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, USB
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	14
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 11x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SSOP (0.209", 5.30mm Width)
Supplier Device Package	20-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f14k50t-i-ss

2.3.1 PRIMARY EXTERNAL OSCILLATOR SHUTDOWN

The Primary External Oscillator can be enabled or disabled via software. To enable software control of the Primary External Oscillator, the PCLKEN bit of the CONFIG1H Configuration register must be set. With the PCLKEN bit set, the Primary External Oscillator is controlled by the PRI_SD bit of the OSCCON2 register. The Primary External Oscillator will be enabled when the PRI_SD bit is set, and disabled when the PRI_SD bit is clear.

Note:	The Primary External Oscillator cannot be shut down when it is selected as the System Clock. To shut down the oscillator, the system clock source must be either the Secondary Oscillator or the Internal Oscillator.

2.3.2 LP, XT AND HS OSCILLATOR MODES

The LP, XT and HS modes support the use of quartz crystal resonators or ceramic resonators connected to OSC1 and OSC2 (Figure 2-2). The mode selects a low, medium or high gain setting of the internal inverter-amplifier to support various resonator types and speed.

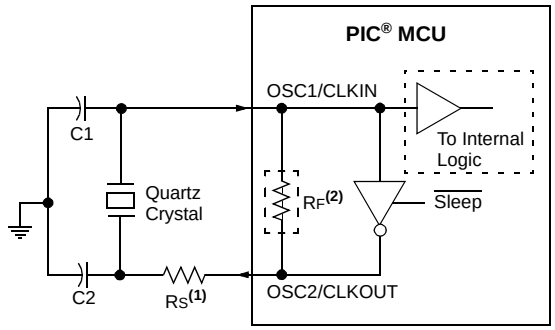
LP Oscillator mode selects the lowest gain setting of the internal inverter-amplifier. LP mode current consumption is the least of the three modes. This mode is best suited to drive resonators with a low drive level specification, for example, tuning fork type crystals.

XT Oscillator mode selects the intermediate gain setting of the internal inverter-amplifier. XT mode current consumption is the medium of the three modes. This mode is best suited to drive resonators with a medium drive level specification.

HS Oscillator mode selects the highest gain setting of the internal inverter-amplifier. HS mode current consumption is the highest of the three modes. This mode is best suited for resonators that require a high drive setting.

Figure 2-2 and Figure 2-3 show typical circuits for quartz crystal and ceramic resonators, respectively.

FIGURE 2-2: QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)



Note 1: A series resistor (RS) may be required for quartz crystals with low drive level.

2: The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

Note 1:	Quartz crystal characteristics vary according to type, package and manufacturer. The user should consult the manufacturer data sheets for specifications and recommended application.
2:	Always verify oscillator performance over the VDD and temperature range that is expected for the application.
3:	For oscillator design assistance, reference the following Microchip Applications Notes: • AN826, "*Crystal Oscillator Basics and Crystal Selection for rfPIC® and PIC® Devices*" (DS00826) • AN849, "*Basic PIC® Oscillator Design*" (DS00849) • AN943, "*Practical PIC® Oscillator Analysis and Design*" (DS00943) • AN949, "*Making Your Oscillator Work*" (DS00949)

2.12 Two-Speed Start-up Mode

Two-Speed Start-Up mode provides additional power savings by minimizing the latency between external Oscillator Start-up Timer (OST) and code execution. In applications that make heavy use of the Sleep mode, Two-Speed Start-up will remove the OST period, which can reduce the overall power consumption of the device.

Two-Speed Start-Up mode is enabled by setting the IESO bit of the CONFIG1H Configuration register. With Two-Speed Start-up enabled, the device will execute instructions using the internal oscillator during the Primary External Oscillator OST period.

When the system clock is set to the Primary External Oscillator and the oscillator is configured for LP, XT or HS modes, the device will not execute code during the OST period. The OST will suspend program execution until 1024 oscillations are counted. Two-Speed Start-Up mode minimizes the delay in code execution by operating from the internal oscillator while the OST is active. The system clock will switch back to the Primary External Oscillator after the OST period has expired.

Two-speed Start-up will become active after:

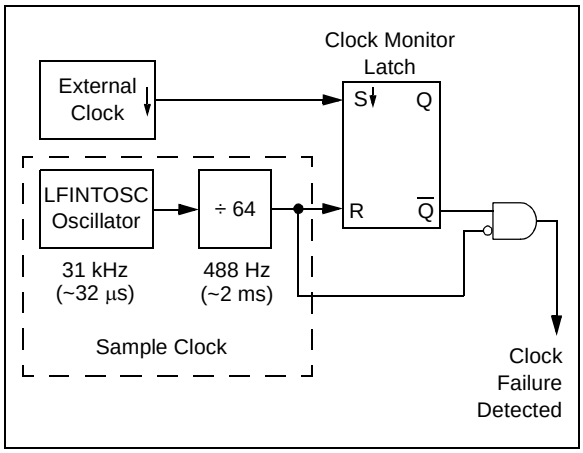
- Power-on Reset (POR)
- Power-up Timer (PWRT), if enabled
- Wake-up from Sleep

The OSTS bit of the OSCCON register reports which oscillator the device is currently using for operation. The device is running from the oscillator defined by the FOSC bits of the CONFIG1H Configuration register when the OSTS bit is set. The device is running from the internal oscillator when the OSTS bit is clear.

2.13 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the device to continue operating should the external oscillator fail. The FSCM can detect oscillator failure any time after the Oscillator Start-up Timer (OST) has expired. The FSCM is enabled by setting the FCMEN bit in the CONFIG1H Configuration register. The FSCM is applicable to all external oscillator modes (LP, XT, HS, EC and RC).

FIGURE 2-6: FSCM BLOCK DIAGRAM



2.13.1 FAIL-SAFE DETECTION

The FSCM module detects a failed oscillator by comparing the external oscillator to the FSCM sample clock. The sample clock is generated by dividing the LFINTOSC by 64. See Figure 2-6. Inside the fail detector block is a latch. The external clock sets the latch on each falling edge of the external clock. The sample clock clears the latch on each rising edge of the sample clock. A failure is detected when an entire half-cycle of the sample clock elapses before the primary clock goes low.

2.13.2 FAIL-SAFE OPERATION

When the external clock fails, the FSCM switches the device clock to an internal clock source and sets the bit flag OSCFIF of the PIR2 register. The OSCFIF flag will generate an interrupt if the OSCFIE bit of the PIE2 register is also set. The device firmware can then take steps to mitigate the problems that may arise from a failed clock. The system clock will continue to be sourced from the internal clock source until the device firmware successfully restarts the external oscillator and switches back to external operation. An automatic transition back to the failed clock source will not occur.

The internal clock source chosen by the FSCM is determined by the IRCF<2:0> bits of the OSCCON register. This allows the internal oscillator to be configured before a failure occurs.

EXAMPLE 4-3: WRITING TO FLASH PROGRAM MEMORY (CONTINUED)

```
                DECFSZ  COUNTER                 ; loop until holding registers are full
                BRA     WRITE_WORD_TO_HREGS
PROGRAM_MEMORY
                BSF     EECON1, EEPGD           ; point to Flash program memory
                BCF     EECON1, CFGS            ; access Flash program memory
                BSF     EECON1, WREN            ; enable write to memory
                BCF     INTCON, GIE             ; disable interrupts
                MOVLW   55h
    Required    MOVWF   EECON2                  ; write 55h
    Sequence    MOVLW   0AAh
                MOVWF   EECON2                  ; write 0AAh
                BSF     EECON1, WR              ; start program (CPU stall)
                DCFSZ   COUNTER2                ; repeat for remaining write blocks
                BRA     WRITE_BYTE_TO_HREGS     ;
                BSF     INTCON, GIE             ; re-enable interrupts
                BCF     EECON1, WREN            ; disable write to memory
```

4.5.2 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

4.5.3 UNEXPECTED TERMINATION OF WRITE OPERATION

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed should be verified and reprogrammed if needed. If the write operation is interrupted by a $\overline{\text{MCLR}}$ Reset or a WDT Time-out Reset during normal operation, the WRERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed.

4.5.4 PROTECTION AGAINST SPURIOUS WRITES

To protect against spurious writes to Flash program memory, the write initiate sequence must also be followed. See **Section 24.0 "Special Features of the CPU"** for more detail.

4.6 Flash Program Operation During Code Protection

See **Section 24.3 "Program Verification and Code Protection"** for details on code protection of Flash program memory.

TABLE 4-3: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					275
TBPLTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								275
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								275
TABLAT	Program Memory Table Latch								275
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	275
EECON2	EEPROM Control Register 2 (not a physical register)								277
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	277
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	278
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	278
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	278

Legend: — = unimplemented, read as '0'. Shaded cells are not used during Flash/EEPROM access.

TABLE 9-14: PORTC I/O SUMMARY (CONTINUED)

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RC5/CCP1/P1A/ T0CKI	RC5	0	O	DIG	LATC<5> data output.
		1	I	ST	PORTC<5> data input.
	CCP1	0	O	DIG	ECCP1 compare or PWM output; takes priority over port data.
		1	I	ST	ECCP1 capture input.
	P1A	0	0	DIG	ECCP1 Enhanced PWM output, channel A. May be configured for tri-state during Enhanced PWM shutdown events. Takes priority over port data
	T0CKI	1	I	ST	Timer0 counter input.
RC6/AN8/$\overline{SS}$/ T13CKI/T1OSCI	RC6	0	O	DIG	LATC<6> data output.
		1	I	ST	PORTC<6> data input.
	AN8	1	I	ANA	A/D input channel 8.
	$\overline{SS}$	1	I	TTL	Slave select input for SSP (MSSP module)
	T13CKI	1	I	ST	Timer1 and Timer3 counter input.
	T1OSCI	x	O	ANA	Timer1 oscillator input; enabled when Timer1 oscillator enabled. Disables digital I/O.
RC7/AN9/SDO/ T1OSCO	RC7	0	O	DIG	LATC<7> data output.
		1	I	ST	PORTC<7> data input.
	AN9	1	I	ANA	A/D input channel 9.
	SDO	0	I	DIG	SPI data output (MSSP module); takes priority over port data.
	T1OSCO	x	O	ANA	Timer1 oscillator output; enabled when Timer1 oscillator enabled. Disables digital I/O.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I^2C/SMB = I^2C/SMBus input buffer; x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

TABLE 9-5: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	278
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	278
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	278
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—	278
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	278
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	276
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON	277
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	276
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	277
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	277
PSTRCON	—	—	—	STRSYNC	STRD	STRC	STRB	STRA	277
SLRCON	—	—	—	—	—	SLRC	SLRB	SLRA	278
REFCON1	D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	---	D1NSS	277
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	275
INTCON2	$\overline{RABPU}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP	275
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	275

14.4.7.1 Steering Synchronization

The STRSYNC bit of the PSTRCON register gives the user two selections of when the steering event will happen. When the STRSYNC bit is '0', the steering event will happen at the end of the instruction that writes to the PSTRCON register. In this case, the output signal at the P1<D:A> pins may be an incomplete PWM waveform. This operation is useful when the user firmware needs to immediately remove a PWM signal from the pin.

When the STRSYNC bit is '1', the effective steering update will happen at the beginning of the next PWM period. In this case, steering on/off the PWM output will always produce a complete PWM waveform.

Figures 14-17 and 14-18 illustrate the timing diagrams of the PWM steering depending on the STRSYNC setting.

FIGURE 14-17: EXAMPLE OF STEERING EVENT AT END OF INSTRUCTION (STRSYNC = 0)

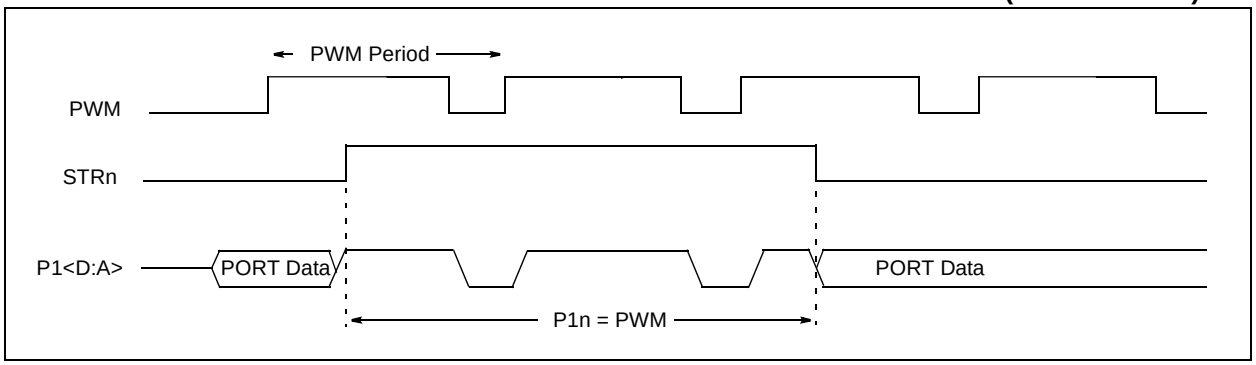
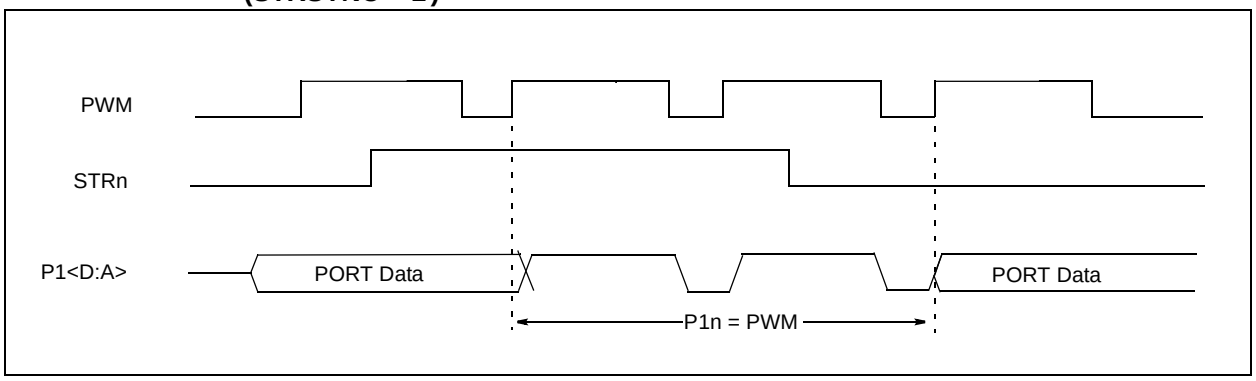


FIGURE 14-18: EXAMPLE OF STEERING EVENT AT BEGINNING OF INSTRUCTION (STRSYNC = 1)



15.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

15.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface, useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C™)
  - Full Master mode
  - Slave mode (with general address call)

The I^2C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

15.2 SPI Mode

The SPI mode allows eight bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

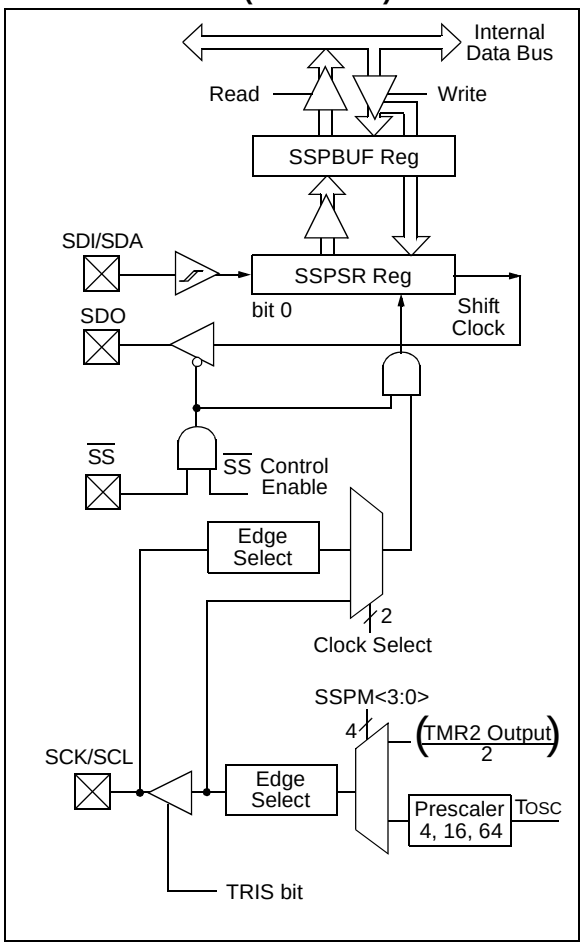
- Serial Data Out – SDO
- Serial Data In – SDI
- Serial Clock – SCK

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select – $\overline{SS}$

Figure 15-1 shows the block diagram of the MSSP module when operating in SPI mode.

FIGURE 15-1: MSSP BLOCK DIAGRAM (SPI MODE)



15.2.2 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON1<5:0> and SSPSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

The MSSP consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR until the received data is ready. Once the eight bits of data have been received, that byte is moved to the SSPBUF register. Then, the Buffer Full detect bit, BF of the SSPSTAT register, and the interrupt flag bit, SSPIF, are set. This double-buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored and the write collision detect bit WCOL of the SSPCON1 register, will be set. User software must clear the WCOL bit to allow the following write(s) to the SSPBUF register to complete successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. The Buffer Full bit, BF of the SSPSTAT register, indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP interrupt is used to determine when the transmission/reception has completed. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur. Example 15-1 shows the loading of the SSPBUF (SSPSR) for data transmission.

The SSPSR is not directly readable or writable and can only be accessed by addressing the SSPBUF register. Additionally, the MSSP Status register (SSPSTAT) indicates the various status conditions.

EXAMPLE 15-1: LOADING THE SSPBUF (SSPSR) REGISTER

```
LOOP    BTFSS   SSPSTAT, BF     ;Has data been received (transmit complete)?
        BRA     LOOP            ;No
        MOVF    SSPBUF, W       ;WREG reg = contents of SSPBUF
        MOVWF   RXDATA          ;Save in user RAM, if data is meaningful
        MOVF    TXDATA, W       ;W reg = contents of TXDATA
        MOVWF   SSPBUF          ;New data to xmit
```

15.3.8 I²C MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start Enable bit, SEN bit of the SSPCON2 register. If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is the Start condition and causes the S bit of the SSPSTAT1 register to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPADD<7:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit of the SSPCON2 register will be automatically cleared by hardware; the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

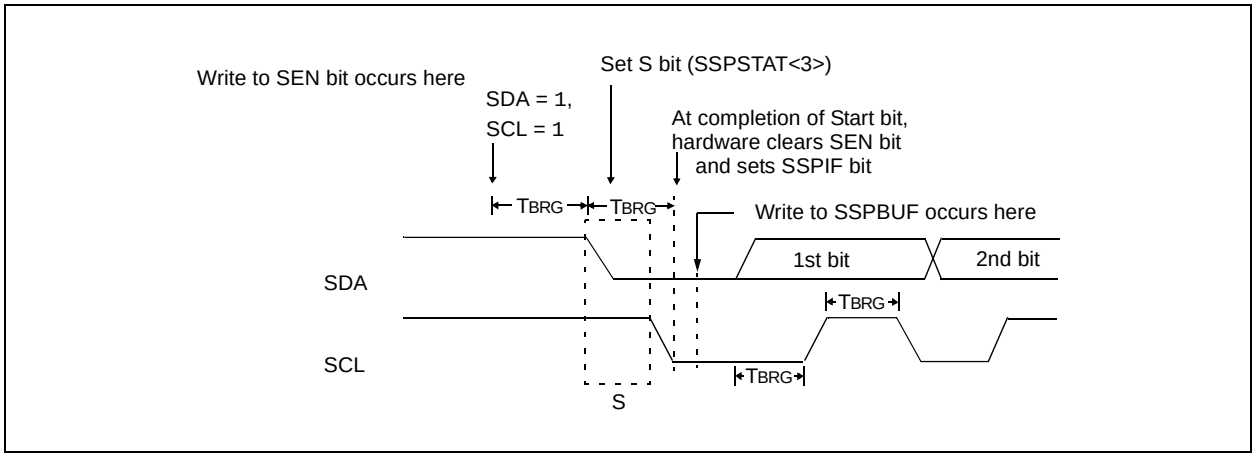
Note:	If at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLIF, is set, the Start condition is aborted and the I²C module is reset into its Idle state.

15.3.8.1 WCOL Status Flag

If the user writes the SSPBUF when a Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write does not occur).

Note:	Because queuing of events is not allowed, writing to the lower five bits of SSPCON2 is disabled until the Start condition is complete.

FIGURE 15-19: FIRST START BIT TIMING



18.2 Comparator Control

Each comparator has a separate control and Configuration register: CM1CON0 for Comparator C1 and CM2CON0 for Comparator C2. In addition, Comparator C2 has a second control register, CM2CON1, for controlling the interaction with Timer1 and simultaneous reading of both comparator outputs.

The CM1CON0 and CM2CON0 registers (see Registers 18-1 and 18-2, respectively) contain the control and Status bits for the following:

- Enable
- Input selection
- Reference selection
- Output selection
- Output polarity
- Speed selection

18.2.1 COMPARATOR ENABLE

Setting the CxON bit of the CMxCON0 register enables the comparator for operation. Clearing the CxON bit disables the comparator resulting in minimum current consumption.

18.2.2 COMPARATOR INPUT SELECTION

The CxCH<1:0> bits of the CMxCON0 register direct one of four analog input pins to the comparator inverting input.

Note:	To use CxIN+ and C12INx- pins as analog inputs, the appropriate bits must be set in the ANSEL register and the corresponding TRIS bits must also be set to disable the output drivers.

18.2.3 COMPARATOR REFERENCE SELECTION

Setting the CxR bit of the CMxCON0 register directs an internal voltage reference or an analog input pin to the non-inverting input of the comparator. See **Section 21.0 "Voltage References"** for more information on the Internal Voltage Reference module.

18.2.4 COMPARATOR OUTPUT SELECTION

The output of the comparator can be monitored by reading either the CxOUT bit of the CMxCON0 register or the MCxOUT bit of the CM2CON1 register. In order to make the output available for an external connection, the following conditions must be true:

- CxOE bit of the CMxCON0 register must be set
- Corresponding TRIS bit must be cleared
- CxON bit of the CMxCON0 register must be set

Both comparators share the same output pin (C12OUT). Priority is determined by the states of the C1OE and C2OE bits.

TABLE 18-1: COMPARATOR OUTPUT PRIORITY

C10E	C2OE	C12OUT
0	0	I/O
0	1	C2OUT
1	0	C1OUT
1	1	C2OUT

Note 1:	The CxOE bit overrides the PORT data latch. Setting the CxON has no impact on the port override.
2:	The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched.

18.2.5 COMPARATOR OUTPUT POLARITY

Inverting the output of the comparator is functionally equivalent to swapping the comparator inputs. The polarity of the comparator output can be inverted by setting the CxPOL bit of the CMxCON0 register. Clearing the CxPOL bit results in a non-inverted output.

Table 18-2 shows the output state versus input conditions, including polarity control.

TABLE 18-2: COMPARATOR OUTPUT STATE VS. INPUT CONDITIONS

Input Condition	CxPOL	CxOUT
CxVIN- > CxVIN+	0	0
CxVIN- < CxVIN+	0	1
CxVIN- > CxVIN+	1	1
CxVIN- < CxVIN+	1	0

18.2.6 COMPARATOR SPEED SELECTION

The trade-off between speed or power can be optimized during program execution with the CxSP control bit. The default state for this bit is '1' which selects the normal speed mode. Device power consumption can be optimized at the cost of slower comparator propagation delay by clearing the CxSP bit to '0'.

18.3 Comparator Response Time

The comparator output is indeterminate for a period of time after the change of an input source or the selection of a new reference voltage. This period is referred to as the response time. The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response time to a comparator input change. See the Comparator and Voltage Reference Specifications in **Section 27.0 "Electrical Specifications"** for more details.

22.2.3 USB STATUS REGISTER (USTAT)

The USB Status register reports the transaction status within the SIE. When the SIE issues a USB transfer complete interrupt, USTAT should be read to determine the status of the transfer. USTAT contains the transfer endpoint number, direction and Ping-Pong Buffer Pointer value (if used).

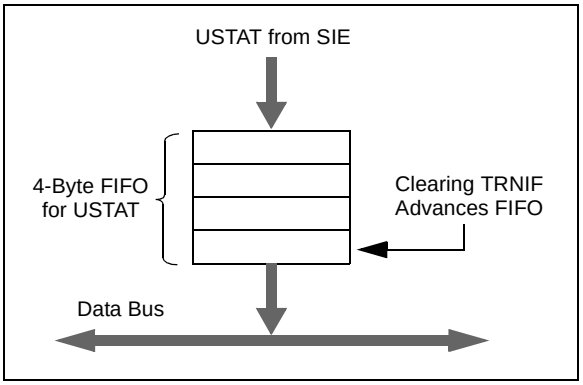
Note:	The data in the USB Status register is valid two SIE clocks after the TRNIF interrupt flag is asserted. In low-speed operation with the system clock operating at 48 MHz, a delay may be required between receiving the TRNIF interrupt and processing the data in the USTAT register.

The USTAT register is actually a read window into a 4-byte status FIFO, maintained by the SIE. It allows the microcontroller to process one transfer while the SIE processes additional endpoints (Figure 22-3). When the SIE completes using a buffer for reading or writing data, it updates the USTAT register. If another USB transfer is performed before a transaction complete interrupt is serviced, the SIE will store the status of the next transfer into the status FIFO.

Clearing the transfer complete flag bit, TRNIF, causes the SIE to advance the FIFO. If the next data in the FIFO holding register is valid, the SIE will reassert the interrupt within 6 TCY of clearing TRNIF. If no additional data is present, TRNIF will remain clear; USTAT data will no longer be reliable.

Note:	If an endpoint request is received while the USTAT FIFO is full, the SIE will automatically issue a NAK back to the host.

FIGURE 22-3: USTAT FIFO



REGISTER 22-3: USTAT: USB STATUS REGISTER

U-0	U-0	R-x	R-x	R-x	R-x	R-x	U-0
—	—	ENDP2	ENDP1	ENDP0	DIR	PPBI(1)	—
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 7-6 **Unimplemented:** Read as '0'

bit 5-3 **ENDP<2:0>:** Encoded Number of Last Endpoint Activity bits
(represents the number of the BDT updated by the last USB transfer)
111 = Endpoint 7
110 = Endpoint 6
....
001 = Endpoint 1
000 = Endpoint 0

bit 2 **DIR:** Last BD Direction Indicator bit
1 = The last transaction was an IN token
0 = The last transaction was an OUT or SETUP token

bit 1 **PPBI:** Ping-Pong BD Pointer Indicator bit(1)
1 = The last transaction was to the Odd BD bank
0 = The last transaction was to the Even BD bank

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

Note 1: This bit is only valid for endpoints with available Even and Odd BD registers.

REGISTER 22-5: BDnSTAT: BUFFER DESCRIPTOR n STATUS REGISTER (BD0STAT THROUGH BD31STAT), CPU MODE (DATA IS WRITTEN TO THE SIDE)

R/W-x	R/W-x	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
UOWN(1)	DTS(2)	—(3)	—(3)	DTSEN	BSTALL	BC9	BC8
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 **UOWN:** USB Own bit(1)
- 0 = The microcontroller core owns the BD and its corresponding buffer

bit 6 **DTS:** Data Toggle Synchronization bit(2)
- 1 = Data 1 packet
- 0 = Data 0 packet

bit 5-4 **Unimplemented:** These bits should always be programmed to '0'.(3)

bit 3 **DTSEN:** Data Toggle Synchronization Enable bit
- 1 = Data toggle synchronization is enabled; data packets with incorrect Sync value will be ignored except for a SETUP transaction, which is accepted even if the data toggle bits do not match
- 0 = No data toggle synchronization is performed

bit 2 **BSTALL:** Buffer Stall Enable bit
- 1 = Buffer stall enabled; STALL handshake issued if a token is received that would use the BD in the given location (UOWN bit remains set, BD value is unchanged)
- 0 = Buffer stall disabled

bit 1-0 **BC<9:8>:** Byte Count 9 and 8 bits

The byte count bits represent the number of bytes that will be transmitted for an IN token or received during an OUT token. Together with BC<7:0>, the valid byte counts are 0-1023.

Note 1: This bit must be initialized by the user to the desired value prior to enabling the USB module.

2: This bit is ignored unless DTSEN = 1.

3: If these bits are set, USB communication may not work. Hence, these bits should always be maintained as '0'.

TABLE 22-2: ASSIGNMENT OF BUFFER DESCRIPTORS FOR THE DIFFERENT BUFFERING MODES

Endpoint	BDs Assigned to Endpoint							
	Mode 0 (No Ping-Pong)		Mode 1 (Ping-Pong on EP0 OUT)		Mode 2 (Ping-Pong on all EPs)		Mode 3 (Ping-Pong on all other EPs, except EP0)	
	Out	In	Out	In	Out	In	Out	In
0	0	1	0 (E), 1 (O)	2	0 (E), 1 (O)	2 (E), 3 (O)	0	1
1	2	3	3	4	4 (E), 5 (O)	6 (E), 7 (O)	2 (E), 3 (O)	4 (E), 5 (O)
2	4	5	5	6	8 (E), 9 (O)	10 (E), 11 (O)	6 (E), 7 (O)	8 (E), 9 (O)
3	6	7	7	8	12 (E), 13 (O)	14 (E), 15 (O)	10 (E), 11 (O)	12 (E), 13 (O)
4	8	9	9	10	16 (E), 17 (O)	18 (E), 19 (O)	14 (E), 15 (O)	16 (E), 17 (O)
5	10	11	11	12	20 (E), 21 (O)	22 (E), 23 (O)	18 (E), 19 (O)	20 (E), 21 (O)
6	12	13	13	14	24 (E), 25 (O)	26 (E), 27 (O)	22 (E), 23 (O)	24 (E), 25 (O)
7	14	15	15	16	28 (E), 29 (O)	30 (E), 31 (O)	26 (E), 27 (O)	28 (E), 29 (O)

Legend: (E) = Even transaction buffer, (O) = Odd transaction buffer

TABLE 22-3: SUMMARY OF USB BUFFER DESCRIPTOR TABLE REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BDnSTAT(1)	UOWN	DTS(4)	PID3(2)	PID2(2)	PID1(2) DTSEN(3)	PID0(2) BSTALL(3)	BC9	BC8
BDnCNT(1)	Byte Count							
BDnADRL(1)	Buffer Address Low							
BDnADRH(1)	Buffer Address High							

Note 1: For buffer descriptor registers, n may have a value of 0 to 31. For the sake of brevity, all 32 registers are shown as one generic prototype. All registers have indeterminate Reset values (xxxx xxxx).

2: Bits 5 through 2 of the BDnSTAT register are used by the SIE to return PID<3:0> values once the register is turned over to the SIE (UOWN bit is set). Once the registers have been under SIE control, the values written for DTSEN and BSTALL are no longer valid.

3: Prior to turning the buffer descriptor over to the SIE (UOWN bit is cleared), bits 5 through 2 of the BDnSTAT register are used to configure the DTSEN and BSTALL settings.

4: This bit is ignored unless DTSEN = 1.

The USB specification limits the power taken from the bus. Each device is ensured 100 mA at approximately 5V (one unit load). Additional power may be requested, up to a maximum of 500 mA. Note that power above one unit load is a request and the host or hub is not obligated to provide the extra current. Thus, a device capable of consuming more than one unit load must be able to maintain a low-power configuration of a one unit load or less, if necessary.

The USB specification also defines a Suspend mode. In this situation, current must be limited to 500 μA, averaged over 1 second. A device must enter a Suspend state after 3 ms of inactivity (i.e., no SOF tokens for 3 ms). A device entering Suspend mode must drop current consumption within 10 ms after Suspend. Likewise, when signaling a wake-up, the device must signal a wake-up within 10 ms of drawing current above the Suspend limit.

22.10.5 ENUMERATION

When the device is initially attached to the bus, the host enters an enumeration process in an attempt to identify the device. Essentially, the host interrogates the device, gathering information such as power consumption, data rates and sizes, protocol and other descriptive information; descriptors contain this information. A typical enumeration process would be as follows:

1. USB Reset: Reset the device. Thus, the device is not configured and does not have an address (address 0).
2. Get Device Descriptor: The host requests a small portion of the device descriptor.
3. USB Reset: Reset the device again.
4. Set Address: The host assigns an address to the device.
5. Get Device Descriptor: The host retrieves the device descriptor, gathering info such as manufacturer, type of device, maximum control packet size.
6. Get configuration descriptors.
7. Get any other descriptors.
8. Set a configuration.

The exact enumeration process depends on the host.

22.10.6 DESCRIPTORS

There are eight different standard descriptor types of which five are most important for this device.

22.10.6.1 Device Descriptor

The device descriptor provides general information, such as manufacturer, product number, serial number, the class of the device and the number of configurations. There is only one device descriptor.

22.10.6.2 Configuration Descriptor

The configuration descriptor provides information on the power requirements of the device and how many different interfaces are supported when in this configuration. There may be more than one configuration for a device (i.e., low-power and high-power configurations).

22.10.6.3 Interface Descriptor

The interface descriptor details the number of endpoints used in this interface, as well as the class of the interface. There may be more than one interface for a configuration.

22.10.6.4 Endpoint Descriptor

The endpoint descriptor identifies the transfer type (**Section 22.10.3 “Transfers”**) and direction, as well as some other specifics for the endpoint. There may be many endpoints in a device and endpoints may be shared in different configurations.

22.10.6.5 String Descriptor

Many of the previous descriptors reference one or more string descriptors. String descriptors provide human readable information about the layer (**Section 22.10.1 “Layered Framework”**) they describe. Often these strings show up in the host to help the user identify the device. String descriptors are generally optional to save memory and are encoded in a unicode format.

22.10.7 BUS SPEED

Each USB device must indicate its bus presence and speed to the host. This is accomplished through a 1.5 kΩ resistor which is connected to the bus at the time of the attachment event.

Depending on the speed of the device, the resistor either pulls up the D+ or D- line to 3.3V. For a low-speed device, the pull-up resistor is connected to the D- line. For a full-speed device, the pull-up resistor is connected to the D+ line.

22.10.8 CLASS SPECIFICATIONS AND DRIVERS

USB specifications include class specifications which operating system vendors optionally support. Examples of classes include Audio, Mass Storage, Communications and Human Interface (HID). In most cases, a driver is required at the host side to ‘talk’ to the USB device. In custom applications, a driver may need to be developed. Fortunately, drivers are available for most common host systems for the most common classes of devices. Thus, these drivers can be reused.

BZ **Branch if Zero**

Syntax: BZ n

Operands: $-128 \le n \le 127$

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: $0 \le k \le 1048575$
$s \in [0,1]$

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	kkkk$_0$
2nd word(k<19:8>)	1111	k_{19}kkk	kkkk	kkkk$_8$

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 (default). 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

SUBLW **Subtract W from literal**

Syntax: SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

Status Affected: N, OV, C, DC, Z

Encoding:

0000	1000	kkkk	kkkk

Description: W is subtracted from the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: `SUBLW 02h`

Before Instruction
W = 01h
C = ?
After Instruction
W = 01h
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBLW 02h`

Before Instruction
W = 02h
C = ?
After Instruction
W = 00h
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBLW 02h`

Before Instruction
W = 03h
C = ?
After Instruction
W = FFh ; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

SUBWF **Subtract W from f**

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

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

Operation: $(f) - (W) \rightarrow dest$

Status Affected: N, OV, C, DC, Z

Encoding:

0101	11da	ffff	ffff

Description: Subtract W from register 'f' (2's complement method). 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 25.2.3 "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: `SUBWF REG, 1, 0`

Before Instruction
REG = 3
W = 2
C = ?
After Instruction
REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBWF REG, 0, 0`

Before Instruction
REG = 2
W = 2
C = ?
After Instruction
REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBWF REG, 1, 0`

Before Instruction
REG = 1
W = 2
C = ?
After Instruction
REG = FFh ;(2's complement)
W = 2
C = 0 ; result is negative
Z = 0
N = 1

25.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F1XK50 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 (page 298) 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

27.2 Standard Operating Conditions

The standard operating conditions for any device are defined as:

Operating Voltage: $V_{DDMIN} \leq V_{DD} \leq V_{DDMAX}$
Operating Temperature: $T_{A\_MIN} \leq T_A \leq T_{A\_MAX}$

V_{DD} — Operating Supply Voltage

PIC18LF1XK50

V_{DDMIN} (Fosc < 16 MHz) ..... +1.8V
V_{DDMIN} (Fosc < 20 MHz) ..... +2.0V
V_{DDMAX} ..... +3.6V

PIC18F1XK50

V_{DDMIN} (Fosc < 16 MHz) ..... +1.8V
V_{DDMIN} (Fosc < 20 MHz) ..... +2.0V
V_{DDMAX} ..... +5.5V

T_A — Operating Ambient Temperature Range

Industrial Temperature

T_{A_MIN} ..... -40°C
T_{A_MAX} ..... +85°C

Extended Temperature

T_{A_MIN} ..... -40°C
T_{A_MAX} ..... +125°C

Unless otherwise noted, V_{IN} = 5V, F_{OSC} = 300 kHz, C_{IN} = 0.1 µF, T_A = 25°C.

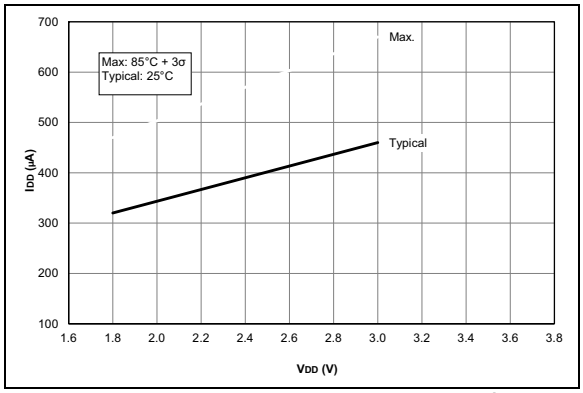


FIGURE 28-13: *IDD, MFINTOSC Mode, FOSC = 500 kHz, PIC18LF1XK50 Only.*

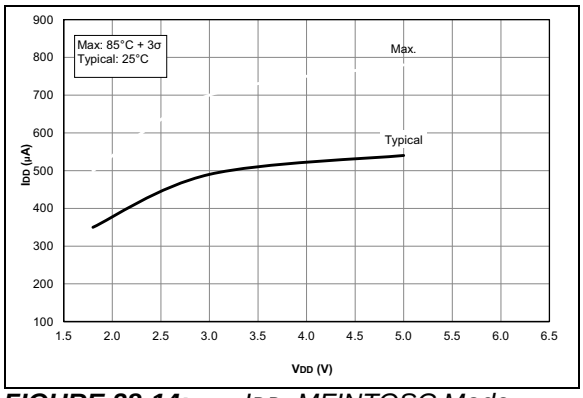


FIGURE 28-14: *IDD, MFINTOSC Mode, FOSC = 500 kHz, PIC18F1XK50 Only.*

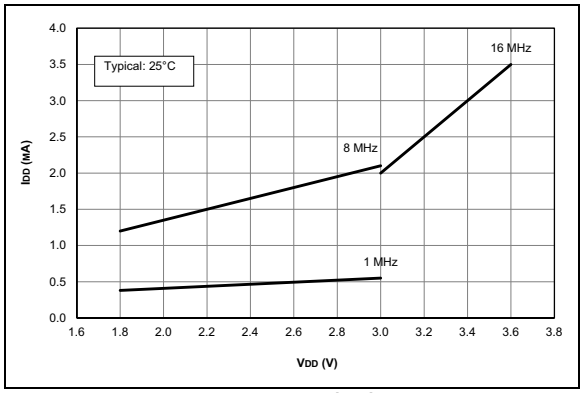


FIGURE 28-15: *IDD Typical, HFINTOSC Mode, PIC18LF1XK50 Only.*

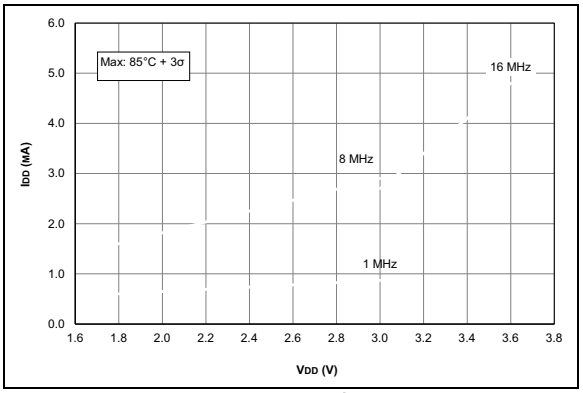


FIGURE 28-16: *IDD Maximum, HFINTOSC Mode, PIC18LF1XK50 Only.*

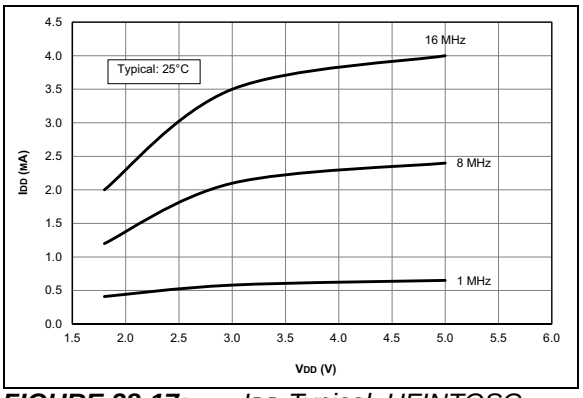


FIGURE 28-17: *IDD Typical, HFINTOSC Mode, PIC18F1XK50 Only.*

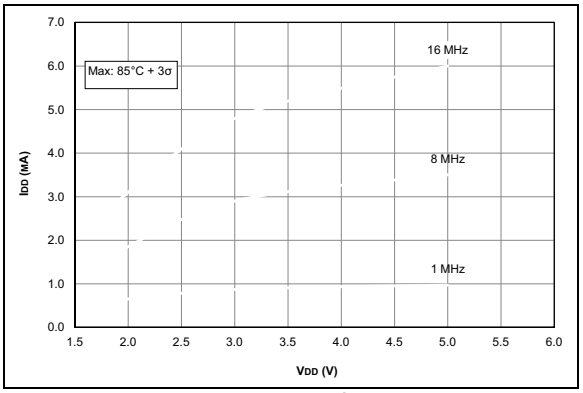


FIGURE 28-18: *IDD Maximum, HFINTOSC Mode, PIC18F1XK50 Only.*

Unless otherwise noted, V_{IN} = 5V, F_{OSC} = 300 kHz, C_{IN} = 0.1 µF, T_A = 25°C.

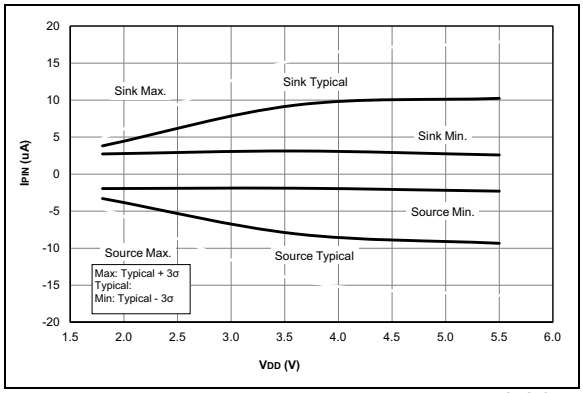


FIGURE 28-55: *CAP Sense Current Sink/ Source Characteristics, Fixed Voltage Reference (CPSRM = 0), High-Current Range (CPSRNG = 11).*

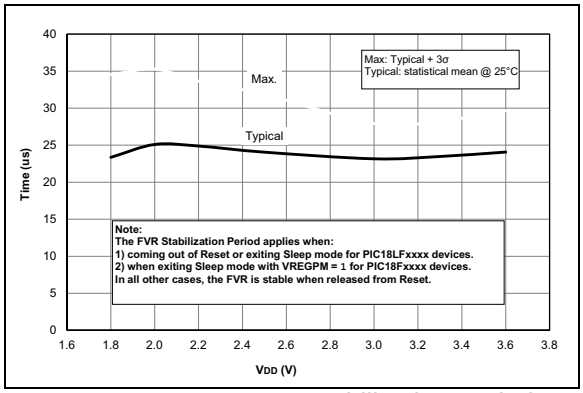


FIGURE 28-58: *FVR Stabilization Period.*

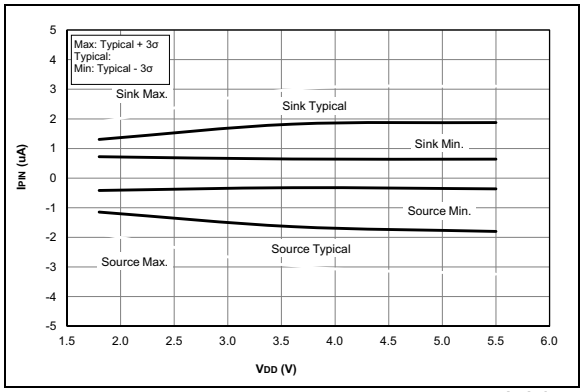


FIGURE 28-56: *CAP Sense Current Sink/ Source Characteristics, Fixed Voltage Reference (CPSRM = 0), Medium-Current Range (CPSRNG = 10).*

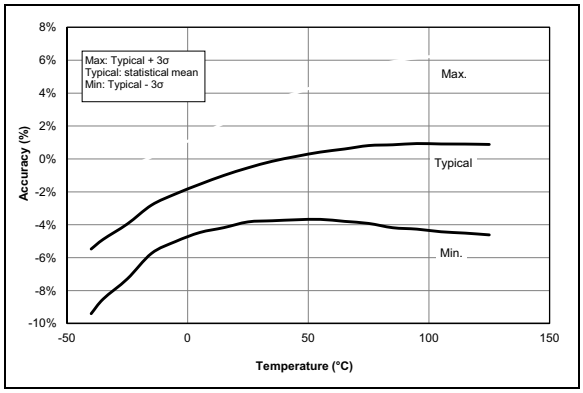


FIGURE 28-59: *HFINTOSC Accuracy, Over Temperature, VDD = 1.8V, PIC18LF1XK50 Only.*

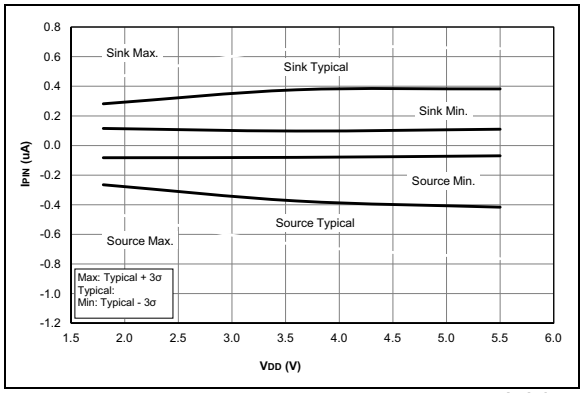


FIGURE 28-57: *CAP Sense Current Sink/ Source Characteristics, Fixed Voltage Reference (CPSRM = 0), Low-Current Range (CPSRNG = 01).*

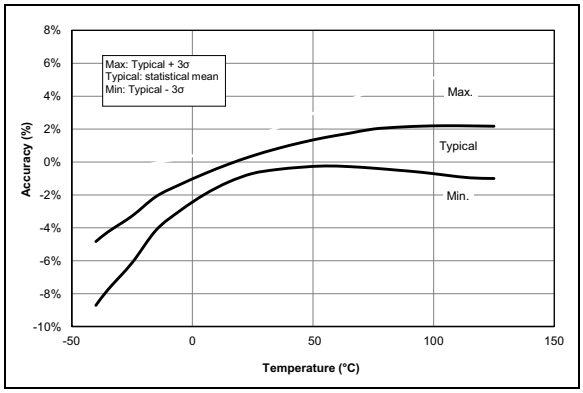
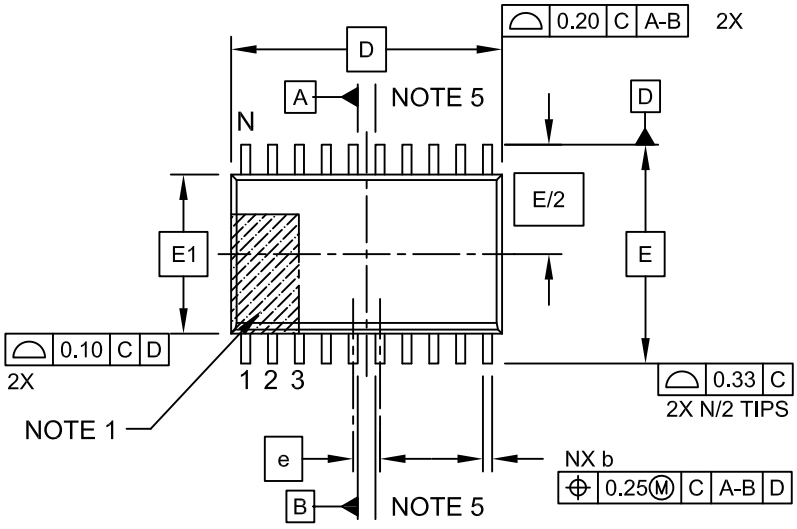


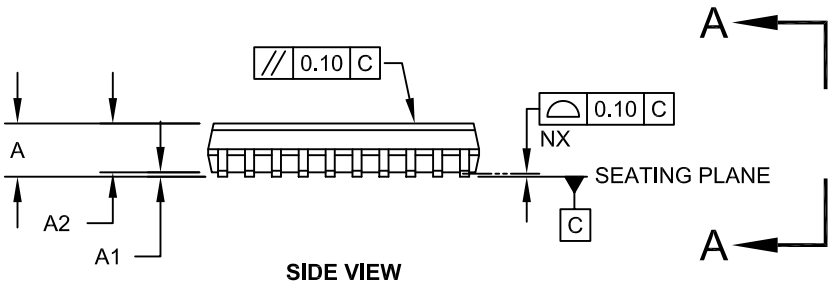
FIGURE 28-60: *HFINTOSC Accuracy, Over Temperature, 2.3V ≤ VDD ≤ 5.5V.*

20-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

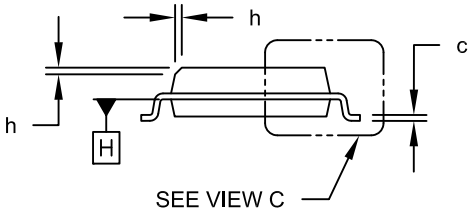
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW



SIDE VIEW



VIEW A-A

Microchip Technology Drawing C04-094C Sheet 1 of 2