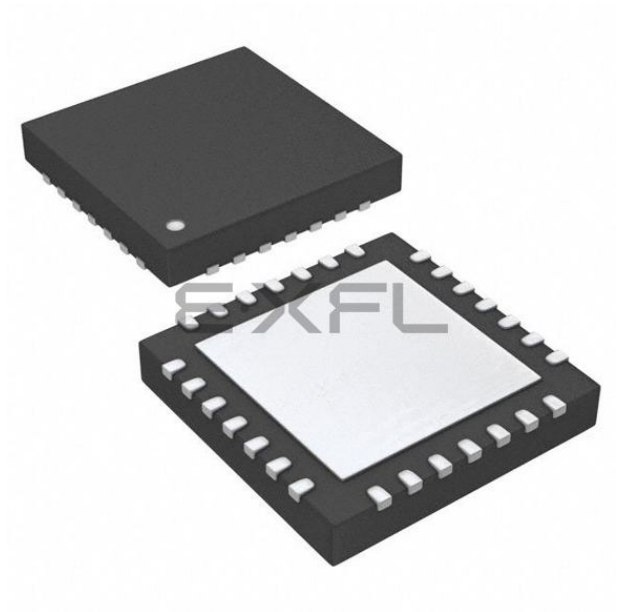


Welcome to E-XFL.COM

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, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	2.15V ~ 3.6V
Data Converters	A/D 10x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f24j11t-i-ml

NOTES:

REGISTER 3-1: OSCTUNE: OSCILLATOR TUNING REGISTER (ACCESS F9Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INTSRC	PLLEN	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0
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 **INTSRC:** Internal Oscillator Low-Frequency Source Select bit
1 = 31.25 kHz device clock derived from 8 MHz INTOSC source (divide-by-256 enabled)
0 = 31 kHz device clock derived directly from INTRC internal oscillator

bit 6 **PLLEN:** Frequency Multiplier Enable bit
1 = PLL enabled
0 = PLL disabled

bit 5-0 **TUN<5:0>:** Frequency Tuning bits
011111 = Maximum frequency
011110
•
•
•
000001
000000 = Center frequency; oscillator module is running at the calibrated frequency
111111
•
•
•
100000 = Minimum frequency

3.3 Clock Sources and Oscillator Switching

Like previous PIC18 enhanced devices, the PIC18F46J11 family includes a feature that allows the device clock source to be switched from the main oscillator to an alternate, low-frequency clock source. PIC18F46J11 family devices offer two alternate clock sources. When an alternate clock source is enabled, the various power-managed operating modes are available.

Essentially, there are three clock sources for these devices:

- Primary Oscillators
- Secondary Oscillators
- Internal Oscillator Block

The **Primary Oscillators** include the External Crystal and Resonator modes, the External Clock modes and the internal oscillator block. The particular mode is defined by the FOSC<2:0> Configuration bits. The details of these modes are covered earlier in this chapter.

The **Secondary Oscillators** are external sources that are not connected to the OSC1 or OSC2 pins. These sources may continue to operate even after the controller is placed in a power-managed mode.

PIC18F46J11 family devices offer the Timer1 oscillator as a secondary oscillator. This oscillator, in all power-managed modes, is often the time base for functions such as a Real-Time Clock (RTC). Most often, a 32.768 kHz watch crystal is connected between the RC0/T1OSO/T1CKI/RP11 and RC1/T1OSI/RP12 pins. Like the HS Oscillator mode circuits, loading capacitors are also connected from each pin to ground. The Timer1 oscillator is discussed in more detail in **Section 13.5 "Timer1 Oscillator"**.

In addition to being a primary clock source, the **postscaled internal clock** is available as a power-managed mode clock source. The INTRC source is also used as the clock source for several special features, such as the WDT and Fail-Safe Clock Monitor (FSCM).

4.0 LOW-POWER MODES

The PIC18F46J11 family devices can manage power consumption through clocking to the CPU and the peripherals. In general, reducing the clock frequency and the amount of circuitry being clocked reduces power consumption.

For managing power in an application, the primary modes of operation are:

- Run Mode
- Idle Mode
- Sleep Mode
- Deep Sleep Mode

Additionally, there is an Ultra Low-Power Wake-up (ULPWU) mode for generating an interrupt-on-change on RA0.

These modes define which portions of the device are clocked and at what speed.

- The Run and Idle modes can use any of the three available clock sources (primary, secondary or internal oscillator blocks).
- The Sleep mode does not use a clock source.

The ULPWU mode on RA0 allows a slow falling voltage to generate an interrupt-on-change on RA0 without excess current consumption. See **Section 4.7 "Ultra Low-Power Wake-up"**.

The power-managed modes include several power-saving features offered on previous PIC® devices, such as clock switching, ULPWU and Sleep mode. In addition, the PIC18F46J11 family devices add a new power-managed Deep Sleep mode.

4.1 Selecting Power-Managed Modes

Selecting a power-managed mode requires these decisions:

- Will the CPU be clocked?
- If so, which clock source will be used?

The IDLEN bit (OSCCON<7>) controls CPU clocking and the SCS<1:0> bits (OSCCON<1:0>) select the clock source. The individual modes, bit settings, clock sources and affected modules are summarized in Table 4-1.

4.1.1 CLOCK SOURCES

The SCS<1:0> bits allow the selection of one of three clock sources for power-managed modes. They are:

- Primary clock source – Defined by the FOSC<2:0> Configuration bits
- Timer1 clock – Provided by the secondary oscillator
- Postscaled internal clock – Derived from the internal oscillator block

4.1.2 ENTERING POWER-MANAGED MODES

Switching from one clock source to another begins by loading the OSCCON register. The SCS<1:0> bits select the clock source.

Changing these bits causes an immediate switch to the new clock source, assuming that it is running. The switch also may be subject to clock transition delays. These delays are discussed in **Section 4.1.3 "Clock Transitions and Status Indicators"** and subsequent sections.

Entry to the power-managed Idle or Sleep modes is triggered by the execution of a SLEEP instruction. The actual mode that results depends on the status of the IDLEN bit.

Depending on the current mode and the mode being switched to, a change to a power-managed mode does not always require setting all of these bits. Many transitions may be done by changing the oscillator select bits, the IDLEN bit or the DSEN bit prior to issuing a SLEEP instruction.

If the IDLEN and DSEN bits are already configured correctly, it may only be necessary to perform a SLEEP instruction to switch to the desired mode.

TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
IPR1	PIC18F2XJ11	PIC18F4XJ11	1111 1111	1111 1111	uuuu uuuu
PIR1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu(3)
PIE1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
RCSTA2	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
OSCTUNE	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
T1GCON	PIC18F2XJ11	PIC18F4XJ11	0000 0x00	0000 0x00	uuuu uxuu
RTCVALH	PIC18F2XJ11	PIC18F4XJ11	0xxx xxxx	0uuu uuuu	0uuu uuuu
RTCVALL	PIC18F2XJ11	PIC18F4XJ11	0xxx xxx	0uuu uuuu	0uuu uuuu
T3GCON	PIC18F2XJ11	PIC18F4XJ11	0000 0x00	uuuu uxuu	uuuu uxuu
TRISE(5)	—	PIC18F4XJ11	---- -111	---- -111	---- -uuu
TRISD(5)	—	PIC18F4XJ11	1111 1111	1111 1111	uuuu uuuu
TRISC	PIC18F2XJ11	PIC18F4XJ11	1111 1111	1111 1111	uuuu uuuu
TRISB	PIC18F2XJ11	PIC18F4XJ11	1111 1111	1111 1111	uuuu uuuu
TRISA	PIC18F2XJ11	PIC18F4XJ11	111- 1111	111- 1111	uuu- uuuu
ALRMCFG	PIC18F2XJ11	PIC18F4XJ11	0000 0000	uuuu uuuu	uuuu uuuu
ALRMRPT	PIC18F2XJ11	PIC18F4XJ11	0000 0000	uuuu uuuu	uuuu uuuu
ALRMVALH	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
ALRMVALL	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATE(5)	—	PIC18F4XJ11	---- -xxx	---- -uuu	---- -uuu
LATD(5)	—	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATC	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA	PIC18F2XJ11	PIC18F4XJ11	xxx- xxxx	uuu- uuuu	uuu- uuuu
DMACON1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
DMACON2	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
HLVDCON	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu
PORTE(5)	—	PIC18F4XJ11	00-- -xxx	uu-- -uuu	uu-- -uuu
PORTD(5)	—	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	PIC18F2XJ11	PIC18F4XJ11	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	PIC18F2XJ11	PIC18F4XJ11	xxx- xxxx	uuu- uuuu	uuu- uuuu
SPBRGH1	PIC18F2XJ11	PIC18F4XJ11	0000 0000	0000 0000	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition.

Note 1: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

4: See Table 5-1 for Reset value for specific condition.

5: Not implemented for PIC18F2XJ11 devices.

6: Not implemented on "LF" devices.

7.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed on 1 byte at a time. A write to program memory is executed on blocks of 64 bytes at a time or 2 bytes at a time. Program memory is erased in blocks of 1024 bytes at a time. A bulk erase operation may not be issued from user code.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a NOP.

7.1 Table Reads and Table Writes

In order to read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

- Table Read (TBLRD)
- Table Write (TBLWT)

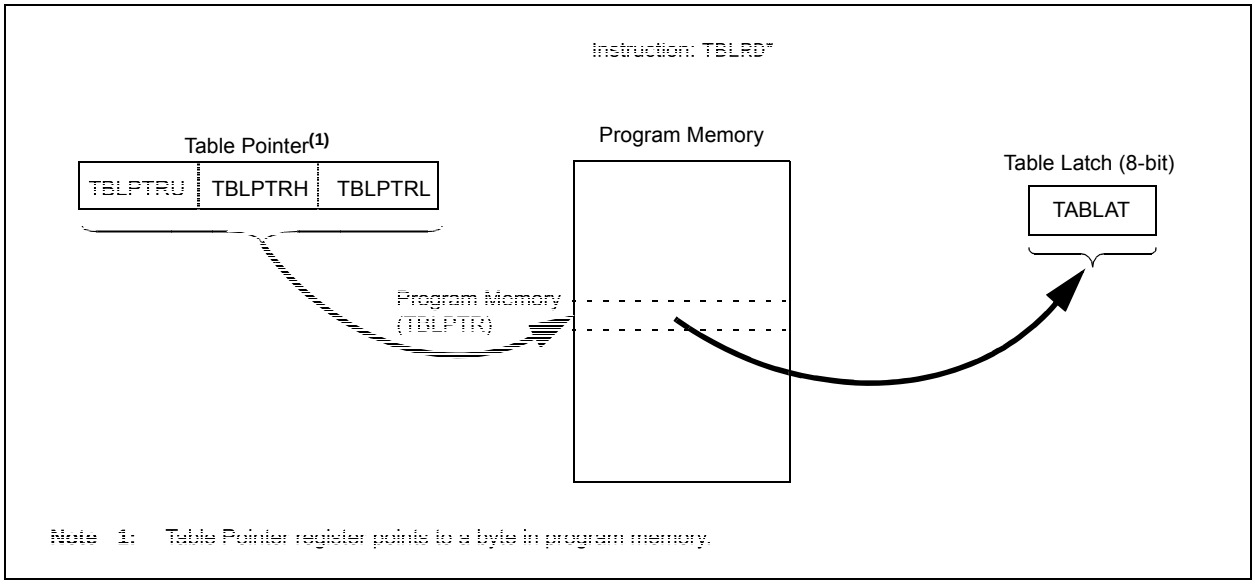
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into the data RAM space. Figure 7-1 illustrates the operation of a table read with program memory and data RAM.

Table write operations store data from the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in **Section 7.5 "Writing to Flash Program Memory"**. Figure 7-2 illustrates the operation of a table write with program memory and data RAM.

Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word-aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word-aligned.

FIGURE 7-1: TABLE READ OPERATION



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 (ACCESS F9Dh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PMPIE(1)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	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 **PMPIE:** Parallel Master Port Read/Write Interrupt Enable bit(1)
1 = Enables the PMP read/write interrupt
0 = Disables the PMP read/write interrupt

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

bit 5 **RC1IE:** EUSART1 Receive Interrupt Enable bit
1 = Enables the EUSART1 receive interrupt
0 = Disables the EUSART1 receive interrupt

bit 4 **TX1IE:** EUSART1 Transmit Interrupt Enable bit
1 = Enables the EUSART1 transmit interrupt
0 = Disables the EUSART1 transmit interrupt

bit 3 **SSP1IE:** Master Synchronous Serial Port 1 Interrupt Enable bit
1 = Enables the MSSP1 interrupt
0 = Disables the MSSP1 interrupt

bit 2 **CCP1IE:** ECCP1 Interrupt Enable bit
1 = Enables the ECCP1 interrupt
0 = Disables the ECCP1 interrupt

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

Note 1: These bits are unimplemented on 28-pin devices.

10.7.6 PERIPHERAL PIN SELECT REGISTERS

The PIC18F46J11 family of devices implements a total of 37 registers for remappable peripheral configuration of 44-pin devices. The 28-pin devices have 31 registers for remappable peripheral configuration.

Note: Input and output register values can only be changed if PPS<IOLOCK> = 0. See Example 10-7 for a specific command sequence.

REGISTER 10-5: PPSCON: PERIPHERAL PIN SELECT INPUT REGISTER 0 (BANKED EFFh)(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	IOLOCK
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 7-1 **Unimplemented:** Read as '0'

bit 0 **IOLOCK:** I/O Lock Enable bit

1 = I/O lock active, RPORx and RPINRx registers are write-protected
0 = I/O lock not active, pin configurations can be changed

Note 1: Register values can only be changed if PPSCON<IOLOCK> = 0.

FIGURE 11-15: WRITE TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS

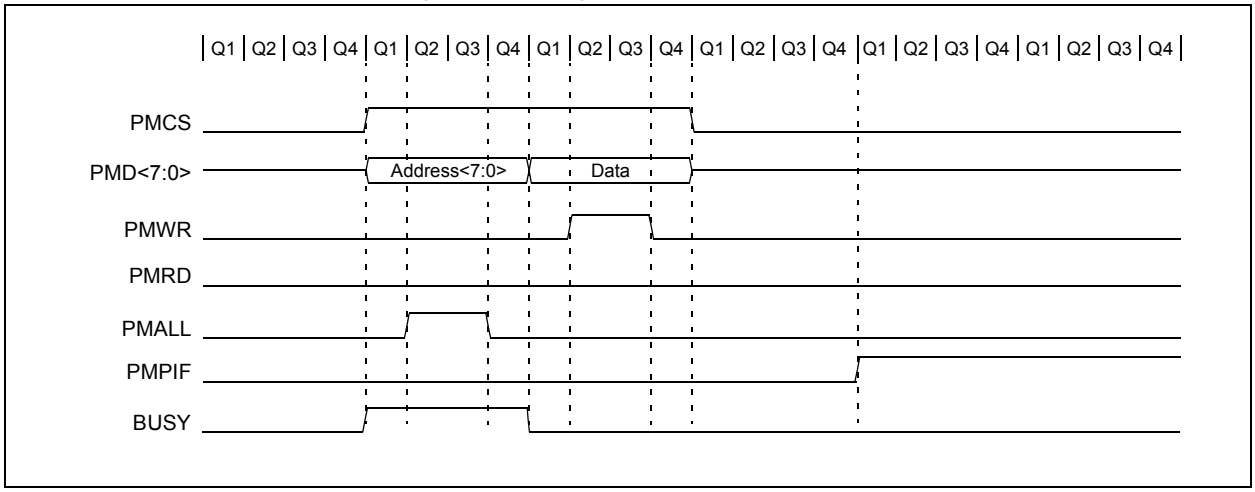


FIGURE 11-16: WRITE TIMING, 8-BIT DATA, WAIT STATES ENABLED, PARTIALLY MULTIPLEXED ADDRESS

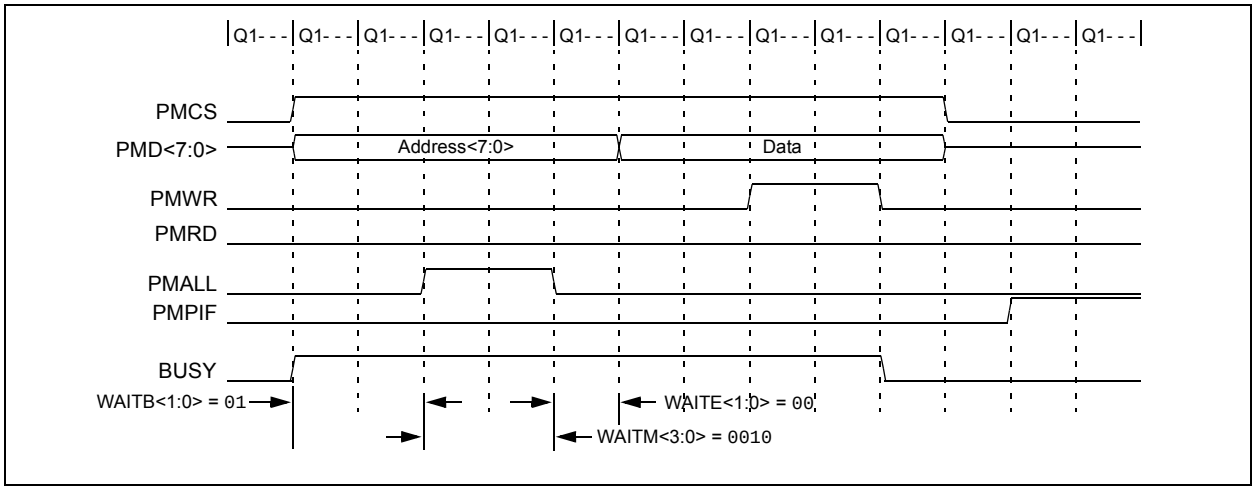
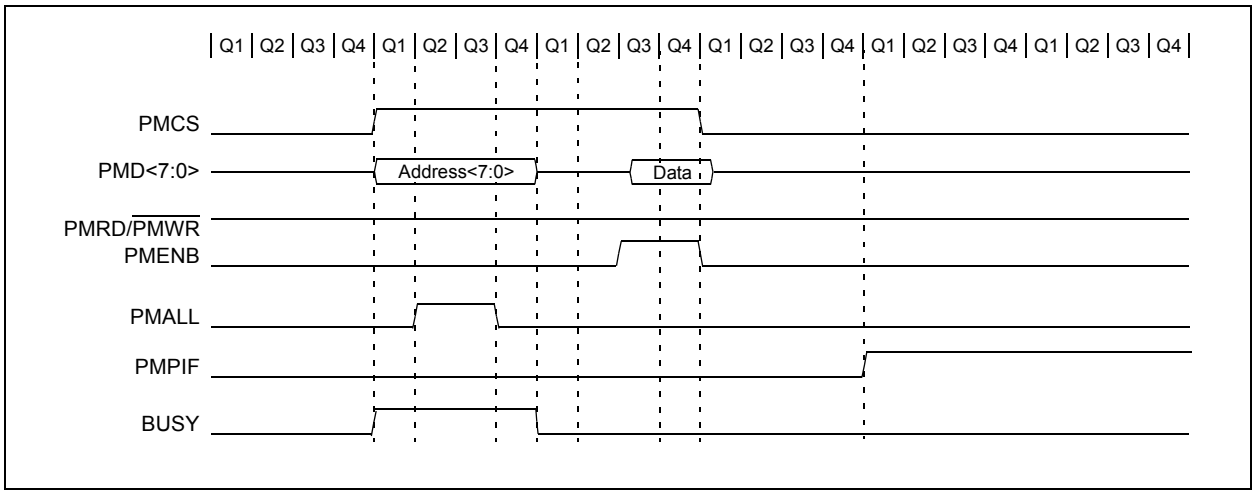


FIGURE 11-17: READ TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS, ENABLE STROBE



19.3.4 ENABLING SPI I/O

To enable the serial port, MSSP Enable bit, SSPEN (SSPxCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, reinitialize the SSPxCON1 registers and then set the SSPEN bit. This configures the SDIx, SDOx, SCKx and $\overline{SSx}$ pins as serial port pins. For the pins to behave as the serial port function, the appropriate TRIS bits, ANCON/PCFG bits and Peripheral Pin Select registers (if using MSSP2) should be correctly initialized prior to setting the SSPEN bit.

A typical SPI serial port initialization process follows:

- Initialize ODCON3 register (optional open-drain output control)
- Initialize remappable pin functions (if using MSSP2, see **Section 10.7 "Peripheral Pin Select (PPS)"**)
- Initialize SCKx LAT value to desired Idle SCK level (if master device)
- Initialize SCKx ANCON/PCFG bit (if Slave mode and multiplexed with ANx function)
- Initialize SCKx TRIS bit as output (Master mode) or input (Slave mode)
- Initialize SDIx ANCON/PCFG bit (if SDIx is multiplexed with ANx function)
- Initialize SDIx TRIS bit
- Initialize $\overline{SSx}$ ANCON/PCFG bit (if Slave mode and multiplexed with ANx function)
- Initialize $\overline{SSx}$ TRIS bit (Slave modes)
- Initialize SDOx TRIS bit
- Initialize SSPxSTAT register
- Initialize SSPxCON1 register
- Set SSPEN bit to enable the module

Any MSSP1 serial port function that is not desired may be overridden by programming the corresponding Data Direction (TRIS) register to the opposite value. If individual MSSP2 serial port functions will not be used, they may be left unmapped.

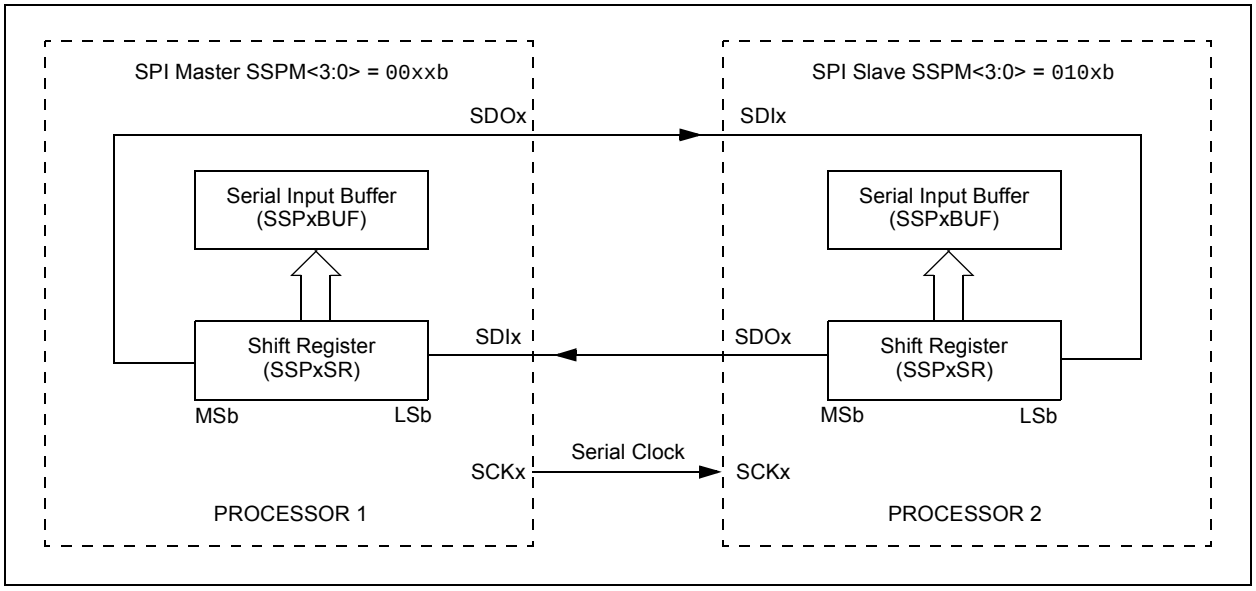
Note:	When MSSP2 is used in SPI Master mode, the SCK2 function must be configured as both an output and input in the PPS module. SCK2 must be initialized as an output pin (by writing 0x0A to one of the RPORx registers). Additionally, SCK2IN must also be mapped to the same pin, by initializing the RPINR22 register. Failure to initialize SCK2/SCK2IN as both output and input will prevent the module from receiving data on the SDI2 pin, as the module uses the SCK2IN signal to latch the received data.

19.3.5 TYPICAL CONNECTION

Figure 19-2 illustrates a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCKx signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to the same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends valid data – Slave sends dummy data
- Master sends valid data – Slave sends valid data
- Master sends dummy data – Slave sends valid data

FIGURE 19-2: SPI MASTER/SLAVE CONNECTION



19.3.9 OPERATION IN POWER-MANAGED MODES

In SPI Master mode, module clocks may be operating at a different speed than when in full-power mode. In the case of Sleep mode, all clocks are halted.

In Idle modes, a clock is provided to the peripherals. That clock can be from the primary clock source, the secondary clock (Timer1 oscillator) or the INTOSC source. See **Section 3.3 "Clock Sources and Oscillator Switching"** for additional information.

In most cases, the speed that the master clocks SPI data is not important; however, this should be evaluated for each system.

If MSSP interrupts are enabled, they can wake the controller from Sleep mode, or one of the Idle modes, when the master completes sending data. If an exit from Sleep or Idle mode is not desired, MSSP interrupts should be disabled.

If the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in any power-managed mode and data to be shifted into the SPI Transmit/Receive Shift register. When all 8 bits have been received, the MSSP interrupt flag bit will be set, and if enabled, will wake the device.

19.3.10 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

19.3.11 BUS MODE COMPATIBILITY

Table 19-1 provides the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

TABLE 19-1: SPI BUS MODES

Standard SPI Mode Terminology	Control Bits State	
	CKP	CKE
0, 0	0	1
0, 1	0	0
1, 0	1	1
1, 1	1	0

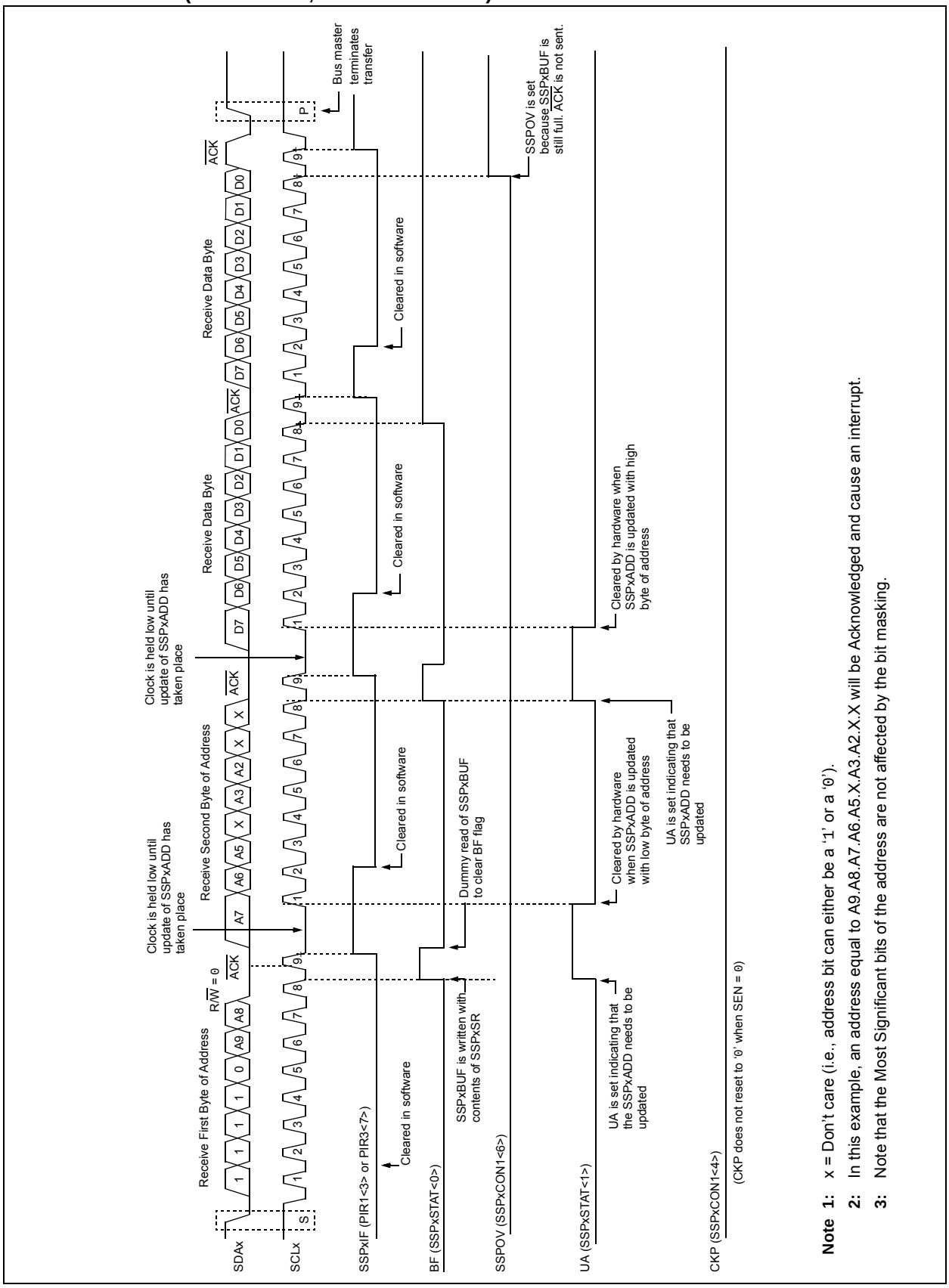
There is also an SMP bit, which controls when the data is sampled.

19.3.12 SPI CLOCK SPEED AND MODULE INTERACTIONS

Because MSSP1 and MSSP2 are independent modules, they can operate simultaneously at different data rates. Setting the SSPM<3:0> bits of the SSPxCON1 register determines the rate for the corresponding module.

An exception is when both modules use Timer2 as a time base in Master mode. In this instance, any changes to the Timer2 module's operation will affect both MSSP modules equally. If different bit rates are required for each module, the user should select one of the other three time base options for one of the modules.

FIGURE 19-11: I²C™ SLAVE MODE TIMING WITH SEN = 0 AND ADMSK<5:1> = 01001 (RECEPTION, 10-BIT ADDRESS)



Note 1: x = Don't care (i.e., address bit can either be a '1' or a '0').

2: In this example, an address equal to A9.A8.A7.A6.A5.X.A3.A2.X.X will be Acknowledged and cause an interrupt.

3: Note that the Most Significant bits of the address are not affected by the bit masking.

REGISTER 20-1: TXSTAx: TRANSMIT STATUS AND CONTROL REGISTER (ACCESS FADh/FA8h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
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 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care.
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-Bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit(1)
1 = Transmit is enabled and the TXx/CKx pin is configured as an output
0 = Transmit is disabled

bit 4 **SYNC:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission completed
Synchronous mode:
Don't care.

bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data
Can be address/data bit or a parity bit.

Note 1: SREN/CREN overrides TXEN in Sync mode.

20.2.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is displayed in Figure 20-6. The data is received on the RXx pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. This mode would typically be used in RS-232 systems.

20.2.2.1 Receiving Data

The receiver data recovery circuit initiates character reception on the falling edge of the first bit. The first bit, also known as the Start bit, is always a zero (after accounting for RXDTP setting). Following the Start bit will be the Least Significant bit of the data character being received. As each bit is received, the value will be sampled and shifted into the Receive Shift Register (RSR). After all 8 or 9 data bits (user selectable option) of the character have been shifted in, one final bit time is measured and the level sampled. This is the Stop bit, which should always be a '1' (after accounting for RXDTP setting). If the data recovery circuit samples a '0' in the Stop bit position then a framing error (FERR) is set for this character, otherwise the framing error is cleared for this character.

Once all data bits of the character and the Stop bit has been received, the data bits in the RSR will immediately be transferred to a two character First-In-First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters before software is required to service the EUSART receiver. The RSR register is not directly accessible by software. Firmware can read data from the FIFO by reading the RCREGx register. Each firmware initiated read from the RCREGx register will advance the FIFO by one character, and will clear the receive interrupt flag (RCxIF), if no additional data exists in the FIFO.

20.2.2.2 Receive Overrun Error

If the user firmware allows the FIFO to become full, and a third character is received before the firmware reads from RCREGx, a buffer overrun error condition will occur. In this case, the hardware will block the RSR contents (the third byte received) from being copied into the receive FIFO, the character will be lost and the OERR status bit in the RCSTAx register will become set. If an OERR condition is allowed to occur, firmware must clear the condition by clearing and then resetting CREN, before additional characters can be successfully received.

Note:	If the receive FIFO is overrun, no additional characters will be received until the overrun condition is cleared.

20.2.2.3 Setting Up Asynchronous Receive

To set up an Asynchronous Reception:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, RCxIE.
4. If 9-bit reception is desired, set bit, RX9.
5. Enable the reception by setting bit, CREN.
6. Flag bit, RCxIF, will be set when reception is complete and an interrupt will be generated if enable bit, RCxIE, was set.
7. Read the RCSTAx register to get the ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREGx register.
9. If any error occurred, clear the error by clearing enable bit, CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

20.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCxIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCxIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCxIE and GIE bits are set.
8. Read the RCSTAx register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREGx to determine if the device is being addressed.

EXAMPLE 25-2: CURRENT CALIBRATION ROUTINE

```
#include <p18cxxx.h>

#define COUNT 500                           //@ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define RCAL .027                           //R value is 4200000 (4.2M)
                                            //scaled so that result is in
                                            //1/100th of uA
#define ADSCALE 1023                        //for unsigned conversion 10 sig bits
#define ADREF 3.3                           //Vdd connected to A/D Vr+

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

//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
CTMUCONLbits.EDG1STAT = 0;                  // Set Edge status bits to zero
CTMUCONLbits.EDG2STAT = 0;
    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

}
```

26.5.2 EXITING FAIL-SAFE OPERATION

The Fail-Safe Clock Monitor condition is terminated by either a device Reset or by entering a power-managed mode. On Reset, the controller starts the primary clock source specified in Configuration Register 2H (with any required start-up delays that are required for the oscillator mode, such as OST or PLL timer). The INTRC oscillator provides the device clock until the primary clock source becomes ready (similar to a Two-Speed Start-up). The clock source is then switched to the primary clock (indicated by the OSTS bit in the OSCCON register becoming set). The FSCM then resumes monitoring the peripheral clock.

The primary clock source may never become ready during start-up. In this case, operation is clocked by the INTRC oscillator. The OSCCON register will remain in its Reset state until a power-managed mode is entered.

26.5.3 FSCM INTERRUPTS IN POWER-MANAGED MODES

By entering a power-managed mode, the clock multiplexer selects the clock source selected by the OSCCON register. FSCM of the power-managed clock source resumes in the power-managed mode.

If an oscillator failure occurs during power-managed operation, the subsequent events depend on whether or not the oscillator failure interrupt is enabled. If enabled (OSCFIF = 1), code execution will be clocked by the INTRC multiplexer. An automatic transition back to the failed clock source will not occur.

If the interrupt is disabled, subsequent interrupts while in Idle mode will cause the CPU to begin executing instructions while being clocked by the INTRC source.

26.5.4 POR OR WAKE-UP FROM SLEEP

The FSCM is designed to detect oscillator failure at any point after the device has exited Power-on Reset (POR) or low-power Sleep mode. When the primary device clock is either the EC or INTRC modes, monitoring can begin immediately following these events.

For HS or HSPLL modes, the situation is somewhat different. Since the oscillator may require a start-up time considerably longer than the FSCM sample clock time, a false clock failure may be detected. To prevent this, the internal oscillator block is automatically configured as the device clock and functions until the primary clock is stable (the OST and PLL timers have timed out). This is identical to Two-Speed Start-up mode. Once the primary clock is stable, the INTRC returns to its role as the FSCM source.

Note:	The same logic that prevents false oscillator failure interrupts on POR, or wake-up from Sleep, will also prevent the detection of the oscillator's failure to start at all following these events. This can be avoided by monitoring the OSTS bit and using a timing routine to determine if the oscillator is taking too long to start. Even so, no oscillator failure interrupt will be flagged.

As noted in **Section 26.4.1 "Special Considerations for Using Two-Speed Start-up"**, it is also possible to select another clock configuration and enter an alternate power-managed mode while waiting for the primary clock to become stable. When the new power-managed mode is selected, the primary clock is disabled.

26.6 Program Verification and Code Protection

For all devices in the PIC18F46J11 family of devices, the on-chip program memory space is treated as a single block. Code protection for this block is controlled by one Configuration bit, CP0. This bit inhibits external reads and writes to the program memory space. It has no direct effect in normal execution mode.

26.6.1 CONFIGURATION REGISTER PROTECTION

The Configuration registers are protected against untoward changes or reads in two ways. The primary protection is the write-once feature of the Configuration bits, which prevents reconfiguration once the bit has been programmed during a power cycle. To safeguard against unpredictable events, Configuration bit changes resulting from individual cell level disruptions (such as ESD events) will cause a parity error and trigger a device Reset. This is seen by the user as a Configuration Mismatch (CM) Reset.

The data for the Configuration registers is derived from the FCW in program memory. When the CP0 bit is set, the source data for device configuration is also protected as a consequence.

COMF **Complement f**

Syntax: COMF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $\bar{f} \rightarrow$ dest

Status Affected: N, Z

Encoding:

0001	11da	ffff	ffff

Description: The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `COMF REG, 0, 0`

Before Instruction
REG = 13h
After Instruction
REG = 13h
W = ECh

CPFSEQ **Compare f with W, Skip if f = W**

Syntax: CPFSEQ f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) = (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	001a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.

If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSEQ REG, 0
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

RRNCF **Rotate Right f (No Carry)**

Syntax: RRNCF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f<n>) → dest<n – 1>,
(f<0>) → dest<7>

Status Affected: N, Z

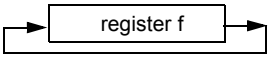
Encoding:

0100	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).

If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1:
```
RRNCF   REG, 1, 0
Before Instruction
    REG     =   1101 0111
After Instruction
    REG     =   1110 1011
```

Example 2:
```
RRNCF   REG, 0, 0
Before Instruction
    W       =   ?
    REG     =   1101 0111
After Instruction
    W       =   1110 1011
    REG     =   1101 0111
```

SETF **Set f**

Syntax: SETF f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: FFh → f

Status Affected: None

Encoding:

0110	100a	ffff	ffff

Description: The contents of the specified register are set to FFh.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example:
```
SETF    REG,1
Before Instruction
    REG     =   5Ah
After Instruction
    REG     =   FFh
```

TABLE 29-9: EXTERNAL CLOCK TIMING REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	FOSC	External CLKI Frequency(1)	DC	48	MHz	EC Oscillator mode
			4	12		ECPLL Oscillator mode
		Oscillator Frequency(1)	4	16	MHz	HS Oscillator mode
			4	12		HSPLL Oscillator mode
1	TOSC	External CLKI Period(1)	20.8	—	ns	EC Oscillator mode
			83.3	—		ECPLL Oscillator mode
		Oscillator Period(1)	62.5	250	ns	HS Oscillator mode
			83.3	250		HSPLL Oscillator mode
2	TCY	Instruction Cycle Time(1)	83.3	DC	ns	TCY = 4/FOSC, Industrial
3	TOSL, TOSH	External Clock in (OSC1) High or Low Time	10	—	ns	EC Oscillator mode
4	TOSR, TOSF	External Clock in (OSC1) Rise or Fall Time	—	7.5	ns	EC Oscillator mode

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time base period for all configurations except PLL. 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 the OSC1/CLKI pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.

TABLE 29-10: PLL CLOCK TIMING SPECIFICATIONS

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
F10	FPLLIN	PLL Input Frequency Range	4	—	12	MHz	
F11	FPLLO	PLL Output Frequency (4x FPLLIN)	16	—	48	MHz	
F12	t_{rc}	PLL Start-up Time (lock time)	—	—	2	ms	

† Data in “Typ” column is at 3.3V, 25°C, unless otherwise stated.

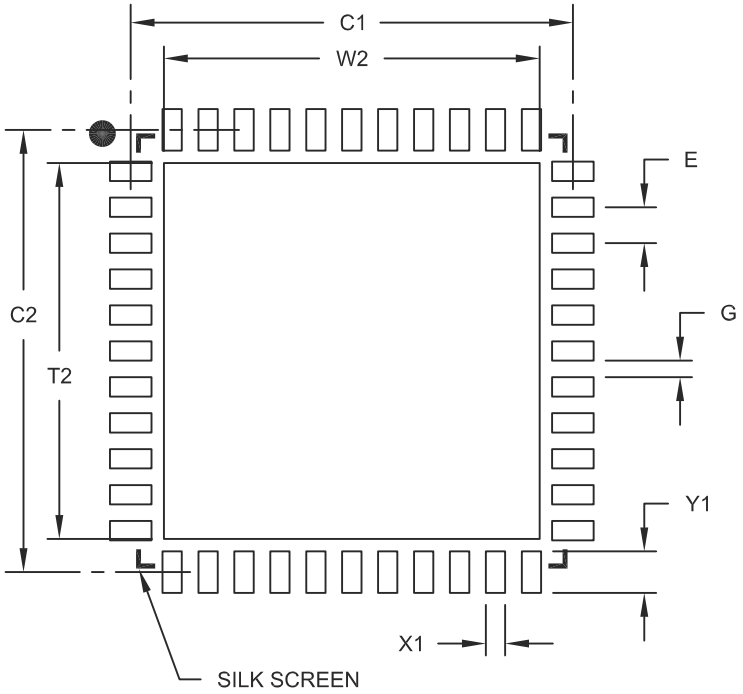
TABLE 29-11: INTERNAL RC ACCURACY (INTOSC AND INTRC SOURCES)

Param No.	Device	Min	Typ	Max	Units	Conditions	
	INTOSC Accuracy @ Freq = 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHz, 250 kHz, 125 kHz, 31 kHz(1)						
	All Devices	-1	+/-0.15	+1	%	0°C to +85°C	VDD = 2.0-3.3V
		-1	+/-0.25	+1	%	-40°C to +85°C	VDD = 2.0-3.6V, VDDCORE = 2.0-2.7V
	INTRC Accuracy @ Freq = 31 kHz(1)						
	All Devices	20.3	—	42.2	kHz	-40°C to +85°C	VDD = 2.0-3.6V, VDDCORE = 2.0-2.7V

Note 1: The accuracy specification of the 31 kHz clock is determined by which source is providing it at a given time. When INTSRC (OSCTUNE<7>) is ‘1’, use the INTOSC accuracy specification. When INTSRC is ‘0’, use the INTRC accuracy specification.

44-Lead Plastic Quad Flat, No Lead Package (ML) – 8x8 mm Body [QFN]

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		0.65 BSC	
Optional Center Pad Width	W2			6.80
Optional Center Pad Length	T2			6.80
Contact Pad Spacing	C1		8.00	
Contact Pad Spacing	C2		8.00	
Contact Pad Width (X44)	X1			0.35
Contact Pad Length (X44)	Y1			0.80
Distance Between Pads	G	0.25		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

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

Microchip Technology Drawing No. C04-2103A

NOTES: