**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 | 48MHz |
| Connectivity | I²C, LINbus, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, LCD, LVD, POR, PWM, WDT |
| Number of I/O | 51 |
| Program Memory Size | 128KB (64K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 3.8K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2V ~ 3.6V |
| Data Converters | A/D 12x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 80-TQFP |
| Supplier Device Package | 80-TQFP (12x12) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic18f87j72t-i-pt |

## 3.6 Effects of Power-Managed Modes on the Various Clock Sources

When PRI_IDLE mode is selected, the designated primary oscillator continues to run without interruption. For all other power-managed modes, the oscillator using the OSC1 pin is disabled. The OSC1 pin (and OSC2 pin if used by the oscillator) will stop oscillating.

In secondary clock modes (SEC_RUN and SEC_IDLE), the Timer1 oscillator is operating and providing the device clock. The Timer1 oscillator may also run in all power-managed modes if required to clock Timer1 or Timer3.

In RC_RUN and RC_IDLE modes, the internal oscillator provides the device clock source. The 31 kHz INTRC output can be used directly to provide the clock and may be enabled to support various special features, regardless of the power-managed mode (see **Section 26.2 "Watchdog Timer (WDT)"** through **Section 26.5 "Fail-Safe Clock Monitor"** for more information on WDT, Fail-Safe Clock Monitor and Two-Speed Start-up).

If the Sleep mode is selected, all clock sources are stopped. Since all the transistor switching currents have been stopped, Sleep mode achieves the lowest current consumption of the device (only leakage currents).

Enabling any on-chip feature that will operate during Sleep will increase the current consumed during Sleep. The INTRC is required to support WDT operation. The Timer1 oscillator may be operating to support a Real-Time Clock (RTC). Other features may be operating that do not require a device clock source (i.e., MSSP slave, INTx pins and others).

Peripherals that may add significant current consumption are listed in **Section 29.1 "DC Characteristics: Power-Down and Supply Current PIC18F87J72 Family (Industrial)"**.

## 3.7 Power-up Delays

Power-up delays are controlled by two timers, so that no external Reset circuitry is required for most applications. The delays ensure that the device is kept in Reset until the device power supply is stable under normal circumstances, and the primary clock is operating and stable. For additional information on power-up delays, see **Section 5.6 "Power-up Timer (PWRT)"**.

The first timer is the Power-up Timer (PWRT), which provides a fixed delay on power-up (parameter 33, Table 29-2); it is always enabled.

The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable (HS modes). The OST does this by counting 1024 oscillator cycles before allowing the oscillator to clock the device.

There is a delay of interval, $T_{CSD}$ (parameter 38, Table 29-2), following POR, while the controller becomes ready to execute instructions.

**TABLE 3-3: OSC1 AND OSC2 PIN STATES IN SLEEP MODE**

| Oscillator Mode | OSC1 Pin | OSC2 Pin |
|---|---|---|
| EC, ECPLL | Floating, pulled by external clock | At logic low (clock/4 output) |
| HS, HSPLL | Feedback inverter disabled at quiescent voltage level | Feedback inverter disabled at quiescent voltage level |
| INTOSC, INTPLL1/2 | I/O pin, RA6, direction controlled by TRISA<6> | I/O pin, RA6, direction controlled by TRISA<7> |

**Note:** See **Section 5.0 "Reset"** for time-outs due to Sleep and $\overline{\text{MCLR}}$ Reset.

#### 4.1.3 CLOCK TRANSITIONS AND STATUS INDICATORS

The length of the transition between clock sources is the sum of two cycles of the old clock source and three to four cycles of the new clock source. This formula assumes that the new clock source is stable.

Two bits indicate the current clock source and its status: OSTS (OSCCON<3>) and T1RUN (T1CON<6>). In general, only one of these bits will be set while in a given power-managed mode. When the OSTS bit is set, the primary clock is providing the device clock. When the T1RUN bit is set, the Timer1 oscillator is providing the clock. If neither of these bits is set, INTRC is clocking the device.

| | |
|---|---|
| **Note:** | Executing a SLEEP instruction does not necessarily place the device into Sleep mode. It acts as the trigger to place the controller into either the Sleep mode, or one of the Idle modes, depending on the setting of the IDLEN bit. |

#### 4.1.4 MULTIPLE SLEEP COMMANDS

The power-managed mode that is invoked with the SLEEP instruction is determined by the setting of the IDLEN bit at the time the instruction is executed. If another SLEEP instruction is executed, the device will enter the power-managed mode specified by IDLEN at that time. If IDLEN has changed, the device will enter the new power-managed mode specified by the new setting.

### 4.2 Run Modes

In the Run modes, clocks to both the core and peripherals are active. The difference between these modes is the clock source.

#### 4.2.1 PRI_RUN MODE

The PRI_RUN mode is the normal, full-power execution mode of the microcontroller. This is also the default mode upon a device Reset unless Two-Speed Start-up is enabled (see **Section 26.4 “Two-Speed Start-up”** for details). In this mode, the OSTS bit is set (see **Section 3.2 “Control Registers”**).

#### 4.2.2 SEC_RUN MODE

The SEC_RUN mode is the compatible mode to the “clock switching” feature offered in other PIC18 devices. In this mode, the CPU and peripherals are clocked from the Timer1 oscillator. This gives users the option of lower power consumption while still using a high-accuracy clock source.

SEC_RUN mode is entered by setting the SCS<1:0> bits to ‘01’. The device clock source is switched to the Timer1 oscillator (see Figure 4-1), the primary oscillator is shut down, the T1RUN bit (T1CON<6>) is set and the OSTS bit is cleared.

| | |
|---|---|
| **Note:** | The Timer1 oscillator should already be running prior to entering SEC_RUN mode. If the T1OSCEN bit is not set when the SCS<1:0> bits are set to ‘01’, entry to SEC_RUN mode will not occur. If the Timer1 oscillator is enabled, but not yet running, device clocks will be delayed until the oscillator has started. In such situations, initial oscillator operation is far from stable and unpredictable operation may result. |

## 9.3 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Enable registers (PIE1, PIE2, PIE3). When IPEN = 0, the PEIE bit must be set to enable any of these peripheral interrupts.

**REGISTER 9-7: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1**

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | ADIE | RC1IE | TX1IE | SSPIE | — | TMR2IE | TMR1IE |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **Unimplemented:** Read as '0'

bit 6 **ADIE:** A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RC1IE:** EUSART Receive Interrupt Enable bit
1 = Enables the EUSART receive interrupt
0 = Disables the EUSART receive interrupt

bit 4 **TX1IE:** EUSART Transmit Interrupt Enable bit
1 = Enables the EUSART transmit interrupt
0 = Disables the EUSART transmit interrupt

bit 3 **SSPIE:** Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

bit 2 **Unimplemented:** Read as '0'

bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE:** TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

**REGISTER 9-12: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3**

| U-0 | R/W-1 | R-1 | R-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| — | LCDIP | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ | |
| -n = Value at POR | ‘1’ = Bit is set | ‘0’ = Bit is cleared | x = Bit is unknown |

bit 7 **Unimplemented:** Read as ‘0’

bit 6 **LCDIP:** LCD Interrupt Priority bit (valid when Type-B waveform with Non-Static mode is selected)
1 = High priority
0 = Low priority

bit 5 **RC2IP:** AUSART Receive Priority Flag bit
1 = High priority
0 = Low priority

bit 4 **TX2IP:** AUSART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **CTMUIP:** CTMU Interrupt Priority bit
1 = High priority
0 = Low priority

bit **CCP2IP:** CCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit **CCP1IP:** CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **RTCCIP:** RTCC Interrupt Priority bit
1 = High priority
0 = Low priority

## 10.6 PORTE, TRISE and LATE Registers

PORTE is a 7-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISE and LATE. All pins on PORTE are digital only and tolerate voltages up to 5.5V.

All pins on PORTE are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output. The RE7 pin is also configurable for open-drain output when CCP2 is active on this pin. Open-drain configuration is selected by setting the CCP2OD control bit (TRISG<6>)

| **Note:** | These pins are configured as digital inputs on any device Reset. |
|---|---|

Each of the PORTE pins has a weak internal pull-up. A single control bit can turn off all the pull-ups. This is performed by clearing bit, REPU (PORTG<6>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on any device Reset.

Pins, RE<6:3>, are multiplexed with the LCD common drives. I/O port functions are only available on those PORTE pins depending on which commons are active. The configuration is determined by the LMUX<1:0> control bits (LCDCON<1:0>). The availability is summarized in Table 10-11.

**TABLE 10-11: PORTE PINS AVAILABLE IN DIFFERENT LCD DRIVE CONFIGURATIONS**

| LCDCON <1:0> | Active LCD Commons | PORTE Available for I/O |
|---|---|---|
| 00 | COM0 | RE6, RE5, RE4 |
| 01 | COM0, COM1 | RE6, RE5 |
| 10 | COM0, COM1 and COM2 | RE6 |
| 11 | All (COM0 through COM3) | None |

Pins, RE1 and RE0, are multiplexed with the functions of LCDBIAS2 and LCDBIAS1. When LCD bias generation is required (i.e., any application where the device is connected to an external LCD), these pins cannot be used as digital I/O.

| **Note:** | The pin corresponding to RE2 of other PIC18F parts has the function of LCDBIAS3 in this device. It cannot be used as digital I/O. |
|---|---|

RE7 is multiplexed with the LCD segment drive (SEG31) controlled by the LCDSE3<7> bit. I/O port function is only available when the segment is disabled.

RE7 can also be configured as the alternate peripheral pin for the CCP2 module. This is done by clearing the CCP2MX Configuration bit.

**EXAMPLE 10-5: INITIALIZING PORTE**

```
CLRF    PORTE   ; Initialize PORTE by
                ; clearing output
                ; data latches
CLRF    LATE    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   03h     ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISE   ; Set RE<1:0> as inputs
                ; RE<7:2> as outputs
```

## 17.0 LIQUID CRYSTAL DISPLAY (LCD) DRIVER MODULE

The Liquid Crystal Display (LCD) driver module generates the timing control to drive a static or multiplexed LCD panel. It also provides control of the LCD pixel data. The module can drive panels of up to 132 pixels (33 segments by four commons).

The LCD driver module supports these features:

- Direct driving of LCD panel
- On-chip bias generator with dedicated charge pump to support a range of fixed and variable bias options
- Up to four commons, with four Multiplexing modes
- Up to 33 segments
- Three LCD clock sources with selectable prescaler, with a fourth source available for use with the LCD charge pump

A simplified block diagram of the module is shown in Figure 17-1.

**FIGURE 17-1: LCD DRIVER MODULE BLOCK DIAGRAM**

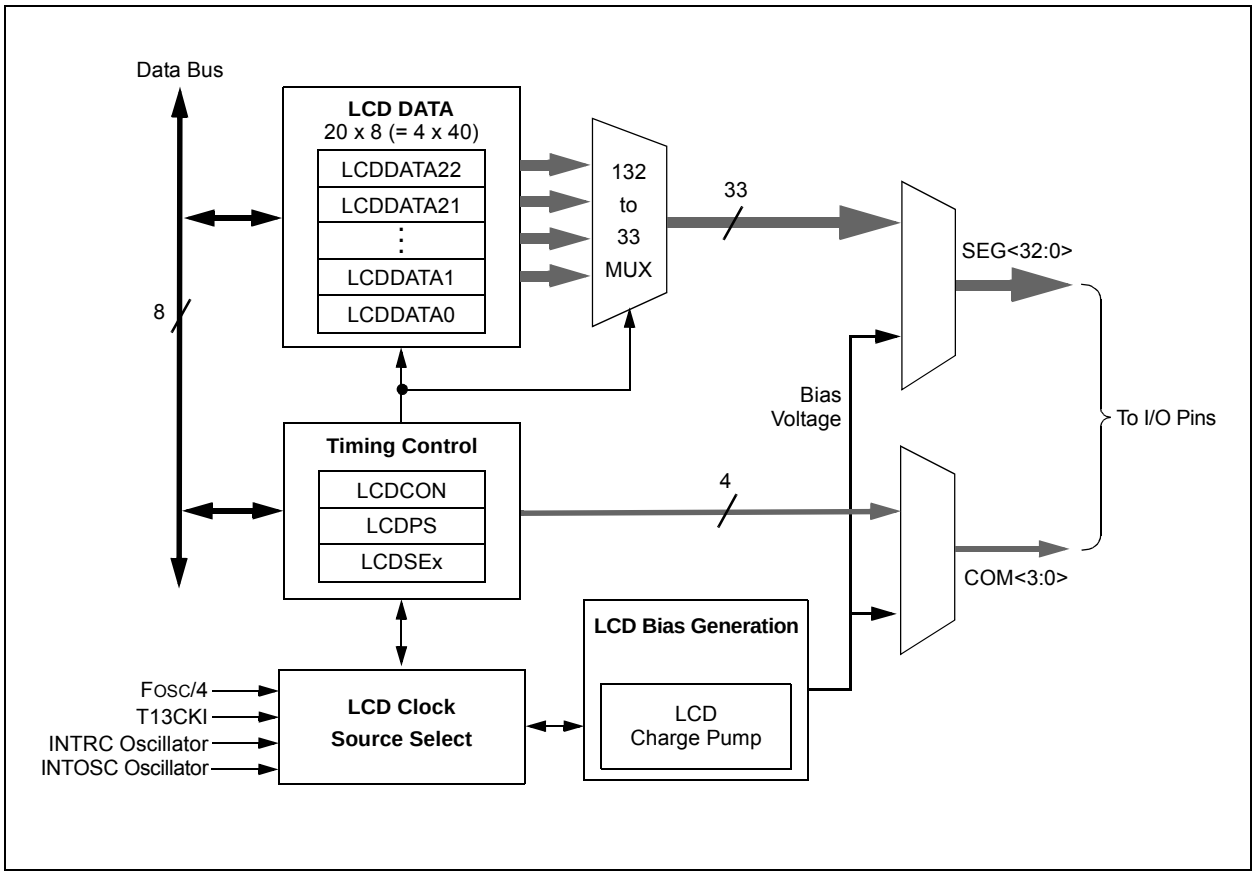
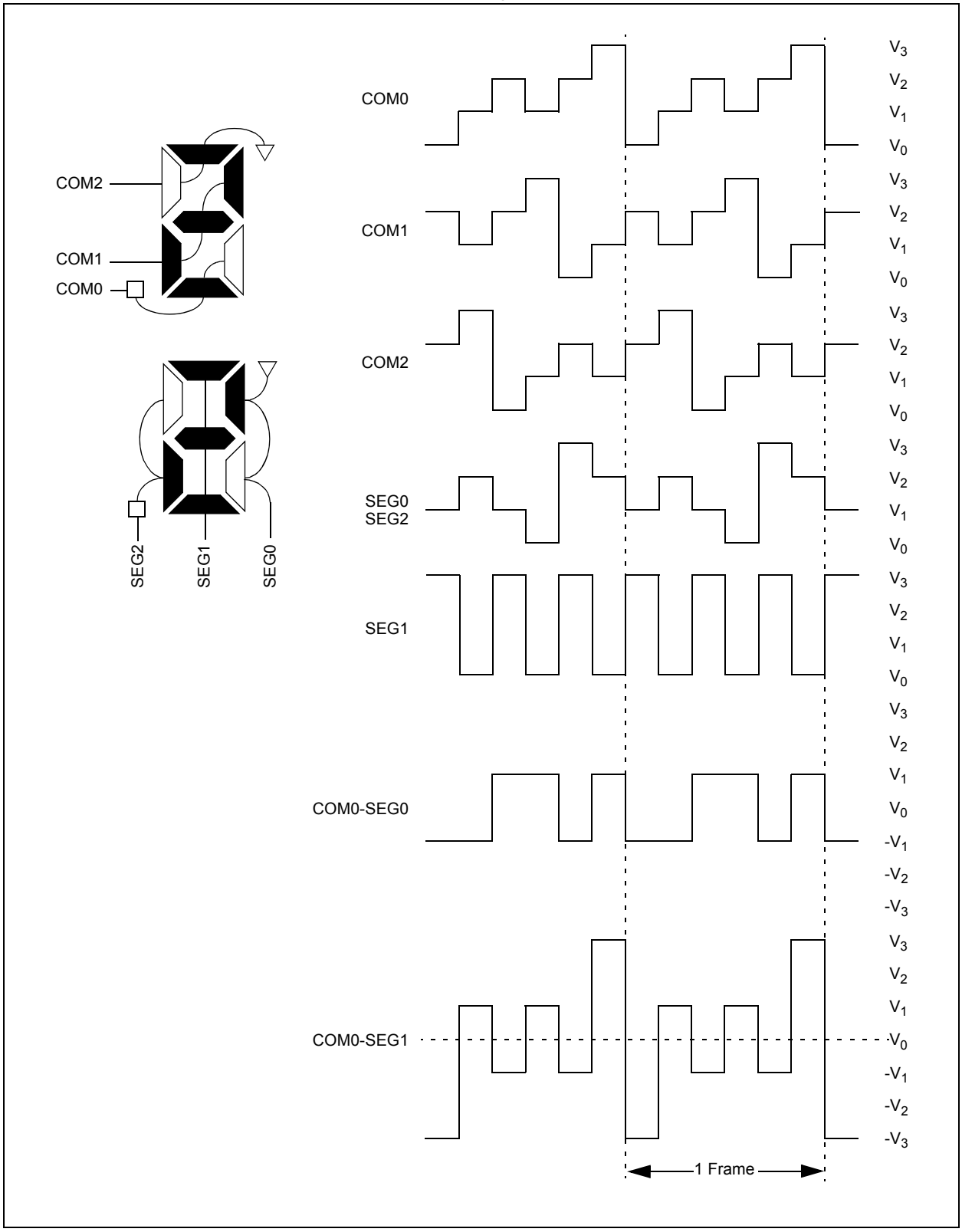


**FIGURE 17-13: TYPE-A WAVEFORMS IN 1/3 MUX, 1/3 BIAS DRIVE**



### 18.4.4 CLOCK STRETCHING

Both 7-Bit and 10-Bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCL pin to be held low at the end of each data receive sequence.

#### 18.4.4.1 Clock Stretching for 7-Bit Slave Receive Mode (SEN = 1)

In 7-Bit Slave Receive mode, on the falling edge of the ninth clock at the end of the $\overline{ACK}$ sequence, if the BF bit is set, the CKP bit in the SSPCON1 register is automatically cleared, forcing the SCL output to be held low. The CKP being cleared to '0' will assert the SCL line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCL line low, the user has time to service the ISR and read the contents of the SSPBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 18-15).

**Note 1:** If the user reads the contents of the SSPBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.

**2:** The CKP bit can be set in software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

#### 18.4.4.2 Clock Stretching for 10-Bit Slave Receive Mode (SEN = 1)

In 10-Bit Slave Receive mode, during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/$\overline{W}$ bit cleared to '0'. The release of the clock line occurs upon updating SSPADD. Clock stretching will occur on each data receive sequence as described in 7-bit mode.

**Note:** If the user polls the UA bit and clears it by updating the SSPADD register before the falling edge of the ninth clock occurs and if the user hasn't cleared the BF bit by reading the SSPBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence.

#### 18.4.4.3 Clock Stretching for 7-Bit Slave Transmit Mode

The 7-Bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock if the BF bit is clear. This occurs regardless of the state of the SEN bit.

The user's ISR must set the CKP bit before transmission is allowed to continue. By holding the SCL line low, the user has time to service the ISR and load the contents of the SSPBUF before the master device can initiate another transmit sequence (see Figure 18-10).

**Note 1:** If the user loads the contents of SSPBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.

**2:** The CKP bit can be set in software regardless of the state of the BF bit.

#### 18.4.4.4 Clock Stretching for 10-Bit Slave Transmit Mode

In 10-Bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-Bit Slave Receive mode. The first two addresses are followed by a third address sequence which contains the high-order bits of the 10-bit address and the R/$\overline{W}$ bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled by the BF flag as in 7-Bit Slave Transmit mode (see Figure 18-13).

**FIGURE 19-12: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**

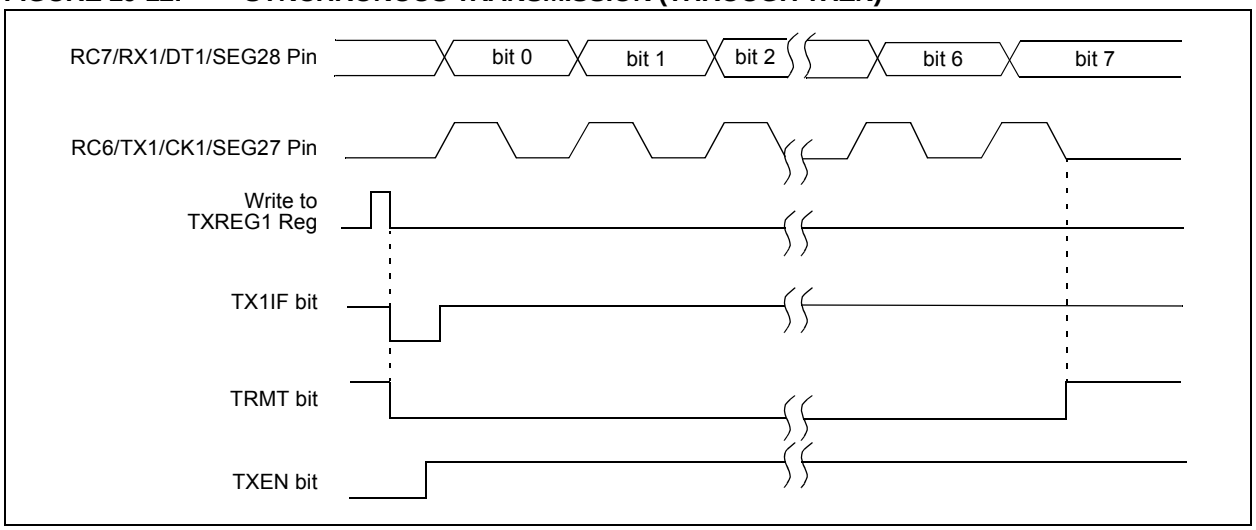



**TABLE 19-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page |
|---|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 45 |
| PIR1 | — | ADIF | RC1IF | TX1IF | SSPIF | — | TMR2IF | TMR1IF | 48 |
| PIE1 | — | ADIE | RC1IE | TX1IE | SSPIE | — | TMR2IE | TMR1IE | 48 |
| IPR1 | — | ADIP | RC1IP | TX1IP | SSPIP | — | TMR2IP | TMR1IP | 48 |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 47 |
| TXREG1 | EUSART Transmit Register | | | | | | | | 47 |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 47 |
| BAUDCON1 | ABDOVF | RCMT | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 49 |
| SPBRGH1 | EUSART Baud Rate Generator Register High Byte | | | | | | | | 49 |
| SPBRG1 | EUSART Baud Rate Generator Register Low Byte | | | | | | | | 47 |
| LATG | U2OD | U1OD | — | LATG4 | LATG3 | LATG2 | LATG1 | LATG0 | 48 |

Legend: — = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

**REGISTER 20-2: RCSTA2: AUSART RECEIVE STATUS AND CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-0 | R-0 | R-x |
|---|---|---|---|---|---|---|---|
| SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| bit 7 | | | | | | | bit 0 |

| Legend: | | | |
|---|---|---|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' | |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX2/DT2 and TX2/CK2 pins as serial port pins; TXEN must also be set to configure TX2/CK2 to transmit)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-Bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit, CREN, is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 9-bit (RX9 = 0):
Don't care.

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be cleared by reading RCREG2 register and receiving next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data
This can be an address/data bit or a parity bit and must be calculated by user firmware.

### 25.1.5 INTERRUPTS

The CTMU sets its interrupt flag (PIR3<2>) whenever the current source is enabled, then disabled. An interrupt is generated only if the corresponding interrupt enable bit (PIE3<2>) is also set. If edge sequencing is not enabled (i.e., Edge 1 must occur before Edge 2), it is necessary to monitor the edge Status bits and determine which edge occurred last and caused the interrupt.

## 25.2 CTMU Module Initialization

The following sequence is a general guideline used to initialize the CTMU module:

1. Select the current source range using the IRNG bits (CTMUICON<1:0>).
2. Adjust the current source trim using the ITRIM bits (CTMUICON<7:2>).
3. Configure the edge input sources for Edge 1 and Edge 2 by setting the EDG1SEL and EDG2SEL bits (CTMUCONL<3:2 and 6:5>).
4. Configure the input polarities for the edge inputs using the EDG1POL and EDG2POL bits (CTMUCONL<4,7>). The default configuration is for negative edge polarity (high-to-low transitions).
5. Enable edge sequencing using the EDGSEQEN bit (CTMUCONH<2>). By default, edge sequencing is disabled.
6. Select the operating mode (Measurement or Time Delay) with the TGEN bit. The default mode is Time/Capacitance Measurement.
7. Configure the module to automatically trigger an A/D conversion when the second edge event has occurred using the CTTRIG bit (CTMUCONH<0>). The conversion trigger is disabled by default.
8. Discharge the connected circuit by setting the IDISSEN bit (CTMUCONH<1>); after waiting a sufficient time for the circuit to discharge, clear IDISSEN.
9. Disable the module by clearing the CTMUEN bit (CTMUCONH<7>).
10. Clear the Edge Status bits, EDG2STAT and EDG1STAT (CTMUCONL<1:0>).
11. Enable both edge inputs by setting the EDGEN bit (CTMUCONH<3>).
12. Enable the module by setting the CTMUEN bit.

Depending on the type of measurement or pulse generation being performed, one or more additional modules may also need to be initialized and configured with the CTMU module:

- Edge Source Generation: In addition to the external edge input pins, CCPx Special Event Triggers can be used as edge sources for the CTMU.
- Capacitance or Time Measurement: The CTMU module uses the A/D Converter to measure the voltage across a capacitor that is connected to one of the analog input channels.
- Pulse Generation: When generating system clock independent output pulses, the CTMU module uses Comparator 2 and the associated comparator voltage reference.

## 25.3 Calibrating the CTMU Module

The CTMU requires calibration for precise measurements of capacitance and time, as well as for accurate time delay. If the application only requires measurement of a relative change in capacitance or time, calibration is usually not necessary. An example of this type of application would include a capacitive touch switch, in which the touch circuit has a baseline capacitance, and the added capacitance of the human body changes the overall capacitance of a circuit.

If actual capacitance or time measurement is required, two hardware calibrations must take place: the current source needs calibration to set it to a precise current, and the circuit being measured needs calibration to measure and/or nullify all other capacitance other than that to be measured.

### 25.3.1 CURRENT SOURCE CALIBRATION

The current source onboard the CTMU module has a range of ±60% nominal for each of three current ranges. Therefore, for precise measurements, it is possible to measure and adjust this current source by placing a high-precision resistor, $R_{CAL}$, onto an unused analog channel. An example circuit is shown in Figure 25-2. The current source measurement is performed using the following steps:

1. Initialize the A/D Converter.
2. Initialize the CTMU.
3. Enable the current source by setting EDG1STAT (CTMUCONL<0>).
4. Issue settling time delay.
5. Perform A/D conversion.
6. Calculate the current source current using $I = V/R_{CAL}$, where $R_{CAL}$ is a high-precision resistance and $V$ is measured by performing an A/D conversion.

**EXAMPLE 25-3: CAPACITANCE CALIBRATION ROUTINE**

```
#include "p18cxxx.h"

#define COUNT 25                        //@ 8MHz INTFRC = 62.5 us.
#define ETIME COUNT*2.5                 //time in uS
#define DELAY for(i=0;i<COUNT;i++)
#define ADSCALE 1023                    //for unsigned conversion 10 sig bits
#define ADREF 3.3                       //Vdd connected to A/D Vr+
#define RCAL .027                       //R value is 4200000 (4.2M)
                                        //scaled so that result is in
                                        //1/100th of uA

int main(void)
{
    int i;
    int j = 0;                          //index for loop
    unsigned int Vread = 0;
    float CTMUISrc, CTMUCap, Vavg, VTot, Vcal;

//assume CTMU and A/D have been setup correctly
//see Example 25-1 for CTMU & A/D setup
setup();

CTMUCONHbits.CTMUEN = 1;                //Enable the CTMU
    for(j=0;j<10;j++)
    {
        CTMUCONHbits.IDISSEN = 1;       //drain charge on the circuit
        DELAY;                          //wait 125us
        CTMUCONHbits.IDISSEN = 0;       //end drain of circuit

        CTMUCONLbits.EDG1STAT = 1;      //Begin charging the circuit
                                        //using CTMU current source
        DELAY;                          //wait for 125us
        CTMUCONLbits.EDG1STAT = 0;      //Stop charging circuit

        PIR1bits.ADIF = 0;              //make sure A/D Int not set
        ADCON0bits.GO=1;                //and begin A/D conv.
        while(!PIR1bits.ADIF);          //Wait for A/D convert complete


        Vread = ADRES;                  //Get the value from the A/D
        PIR1bits.ADIF = 0;              //Clear A/D Interrupt Flag
        VTot += Vread;                  //Add the reading to the total
    }

    Vavg = (float)(VTot/10.000);        //Average of 10 readings
    Vcal = (float)(Vavg/ADSCALE*ADREF);
    CTMUISrc = Vcal/RCAL;               //CTMUISrc is in 1/100ths of uA
    CTMUCap = (CTMUISrc*ETIME/Vcal)/100;
}
```

**TABLE 27-2: PIC18F87J72 FAMILY INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|
| | | | MSb | | | LSb | | |
| **LITERAL OPERATIONS** | | | | | | | | |
| ADDLW k | Add Literal and WREG | 1 | 0000 | 1111 | kkkk | kkkk | C, DC, Z, OV, N | |
| ANDLW k | AND Literal with WREG | 1 | 0000 | 1011 | kkkk | kkkk | Z, N | |
| IORLW k | Inclusive OR Literal with WREG | 1 | 0000 | 1001 | kkkk | kkkk | Z, N | |
| LFSR f, k | Move literal (12-bit) 2nd word | 2 | 1110 | 1110 | 00ff | kkkk | None | |
| | to FSR(f) 1st word | | 1111 | 0000 | kkkk | kkkk | | |
| MOVLB k | Move Literal to BSR<3:0> | 1 | 0000 | 0001 | 0000 | kkkk | None | |
| MOVLW k | Move Literal to WREG | 1 | 0000 | 1110 | kkkk | kkkk | None | |
| MULLW k | Multiply Literal with WREG | 1 | 0000 | 1101 | kkkk | kkkk | None | |
| RETLW k | Return with Literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | |
| SUBLW k | Subtract WREG from Literal | 1 | 0000 | 1000 | kkkk | kkkk | C, DC, Z, OV, N | |
| XORLW k | Exclusive OR Literal with WREG | 1 | 0000 | 1010 | kkkk | kkkk | Z, N | |
| **DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS** | | | | | | | | |
| TBLRD* | Table Read | 2 | 0000 | 0000 | 0000 | 1000 | None | |
| TBLRD*+ | Table Read with Post-Increment | | 0000 | 0000 | 0000 | 1001 | None | |
| TBLRD*- | Table Read with Post-Decrement | | 0000 | 0000 | 0000 | 1010 | None | |
| TBLRD+* | Table Read with Pre-Increment | | 0000 | 0000 | 0000 | 1011 | None | |
| TBLWT* | Table Write | 2 | 0000 | 0000 | 0000 | 1100 | None | |
| TBLWT*+ | Table Write with Post-Increment | | 0000 | 0000 | 0000 | 1101 | None | |
| TBLWT*- | Table Write with Post-Decrement | | 0000 | 0000 | 0000 | 1110 | None | |
| TBLWT+* | Table Write with Pre-Increment | | 0000 | 0000 | 0000 | 1111 | None | |

**Note 1:** When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

**2:** If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

**3:** If the Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

**4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

**BZ** **Branch if Zero**

Syntax: BZ n

Operands: -128 ≤ n ≤ 127

Operation: if Zero bit is '1',
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0000 | nnnn | nnnn |

Description: If the Zero bit is '1', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example:
```
HERE    BZ   Jump
Before Instruction
    PC      =   address (HERE)
After Instruction
    If Zero     =   1;
        PC      =   address (Jump)
    If Zero     =   0;
        PC      =   address (HERE + 2)
```

**CALL** **Subroutine Call**

Syntax: CALL k {,s}

Operands: 0 ≤ k ≤ 1048575
s ∈ [0,1]

Operation: (PC) + 4 → TOS,
k → PC<20:1>;
if s = 1
(W) → WS,
(STATUS) → STATUSS,
(BSR) → BSRS

Status Affected: None

Encoding:

| | | | | |
|---|---|---|---|---|
| 1st word (k<7:0>) | 1110 | 110s | $k_7$kkk | kkk$k_0$ |
| 2nd word(k<19:8>) | 1111 | $k_{19}$kkk | kkkk | kkk$k_8$ |

Description: Subroutine call of entire 2-Mbyte memory range. First, return address (PC+ 4) is pushed onto the return stack. If 's' = 1, the W, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs. Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a 2-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<7:0>, | Push PC to stack | Read literal 'k'<19:8>, Write to PC |
| No operation | No operation | No operation | No operation |

Example:
```
HERE    CALL   THERE,1
Before Instruction
    PC      =   address (HERE)
After Instruction
    PC      =   address (THERE)
    TOS     =   address (HERE + 4)
    WS      =   W
    BSRS    =   BSR
    STATUSS =   STATUS
```

**MOVSS** **Move Indexed to Indexed**

Syntax: MOVSS [$z_s$], [$z_d$]

Operands: $0 \le z_s \le 127$
$0 \le z_d \le 127$

Operation: ((FSR2) + $z_s$) → ((FSR2) + $z_d$)

Status Affected: None

Encoding:

| | | | | |
|---|---|---|---|---|
| 1st word (source) | 1110 | 1011 | 1zzz | $zzzz_s$ |
| 2nd word (dest.) | 1111 | xxxx | xzzz | $zzzz_d$ |

Description: The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets, '$z_s$' or '$z_d$', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).

The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

If the resultant source address points to an Indirect Addressing register, the value returned will be 00h. If the resultant destination address points to an Indirect Addressing register, the instruction will execute as a NOP.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Determine source addr | Determine source addr | Read source reg |
| Decode | Determine dest addr | Determine dest addr | Write to dest reg |

Example: MOVSS [05h], [06h]

Before Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 11h
After Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 33h

**PUSHL** **Store Literal at FSR2, Decrement FSR2**

Syntax: PUSHL k

Operands: $0 \le k \le 255$

Operation: k → (FSR2),
FSR2 – 1 → FSR2

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1111 | 1010 | kkkk | kkkk |

Description: The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation.

This instruction allows users to push values onto a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read 'k' | Process data | Write to destination |

Example: PUSHL 08h

Before Instruction
FSR2H:FSR2L = 01ECh
Memory (01ECh) = 00h

After Instruction
FSR2H:FSR2L = 01EBh
Memory (01ECh) = 08h

**TABLE 29-7: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)**

| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|
| 83 | TscH2ssH, TscL2ssH | $\overline{SS}$ ↑ after SCK Edge | 1.5 Tcy + 40 | — | ns | |

**Note 1:** Requires the use of Parameter #73A.

**2:** Only if Parameter #71A and #72A are used.

**FIGURE 29-12: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)**

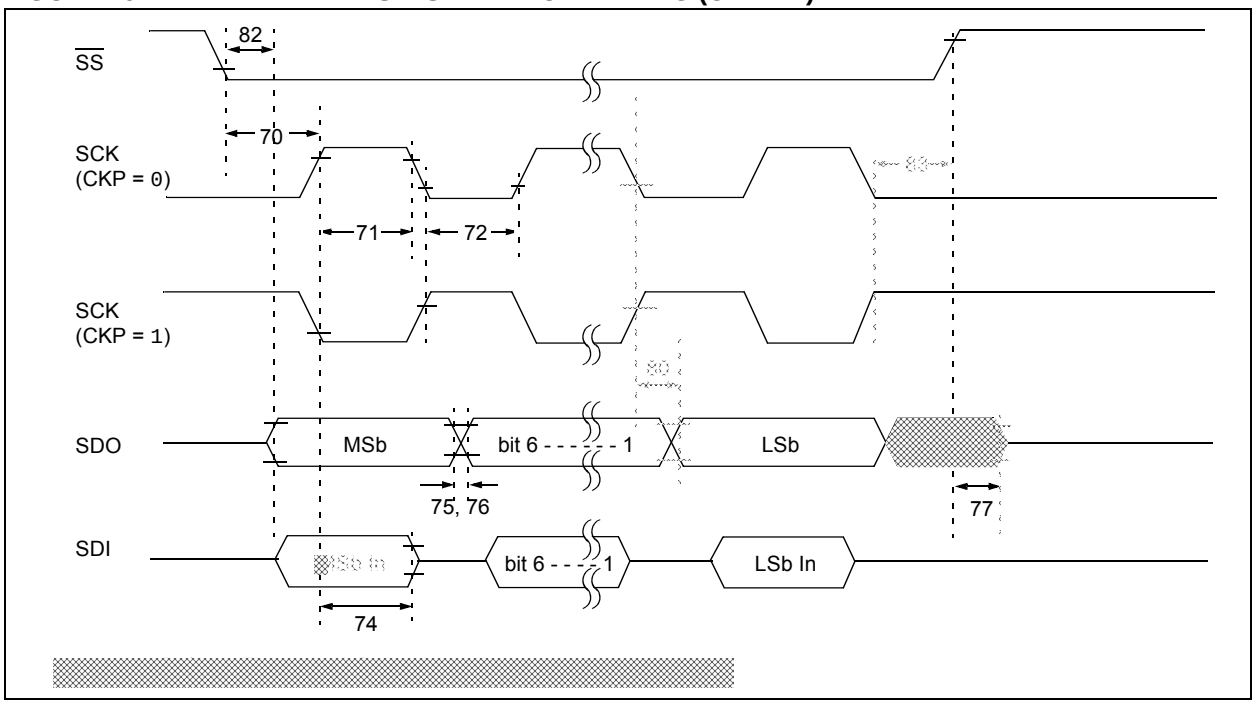



**TABLE 29-8: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)**

| Param No. | Symbol | Characteristic | | Min. | Max. | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 70 | TssL2scH, TssL2scL | $\overline{SS}$ ↓ to SCK ↓ or SCK ↑ Input | | 3 Tcy | — | ns | |
| 70A | TssL2WB | $\overline{SS}$ to Write to SSPBUF | | 3 Tcy | — | ns | |
| 71 | TscH | SCK Input High Time (Slave mode) | Continuous | 1.25 Tcy + 30 | — | ns | |
| 71A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 72 | TscL | SCK Input Low Time (Slave mode) | Continuous | 1.25 Tcy + 30 | — | ns | |
| 72A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 73A | Tb2b | Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2 | | 1.5 Tcy + 40 | — | ns | **(Note 2)** |
| 74 | TscH2diL, TscL2diL | Hold Time of SDI Data Input to SCK Edge | | 100 | — | ns | |
| 75 | TdoR | SDO Data Output Rise Time | | — | 25 | ns | |

**Note 1:** Requires the use of Parameter #73A.

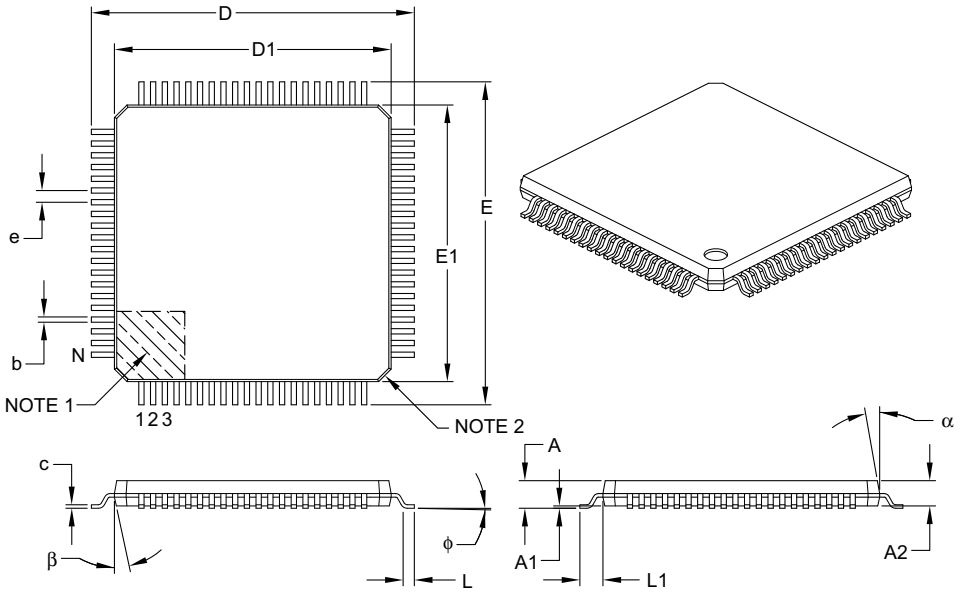
**2:** Only if Parameter #71A and #72A are used.

## 30.2 Package Details

The following sections give the technical details of the packages.

### 80-Lead Plastic Thin Quad Flatpack (PT) – 12x12x1 mm Body, 2.00 mm [TQFP]

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




| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Leads | N | | 80 | |
| Lead Pitch | e | | 0.50 BSC | |
| Overall Height | A | – | – | 1.20 |
| Molded Package Thickness | A2 | 0.95 | 1.00 | 1.05 |
| Standoff | A1 | 0.05 | – | 0.15 |
| Foot Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | | 1.00 REF | |
| Foot Angle | ϕ | 0° | 3.5° | 7° |
| Overall Width | E | | 14.00 BSC | |
| Overall Length | D | | 14.00 BSC | |
| Molded Package Width | E1 | | 12.00 BSC | |
| Molded Package Length | D1 | | 12.00 BSC | |
| Lead Thickness | c | 0.09 | – | 0.20 |
| Lead Width | b | 0.17 | 0.22 | 0.27 |
| Mold Draft Angle Top | α | 11° | 12° | 13° |
| Mold Draft Angle Bottom | β | 11° | 12° | 13° |

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

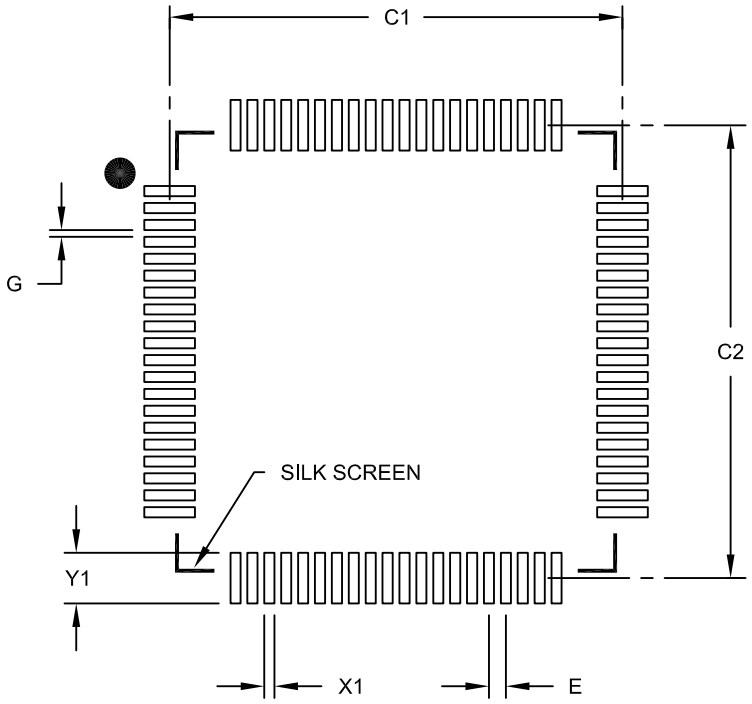
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-092B

80-Lead Plastic Thin Quad Flatpack (PT) - 12x12x1mm Body, 2.00 mm Footprint [TQFP]

**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 | | 0.50 BSC | |
| Contact Pad Spacing | C1 | | 13.40 | |
| Contact Pad Spacing | C2 | | 13.40 | |
| Contact Pad Width (X80) | X1 | | | 0.30 |
| Contact Pad Length (X80) | Y1 | | | 1.50 |
| 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. C04-2092B

### B.4.5.1 ADC Resolution as a Function of OSR

The ADC resolution is a function of the OSR (**Section B.4.4 "SINC3 Filter"**). The resolution is the same for both channels. No matter what the resolution is, the ADC output data is always presented in 24-bit words, with added zeros at the end, if the OSR is not large enough to produce 24-bit resolution (left justification).

**TABLE B-5: OSR = 256 OUTPUT CODE EXAMPLES**

| ADC Output Code (MSB First) | Hexadecimal | Decimal |
|---|---|---|
| 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0x7FFFFF | + 8,388,607 |
| 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 | 0x7FFFFE | + 8,388,606 |
| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0x000000 | 0 |
| 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0xFFFFFF | -1 |
| 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | 0x800001 | - 8,388,607 |
| 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0x800000 | - 8,388,608 |

**TABLE B-6: OSR = 128 OUTPUT CODE EXAMPLES**

| ADC Output Code (MSB First) | Hexadecimal | Decimal 23-Bit Resolution |
|---|---|---|
| 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 | 0x7FFFFE | + 4,194,303 |
| 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 | 0x7FFFFC | + 4,194,302 |
| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0x000000 | 0 |
| 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 | 0xFFFFFE | -1 |
| 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 | 0x800002 | - 4,194,303 |
| 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0x800000 | - 4,194,304 |

**TABLE B-7: OSR = 64 OUTPUT CODE EXAMPLES**

| ADC Output Code (MSB First) | Hexadecimal | Decimal 20-Bit resolution |
|---|---|---|
| 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 | 0x7FFFF0 | + 524, 287 |
| 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 | 0x7FFFE0 | + 524, 286 |
| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0x000000 | 0 |
| 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 | 0xFFFFF0 | -1 |
| 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 | 0x800010 | - 524,287 |
| 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0x800000 | - 524, 288 |

**TABLE B-8: OSR = 32 OUTPUT CODE EXAMPLES**

| ADC Output Code (MSB First) | Hexadecimal | Decimal 17-Bit resolution |
|---|---|---|
| 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 | 0x7FFF80 | + 65, 535 |
| 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 | 0x7FFF00 | + 65, 534 |
| 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0x000000 | 0 |
| 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 | 0xFFFF80 | -1 |
| 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 | 0x800080 | - 65,535 |
| 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | 0x800000 | - 65, 536 |

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.



*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

**QUALITY MANAGEMENT SYSTEM**
**CERTIFIED BY DNV**
**═ ISO/TS 16949 ═**

**Trademarks**



ISBN: 978-1-5224-0933-5