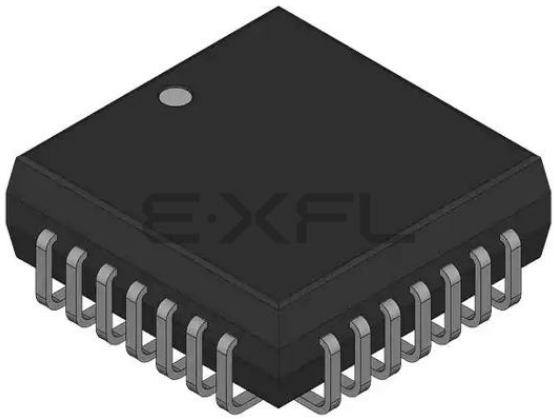


**Welcome to E-XFL.COM**

## 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 | Active |
| Core Processor | HC08 |
| Core Size | 8-Bit |
| Speed | 8MHz |
| Connectivity | I²C, IRSCI, SCI, SPI |
| Peripherals | LED, LVD, POR, PWM |
| Number of I/O | 32 |
| Program Memory Size | 64KB (64K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 2K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.5V |
| Data Converters | A/D 8x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 48-LQFP |
| Supplier Device Package | 48-LQFP (7x7) |
| Purchase URL | https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc908ap64cfae |

**Table 1-2. Pin Functions**

| PIN NAME | PIN DESCRIPTION | IN/OUT | VOLTAGE LEVEL |
|---|---|---|---|
| PTB0/SDA<br>PTB1/SCL<br>PTB2/TxD<br>PTB3/RxD<br>PTB4/T1CH0<br>PTB5/T1CH1<br>PTB6/T2CH0<br>PTB7/T2CH1 | 8-bit general purpose I/O port; PTB0–PTB3 are open drain when configured as output. PTB4–PTB7 have schmitt trigger inputs. | In/Out | $V_{DD}$ |
| | PTB0 as SDA of MMIIC. | In/Out | $V_{DD}$ |
| | PTB1 as SCL of MMIIC. | In/Out | $V_{DD}$ |
| | PTB2 as TxD of SCI; open drain output. | Out | $V_{DD}$ |
| | PTB3 as RxD of SCI. | In | $V_{DD}$ |
| | PTB4 as T1CH0 of TIM1. | In/Out | $V_{DD}$ |
| | PTB5 as T1CH1 of TIM1. | In/Out | $V_{DD}$ |
| | PTB6 as T2CH0 of TIM2. | In/Out | $V_{DD}$ |
| | PTB7 as T2CH1 of TIM2. | In/Out | $V_{DD}$ |
| PTC0/$\overline{IRQ2}$<br>PTC1<br>PTC2/MISO<br>PTC3/MOSI<br>PTC4/$\overline{SS}$<br>PTC5/SPSCK<br>PTC6/SCTxD<br>PTC7/SCRxD | 8-bit general purpose I/O port; PTC6 and PTC7 are open drain when configured as output. | In/Out | $V_{DD}$ |
| | PTC0 is shared with $\overline{IRQ2}$ and has schmitt trigger input. | In | $V_{DD}$ |
| | PTC2 as MISO of SPI. | In | $V_{DD}$ |
| | PTC3 as MOSI of SPI. | Out | $V_{DD}$ |
| | PTC4 as $\overline{SS}$ of SPI. | In | $V_{DD}$ |
| | PTC5 as SPSCK of SPI. | In/Out | $V_{DD}$ |
| | PTC6 as SCTxD of IRSCI; open drain output. | Out | $V_{DD}$ |
| | PTC7 as SCRxD of IRSCI. | In | $V_{DD}$ |
| PTD0/KBI0<br>:<br>PTD7/KBI7 | 8-bit general purpose I/O port with schmitt trigger inputs. | In/Out | $V_{DD}$ |
| | Pins as keyboard interrupts (with pullup), KBI0–KBI7. | In | $V_{DD}$ |

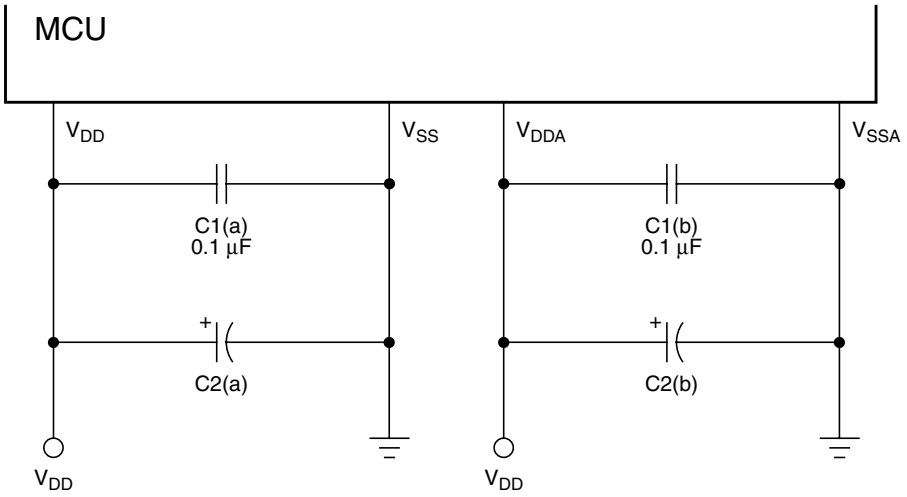
1. See Chapter 22 Electrical Specifications for $V_{REG}$ tolerance.

## 1.6 Power Supply Bypassing (VDD, VDDA, VSS, VSSA)

$V_{DD}$ and $V_{SS}$ are the power supply and ground pins, the MCU operates from a single power supply together with an on chip voltage regulator.

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 Figure 1-5 shows. Place the bypass capacitors as close to the MCU power pins as possible. Use high-frequency-response ceramic capacitor for $C_{BYPASS}$, $C_{BULK}$ are optional bulk current bypass capacitors for use in applications that require the port pins to source high current level.

$V_{DDA}$ and $V_{SSA}$ are the power supply and ground pins for the analog circuits of the MCU. These pins should be decoupled as per the digital power supply pins.




NOTE: Component values shown represent typical applications.

**Figure 1-5. Power Supply Bypassing**

## 1.7 Regulator Power Supply Configuration (VREG)

$V_{REG}$ is the output from the on-chip regulator. All internal logics, except for the I/O pads, are powered by $V_{REG}$ output. $V_{REG}$ requires an external ceramic bypass capacitor of 100 nF as Figure 1-6 shows. Place the bypass capacitor as close to the $V_{REG}$ pin as possible.

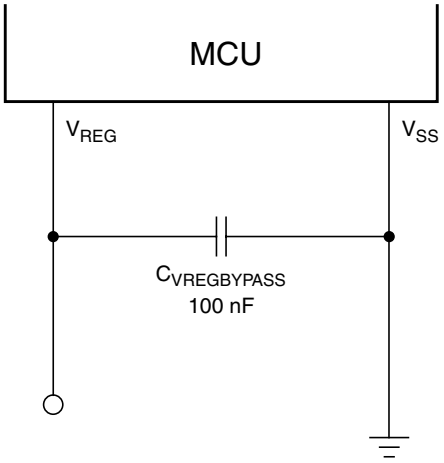



**Figure 1-6. Regulator Power Supply Bypassing**

## Table 2-1. Vector Addresses

| Priority | INT Flag | Address | Vector |
|---|---|---|---|
| Lowest | — | $FFD0 | Reserved |
| | | $FFD1 | Reserved |
| | IF21 | $FFD2 | TBM Vector (High) |
| | | $FFD3 | TBM Vector (Low) |
| | IF20 | $FFD4 | SCI2 (IRSCI) Transmit Vector (High) |
| | | $FFD5 | SCI2 (IRSCI) Transmit Vector (Low) |
| | IF19 | $FFD6 | SCI2 (IRSCI) Receive Vector (High) |
| | | $FFD7 | SCI2 (IRSCI) Receive Vector (Low) |
| | IF18 | $FFD8 | SCI2 (IRSCI) Error Vector (High) |
| | | $FFD9 | SCI2 (IRSCI) Error Vector (Low) |
| | IF17 | $FFDA | SPI Transmit Vector (High) |
| | | $FFDB | SPI Transmit Vector (Low) |
| | IF16 | $FFDC | SPI Receive Vector (High) |
| | | $FFDD | SPI Receive Vector (Low) |
| | IF15 | $FFDE | ADC Conversion Complete Vector (High) |
| | | $FFDF | ADC Conversion Complete Vector (Low) |
| | IF14 | $FFE0 | Keyboard Vector (High) |
| | | $FFE1 | Keyboard Vector (Low) |
| | IF13 | $FFE2 | SCI Transmit Vector (High) |
| | | $FFE3 | SCI Transmit Vector (Low) |
| | IF12 | $FFE4 | SCI Receive Vector (High) |
| | | $FFE5 | SCI Receive Vector (Low) |
| | IF11 | $FFE6 | SCI Error Vector (High) |
| | | $FFE7 | SCI Error Vector (Low) |
| | IF10 | $FFE8 | MMIIC Interrupt Vector (High) |
| | | $FFE9 | MMIIC Interrupt Vector (Low) |
| | IF9 | $FFEA | TIM2 Overflow Vector (High) |
| | | $FFEB | TIM2 Overflow Vector (Low) |

## 3.2 Functional Description

The configuration registers and the mask option register are used in the initialization of various options. These two types of registers are configured differently:

- Configuration registers — Write-once registers after reset
- Mask option register — FLASH register (write by programming)

The configuration registers can be written once after each reset. All of the configuration register bits are cleared during reset. Since the various options affect the operation of the MCU, it is recommended that these registers be written immediately after reset. The configuration registers are located at $001D and $001F. The configuration registers may be read at anytime.

***NOTE***

*The CONFIG registers are not in the FLASH memory but are special registers containing one-time writable latches after each reset. Upon a reset, the CONFIG registers default to predetermined settings as shown in Figure 3-2 and Figure 3-3.*

The mask option register (MOR) is used for selecting one of the three clock options for the MCU. The MOR is a byte located in FLASH memory, and is written to by a FLASH programming routine.

## 3.3 Configuration Register 1 (CONFIG1)

Address: $001F

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | COPRS | LVISTOP | LVIRSTD | LVIPWRD | LVIREGD | SSREC | STOP | COPD |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 3-2. Configuration Register 1 (CONFIG1)**

**COPRS — COP Rate Select Bit**

COPRS selects the COP time out period. Reset clears COPRS. (See Chapter 19 Computer Operating Properly (COP).)

1 = COP time out period = $2^{13} – 2^4$ ICLK cycles
0 = COP time out period = $2^{18} – 2^4$ ICLK cycles

**LVISTOP — LVI Enable in Stop Mode Bit**

When the LVIPWRD or LVIREGD bit is clear, setting the LVISTOP bit enables the LVI to operate during stop mode. Reset clears LVISTOP. (See Chapter 20 Low-Voltage Inhibit (LVI).)

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

***NOTE***

*If LVISTOP=0, set LVIRSTD=1 before entering stop mode.*

**LVIRSTD — LVI Reset Disable Bit**

LVIRSTD disables the reset signal from the LVI module. (See Chapter 20 Low-Voltage Inhibit (LVI).)

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

### 6.4.3 PLL Analog Ground Pin ($V_{SSA}$)

$V_{SSA}$ is a ground pin used by the analog portions of the PLL. Connect the $V_{SSA}$ pin to the same voltage potential as the $V_{SS}$ pin.

***NOTE***

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

### 6.4.4 Oscillator Output Frequency Signal (CGMXCLK)

CGMXCLK is the oscillator output signal. It runs at the full speed of the oscillator, and is generated directly from the crystal oscillator circuit, the RC oscillator circuit, or the internal oscillator circuit.

### 6.4.5 CGM Reference Clock (CGMRCLK)

CGMRCLK is a buffered version of CGMXCLK, this clock is the reference clock for the phase-locked-loop circuit.

### 6.4.6 CGM VCO Clock Output (CGMVCLK)

CGMVCLK is the clock output from the VCO.

### 6.4.7 CGM Base Clock Output (CGMOUT)

CGMOUT is the clock output of the CGM. This signal goes to the SIM, which generates the MCU clocks. CGMOUT is a 50 percent 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 divided VCO clock, CGMPCLK, divided by two.

### 6.4.8 CGM CPU Interrupt (CGMINT)

CGMINT is the interrupt signal generated by the PLL lock detector.

## 6.5 CGM Registers

The following registers control and monitor operation of the CGM:

- PLL control register (PCTL)
  (See 6.5.1 PLL Control Register.)
- PLL bandwidth control register (PBWC)
  (See 6.5.2 PLL Bandwidth Control Register.)
- PLL multiplier select registers (PMSH and PMSL)
  (See 6.5.3 PLL Multiplier Select Registers.)
- PLL VCO range select register (PMRS)
  (See 6.5.4 PLL VCO Range Select Register.)
- PLL reference divider select register (PMDS)
  (See 6.5.5 PLL Reference Divider Select Register.)

### 6.5.1 PLL Control Register

The PLL control register (PCTL) contains the interrupt enable and flag bits, the on/off switch, the base clock selector bit, the prescaler bits, and the VCO power-of-two range selector bits.

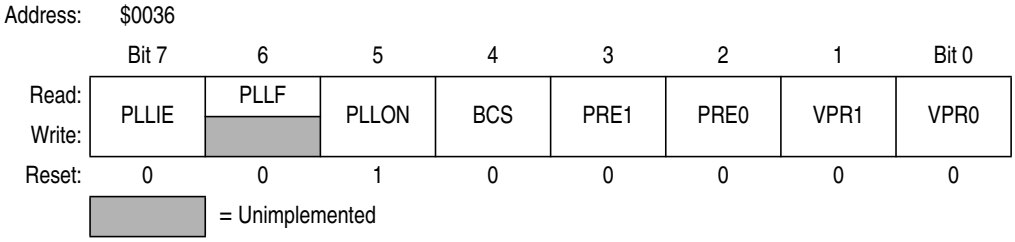



**Figure 6-4. PLL Control Register (PCTL)**

**PLLIE — PLL Interrupt Enable Bit**

This read/write bit enables the PLL to generate an 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 interrupts enabled
0 = PLL interrupts disabled

**PLLF — PLL Interrupt Flag Bit**

This read-only bit is set whenever the LOCK bit toggles. PLLF generates an 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. 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 6.3.8 Base Clock Selector Circuit.) 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 oscillator output, CGMXCLK, or the divided VCO clock, CGMPCLK, 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 CGMPCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. (See 6.3.8 Base Clock Selector Circuit.) Reset clears the BCS bit.

1 = CGMPCLK divided by two drives CGMOUT
0 = CGMXCLK divided by two drives CGMOUT

**Table 7-3. Interrupt Sources**

| Priority | INT Flag | Vector Address | Interrupt Source |
|---|---|---|---|
| Lowest | — | $FFD0 | Reserved |
| | | $FFD1 | |
| | IF21 | $FFD2 | Timebase |
| | | $FFD3 | |
| | IF20 | $FFD4 | Infrared SCI Transmit |
| | | $FFD5 | |
| | IF19 | $FFD6 | Infrared SCI Receive |
| | | $FFD7 | |
| | IF18 | $FFD8 | Infrared SCI Error |
| | | $FFD9 | |
| | IF17 | $FFDA | SPI Transmit |
| | | $FFDB | |
| | IF16 | $FFDC | SPI Receive |
| | | $FFDD | |
| | IF15 | $FFDE | ADC Conversion Complete |
| | | $FFDF | |
| | IF14 | $FFE0 | Keyboard |
| | | $FFE1 | |
| | IF13 | $FFE2 | SCI Transmit |
| | | $FFE3 | |
| | IF12 | $FFE4 | SCI Receive |
| | | $FFE5 | |
| | IF11 | $FFE6 | SCI Error |
| | | $FFE7 | |
| | IF10 | $FFE8 | MMIIC |
| | | $FFE9 | |
| | IF9 | $FFEA | TIM2 Overflow |
| | | $FFEB | |
| | IF8 | $FFEC | TIM2 Channel 1 |
| | | $FFED | |
| | IF7 | $FFEE | TIM2 Channel 0 |
| | | $FFEF | |
| | IF6 | $FFF0 | TIM1 Overflow |
| | | $FFF1 | |
| | IF5 | $FFF2 | TIM1 Channel 1 |
| | | $FFF3 | |
| | IF4 | $FFF4 | TIM1 Channel 0 |
| | | $FFF5 | |
| | IF3 | $FFF6 | PLL |
| | | $FFF7 | |
| | IF2 | $FFF8 | $\overline{\text{IRQ2}}$ |
| | | $FFF9 | |
| | IF1 | $FFFA | IRQ1 |
| | | $FFFB | |
| | — | $FFFC | SWI |
| | | $FFFD | |
| Highest | — | $FFFE | Reset |
| | | $FFFF | |

### 8.3.5 Commands

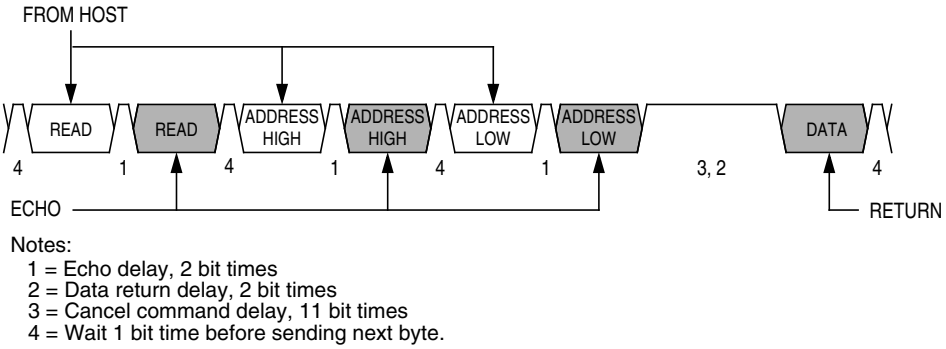
The monitor ROM firmware uses these commands:

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

The monitor ROM firmware echoes each received byte back to the PTA0 pin for error checking. An 11-bit delay at the end of each command allows the host to send a break character to cancel the command. A delay of two bit times occurs before each echo and before READ, IREAD, or READSP data is returned. The data returned by a read command appears after the echo of the last byte of the command.

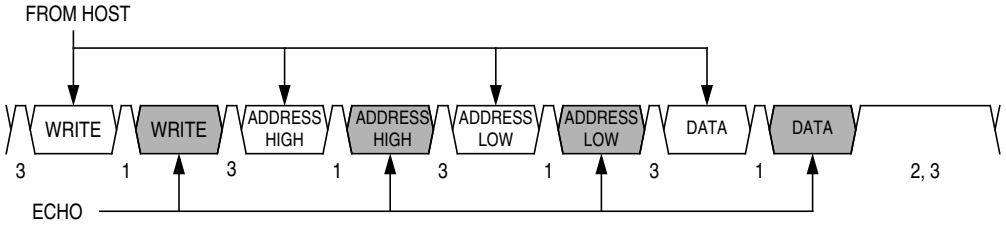
***NOTE***

*Wait one bit time after each echo before sending the next byte.*




Notes:
1 = Echo delay, 2 bit times
2 = Data return delay, 2 bit times
3 = Cancel command delay, 11 bit times
4 = Wait 1 bit time before sending next byte.

**Figure 8-5. Read Transaction**




Notes:
1 = Echo delay, 2 bit times
2 = Cancel command delay, 11 bit times
3 = Wait 1 bit time before sending next byte.

**Figure 8-6. Write Transaction**

A brief description of each monitor mode command is given in Table 8-4 through Table 8-9.

Upon power-on reset, if the received bytes of the security code 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 a FLASH location returns an invalid value and trying to execute code from FLASH causes an illegal address reset. After receiving the eight security bytes from the host, the MCU transmits a break character, signifying 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 bits.*

To determine whether the security code entered is correct, check to see if bit 6 of RAM address $60 is set. If it is, then the correct security code has been entered and FLASH can be accessed.

If the security sequence fails, the device should be reset by a power-on reset and brought up in monitor mode to attempt another entry. After failing the security sequence, the FLASH module can also be mass erased by executing an erase routine that was downloaded into internal RAM. The mass erase operation clears the security code locations so that all eight security bytes become $FF (blank).

## 8.5 ROM-Resident Routines

Seven routines stored in the monitor ROM area (thus ROM-resident) are provided for FLASH memory manipulation. Five of the seven routines are intended to simplify FLASH program, erase, and load operations. The other two routines are intended to simplify the use of the FLASH memory as EEPROM. Table 8-10 shows a summary of the ROM-resident routines.

**Table 8-10. Summary of ROM-Resident Routines**

| Routine Name | Routine Description | Call Address | Stack Used (bytes) |
|---|---|---|---|
| **PRGRNGE** | Program a range of locations | $FC34 | 15 |
| **ERARNGE** | Erase a page or the entire array | $FCE4 | 9 |
| **LDRNGE** | Loads data from a range of locations | $FC00 | 7 |
| **MON_PRGRNGE** | Program a range of locations in monitor mode | $FF24 | 17 |
| **MON_ERARNGE** | Erase a page or the entire array in monitor mode | $FF28 | 11 |
| **EE_WRITE** | Emulated EEPROM write. Data size ranges from 7 to 15 bytes at a time. | $FF36 | 30 |
| **EE_READ** | Emulated EEPROM read. Data size ranges from 7 to 15 bytes at a time. | $FD5B | 18 |

The routines are designed to be called as stand-alone subroutines in the user program or monitor mode. The parameters that are passed to a routine are in the form of a contiguous data block, stored in RAM. The index register (H:X) is loaded with the address of the first byte of the data block (acting as a pointer), and the subroutine is called (JSR). Using the start address as a pointer, multiple data blocks can be used, any area of RAM be used. A data block has the control and data bytes in a defined order, as shown in Figure 8-9.

$00FF (255) to the TIM counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000. See 9.9.1 TIM Status and Control Register.

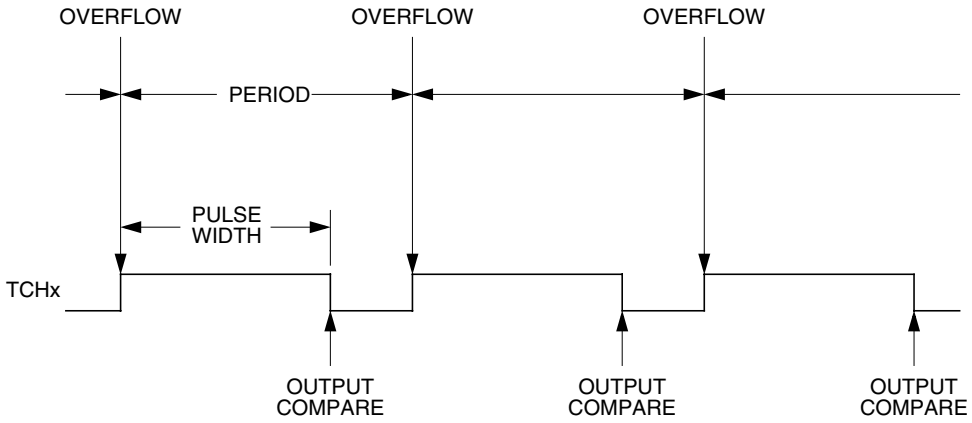



**Figure 9-3. PWM Period and Pulse Width**

The value in the TIM 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 TIM channel registers produces a duty cycle of 128/256 or 50%.

#### *9.4.4.1 Unbuffered PWM Signal Generation*

Any output compare channel can generate unbuffered PWM pulses as described in 9.4.4 Pulse Width Modulation (PWM). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM channel registers.

An unsynchronized write to the TIM 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 TIM overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIM may pass the new value before it is written.

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 TIM overflow interrupts and write the new value in the TIM overflow interrupt routine. The TIM 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.*

***NOTE***

*Before enabling a TIM channel register for input capture operation, make sure that the TCHx 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 TIM 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 TIM counter overflow

0 = Channel x pin does not toggle on TIM counter overflow

***NOTE***

*When TOVx is set, a TIM counter overflow takes precedence over a channel x output compare if both occur at the same time.*

**CHxMAX — Channel x Maximum Duty Cycle Bit**

When the TOVx bit is at logic 1, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 9-11 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.

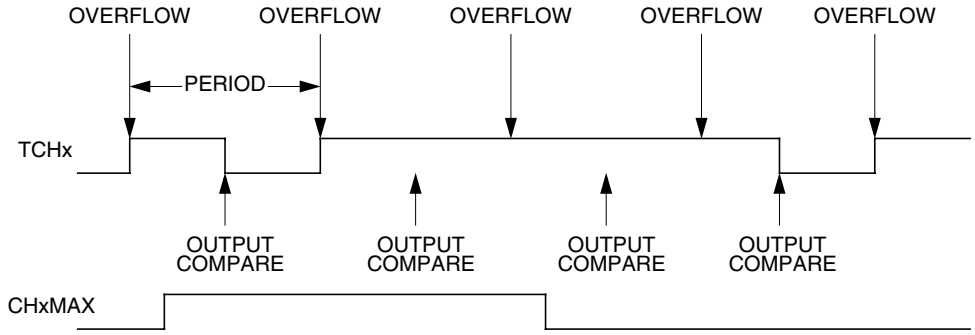



**Figure 9-11. CHxMAX Latency**

## 9.9.5 TIM Channel Registers

These read/write registers contain the captured TIM counter value of the input capture function or the output compare value of the output compare function. The state of the TIM channel registers after reset is unknown.

In input capture mode (MSxB:MSxA = 0:0), reading the high byte of the TIM channel x registers (TCHxH) inhibits input captures until the low byte (TCHxL) is read.

In output compare mode (MSxB:MSxA ≠ 0:0), writing to the high byte of the TIM channel x registers (TCHxH) inhibits output compares until the low byte (TCHxL) is written.

#### 11.4.3.4 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.

#### 11.4.3.5 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 11-7 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.

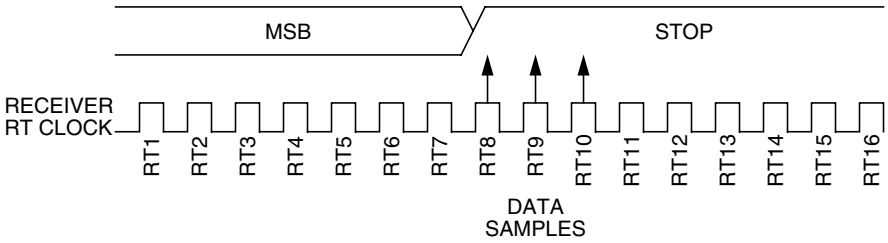



**Figure 11-7. Slow Data**

For an 8-bit character, data sampling of the stop bit takes the receiver
9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles.

With the misaligned character shown in Figure 11-7, the receiver counts 154 RT cycles at the point when the count of the transmitting device is
9 bit 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 × 16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in Figure 11-7, the receiver counts 170 RT cycles at the point when the count of the transmitting device is
10 bit times × 16 RT cycles + 3 RT cycles = 163 RT cycles.

$$\left|\frac{170 - 176}{170}\right| \times 100 = 3.53\%$$

#### 12.5.3.6 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 IRSCC2 puts the receiver into a standby state during which receiver interrupts are disabled.

Depending on the state of the WAKE bit in IRSCC1, 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**

*Clearing the WAKE bit after the RxD pin has been idle may cause the receiver to wake up immediately.*

#### 12.5.3.7 Receiver Interrupts

The following sources can generate CPU interrupt requests from the SCI receiver:

- SCI receiver full (SCRF) — The SCRF bit in IRSCS1 indicates that the receive shift register has transferred a character to the IRSCDR. SCRF can generate a receiver interrupt request. Setting the SCI receive interrupt enable bit, SCRIE, in IRSCC2 enables the SCRF bit to generate receiver CPU interrupts.
- Idle input (IDLE) — The IDLE bit in IRSCS1 indicates that 10 or 11 consecutive logic 1s shifted in from the RxD pin. The idle line interrupt enable bit, ILIE, in IRSCC2 enables the IDLE bit to generate CPU interrupt requests.

#### 12.5.3.8 Error Interrupts

The following receiver error flags in IRSCS1 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 IRSCDR. The previous character remains in the IRSCDR, and the new character is lost. The overrun interrupt enable bit, ORIE, in IRSCC3 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 IRSCC3 enables NF to generate SCI error CPU interrupt requests.

***NOTE***

*Writing to the TE bit is not allowed when the enable SCI bit (ENSCI) is clear. ENSCI is in SCI 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. 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 IRSCC1 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.*

### 12.9.3 IRSCI Control Register 3

IRSCI 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

Address: $0042

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | R8 | T8 | DMARE | DMATE | ORIE | NEIE | FEIE | PEIE |
| Write: | | | | | | | | |
| Reset: | U | U | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented U = Unaffected

**Figure 12-14. IRSCI Control Register 3 (IRSCC3)**

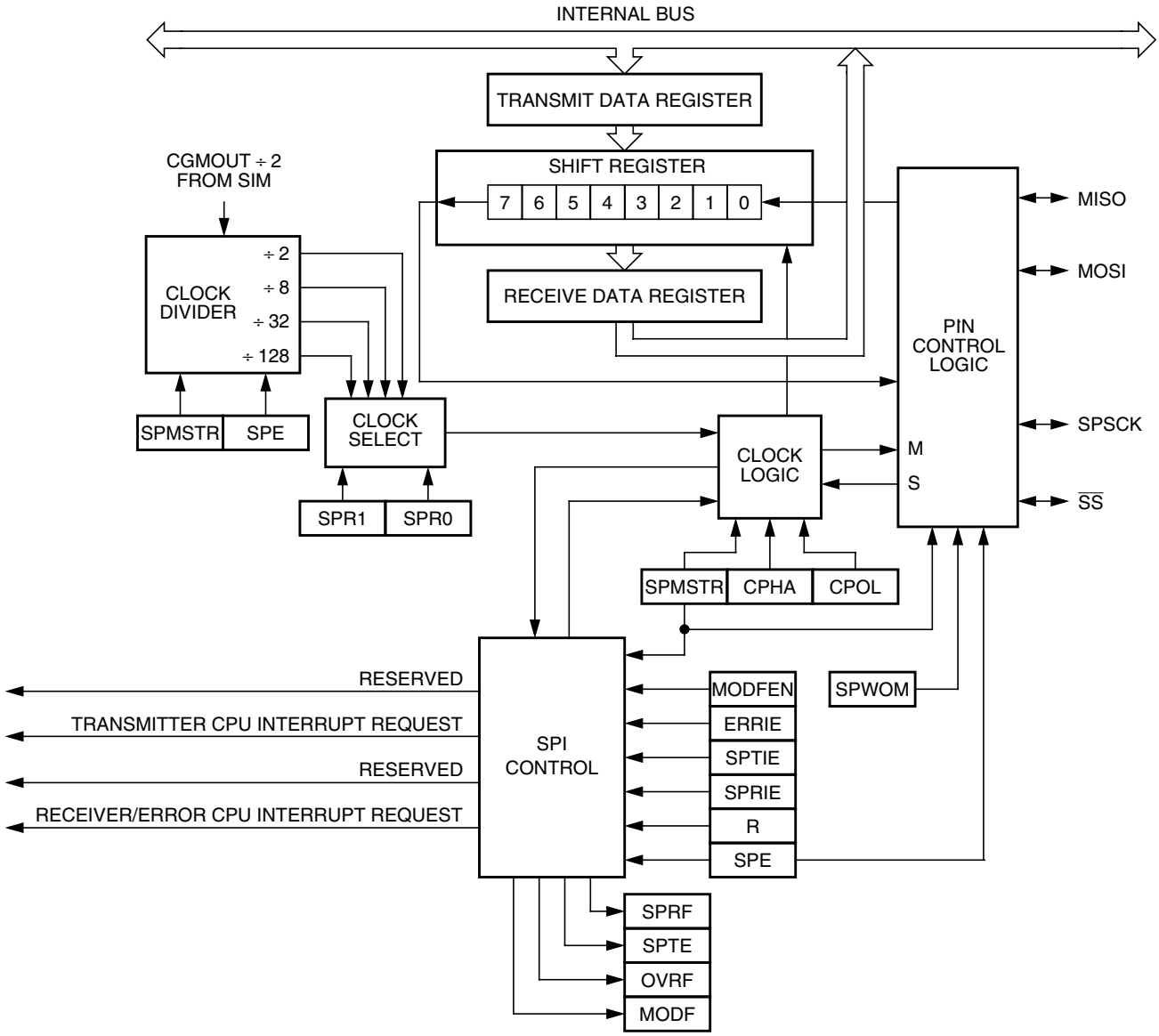



**Figure 13-2. SPI Module Block Diagram**

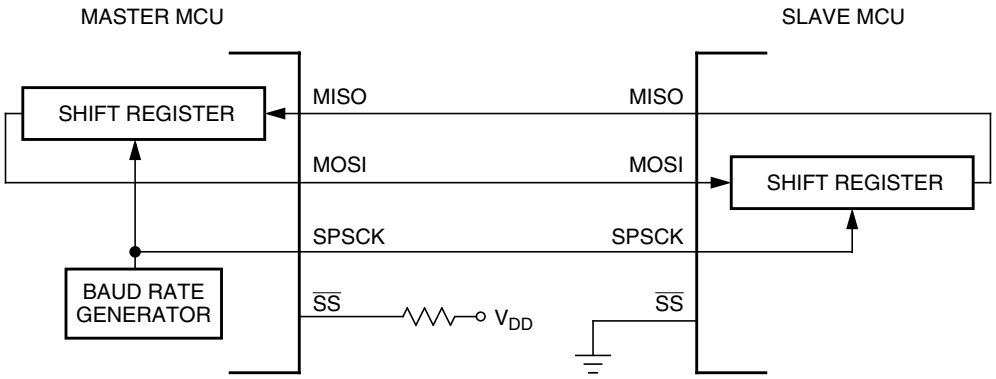



**Figure 13-3. Full-Duplex Master-Slave Connections**

## 14.3 I/O Pins

The MMIIC module uses two I/O pins, shared with standard port I/O pins. The full name of the MMIIC I/O pins are listed in Table 14-1. The generic pin name appear in the text that follows.

The SDA and SDL pins are open-drain. When configured as general purpose output pins (PTB0 and PTB1), pullup resistors must be connected to these pins.

**Table 14-1. Pin Name Conventions**

| MMIIC Generic Pin Names: | Full MCU Pin Names: | Pin Selected for MMIIC Function By: |
|---|---|---|
| SDA | PTB0/SDA | MMEN bit in MMCR1 ($0049) |
| SCL | PTB1/SCL | |

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0048 | MMIIC Address Register (MMADR) | Read: | MMAD7 | MMAD6 | MMAD5 | MMAD4 | MMAD3 | MMAD2 | MMAD1 | MMEXTAD |
| | | Write: | | | | | | | | |
| | | Reset: | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| $0049 | MMIIC Control Register 1 (MMCR1) | Read: | MMEN | MMIEN | 0 | 0 | MMTXAK | REPSEN | MMCRCBYTE | 0 |
| | | Write: | | | MMCLRBB | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $004A | MMIIC Control Register 2 (MMCR2) | Read: | MMALIF | MMNAKIF | MMBB | MMAST | MMRW | 0 | 0 | MMCRCEF |
| | | Write: | 0 | 0 | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Unaffected |
| $004B | MMIIC Status Register (MMSR) | Read: | MMRXIF | MMTXIF | MMATCH | MMSRW | MMRXAK | MMCRCBF | MMTXBE | MMRXBF |
| | | Write: | 0 | 0 | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| $004C | MMIIC Data Transmit Register (MMDTR) | Read: | MMTD7 | MMTD6 | MMTD5 | MMTD4 | MMTD3 | MMTD2 | MMTD1 | MMTD0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $004D | MMIIC Data Receive Register (MDDRR) | Read: | MMRD7 | MMRD6 | MMRD5 | MMRD4 | MMRD3 | MMRD2 | MMRD1 | MMRD0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $004E | MMIIC CRC Data Register (MMCRDR) | Read: | MMCRCD7 | MMCRCD6 | MMCRCD5 | MMCRCD4 | MMCRCD3 | MMCRCD2 | MMCRCD1 | MMCRCD0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $004F | MMIIC Frequency Divider Register (MMFDR) | Read: | 0 | 0 | 0 | 0 | 0 | MMBR2 | MMBR1 | MMBR0 |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

= Unimplemented

**Figure 14-1. MMIIC I/O Register Summary**

## 14.4 Multi-Master IIC System Configuration

The multi-master IIC system uses a serial data line SDA and a serial clock line SCL for data transfer. All devices connected to it must have open collector (drain) outputs and the logical-AND function is performed on both lines by two pull-up resistors.

## 22.11 3V Oscillator Characteristics

**Table 22-10. Oscillator Specifications (3V)**

| Characteristic[1] | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| Internal oscillator clock frequency | $f_{ICLK}$ | 64k | 88k[2] | 104k | Hz |
| External reference clock to OSC1[3] | $f_{OSC}$ | dc | | 32M | Hz |
| Crystal reference frequency | $f_{XTALCLK}$ | 30 | 32.768 | 100 | kHz |
| Crystal load capacitance[4] | $C_L$ | — | 12.5 | — | pF |
| Crystal fixed capacitance[5] | $C_1$ | — | 15 | — | pF |
| Crystal tuning capacitance[6] | $C_2$ | — | 15 | — | pF |
| Feedback bias resistor | $R_B$ | 1 | 10 | 22 | MΩ |
| Series resistor | $R_S$ | 100 | 330 | 470 | kΩ |
| External RC clock frequency | $f_{RCCLK}$ | | | 7.6M | Hz |
| RC oscillator external R | $R_{EXT}$ | See Figure 22-1 | | | Ω |
| RC oscillator external C | $C_{EXT}$ | — | 10 | — | pF |

1. The oscillator circuit operates at $V_{REG}$.
2. Typical value reflect average measurements at midpoint of voltage range, 25 °C only.
3. No more than 10% duty cycle deviation from 50%. The max. frequency is limited by an EMC filter.
4. Crystal manufacturer value.
5. Capacitor on OSC1 pin. Does not include parasitic capacitance due to package, pin, and board.
6. Capacitor on OSC2 pin. Does not include parasitic capacitance due to package, pin, and board.

## 22.13 MMIIC Electrical Characteristics

**Table 22-12. MMIIC DC Electrical Characteristics**

| Characteristic[1] | Symbol | Min | Typ | Max | Unit | Comments |
|---|---|---|---|---|---|---|
| Input low | $V_{IL}$ | –0.5 | — | 0.8 | V | Data, clock input low. |
| Input high | $V_{IH}$ | 2.1 | — | 5.5 | V | Data, clock input high. |
| Output low | $V_{OL}$ | — | — | 0.4 | V | Data, clock output low; @$I_{PULLUP,MAX}$ |
| Input leakage | $I_{LEAK}$ | — | — | ± 5 | µA | Input leakage current |
| Pullup current | $I_{PULLUP}$ | 100 | — | 350 | µA | Current through pull-up resistor or current source. See note.[2] |

1. $V_{DD}$ = 2.7 to 5.5Vdc, $V_{SS}$ = 0 Vdc, $T_A$ = $T_L$ to $T_H$, unless otherwise noted.
2. The $I_{PULLUP}$ (max) specification is determined primarily by the need to accommodate a maximum of 1.1kΩ equivalent series resistor of removable SMBus devices, such as the smart battery, while maintaining the $V_{OL}$ (max) of the bus.

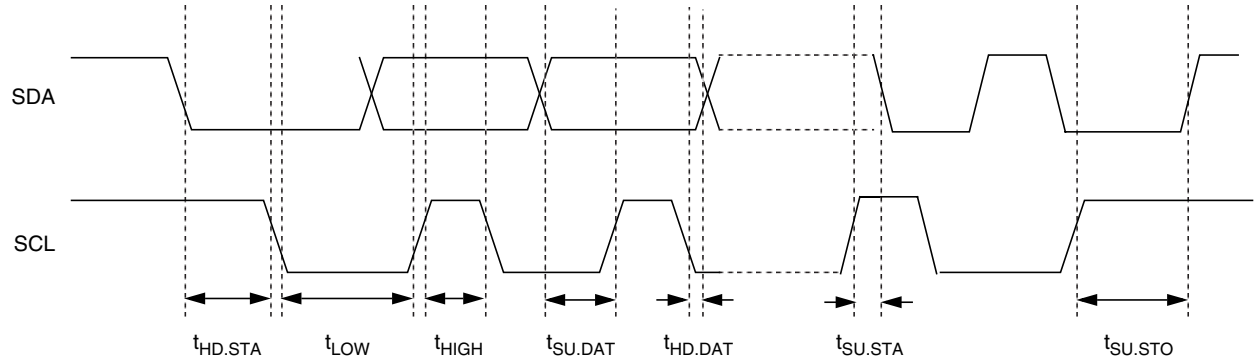



**Figure 22-2. MMIIC Signal Timings**

See Table 22-13 for MMIIC timing parameters.