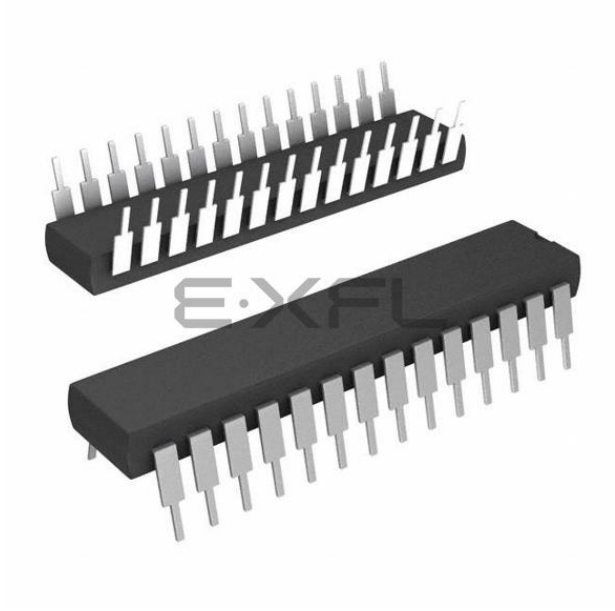


Welcome to E-XFL.COM

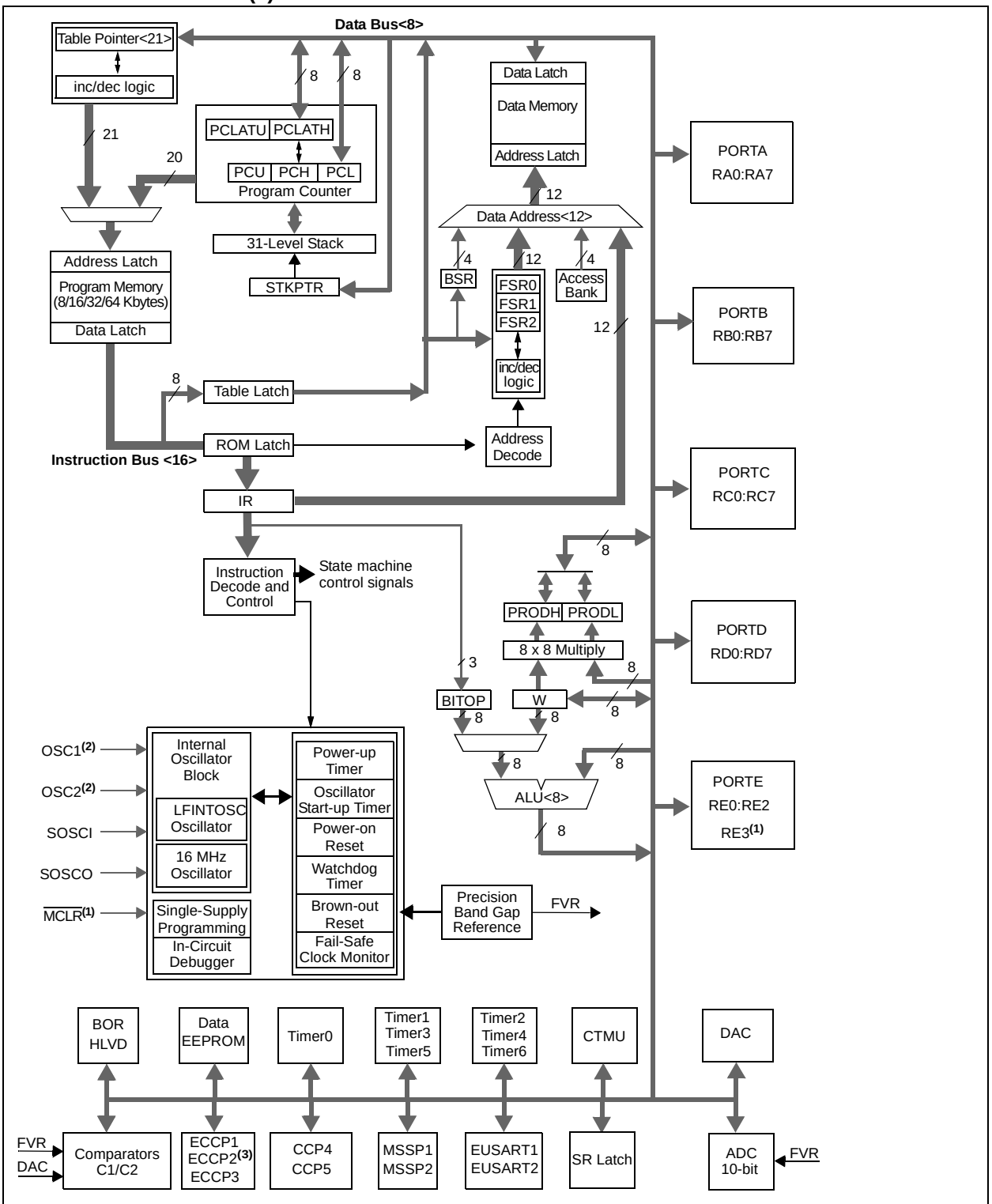
What is "Embedded - Microcontrollers"?

"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	64MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, HLVD, POR, PWM, WDT
Number of I/O	24
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 19x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.300", 7.62mm)
Supplier Device Package	28-SPDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf26k22-i-sp

FIGURE 1-1: PIC18(L)F2X/4XK22 FAMILY BLOCK DIAGRAM



Note 1: RE3 is only available when $\overline{\text{MCLR}}$ functionality is disabled.

2: OSC1/CLKIN and OSC2/CLKOUT are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Module (With Fail-Safe Clock Monitor)"** for additional information.

3: Full-Bridge operation for PIC18(L)F4XK22, half-bridge operation for PIC18(L)F2XK22.

TABLE 1-2: PIC18(L)F2XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Number		Pin Name	Pin Type	Buffer Type	Description
PDIP, SOIC	QFN, UQFN				
9	6	RA7/CLKI/OSC1			
		RA7	I/O	TTL	Digital I/O.
		CLKI	I	CMOS	External clock source input. Always associated with pin function OSC1.
		OSC1	I	ST	Oscillator crystal input or external clock source input ST buffer when configured in RC mode; CMOS otherwise.
21	18	RB0/INT0/CCP4/FLT0/SRI/$\overline{SS2}$/AN12			
		RB0	I/O	TTL	Digital I/O.
		INT0	I	ST	External interrupt 0.
		CCP4	I/O	ST	Capture 4 input/Compare 4 output/PWM 4 output.
		FLT0	I	ST	PWM Fault input for ECCP Auto-Shutdown.
		SRI	I	ST	SR latch input.
		$\overline{SS2}$	I	TTL	SPI slave select input (MSSP).
		AN12	I	Analog	Analog input 12.
22	19	RB1/INT1/P1C/SCK2/SCL2/C12IN3-/AN10			
		RB1	I/O	TTL	Digital I/O.
		INT1	I	ST	External interrupt 1.
		P1C	O	CMOS	Enhanced CCP1 PWM output.
		SCK2	I/O	ST	Synchronous serial clock input/output for SPI mode (MSSP).
		SCL2	I/O	ST	Synchronous serial clock input/output for I^2C mode (MSSP).
		C12IN3-	I	Analog	Comparators C1 and C2 inverting input.
		AN10	I	Analog	Analog input 10.
23	20	RB2/INT2/CTED1/P1B/SDI2/SDA2/AN8			
		RB2	I/O	TTL	Digital I/O.
		INT2	I	ST	External interrupt 2.
		CTED1	I	ST	CTMU Edge 1 input.
		P1B	O	CMOS	Enhanced CCP1 PWM output.
		SDI2	I	ST	SPI data in (MSSP).
		SDA2	I/O	ST	I^2C data I/O (MSSP).
		AN8	I	Analog	Analog input 8.
24	21	RB3/CTED2/P2A/CCP2/SDO2/C12IN2-/AN9			
		RB3	I/O	TTL	Digital I/O.
		CTED2	I	ST	CTMU Edge 2 input.
		P2A	O	CMOS	Enhanced CCP2 PWM output.
		CCP2**(2)**	I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
		SDO2	O	—	SPI data out (MSSP).
		C12IN2-	I	Analog	Comparators C1 and C2 inverting input.
		AN9	I	Analog	Analog input 9.

Legend: TTL = TTL compatible input CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power.

Note 1: Default pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are set.

2: Alternate pin assignment for P2B, T3CKI, CCP3 and CCP2 when Configuration bits PB2MX, T3CMX, CCP3MX and CCP2MX are clear.

3.5 Exiting Idle and Sleep Modes

An exit from Sleep mode or any of the Idle modes is triggered by any one of the following:

- an interrupt
- a Reset
- a Watchdog Time-out

This section discusses the triggers that cause exits from power-managed modes. The clocking subsystem actions are discussed in each of the power-managed modes (see **Section 3.2 “Run Modes”, Section 3.3 “Sleep Mode”** and **Section 3.4 “Idle Modes”**).

3.5.1 EXIT BY INTERRUPT

Any of the available interrupt sources can cause the device to exit from an Idle mode or the Sleep mode to a Run mode. To enable this functionality, an interrupt source must be enabled by setting its enable bit in one of the INTCON or PIE registers. The exit sequence is initiated when the corresponding interrupt flag bit is set.

The instruction immediately following the SLEEP instruction is executed on all exits by interrupt from Idle or Sleep modes. Code execution then branches to the interrupt vector if the GIE/GIEH bit of the INTCON register is set, otherwise code execution continues without branching (see **Section 9.0 “Interrupts”**).

A fixed delay of interval TCSD following the wake event is required when leaving Sleep and Idle modes. This delay is required for the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

3.5.2 EXIT BY WDT TIME-OUT

A WDT time-out will cause different actions depending on which power-managed mode the device is in when the time-out occurs.

If the device is not executing code (all Idle modes and Sleep mode), the time-out will result in an exit from the power-managed mode (see **Section 3.2 “Run Modes”** and **Section 3.3 “Sleep Mode”**). If the device is executing code (all Run modes), the time-out will result in a WDT Reset (see **Section 24.3 “Watchdog Timer (WDT)”**).

The WDT timer and postscaler are cleared by any one of the following:

- executing a SLEEP instruction
- executing a CLRWDT instruction
- the loss of the currently selected clock source when the Fail-Safe Clock Monitor is enabled
- modifying the IRCF bits in the OSCCON register when the internal oscillator block is the device clock source

3.5.3 EXIT BY RESET

Exiting Sleep and Idle modes by Reset causes code execution to restart at address ‘0’. See **Section 4.0 “Reset”** for more details.

The exit delay time from Reset to the start of code execution depends on both the clock sources before and after the wake-up and the type of oscillator.

3.5.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

Certain exits from power-managed modes do not invoke the OST at all. There are two cases:

- PRI_IDLE mode, where the primary clock source is not stopped and
- the primary clock source is not any of the LP, XT, HS or HSPLL modes.

In these instances, the primary clock source either does not require an oscillator start-up delay since it is already running (PRI_IDLE), or normally does not require an oscillator start-up delay (RC, EC, INTOSC, and INTOSCIO modes). However, a fixed delay of interval TCSD following the wake event is still required when leaving Sleep and Idle modes to allow the CPU to prepare for execution. Instruction execution resumes on the first clock cycle following this delay.

3.6 Selective Peripheral Module Control

Idle mode allows users to substantially reduce power consumption by stopping the CPU clock. Even so, peripheral modules still remain clocked, and thus, consume power. There may be cases where the application needs what IDLE mode does not provide: the allocation of power resources to the CPU processing with minimal power consumption from the peripherals. PIC18(L)F2X/4XK22 family devices address this requirement by allowing peripheral modules to be selectively disabled, reducing or eliminating their power consumption. This can be done with control bits in the Peripheral Module Disable (PMD) registers. These bits generically named XXXMD are located in control registers PMD0, PMD1 or PMD2.

Setting the PMD bit for a module disables all clock sources to that module, reducing its power consumption to an absolute minimum. In this state, power to the control and status registers associated with the peripheral is removed. Writes to these registers have no effect and read values are invalid. Clearing a set PMD bit restores power to the associated control and status registers, thereby setting those registers to their default values.

3.7 Register Definitions: Peripheral Module Disable

REGISTER 3-1: PMD0: PERIPHERAL MODULE DISABLE REGISTER 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
UART2MD	UART1MD	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD
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 **UART2MD:** UART2 Peripheral Module Disable Control bit
1 = Module is disabled, Clock Source is disconnected, module does not draw digital power
0 = Module is enabled, Clock Source is connected, module draws digital power

bit 6 **UART1MD:** UART1 Peripheral Module Disable Control bit
1 = Module is disabled, Clock Source is disconnected, module does not draw digital power
0 = Module is enabled, Clock Source is connected, module draws digital power

bit 5 **TMR6MD:** Timer6 Peripheral Module Disable Control bit
1 = Module is disabled, Clock Source is disconnected, module does not draw digital power
0 = Module is enabled, Clock Source is connected, module draws digital power

bit 4 **TMR5MD:** Timer5 Peripheral Module Disable Control bit
1 = Module is disabled, Clock Source is disconnected, module does not draw digital power
0 = Module is enabled, Clock Source is connected, module draws digital power

bit 3 **TMR4MD:** Timer4 Peripheral Module Disable Control bit
1 = Module is disabled, Clock Source is disconnected, module does not draw digital power
0 = Module is enabled, Clock Source is connected, module draws digital power

bit 2 **TMR3MD:** Timer3 Peripheral Module Disable Control bit
1 = Module is disabled, Clock Source is disconnected, module does not draw digital power
0 = Module is enabled, Clock Source is connected, module draws digital power

bit 1 **TMR2MD:** Timer2 Peripheral Module Disable Control bit
1 = Module is disabled, Clock Source is disconnected, module does not draw digital power
0 = Module is enabled, Clock Source is connected, module draws digital power

bit 0 **TMR1MD:** Timer1 Peripheral Module Disable Control bit
1 = Module is disabled, Clock Source is disconnected, module does not draw digital power
0 = Module is enabled, Clock Source is connected, module draws digital power

EXAMPLE 5-1: FAST REGISTER STACK CODE EXAMPLE

```
CALL SUB1, FAST    ;STATUS, WREG, BSR
                   ;SAVED IN FAST REGISTER
                   ;STACK
       •
       •

SUB1   •
       •
   RETURN, FAST    ;RESTORE VALUES SAVED
                   ;IN FAST REGISTER STACK
```

5.2.2 LOOK-UP TABLES IN PROGRAM MEMORY

There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented in two ways:

- Computed GOTO
- Table Reads

5.2.2.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in Example 5-2.

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW nn instructions. The W register is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW nn instructions that returns the value 'nn' to the calling function.

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of two (LSb = 0).

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

EXAMPLE 5-2: COMPUTED GOTO USING AN OFFSET VALUE

```
        MOVF    OFFSET, W
        CALL    TABLE
ORG     nn00h
TABLE   ADDWF   PCL
        RETLW   nnh
        RETLW   nnh
        RETLW   nnh
        .
        .
        .
```

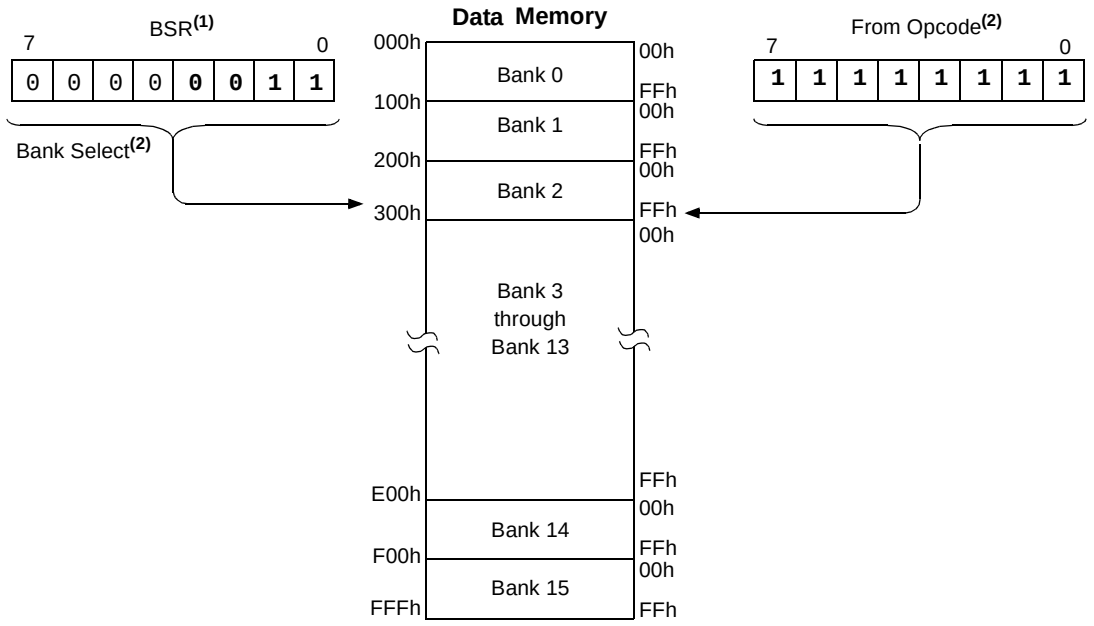
5.2.2.2 Table Reads and Table Writes

A better method of storing data in program memory allows two bytes of data to be stored in each instruction location.

Look-up table data may be stored two bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from or written to program memory. Data is transferred to or from program memory one byte at a time.

Table read and table write operations are discussed further in **Section 6.1 "Table Reads and Table Writes"**.

FIGURE 5-9: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)



Note 1: The Access RAM bit of the instruction can be used to force an override of the selected bank (BSR<3:0>) to the registers of the Access Bank.

2: The `MOVFF` instruction embeds the entire 12-bit address in the instruction.

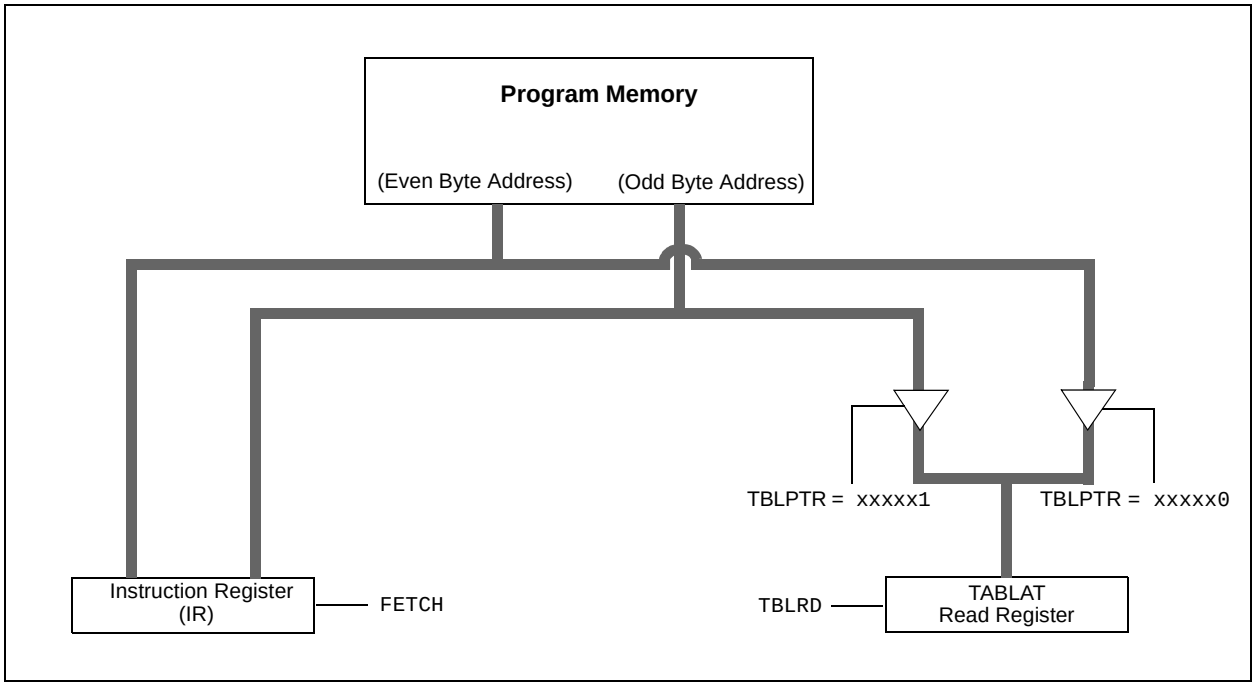
6.4 Reading the Flash Program Memory

The TBLRD instruction retrieves data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

TBLPTR points to a byte address in program space. Executing TBLRD places the byte pointed to into TABLAT. In addition, TBLPTR can be modified automatically for the next table read operation.

The internal program memory is typically organized by words. The Least Significant bit of the address selects between the high and low bytes of the word. Figure 6-4 shows the interface between the internal program memory and the TABLAT.

FIGURE 6-4: READS FROM FLASH PROGRAM MEMORY



EXAMPLE 6-1: READING A FLASH PROGRAM MEMORY WORD

```
            MOVLW     CODE_ADDR_UPPER         ; Load TBLPTR with the base
            MOVWF     TBLPTRU                 ; address of the word
            MOVLW     CODE_ADDR_HIGH
            MOVWF     TBLPTRH
            MOVLW     CODE_ADDR_LOW
            MOVWF     TBLPTRL
READ_WORD
            TBLRD*+                           ; read into TABLAT and increment
            MOVF      TABLAT, W               ; get data
            MOVWF     WORD_EVEN
            TBLRD*+                           ; read into TABLAT and increment
            MOVFW     TABLAT, W               ; get data
            MOVF      WORD_ODD
```

12.7.2.3 Comparator C1 Gate Operation

The output resulting from a Comparator 1 operation can be selected as a source for Timer1/3/5 Gate Control. The Comparator 1 output (sync_C1OUT) can be synchronized to the Timer1/3/5 clock or left asynchronous. For more information see **Section 18.8.4 "Synchronizing Comparator Output to Timer1"**.

12.7.2.4 Comparator C2 Gate Operation

The output resulting from a Comparator 2 operation can be selected as a source for Timer1/3/5 Gate Control. The Comparator 2 output (sync_C2OUT) can be synchronized to the Timer1/3/5 clock or left asynchronous. For more information see **Section 18.8.4 "Synchronizing Comparator Output to Timer1"**.

12.7.3 TIMER1/3/5 GATE TOGGLE MODE

When Timer1/3/5 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1/3/5 gate signal, as opposed to the duration of a single level pulse.

The Timer1/3/5 Gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See Figure 12-5 for timing details.

Timer1/3/5 Gate Toggle mode is enabled by setting the TxGTM bit of the TxGCON register. When the TxGTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

Note:	Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

12.7.4 TIMER1/3/5 GATE SINGLE-PULSE MODE

When Timer1/3/5 Gate Single-Pulse mode is enabled, it is possible to capture a single-pulse gate event. Timer1/3/5 Gate Single-Pulse mode is first enabled by setting the TxGSPM bit in the TxGCON register. Next, the TxGGO/$\overline{\text{DONE}}$ bit in the TxGCON register must be set. The Timer1/3/5 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the TxGGO/$\overline{\text{DONE}}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1/3/5 until the TxGGO/$\overline{\text{DONE}}$ bit is once again set in software.

Clearing the TxGSPM bit of the TxGCON register will also clear the TxGGO/$\overline{\text{DONE}}$ bit. See Figure 12-6 for timing details.

Enabling the Toggle mode and the Single-Pulse mode simultaneously will permit both sections to work together. This allows the cycle times on the Timer1/3/5 Gate source to be measured. See Figure 12-7 for timing details.

12.7.5 TIMER1/3/5 GATE VALUE STATUS

When Timer1/3/5 Gate Value Status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the TxGVAL bit in the TxGCON register. The TxGVAL bit is valid even when the Timer1/3/5 Gate is not enabled (TMRxGE bit is cleared).

12.7.6 TIMER1/3/5 GATE EVENT INTERRUPT

When Timer1/3/5 Gate Event Interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of TxGVAL occurs, the TMRxGIF flag bit in the PIR3 register will be set. If the TMRxGIE bit in the PIE3 register is set, then an interrupt will be recognized.

The TMRxGIF flag bit operates even when the Timer1/3/5 Gate is not enabled (TMRxGE bit is cleared).

For more information on selecting high or low priority status for the Timer1/3/5 Gate Event Interrupt see **Section 9.0 "Interrupts"**.

13.0 TIMER2/4/6 MODULE

There are three identical 8-bit Timer2-type modules available. To maintain pre-existing naming conventions, the Timers are called Timer2, Timer4 and Timer6 (also Timer2/4/6).

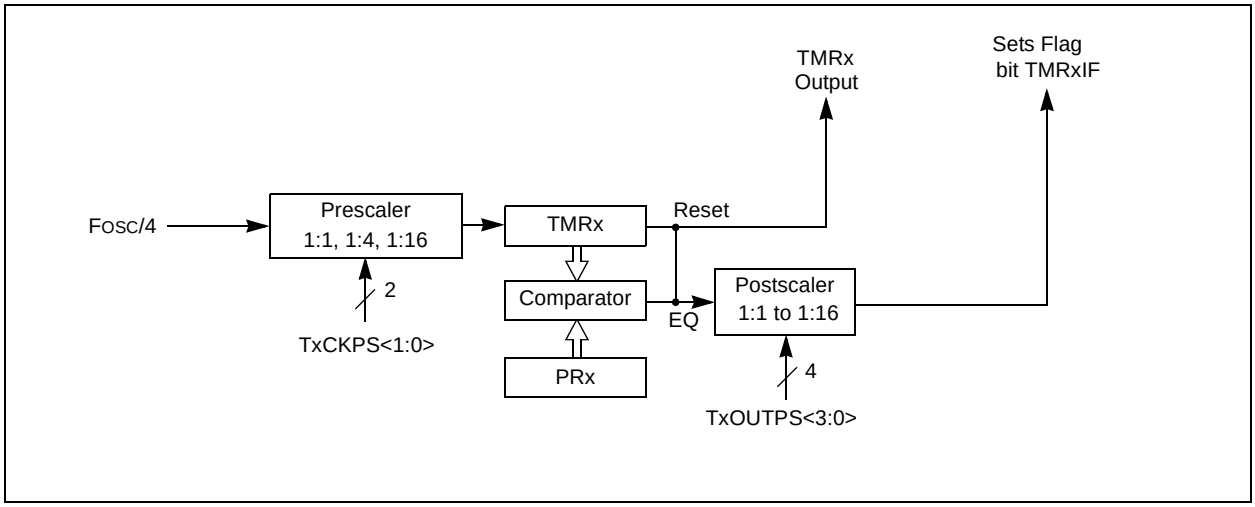
Note:	The 'x' variable used in this section is used to designate Timer2, Timer4, or Timer6. For example, TxCON references T2CON, T4CON, or T6CON. PRx references PR2, PR4, or PR6.

The Timer2/4/6 module incorporates the following features:

- 8-bit Timer and Period registers (TMRx and PRx, respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMRx match with PRx, respectively
- Optional use as the shift clock for the MSSPx modules (Timer2 only)

See Figure 13-1 for a block diagram of Timer2/4/6.

FIGURE 13-1: TIMER2/4/6 BLOCK DIAGRAM



The I^2C interface supports the following modes and features:

- Master mode
- Slave mode
- Byte NACKing (Slave mode)
- Limited Multi-master support
- 7-bit and 10-bit addressing
- Start and Stop interrupts
- Interrupt masking
- Clock stretching
- Bus collision detection
- General call address matching
- Address masking
- Address Hold and Data Hold modes
- Selectable SDAx hold times

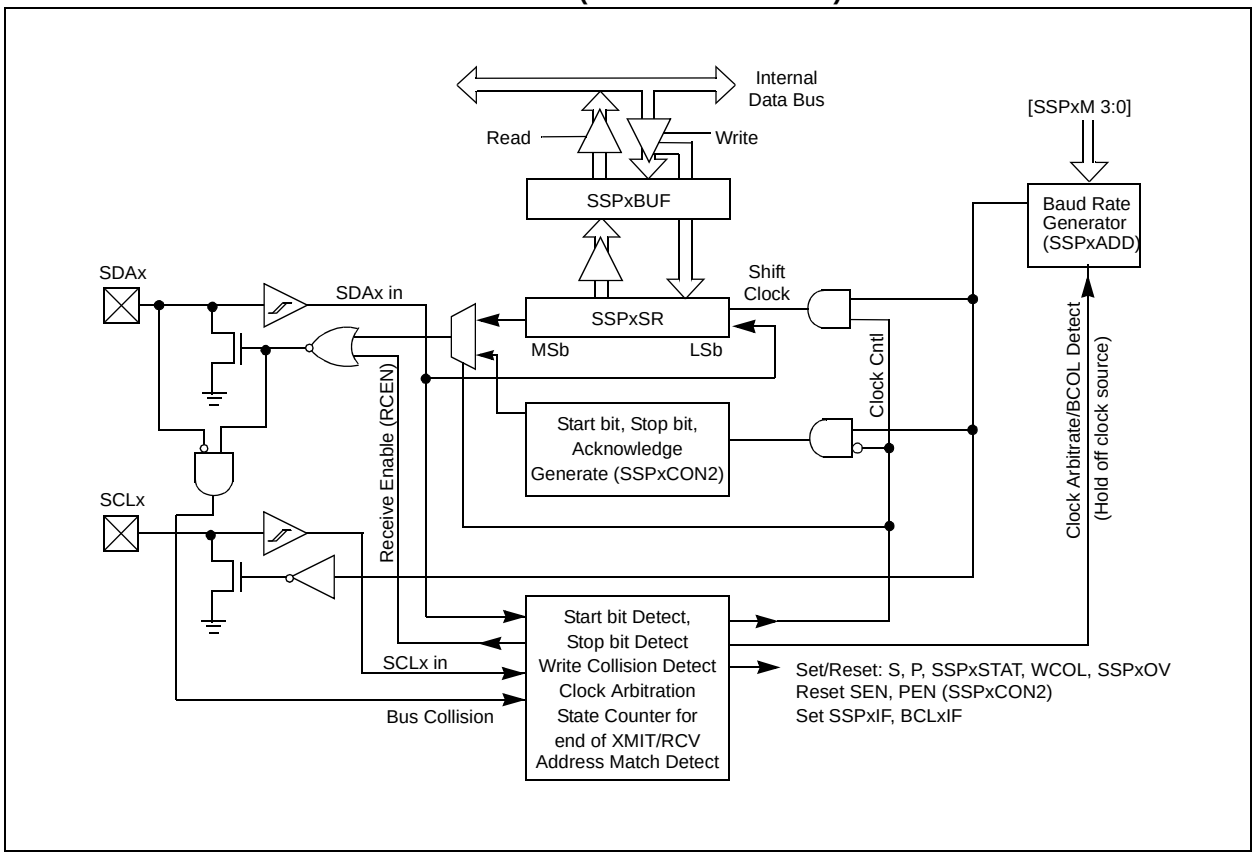
Figure 15-2 is a block diagram of the I^2C interface module in Master mode. Figure 15-3 is a diagram of the I^2C interface module in Slave mode.

The PIC18(L)F2X/4XK22 has two MSSP modules, MSSP1 and MSSP2, each module operating independently from the other.

Note 1: In devices with more than one MSSP module, it is very important to pay close attention to SSPxCONx register names. SSP1CON1 and SSP1CON2 registers control different operational aspects of the same module, while SSP1CON1 and SSP2CON1 control the same features for two different modules.

2: Throughout this section, generic references to an MSSP module in any of its operating modes may be interpreted as being equally applicable to MSSP1 or MSSP2. Register names, module I/O signals, and bit names may use the generic designator 'x' to indicate the use of a numeral to distinguish a particular module when required.

FIGURE 15-2: MSSPx BLOCK DIAGRAM (I^2C MASTER MODE)



15.5.3 SLAVE TRANSMISSION

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{W}$ bit of the SSPxSTAT register is set. The received address is loaded into the SSPxBUF register, and an $\overline{ACK}$ pulse is sent by the slave on the ninth bit.

Following the $\overline{ACK}$, slave hardware clears the CKP bit and the SCLx pin is held low (see **Section 15.5.6 "Clock Stretching"** for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data.

The transmit data must be loaded into the SSPxBUF register which also loads the SSPxSR register. Then the SCLx pin should be released by setting the CKP bit of the SSPxCON1 register. The eight data bits are shifted out on the falling edge of the SCLx input. This ensures that the SDAx signal is valid during the SCLx high time.

The $\overline{ACK}$ pulse from the master-receiver is latched on the rising edge of the ninth SCLx input pulse. This $\overline{ACK}$ value is copied to the ACKSTAT bit of the SSPxCON2 register. If ACKSTAT is set (not $\overline{ACK}$), then the data transfer is complete. In this case, when the not $\overline{ACK}$ is latched by the slave, the slave goes Idle and waits for another occurrence of the Start bit. If the SDAx line was low ($\overline{ACK}$), the next transmit data must be loaded into the SSPxBUF register. Again, the SCLx pin must be released by setting bit CKP.

An MSSPx interrupt is generated for each data transfer byte. The SSPxIF bit must be cleared by software and the SSPxSTAT register is used to determine the status of the byte. The SSPxIF bit is set on the falling edge of the ninth clock pulse.

15.5.3.1 Slave Mode Bus Collision

A slave receives a Read request and begins shifting data out on the SDAx line. If a bus collision is detected and the SBCDE bit of the SSPxCON3 register is set, the BCLxIF bit of the PIRx register is set. Once a bus collision is detected, the slave goes Idle and waits to be addressed again. User software can use the BCLxIF bit to handle a slave bus collision.

15.5.3.2 7-bit Transmission

A master device can transmit a read request to a slave, and then clock data out of the slave. The list below outlines what software for a slave will need to do to accomplish a standard transmission. Figure 15-18 can be used as a reference to this list.

1. Master sends a Start condition on SDAx and SCLx.
2. S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Matching address with R/$\overline{W}$ bit set is received by the slave setting SSPxIF bit.
4. Slave hardware generates an $\overline{ACK}$ and sets SSPxIF.
5. SSPxIF bit is cleared by user.
6. Software reads the received address from SSPxBUF, clearing BF.
7. R/$\overline{W}$ is set so CKP was automatically cleared after the $\overline{ACK}$.
8. The slave software loads the transmit data into SSPxBUF.
9. CKP bit is set releasing SCLx, allowing the master to clock the data out of the slave.
10. SSPxIF is set after the $\overline{ACK}$ response from the master is loaded into the ACKSTAT register.
11. SSPxIF bit is cleared.
12. The slave software checks the ACKSTAT bit to see if the master wants to clock out more data.

Note 1: If the master $\overline{ACK}$s the clock will be stretched.

2: ACKSTAT is the only bit updated on the rising edge of SCLx (9th) rather than the falling.

13. Steps 9-13 are repeated for each transmitted byte.
14. If the master sends a not $\overline{ACK}$; the clock is not held, but SSPxIF is still set.
15. The master sends a Restart condition or a Stop.
16. The slave is no longer addressed.

REGISTER 15-3: SSPxCON1: SSPx CONTROL REGISTER 1 (CONTINUED)

bit 3-0 **SSPxM<3:0>:** Synchronous Serial Port Mode Select bits
0000 = SPI Master mode, clock = FOSC/4
0001 = SPI Master mode, clock = FOSC/16
0010 = SPI Master mode, clock = FOSC/64
0011 = SPI Master mode, clock = TMR2 output/2
0100 = SPI Slave mode, clock = SCKx pin, $\overline{SSx}$ pin control enabled
0101 = SPI Slave mode, clock = SCKx pin, $\overline{SSx}$ pin control disabled, $\overline{SSx}$ can be used as I/O pin
0110 = I^2C Slave mode, 7-bit address
0111 = I^2C Slave mode, 10-bit address
1000 = I^2C Master mode, clock = FOSC / (4 * (SSPxADD+1))**(4)**
1001 = Reserved
1010 = SPI Master mode, clock = FOSC/(4 * (SSPxADD+1))
1011 = I^2C firmware controlled Master mode (slave idle)
1100 = Reserved
1101 = Reserved
1110 = I^2C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1111 = I^2C Slave mode, 10-bit address with Start and Stop bit interrupts enabled

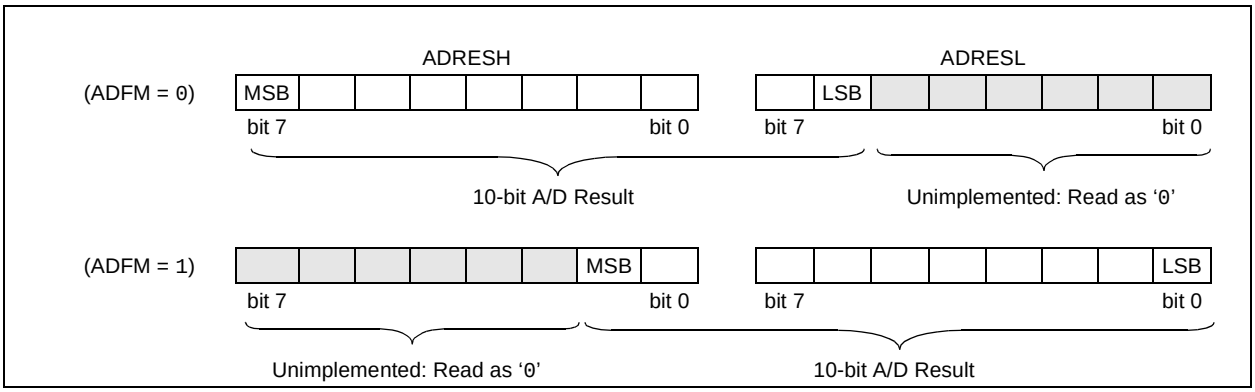
Note 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register.
2: When enabled, these pins must be properly configured as input or output.
3: When enabled, the SDAx and SCLx pins must be configured as inputs.
4: SSPxADD values of 0, 1 or 2 are not supported for I^2C mode.

17.1.7 RESULT FORMATTING

The 10-bit A/D conversion result can be supplied in two formats, left justified or right justified. The ADFM bit of the ADCON2 register controls the output format.

Figure 17-2 shows the two output formats.

FIGURE 17-2: 10-BIT A/D CONVERSION RESULT FORMAT



24.5.2 DATA EEPROM CODE PROTECTION

The entire data EEPROM is protected from external reads and writes by two bits: CPD and WRTD. CPD inhibits external reads and writes of data EEPROM. WRTD inhibits internal and external writes to data EEPROM. The CPU can always read data EEPROM under normal operation, regardless of the protection bit settings.

24.5.3 CONFIGURATION REGISTER PROTECTION

The Configuration registers can be write-protected. The WRTC bit controls protection of the Configuration registers. In Normal execution mode, the WRTC bit is readable only. WRTC can only be written via ICSP or an external programmer.

24.6 ID Locations

Eight memory locations (200000h-200007h) are designated as ID locations, where the user can store checksum or other code identification numbers. These locations are both readable and writable during normal execution through the TBLRD and TBLWT instructions or during program/verify. The ID locations can be read when the device is code-protected.

24.7 In-Circuit Serial Programming

PIC18(L)F2X/4XK22 devices can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data and three other lines for power, ground and the programming voltage. This allows customers to manufacture boards with unprogrammed devices and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

24.8 In-Circuit Debugger

When the DEBUG Configuration bit is programmed to a '0', the In-Circuit Debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® IDE. When the microcontroller has this feature enabled, some resources are not available for general use. Table 24-6 shows which resources are required by the background debugger.

TABLE 24-6: DEBUGGER RESOURCES

I/O pins:	RB6, RB7

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to the following pins:

- $\overline{MCLR}$/VPP/RE3
- VDD
- VSS
- RB7
- RB6

This will interface to the In-Circuit Debugger module available from Microchip or one of the third party development tool companies.

24.9 Single-Supply ICSP Programming

The LVP Configuration bit enables Single-Supply ICSP Programming (formerly known as Low-Voltage ICSP Programming or LVP). When Single-Supply Programming is enabled, the microcontroller can be programmed without requiring high voltage being applied to the $\overline{MCLR}$/VPP/RE3 pin. See "*PIC18(L)F2XK22/4XK22 Flash Memory Programming*" (DS41398) for more details about low voltage programming.

Note 1: High-voltage programming is always available, regardless of the state of the LVP bit, by applying VIHH to the $\overline{MCLR}$ pin.

2: By default, Single-Supply ICSP is enabled in unprogrammed devices (as supplied from Microchip) and erased devices.

3: While in Low-Voltage ICSP mode, $\overline{MCLR}$ is always enabled, regardless of the MCLRE bit, and the RE3 pin can no longer be used as a general purpose input.

The LVP bit may be set or cleared only when using standard high-voltage programming (VIHH applied to the $\overline{MCLR}$/VPP/RE3 pin). Once LVP has been disabled, only the standard high-voltage programming is available and must be used to program the device.

Memory that is not code-protected can be erased using either a block erase, or erased row by row, then written at any specified VDD. If code-protected memory is to be erased, a block erase is required.

25.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F2X/4XK22 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features of the extended instruction set are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers, or use them for indexed addressing. Two of the instructions, ADDFSR and SUBFSR, each have an additional special instantiation for using FSR2. These versions (ADDULNK and SUBULNK) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 25-3. Detailed descriptions are provided in **Section 25.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 25-1 apply to both the standard and extended PIC18 instruction sets.

Note:	The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

25.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets (“[ ]”). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 25.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”**.

Note:	In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces (“{ }”).

TABLE 25-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected
				MSb			LSb	
ADDFSR	f, k	Add literal to FSR	1	1110	1000	ffkk	kkkk	None
ADDULNK	k	Add literal to FSR2 and return	2	1110	1000	11kk	kkkk	None
CALLW		Call subroutine using WREG	2	0000	0000	0001	0100	None
MOVSF	z_s, f_d	Move z_s (source) to 1st word	2	1110	1011	0zzz	zzzz	None
		f_d (destination) 2nd word		1111	ffff	ffff	ffff	
MOVSS	z_s, z_d	Move z_s (source) to 1st word	2	1110	1011	1zzz	zzzz	None
		z_d (destination) 2nd word		1111	xxxx	xzzz	zzzz	
PUSHL	k	Store literal at FSR2, decrement FSR2	1	1110	1010	kkkk	kkkk	None
SUBFSR	f, k	Subtract literal from FSR	1	1110	1001	ffkk	kkkk	None
SUBULNK	k	Subtract literal from FSR2 and return	2	1110	1001	11kk	kkkk	None

25.2.2 EXTENDED INSTRUCTION SET

ADDFSR **Add Literal to FSR**

Syntax: ADDFSR f, k

Operands: $0 \le k \le 63$
$f \in [0, 1, 2]$

Operation: FSR(f) + k → FSR(f)

Status Affected: None

Encoding:

1110	1000	ffkk	kkkk

Description: The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to FSR

Example: `ADDFSR 2, 23h`

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 0422h

ADDULNK **Add Literal to FSR2 and Return**

Syntax: ADDULNK k

Operands: $0 \le k \le 63$

Operation: FSR2 + k → FSR2,
(TOS) → PC

Status Affected: None

Encoding:

1110	1000	11kk	kkkk

Description: The 6-bit literal 'k' is added to the contents of FSR2. A RETURN is then executed by loading the PC with the TOS.
The instruction takes two cycles to execute; a NOP is performed during the second cycle.
This may be thought of as a special case of the ADDFSR instruction, where f = 3 (binary '11'); it operates only on FSR2.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to FSR
No Operation	No Operation	No Operation	No Operation

Example: `ADDULNK 23h`

Before Instruction
FSR2 = 03FFh
PC = 0100h
After Instruction
FSR2 = 0422h
PC = (TOS)

Note: All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction syntax then becomes: {label} instruction argument(s).

27.4 DC Characteristics: RC Idle Supply Current, PIC18(L)F2X/4XK22

PIC18LF2X/4XK22		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C					
PIC18F2X/4XK22		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +125°C					
Param No.	**Device Characteristics**	**Typ**	**Max**	**Units**	**Conditions**		
D045	Supply Current (IDD)(1),(2)	0.5	18	µA	-40°C	VDD = 1.8V	FOSC = 31 kHz (**RC_IDLE** mode, LFINTOSC source)
		0.6	18	µA	+25°C		
		0.7	—	µA	+60°C		
		0.75	20	µA	+85°C		
		2.3	22	µA	+125°C		
D046		1.1	20	µA	-40°C	VDD = 3.0V	
		1.2	20	µA	+25°C		
		1.3	—	µA	+60°C		
		1.4	22	µA	+85°C		
		3.2	25	µA	+125°C		
D047		17	30	µA	-40°C	VDD = 2.3V	FOSC = 31 kHz (**RC_IDLE** mode, LFINTOSC source)
		13	30	µA	+25°C		
		14	30	µA	+85°C		
		15	45	µA	+125°C		
D048		19	35	µA	-40°C	VDD = 3.0V	
		15	35	µA	+25°C		
		16	35	µA	+85°C		
		17	50	µA	+125°C		
D049		21	40	µA	-40°C	VDD = 5.0V	
		15	40	µA	+25°C		
		16	40	µA	+85°C		
		18	60	µA	+125°C		
D050		0.11	0.20	mA	-40°C to +125°C	VDD = 1.8V	FOSC = 500 kHz (**RC_IDLE** mode, MFINTOSC source)
D051		0.12	0.25	mA	-40°C to +125°C	VDD = 3.0V	
D052		0.14	0.21	mA	-40°C to +125°C	VDD = 2.3V	FOSC = 500 kHz (**RC_IDLE** mode, MFINTOSC source)
D053		0.15	0.25	mA	-40°C to +125°C	VDD = 3.0V	
D054		0.20	0.31	mA	-40°C to +125°C	VDD = 5.0V	

Note 1: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.

Test condition: All Peripheral Module Control bits in PMD0, PMD1 and PMD2 set to '1'.

2: The test conditions for all IDD measurements in active operation mode are:

All I/O pins set as outputs driven to Vss;

$\overline{MCLR}$ = VDD;

OSC1 = external square wave, from rail-to-rail (PRI_RUN and PRI_IDLE only).

27.5 DC Characteristics: Primary Run Supply Current, PIC18(L)F2X/4XK22

PIC18LF2X/4XK22		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C					
PIC18F2X/4XK22		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +125°C					
Param No.	**Device Characteristics**	**Typ**	**Max**	**Units**	**Conditions**		
D070	Supply Current (IDD)[1],[2]	0.11	0.20	mA	-40°C to +125°C	VDD = 1.8V	FOSC = 1 MHz (**PRI_RUN** mode, ECM source)
D071		0.17	0.25	mA	-40°C to +125°C	VDD = 3.0V	
D072		0.15	0.25	mA	-40°C to +125°C	VDD = 2.3V	FOSC = 1 MHz (**PRI_RUN** mode, ECM source)
D073		0.20	0.30	mA	-40°C to +125°C	VDD = 3.0V	
D074		0.25	0.35	mA	-40°C to +125°C	VDD = 5.0V	
D075		1.45	2.0	mA	-40°C to +125°C	VDD = 1.8V	FOSC = 20 MHz (**PRI_RUN** mode, ECH source)
D076		2.60	3.5	mA	-40°C to +125°C	VDD = 3.0V	
D077		1.95	2.5	mA	-40°C to +125°C	VDD = 2.3V	FOSC = 20 MHz (**PRI_RUN** mode, ECH source)
D078		2.65	3.5	mA	-40°C to +125°C	VDD = 3.0V	
D079		2.95	4.5	mA	-40°C to +125°C	VDD = 5.0V	
D080		7.5	10	mA	-40°C to +125°C	VDD = 3.0V	FOSC = 64 MHz (**PRI_RUN**, ECH oscillator)
D081		7.5	10	mA	-40°C to +125°C	VDD = 3.0V	FOSC = 64 MHz (**PRI_RUN** mode, ECH source)
D082		8.5	11.5	mA	-40°C to +125°C	VDD = 5.0V	
D083		1.0	1.5	mA	-40°C to +125°C	VDD = 1.8V	FOSC = 4 MHz 16 MHz Internal (**PRI_RUN** mode, ECM + PLL source)
D084		1.8	3.0	mA	-40°C to +125°C	VDD = 3.0V	
D085		1.4	2.0	mA	-40°C to +125°C	VDD = 2.3V	FOSC = 4 MHz 16 MHz Internal (**PRI_RUN** mode, ECM + PLL source)
D086		1.85	2.5	mA	-40°C to +125°C	VDD = 3.0V	
D087		2.1	3.0	mA	-40°C to +125°C	VDD = 5.0V	
D088		6.35	9.0	mA	-40°C to +125°C	VDD = 3.0V	FOSC = 16 MHz 64 MHz Internal (**PRI_RUN** mode, ECH + PLL source)
D089		6.35	9.0	mA	-40°C to +125°C	VDD = 3.0V	FOSC = 16 MHz 64 MHz Internal (**PRI_RUN** mode, ECH + PLL source)
D090		7.0	10	mA	-40°C to +125°C	VDD = 5.0V	

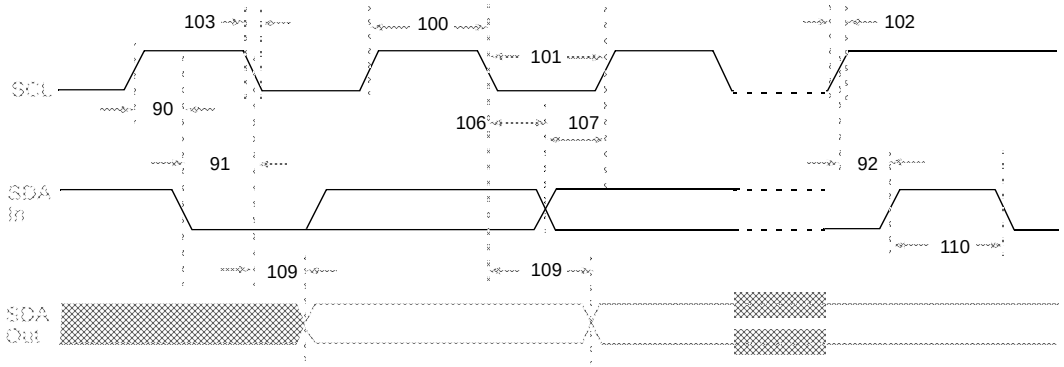
Note 1: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
Test condition: All Peripheral Module Control bits in PMD0, PMD1 and PMD2 set to '1'.

2: The test conditions for all IDD measurements in active operation mode are:
All I/O pins set as outputs driven to Vss;
MCLR = VDD;
OSC1 = external square wave, from rail-to-rail (PRI_RUN and PRI_IDLE only).

TABLE 27-15: I²C BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	Start Condition Setup Time	100 kHz mode	4700	—	ns	Only relevant for Repeated Start condition
			400 kHz mode	600	—		
91	THD:STA	Start Condition Hold Time	100 kHz mode	4000	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	600	—		
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	4700	—	ns	
			400 kHz mode	600	—		
93	THD:STO	Stop Condition Hold Time	100 kHz mode	4000	—	ns	
			400 kHz mode	600	—		

FIGURE 27-18: I²C BUS DATA TIMING



Note: Refer to Figure 27-6 for load conditions.

FIGURE 28-46: PIC18F2X/4XK22 TYPICAL IDD: RC_IDLE HF-INTOSC with PLL

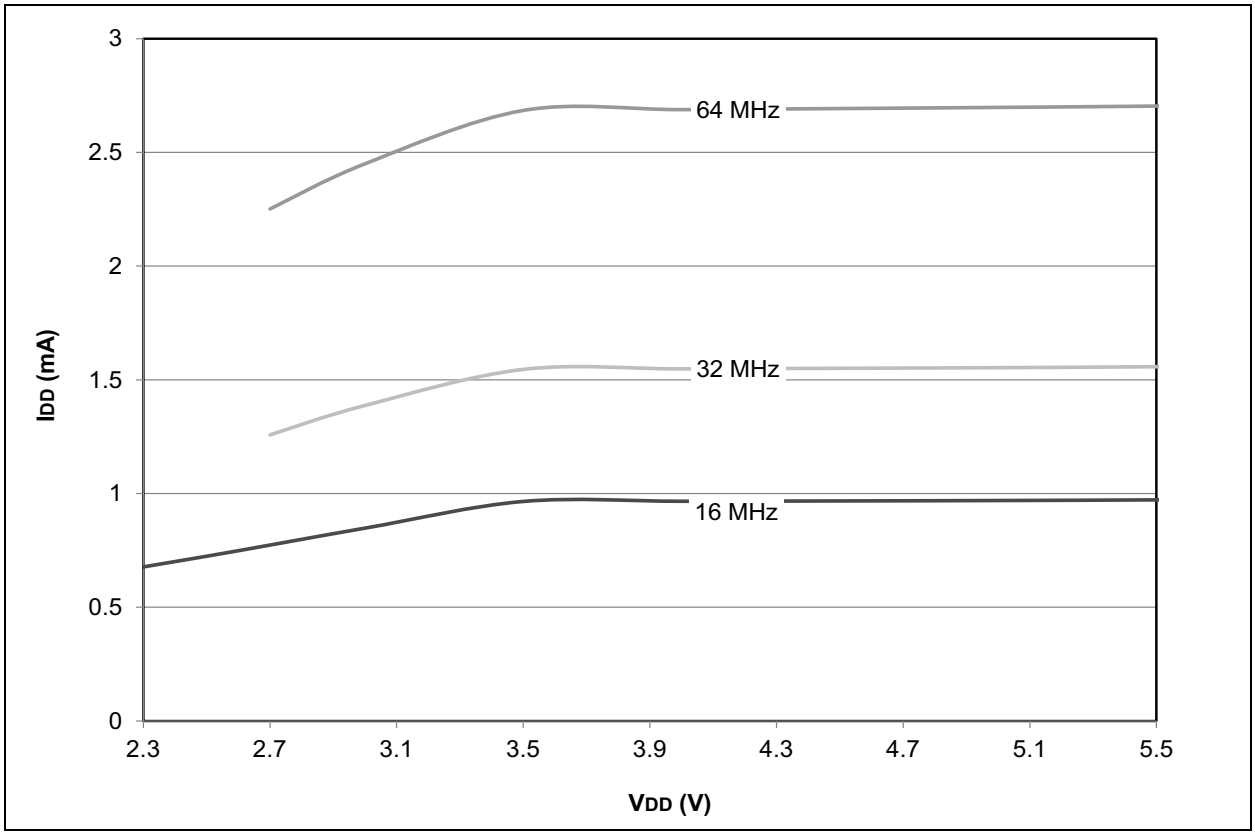


FIGURE 28-47: PIC18F2X/4XK22 MAXIMUM IDD: RC_IDLE HF-INTOSC with PLL