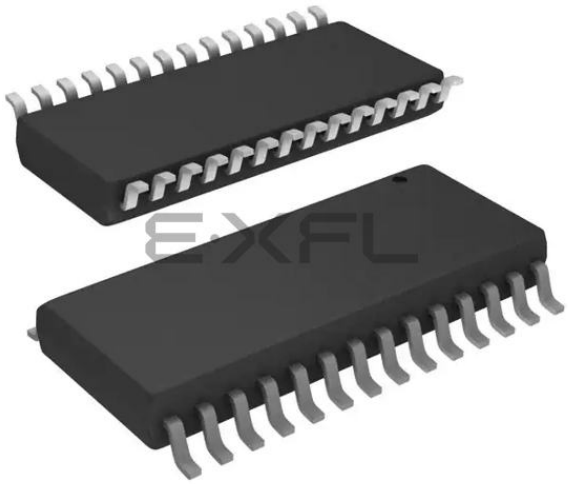


**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

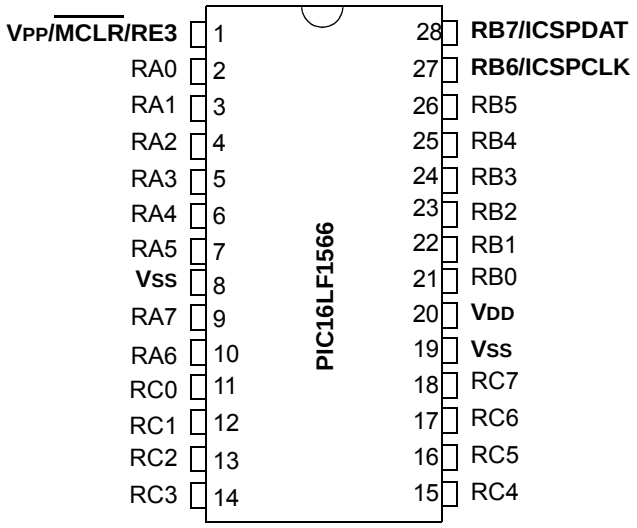
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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 | PIC |
| Core Size | 8-Bit |
| Speed | 32MHz |
| Connectivity | I²C, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, WDT |
| Number of I/O | 25 |
| Program Memory Size | 14KB (8K x 14) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 1K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 23x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 28-SOIC (0.295", 7.50mm Width) |
| Supplier Device Package | 28-SOIC |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic16lf1566t-i-so |

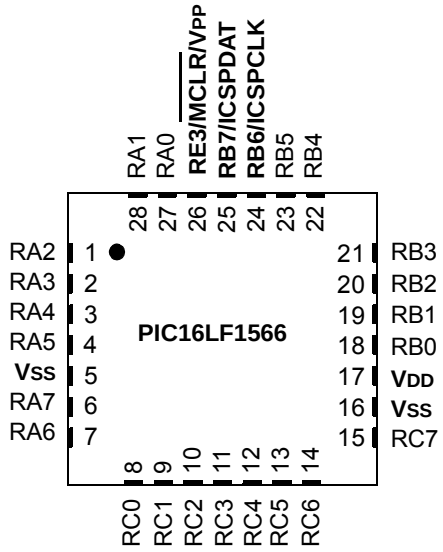
## PIN DIAGRAMS

**FIGURE 1: 28-PIN SPDIP, SOIC, SSOP DIAGRAM FOR PIC16LF1566**




**Note:** See Table 2 for the pin allocation tables.

**FIGURE 2: 28-PIN UQFN DIAGRAM FOR PIC16LF1566**




**Note:** See Table 2 or the pin allocation tables.

**TABLE 1-2: PIC16LF1566 PINOUT DESCRIPTION (CONTINUED)**

| Name | Function | Input Type | Output Type | Description |
|---|---|---|---|---|
| RB3/AN28/PWM23 | RB3 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN28 | AN | — | ADC Channel Input for ADC2. |
| | PWM23 | — | CMOS | PWM Output for PWM2. |
| RB4/AN18/AD1GRDA(1)/AD2GRDA(1) | RB4 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN18 | AN | — | ADC Channel Input for ADC1. |
| | AD1GRDA | — | CMOS | ADC1 Guard Ring Output A. |
| | AD2GRDA | — | CMOS | ADC2 Guard Ring Output A. |
| RB5/AN29/AD1GRDA(1)/AD2GRDA(1)/T1G | RB5 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN29 | AN | — | ADC Channel Input for ADC2. |
| | AD1GRDA | — | CMOS | ADC1 Guard Ring Output A. |
| | AD2GRDA | — | CMOS | ADC2 Guard Ring Output A. |
| | T1G | ST | — | Timer1 Gate Input. |
| RB6/AN19/AD1GRDB(1)/AD2GRDB(1)/ ICSPCLK/ICDCLK | RB6 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN19 | AN | — | ADC Channel Input for ADC1. |
| | AD1GRDB | — | CMOS | ADC1 Guard Ring Output B. |
| | AD2GRDB | — | CMOS | ADC2 Guard Ring Output B. |
| | ICSPCLK | ST | CMOS | ICSP™ Programming Clock. |
| | ICDCLK | ST | CMOS | In-Circuit Debug Clock. |
| RB7/AN40/AD1GRDB(1)/AD2GRDB(1)/ ICSPDAT/ICDDAT | RB7 | TTL | CMOS | General Purpose I/O with IOC and WPU. |
| | AN40 | AN | — | ADC Channel Input for ADC2. |
| | AD1GRDB | — | CMOS | ADC1 Guard Ring Output B. |
| | AD2GRDB | — | CMOS | ADC2 Guard Ring Output B. |
| | ICSPDAT | ST | CMOS | ICSP™ Data I/O. |
| | ICDDAT | ST | CMOS | In-Circuit Debug Data. |
| RC0/AN12/T1CKI/SDO2 | RC0 | TTL | CMOS | General Purpose I/O. |
| | AN12 | AN | — | ADC Channel Input for ADC1. |
| | T1CKI | ST | — | Timer1 Clock Input. |
| | SDO2 | — | CMOS | SPI Data Output for MSSP2. |
| RC1/AN23/PWM2/SCL2/SCK2 | RC1 | TTL | CMOS | General Purpose I/O. |
| | AN23 | AN | — | ADC Channel Input for ADC2. |
| | PWM2 | — | CMOS | PWM Output for PWM2. |
| | SCL2 | $I^2C$ | OD | $I^2C$ Clock for MSSP2. |
| | SCK2 | ST | CMOS | SPI Clock for MSSP2. |
| RC2/AN13/PWM1/SDA2/SDI2 | RC2 | TTL | CMOS | General Purpose I/O. |
| | AN13 | AN | — | ADC Channel Input for ADC1. |
| | PWM1 | — | CMOS | PWM Output for PWM1. |
| | SDA2 | $I^2C$ | OD | $I^2C$ Data for MSSP2. |
| | SDI2 | CMOS | — | SPI Data Input for MSSP2. |
| RC3/AN24/SCL1/SCK1 | RC3 | TTL | CMOS | General Purpose I/O. |
| | AN24 | AN | — | ADC Channel Input for ADC2. |
| | SCL1 | $I^2C$ | OD | $I^2C$ Clock for MSSP1. |
| | SCK1 | ST | CMOS | SPI Clock for MSSP1. |

**TABLE 3-7: PIC16LF1566/1567 MEMORY MAP, BANKS 24-31**

| | BANK 24 | | BANK 25 | | BANK 26 | | BANK 27 | | BANK 28 | | BANK 29 | | BANK 30 | | BANK 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C00h – C0Bh | CPU Core Register, see Table 3-2 for specifics | | | | | | | | | | | | | | |
| C0Ch | — | C8Ch | — | D0Ch | — | D8Ch | — | E0Ch | — | E8Ch | — | F0Ch | — | F8Ch | See Table 3-8 and Table 3-9 for register mapping details |
| C0Dh | — | C8Dh | — | D0Dh | — | D8Dh | — | E0Dh | — | E8Dh | — | F0Dh | — | F8Dh | |
| C0Eh | — | C8Eh | — | D0Eh | — | D8Eh | — | E0Eh | — | E8Eh | — | F0Eh | — | F8Eh | |
| C0Fh | — | C8Fh | — | D0Fh | — | D8Fh | — | E0Fh | — | E8Fh | — | F0Fh | — | F8Fh | |
| C10h | — | C90h | — | D10h | — | D90h | — | E10h | — | E90h | — | F10h | — | F90h | |
| C11h | — | C91h | — | D11h | — | D91h | — | E11h | — | E91h | — | F11h | — | F91h | |
| C12h | — | C92h | — | D12h | — | D92h | — | E12h | — | E92h | — | F12h | — | F92h | |
| C13h | — | C93h | — | D13h | — | D93h | — | E13h | — | E93h | — | F13h | — | F93h | |
| C14h | — | C94h | — | D14h | — | D94h | — | E14h | — | E94h | — | F14h | — | F94h | |
| C15h | — | C95h | — | D15h | — | D95h | — | E15h | — | E95h | — | F15h | — | F95h | |
| C16h | — | C96h | — | D16h | — | D96h | — | E16h | — | E96h | — | F16h | — | F96h | |
| C17h | — | C97h | — | D17h | — | D97h | — | E17h | — | E97h | — | F17h | — | F97h | |
| C18h | — | C98h | — | D18h | — | D98h | — | E18h | — | E98h | — | F18h | — | F98h | |
| C19h | — | C99h | — | D19h | — | D99h | — | E19h | — | E99h | — | F19h | — | F99h | |
| C1Ah | — | C9Ah | — | D1Ah | — | D9Ah | — | E1Ah | — | E9Ah | — | F1Ah | — | F9Ah | |
| C1Bh | — | C9Bh | — | D1Bh | — | D9Bh | — | E1Bh | — | E9Bh | — | F1Bh | — | F9Bh | |
| C1Ch | — | C9Ch | — | D1Ch | — | D9Ch | — | E1Ch | — | E9Ch | — | F1Ch | — | F9Ch | |
| C1Dh | — | C9Dh | — | D1Dh | — | D9Dh | — | E1Dh | — | E9Dh | — | F1Dh | — | F9Dh | |
| C1Eh | — | C9Eh | — | D1Eh | — | D9Eh | — | E1Eh | — | E9Eh | — | F1Eh | — | F9Eh | |
| C1Fh | — | C9Fh | — | D1Fh | — | D9Fh | — | E1Fh | — | E9Fh | — | F1Fh | — | F9Fh | |
| C20h – C6Fh | Unimplemented Read as '0' | CA0h – CEFh | Unimplemented Read as '0' | D20h – D6Fh | Unimplemented Read as '0' | DA0h – DEFh | Unimplemented Read as '0' | E20h – E6Fh | Unimplemented Read as '0' | EA0h – EEFh | Unimplemented Read as '0' | F20h – F6Fh | Unimplemented Read as '0' | FA0h – FEFh | |
| C70h – CFFh | Accesses 70h – 7Fh | CF0h – CFFh | Accesses 70h – 7Fh | D70h – D7Fh | Accesses 70h – 7Fh | DF0h – DFFh | Accesses 70h – 7Fh | E70h – E7Fh | Accesses 70h – 7Fh | EF0h – EFFh | Accesses 70h – 7Fh | F70h – F7Fh | Accesses 70h – 7Fh | F0h – FFFh | Accesses 70h – 7Fh |

**Legend:** = Unimplemented data memory locations, read as '0'.

**TABLE 3-11: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)**

| Addr. | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on: POR, BOR | Value on all other Resets |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Bank 2** | | | | | | | | | | | |
| 100h | INDF0(1) | Addressing this location uses contents of FSR0H/FSR0L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 101h | INDF1(1) | Addressing this location uses contents of FSR1H/FSR1L to address data memory (not a physical register) | | | | | | | | xxxx xxxx | uuuu uuuu |
| 102h | PCL(1) | Program Counter (PC) Least Significant Byte | | | | | | | | 0000 0000 | 0000 0000 |
| 103h | STATUS(1) | — | — | — | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C | ---1 1000 | ---q quuu |
| 104h | FSR0L(1) | Indirect Data Memory Address 0 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 105h | FSR0H(1) | Indirect Data Memory Address 0 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 106h | FSR1L(1) | Indirect Data Memory Address 1 Low Pointer | | | | | | | | 0000 0000 | uuuu uuuu |
| 107h | FSR1H(1) | Indirect Data Memory Address 1 High Pointer | | | | | | | | 0000 0000 | 0000 0000 |
| 108h | BSR(1) | — | — | — | BSR<4:0> | | | | | ---0 0000 | ---0 0000 |
| 109h | WREG(1) | Working Register | | | | | | | | 0000 0000 | uuuu uuuu |
| 10Ah | PCLATH(1) | — | Write Buffer for the upper 7 bits of the Program Counter | | | | | | | -000 0000 | -000 0000 |
| 10Bh | INTCON(1) | GIE | PEIE | TMR0IE | INTE | IOCIE | TMR0IF | INTF | IOCIF | 0000 0000 | 0000 0000 |
| 10Ch | LATA | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 | xxxx xxxx | uuuu uuuu |
| 10Dh | LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | xxxx xxxx | uuuu uuuu |
| 10Eh | LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 | xxxx xxxx | uuuu uuuu |
| 10Fh | Unimplemented | | | | | | | | | | |
| | LATD(2) | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | xxxx xxxx | uuuu uuuu |
| 110h | Unimplemented | | | | | | | | | | |
| | LATE(2) | — | — | — | — | — | LATE2 | LATE1 | LATE0 | -----xxx | ---- -uuu |
| 111h | — | Unimplemented | | | | | | | | — | — |
| 112h | — | Unimplemented | | | | | | | | — | — |
| 113h | — | Unimplemented | | | | | | | | — | — |
| 114h | — | Unimplemented | | | | | | | | — | — |
| 115h | — | Unimplemented | | | | | | | | — | — |
| 116h | BORCON | SBOREN | BORFS | — | — | — | — | — | BORRDY | 10-- ---q | uu-- ---u |
| 117h | FVRCON | FVREN | FVRRDY | TSEN | TSRNG | — | — | ADFVR<1:0> | | 0q00 --00 | 0q00 --00 |
| 118h | — | Unimplemented | | | | | | | | — | — |
| 119h | — | Unimplemented | | | | | | | | — | — |
| 11Ah | — | Unimplemented | | | | | | | | — | — |
| 11Bh | — | Unimplemented | | | | | | | | — | — |
| 11Ch | — | Unimplemented | | | | | | | | — | — |
| 11Dh | APFCON | — | — | SSSEL | — | — | — | GRDBSEL | GRDASEL | --0---00 | --0---00 |
| 11Eh | — | Unimplemented | | | | | | | | — | — |
| 11Fh | — | Unimplemented | | | | | | | | — | — |

**Legend:** x = unknown, u = unchanged, q = depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations unimplemented, read as '0'.

**Note 1:** These registers can be accessed from any bank.
**2:** PIC16LF1567.
**3:** These registers/bits are available at two address locations, in Bank 1 and Bank 14.
**4:** PIC16LF1566 only.
**5:** Unimplemented, read as '1'.

## 5.2 Clock Source Types

Clock sources can be classified as external or internal.

External clock sources rely on external circuitry for the clock source to function. Examples are: oscillator modules (EC mode).

Internal clock sources are contained within the oscillator module. The oscillator block has two internal oscillators that are used to generate two system clock sources: the 16 MHz High-Frequency Internal Oscillator (HFINTOSC) and the 31 kHz Low-Frequency Internal Oscillator (LFINTOSC).

The system clock can be selected between external or internal clock sources via the System Clock Select (SCS) bits in the OSCCON register. See **Section 5.3 "Clock Switching"** for additional information.

### 5.2.1 EXTERNAL CLOCK SOURCES

An external clock source can be used as the device system clock by performing one of the following actions:

- Program the FOSC<1:0> bits in the Configuration Words to select an external clock source that will be used as the default system clock upon a device Reset.
- Clear the SCS<1:0> bits in the OSCCON register to switch the system clock source to:
  - An external clock source determined by the value of the FOSC bits.

See **Section 5.3 "Clock Switching"** for more information.

#### 5.2.1.1 EC Mode

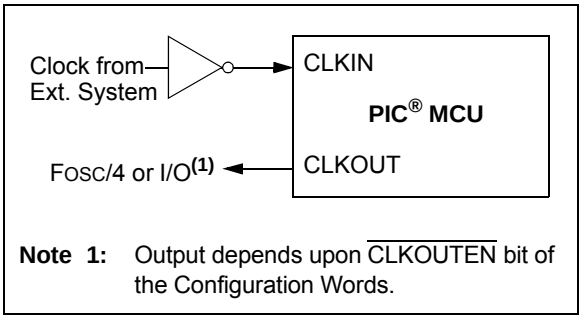
The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in this mode, an external clock source is connected to the CLKIN input. CLKOUT is available for general purpose I/O or CLKOUT. Figure 5-2 shows the pin connections for EC mode.

EC mode has three power modes to select from through Configuration Words:

- High power, 4-20 MHz (FOSC = 11)
- Medium power, 0.5-4 MHz (FOSC = 10)
- Low power, 0-0.5 MHz (FOSC = 01)

When EC mode is selected, there is no delay in operation after a Power-on Reset (POR) or wake-up from Sleep. Because the PIC® MCU design is fully static, stopping the external clock input will have the effect of halting the device while leaving all data intact. Upon restarting the external clock, the device will resume operation as if no time had elapsed.

**FIGURE 5-2: EXTERNAL CLOCK (EC) MODE OPERATION**




**Note 1:** Output depends upon $\overline{\text{CLKOUTEN}}$ bit of the Configuration Words.

**REGISTER 15-3: ADCON1(1)/ADCOMCON(2): ADC CONTROL REGISTER 1**

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---|---|---|---|---|---|---|---|
| ADFM | ADCS<2:0> | | | ADNREF | GO/$\overline{\text{DONE}}$_ALL | ADPREF<1:0> | |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7 **ADFM:** ADC Result Format Select bit
- 1 = Right justified. Six Most Significant bits of ADxRESxH are set to '0' when the conversion result is loaded.
- 0 = Left justified. Six Least Significant bits of ADxRESxL are set to '0' when the conversion result is loaded.

bit 6-4 **ADCS<2:0>:** ADC Conversion Clock Select bits
- 111 = FRC (clock supplied from an internal RC oscillator)
- 110 = FOSC/64
- 101 = FOSC/16
- 100 = FOSC/4
- 011 = FRC (clock supplied from an internal RC oscillator)
- 010 = FOSC/32
- 001 = FOSC/8
- 000 = FOSC/2

bit 3 **ADNREF:** ADC Negative Voltage Reference Configuration bit
- 1 = VREFL is connected to external VREF- pin(4)
- 0 = VREFL is connected to AVSS.

bit 2 **GO/$\overline{\text{DONE}}$_ALL(3):** Synchronized ADC Conversion Status bit
- 1 = Synchronized ADC conversion in progress. Setting this bit starts conversion in any ADC with ADxON = 1.
- 0 = Synchronized ADC conversion completed/ not in progress.

bit 1-0 **ADPREF<1:0>:** ADC Positive Voltage Reference Configuration bits
- 11 = VREFH is connected to internal Fixed Voltage Reference.
- 10 = VREFH is connected to external VREF+ pin(4)
- 01 = Reserved
- 00 = VREFH is connected to VDD

**Note 1:** Bank 1 name is ADCON1.

**2:** Bank 14 name is ADCOMCON.

**3:** Setting this bit triggers the GO/$\overline{\text{DONE}}$x bits in both ADCs. Each ADC will run a conversion according to its control register settings. This bit reads as an OR of the individual GO/$\overline{\text{DONE}}$x bits.

**4:** When selecting the VREF+ or VREF- pin as the source of the positive or negative reference, be aware that a minimum voltage specification exists. See **Section 25.0 "Electrical Specifications"** for details.

Each secondary channel is forced to input. The ANSELx bit for secondary channel is still under user control. During the precharge stage, the output drivers on each secondary channel will be overridden by the hardware CVD module and do exactly what the output drivers on the ADC's primary channel are configured to do.

Both the primary and secondary channels are connected to the ADC as soon as the channels are selected by the CHx<4:0> bits of the AADxCON0 register and the bits in the AADxCH register.

**FIGURE 16-5: HARDWARE CVD SEQUENCE TIMING DIAGRAM**

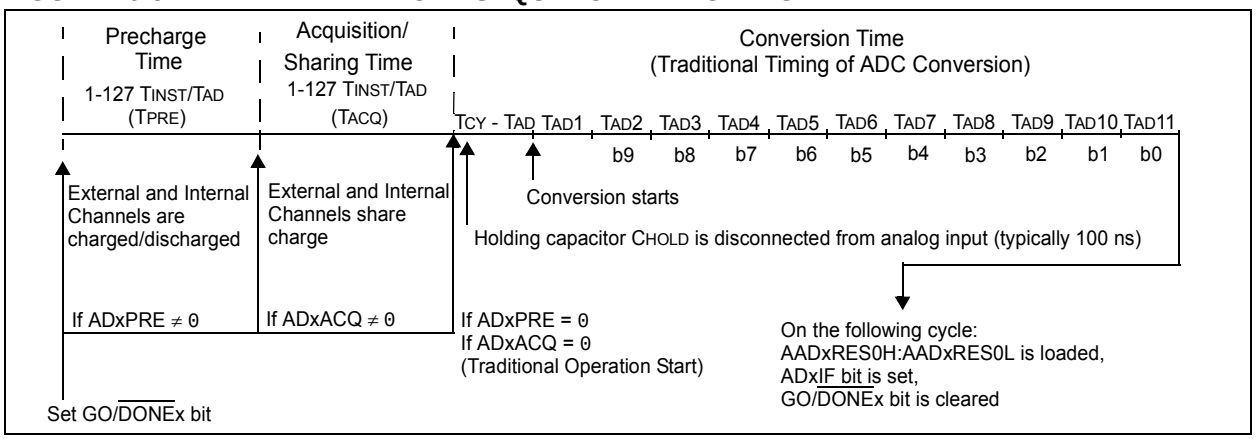
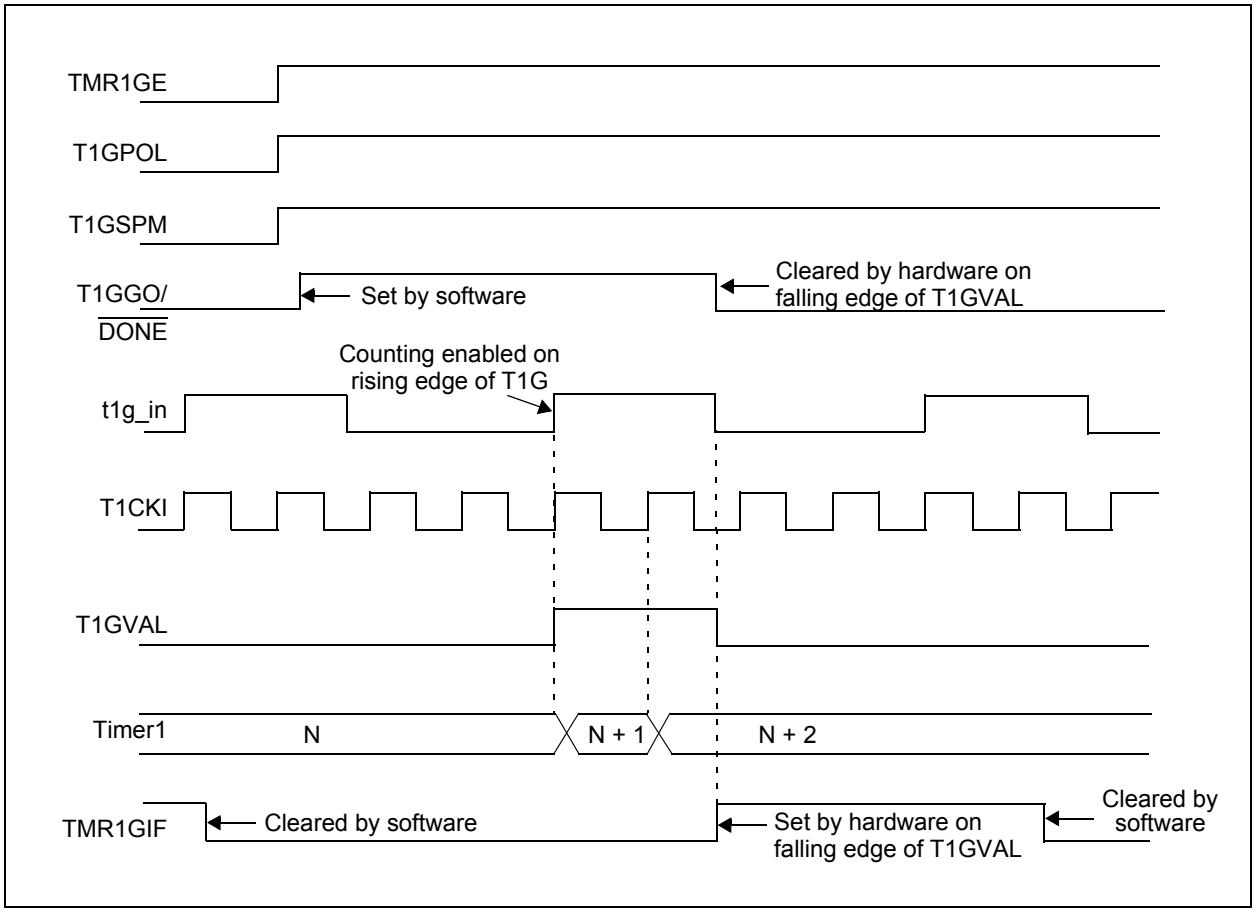


**FIGURE 18-5: TIMER1 GATE SINGLE-PULSE MODE**



**REGISTER 18-2: T1GCON: TIMER1 GATE CONTROL REGISTER**

| R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W/HC-0/u | R-x/x | U-0 | R/W-0/u |
|---|---|---|---|---|---|---|---|
| TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/$\overline{\text{DONE}}$ | T1GVAL | — | T1GSS |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | | |
|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | HC = Bit is cleared by hardware |

bit 7 **TMR1GE:** Timer1 Gate Enable bit
If TMR1ON = 0:
This bit is ignored
If TMR1ON = 1:
1 = Timer1 counting is controlled by the Timer1 gate function
0 = Timer1 counts regardless of Timer1 gate function

bit 6 **T1GPOL:** Timer1 Gate Polarity bit
1 = Timer1 gate is active-high (Timer1 counts when gate is high)
0 = Timer1 gate is active-low (Timer1 counts when gate is low)

bit 5 **T1GTM:** Timer1 Gate Toggle Mode bit
1 = Timer1 Gate Toggle mode is enabled
0 = Timer1 Gate Toggle mode is disabled and toggle flip-flop is cleared
Timer1 gate flip-flop toggles on every rising edge.

bit 4 **T1GSPM:** Timer1 Gate Single-Pulse Mode bit
1 = Timer1 gate Single-Pulse mode is enabled and is controlling Timer1 gate
0 = Timer1 gate Single-Pulse mode is disabled

bit 3 **T1GGO/$\overline{\text{DONE}}$:** Timer1 Gate Single-Pulse Acquisition Status bit
1 = Timer1 gate single-pulse acquisition is ready, waiting for an edge
0 = Timer1 gate single-pulse acquisition has completed or has not been started

bit 2 **T1GVAL:** Timer1 Gate Value Status bit
Indicates the current state of the Timer1 gate that could be provided to TMR1H:TMR1L.
Unaffected by Timer1 Gate Enable (TMR1GE).

bit 1 **Unimplemented:** Read as ‘0’

bit 0 **T1GSS:** Timer1 Gate Source Select bits
01 = Timer0 overflow output (T0_overflow)
00 = Timer1 gate pin (T1G)

### 20.2.3 SPI MASTER MODE

The master can initiate the data transfer at any time because it controls the SCKx line. The master determines when the slave (Processor 2, Figure 20-5) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPxBUF register is written to. If the SPI is only going to receive, the SDOx output could be disabled (programmed as an input). The SSPxSR register will continue to shift in the signal present on the SDIx pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPxBUF register as if a normal received byte (interrupts and Status bits appropriately set).

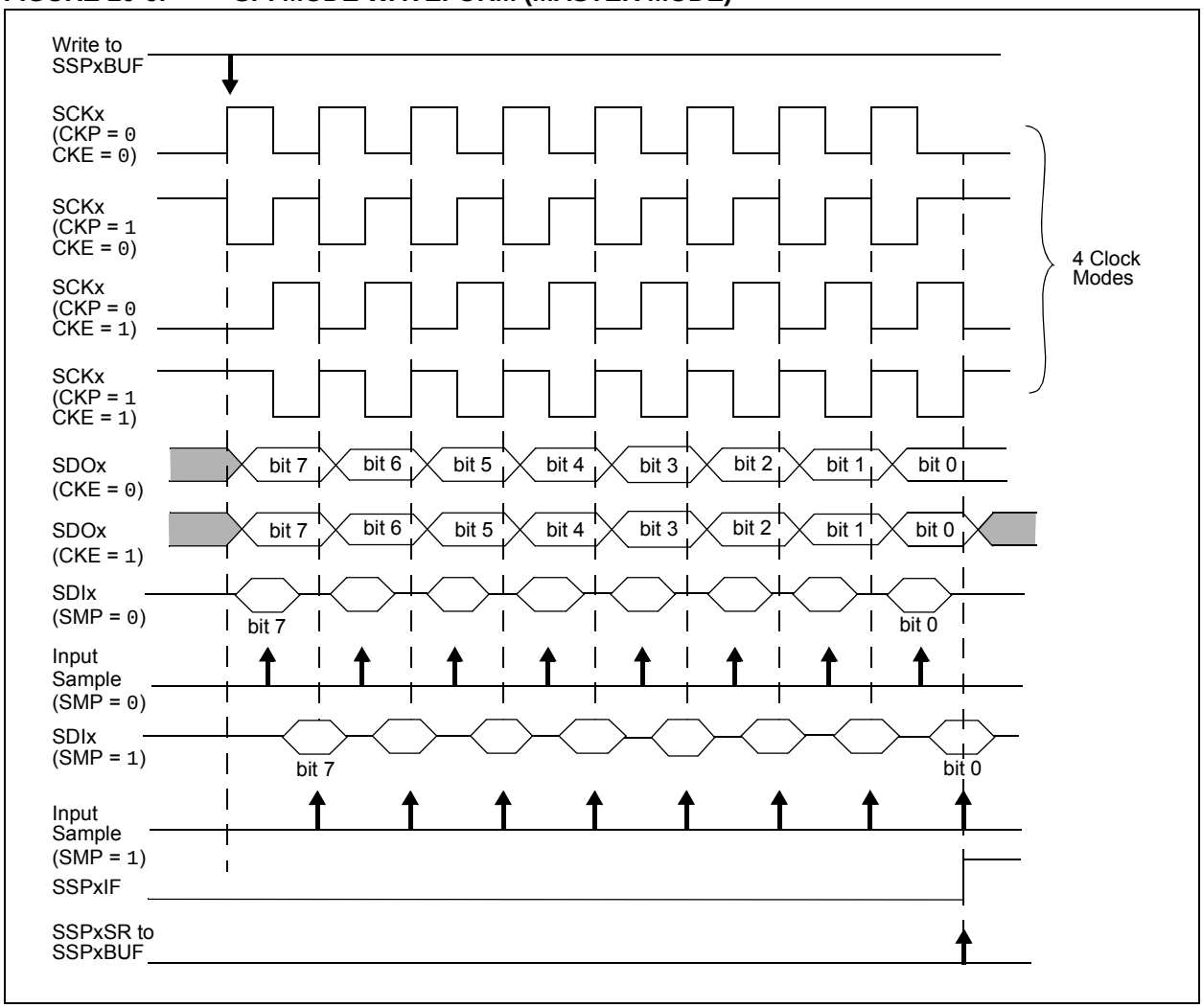
The clock polarity is selected by appropriately programming the CKP bit of the SSPxCON1 register and the CKE bit of the SSPxSTAT register. This then, would give waveforms for SPI communication as shown in Figure 20-6, Figure 20-8, Figure 20-9 and Figure 20-10, where the MSb is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 * TCY)
- FOSC/64 (or 16 * TCY)
- Timer2 output/2
- Fosc/(4 * (SSPxADD + 1))

Figure 20-6 shows the waveforms for Master mode.

When the CKE bit is set, the SDOx data is valid before there is a clock edge on SCKx. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPxBUF is loaded with the received data is shown.

**FIGURE 20-6: SPI MODE WAVEFORM (MASTER MODE)**



When one device is transmitting a logical one, or letting the line float, and a second device is transmitting a logical zero, or holding the line low, the first device can detect that the line is not a logical one. This detection, when used on the SCLx line, is called clock stretching. Clock stretching gives slave devices a mechanism to control the flow of data. When this detection is used on the SDAx line, it is called arbitration. Arbitration ensures that there is only one master device communicating at any single time.

### 20.3.1 CLOCK STRETCHING

When a slave device has not completed processing data, it can delay the transfer of more data through the process of clock stretching. An addressed slave device may hold the SCLx clock line low after receiving or sending a bit, indicating that it is not yet ready to continue. The master that is communicating with the slave will attempt to raise the SCLx line in order to transfer the next bit, but will detect that the clock line has not yet been released. Because the SCLx connection is open-drain, the slave has the ability to hold that line low until it is ready to continue communicating.

Clock stretching allows receivers that cannot keep up with a transmitter to control the flow of incoming data.

### 20.3.2 ARBITRATION

Each master device must monitor the bus for Start and Stop bits. If the device detects that the bus is busy, it cannot begin a new message until the bus returns to an Idle state.

However, two master devices may try to initiate a transmission on or about the same time. When this occurs, the process of arbitration begins. Each transmitter checks the level of the SDAx data line and compares it to the level that it expects to find. The first transmitter to observe that the two levels do not match, loses arbitration, and must stop transmitting on the SDAx line.

For example, if one transmitter holds the SDAx line to a logical one (lets it float) and a second transmitter holds it to a logical zero (pulls it low), the result is that the SDAx line will be low. The first transmitter then observes that the level of the line is different than expected and concludes that another transmitter is communicating.

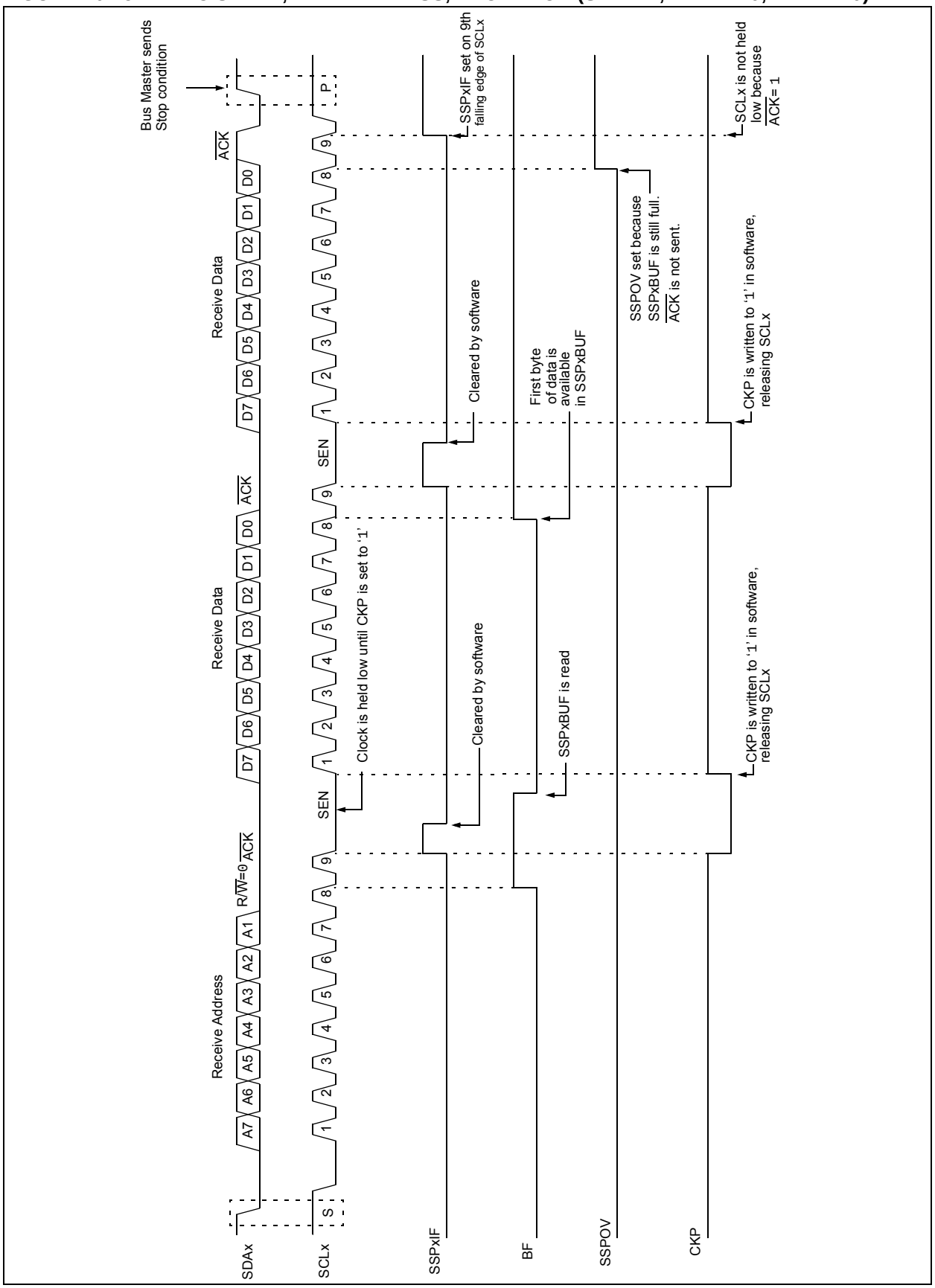
The first transmitter to notice this difference is the one that loses arbitration and must stop driving the SDAx line. If this transmitter is also a master device, it also must stop driving the SCLx line. It then can monitor the lines for a Stop condition before trying to reissue its transmission. In the meantime, the other device that has not noticed any difference between the expected and actual levels on the SDAx line continues with its original transmission. It can do so without any complications, because so far, the transmission appears exactly as expected with no other transmitter disturbing the message.

Slave Transmit mode can also be arbitrated, when a master addresses multiple slaves, but this is less common.

If two master devices are sending a message to two different slave devices at the address stage, the master sending the lower slave address always wins arbitration. When two master devices send messages to the same slave address, and addresses can sometimes refer to multiple slaves, the arbitration process must continue into the data stage.

Arbitration usually occurs very rarely, but it is a necessary process for proper multi-master support.

**FIGURE 20-15: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)**



### 20.5.6 CLOCK STRETCHING

Clock stretching occurs when a device on the bus holds the SCLx line low effectively pausing communication. The slave may stretch the clock to allow more time to handle data or prepare a response for the master device. A master device is not concerned with stretching as anytime it is active on the bus and not transferring data it is stretching. Any stretching done by a slave is invisible to the master software and handled by the hardware that generates SCLx.

The CKP bit of the SSPxCON1 register is used to control stretching in software. Any time the CKP bit is cleared, the module will wait for the SCLx line to go low and then hold it. Setting CKP will release SCLx and allow more communication.

#### 20.5.6.1 Normal Clock Stretching

Following an $\overline{ACK}$ if the R/$\overline{W}$ bit of SSPxSTAT is set, a read request, the slave hardware will clear CKP. This allows the slave time to update SSPxBUF with data to transfer to the master. If the SEN bit of SSPxCON2 is set, the slave hardware will always stretch the clock after the $\overline{ACK}$ sequence. Once the slave is ready; CKP is set by software and communication resumes.

**Note 1:** The BF bit has no effect on if the clock will be stretched or not. This is different than previous versions of the module that would not stretch the clock, clear CKP, if SSPxBUF was read before the ninth falling edge of SCLx.

**2:** Previous versions of the module did not stretch the clock for a transmission if SSPxBUF was loaded before the ninth falling edge of SCLx. It is now always cleared for read requests.

#### 20.5.6.2 10-bit Addressing Mode

In 10-bit Addressing mode, when the UA bit is set, the clock is always stretched. This is the only time the SCLx is stretched without CKP being cleared. SCLx is released immediately after a write to SSPxADD.

**Note:** Previous versions of the module did not stretch the clock if the second address byte did not match.

#### 20.5.6.3 Byte NACKing

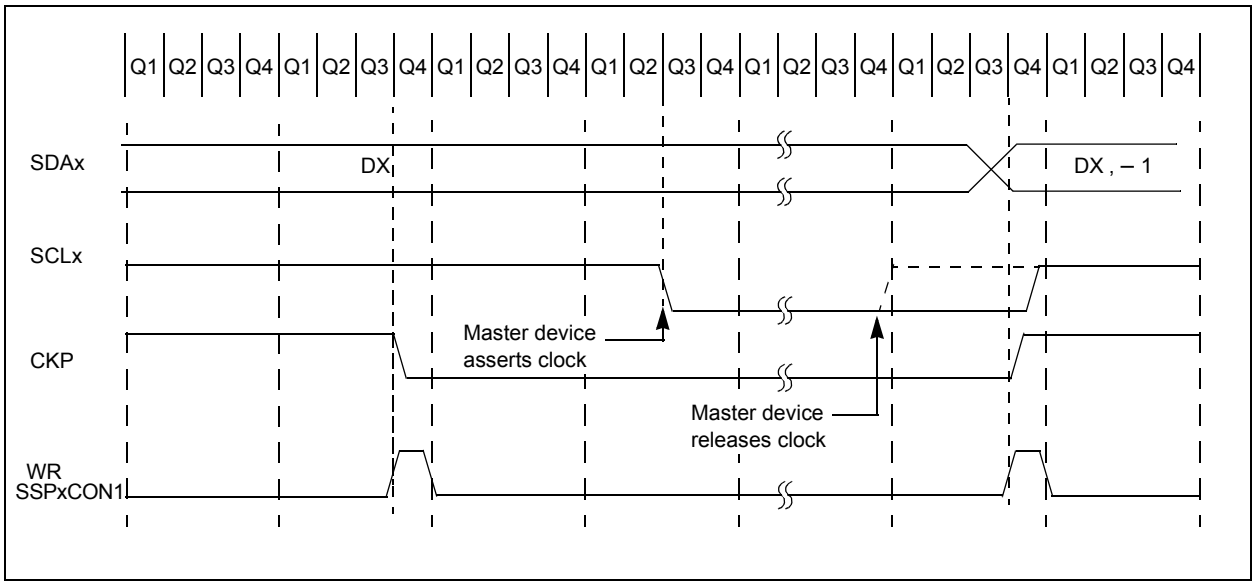
When the AHEN bit of SSPxCON3 is set; CKP is cleared by hardware after the eighth falling edge of SCLx for a received matching address byte. When the DHEN bit of SSPxCON3 is set; CKP is cleared after the eighth falling edge of SCLx for received data.

Stretching after the eighth falling edge of SCLx allows the slave to look at the received address or data and decide if it wants to ACK the received data.

### 20.5.7 CLOCK SYNCHRONIZATION AND THE CKP BIT

Any time the CKP bit is cleared, the module will wait for the SCLx line to go low and then hold it. However, clearing the CKP bit will not assert the SCLx output low until the SCLx output is already sampled low. Therefore, the CKP bit will not assert the SCLx line until an external $I^2C$ master device has already asserted the SCLx line. The SCLx output will remain low until the CKP bit is set and all other devices on the $I^2C$ bus have released SCLx. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCLx (see Figure 20-23).

**FIGURE 20-23: CLOCK SYNCHRONIZATION TIMING**



#### 21.1.2.8 Asynchronous Reception Set-up:

1. Initialize the SPBRGH:SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 21.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSELx bit for the RX pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
5. If 9-bit reception is desired, set the RX9 bit.
6. Enable reception by setting the CREN bit.
7. The RCIF interrupt flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
8. Read the RCSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
9. Get the received eight Least Significant data bits from the receive buffer by reading the RCREG register.
10. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

#### 21.1.2.9 9-Bit Address Detection Mode Set-up

This mode would typically be used in RS-485 systems. To set up an asynchronous reception with address detect enable:

1. Initialize the SPBRGH:SPBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see **Section 21.4 "EUSART Baud Rate Generator (BRG)"**).
2. Clear the ANSELx bit for the RX pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCIE bit of the PIE1 register and the GIE and PEIE bits of the INTCON register.
5. Enable 9-bit reception by setting the RX9 bit.
6. Enable address detection by setting the ADDEN bit.
7. Enable reception by setting the CREN bit.
8. The RCIF interrupt flag bit will be set when a character with the ninth bit set is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCIE interrupt enable bit was also set.
9. Read the RCSTA register to get the error flags. The ninth data bit will always be set.
10. Get the received eight Least Significant data bits from the receive buffer by reading the RCREG register. Software determines if this is the device's address.
11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
12. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

**TABLE 21-3: BAUD RATE FORMULAS**

| Configuration Bits | | | BRG/EUSART Mode | Baud Rate Formula |
|---|---|---|---|---|
| SYNC | BRG16 | BRGH | | |
| 0 | 0 | 0 | 8-bit/Asynchronous | FOSC/[64 (n+1)] |
| 0 | 0 | 1 | 8-bit/Asynchronous | FOSC/[16 (n+1)] |
| 0 | 1 | 0 | 16-bit/Asynchronous | |
| 0 | 1 | 1 | 16-bit/Asynchronous | FOSC/[4 (n+1)] |
| 1 | 0 | x | 8-bit/Synchronous | |
| 1 | 1 | x | 16-bit/Synchronous | |

**Legend:** x = Don't care, n = value of SPBRGH:SPBRGL register pair.

**TABLE 21-4: SUMMARY OF REGISTERS ASSOCIATED WITH THE BAUD RATE GENERATOR**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 266 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 265 |
| SPBRGL | BRG<7:0> | | | | | | | | 267* |
| SPBRGH | BRG<15:8> | | | | | | | | 267* |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 264 |

**Legend:** — = unimplemented location, read as '0'. Shaded cells are not used for the Baud Rate Generator.

* Page provides register information.

### 21.4.4 BREAK CHARACTER SEQUENCE

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN bus standard. A Break character consists of a Start bit, followed by 12 '0' bits and a Stop bit.

To send a Break character, set the SENDB and TXEN bits of the TXSTA register. The Break character transmission is then initiated by a write to the TXREG. The value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

The TRMT bit of the TXSTA register indicates when the transmit operation is active or idle, just as it does during normal transmission. See Figure 21-9 for the timing of the Break character sequence.

#### 21.4.4.1 Break and Sync Transmit Sequence

The following sequence will start a message frame header made up of a Break, followed by an auto-baud Sync byte. This sequence is typical of a LIN bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to enable the Break sequence.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware and the Sync character is then transmitted.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

### 21.4.5 RECEIVING A BREAK CHARACTER

The Enhanced EUSART module can receive a Break character in two ways.

The first method to detect a Break character uses the FERR bit of the RCSTA register and the received data as indicated by RCREG. The Baud Rate Generator is assumed to have been initialized to the expected baud rate.

A Break character has been received when;

- RCIF bit is set
- FERR bit is set
- RCREG = 00h

The second method uses the Auto-Wake-up feature described in **Section 21.4.3 "Auto-Wake-up on Break"**. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCIF interrupt, and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Detect feature. For both methods, the user can set the ABDEN bit of the BAUDCON register before placing the EUSART in Sleep mode.

**FIGURE 21-9: SEND BREAK CHARACTER SEQUENCE**

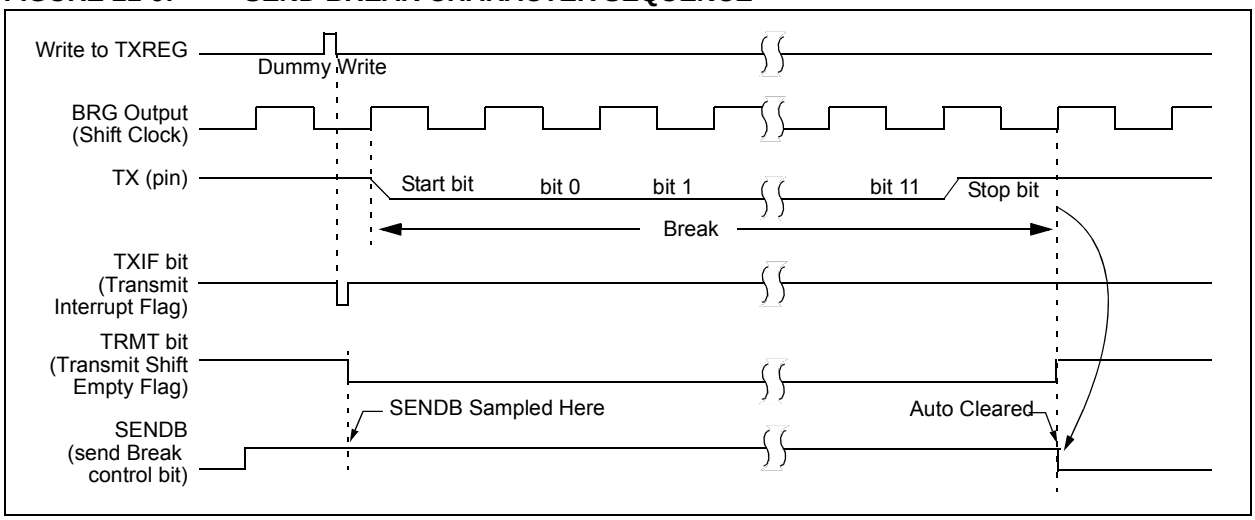


**TABLE 22-4: SUMMARY OF REGISTERS ASSOCIATED WITH PWM**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---|---|---|---|---|---|---|---|---|---|
| PR2 | Timer2 module Period Register | | | | | | | | 193* |
| PWM1CON | PWM1EN | PWM1OE | PWM1OUT | PWM1POL | — | — | — | — | 286 |
| PWM1DCH | PWM1DCH<7:0> | | | | | | | | 287 |
| PWM1DCL | PWM1DCL<7:6> | | — | — | — | — | — | — | 287 |
| PWM2CON | PWM2EN | PWM2OE | PWM2OUT | PWM2POL | — | — | — | — | 286 |
| PWM2DCH | PWM2DCH<7:0> | | | | | | | | 287 |
| PWM2DCL | PWM2DCL<7:6> | | — | — | — | — | — | — | 287 |
| T2CON | — | T2OUTPS<3:0> | | | | TMR2ON | T2CKPS<1:0> | | 195 |
| TMR2 | Timer2 module Register | | | | | | | | 193* |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 115 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 123 |

**Legend:** - = Unimplemented locations, read as '0', u = unchanged, x = unknown. Shaded cells are not used by the PWM.

* Page provides register information.

**TABLE 25-7: CLOCK OSCILLATOR TIMING REQUIREMENTS**

| **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +125°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| OS01 | FOSC | External CLKIN Frequency(1) | DC | — | 0.5 | MHz | EC Oscillator mode (low) |
| | | | DC | — | 4 | MHz | EC Oscillator mode (medium) |
| | | | DC | — | 20 | MHz | EC Oscillator mode (high) |
| OS02 | TOSC | External CLKIN Period(1) | 50 | — | ∞ | ns | EC mode |
| OS03 | TCY | Instruction Cycle Time(1) | 200 | — | DC | ns | TCY = FOSC/4 |

* These parameters are characterized but not tested.

† Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min" values with an external clock applied to CLKIN pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

**TABLE 25-8: OSCILLATOR PARAMETERS**

| **Standard Operating Conditions (unless otherwise stated)**<br>Operating Temperature -40°C ≤ TA ≤ +125°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| OS08 | HFOSC | Internal Calibrated HFINTOSC Frequency(1) | — | 16.0 | — | MHz | 0°C ≤ TA ≤ +85°C |
| OS08A | HFTOL | Frequency Tolerance | — | ±3 | — | % | 25°C, 16 MHz |
| | | | — | ±6 | — | % | 0°C ≤ TA ≤ +85°C, 16 MHz |
| OS09 | LFOSC | Internal LFINTOSC Frequency | — | 31 | — | kHz | -40°C ≤ TA ≤ +125°C |
| OS10* | TWARM | HFINTOSC<br>Wake-up from Sleep Start-up Time | — | 5 | 15 | µs | |
| | | LFINTOSC<br>Wake-up from Sleep Start-up Time | — | 0.5 | — | ms | |

* These parameters are characterized but not tested.

† Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** To ensure these oscillator frequency tolerances, VDD and VSS must be capacitively decoupled as close to the device as possible. 0.1 µF and 0.01 µF values in parallel are recommended.

## 27.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

## 27.12 Third-Party Development Tools

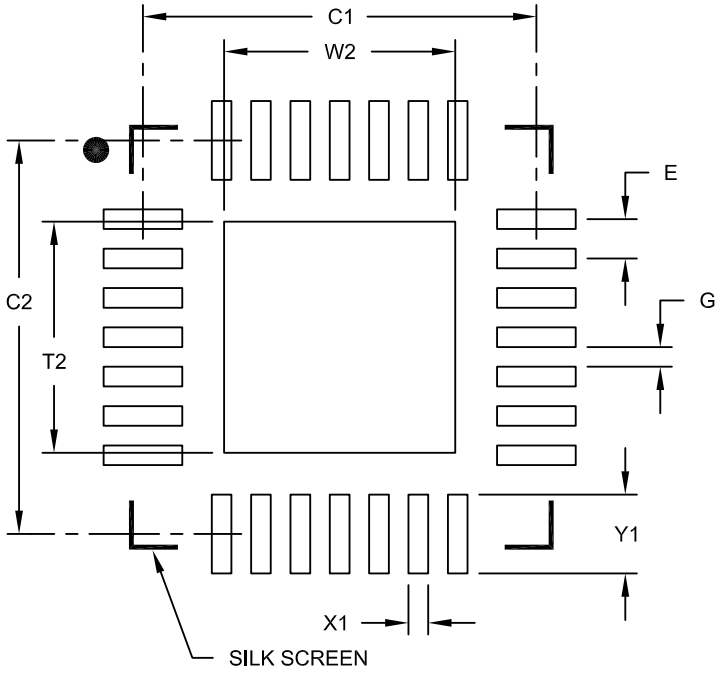
Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®

28-Lead Ultra Thin Plastic Quad Flat, No Lead Package (MV) - 4x4 mm Body [UQFN]
With 0.40 mm Contact Length

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.40 BSC | |
| Optional Center Pad Width | W2 | | | 2.35 |
| Optional Center Pad Length | T2 | | | 2.35 |
| Contact Pad Spacing | C1 | | 4.00 | |
| Contact Pad Spacing | C2 | | 4.00 | |
| Contact Pad Width (X28) | X1 | | | 0.20 |
| Contact Pad Length (X28) | Y1 | | | 0.80 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2152A