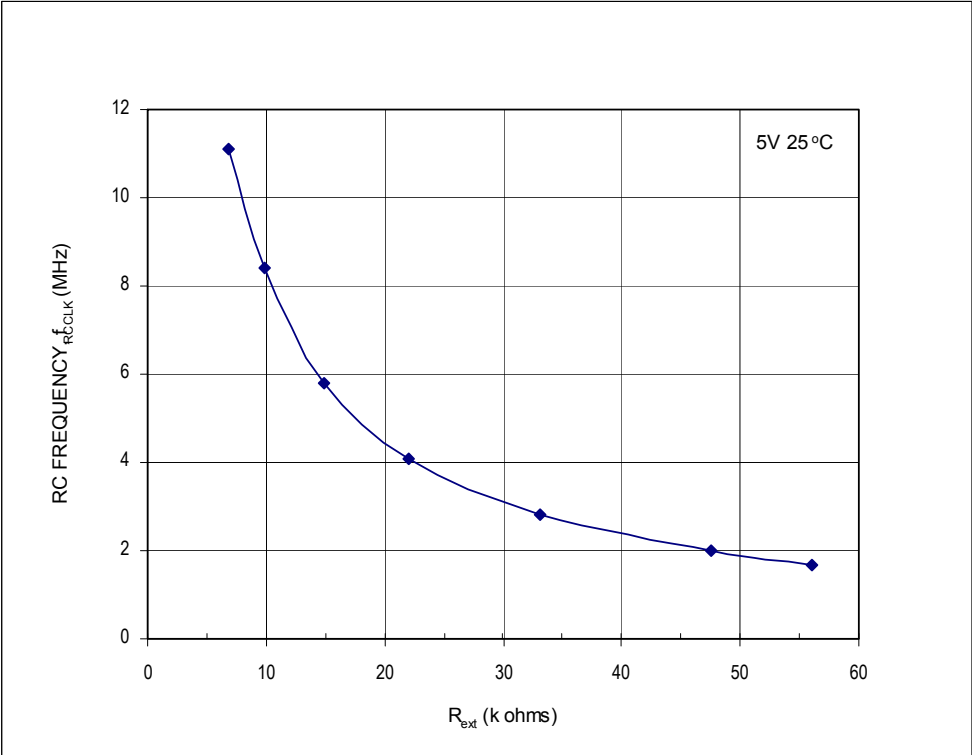


Figure 18-7. RC versus Frequency (5 Volts @ 25°C)

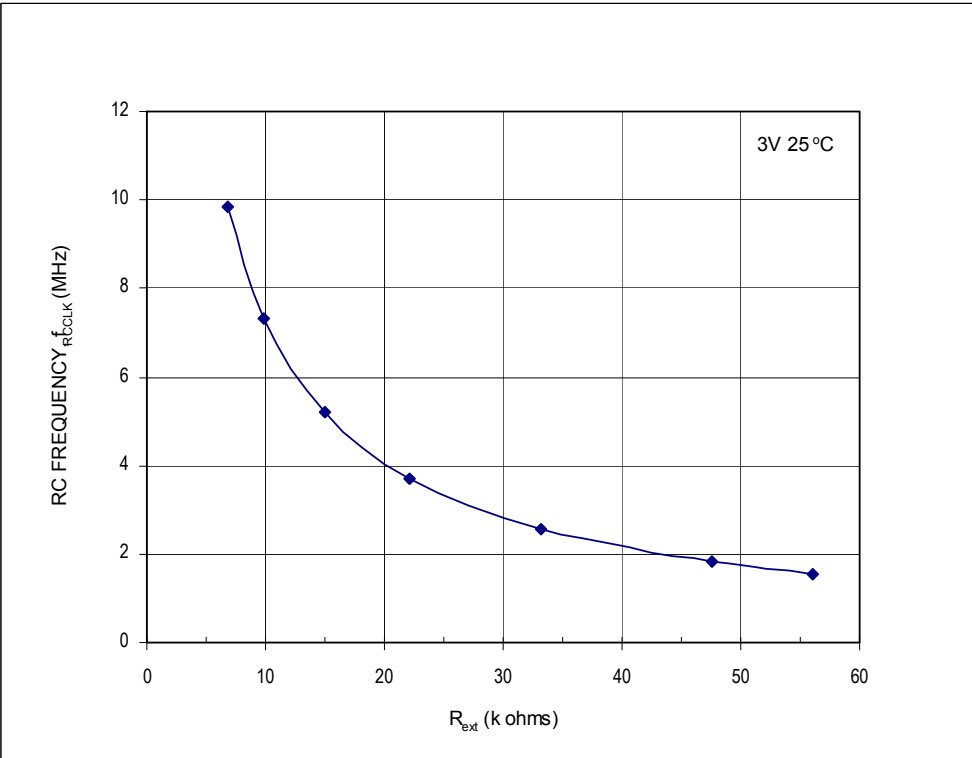


Figure 18-8. RC versus Frequency (3 Volts @ 25°C)