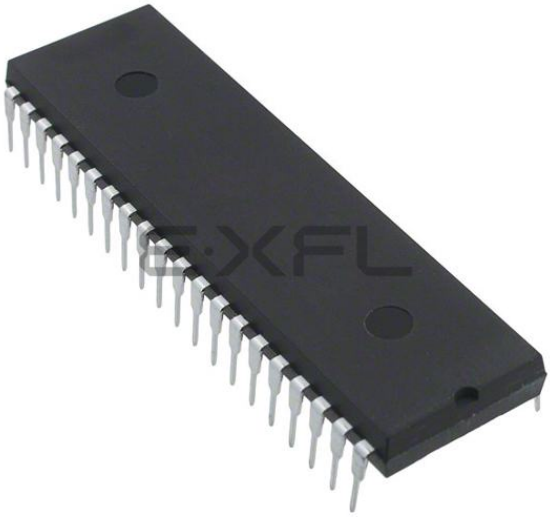


Welcome to E-XFL.COM

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	ECANbus, I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	35
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.6K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 11x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	40-DIP (0.600", 15.24mm)
Supplier Device Package	40-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f46k80-i-p

REGISTER 3-3: OSCTUNE: OSCILLATOR TUNING REGISTER

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 is derived from 16 MHz INTOSC source (divide-by-512 enabled, HF-INTOSC)
0 = 31 kHz device clock is derived from INTOSC 31 kHz oscillator (LF-INTOSC)

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

bit 5-0 **TUN<5:0>:** Fast RC Oscillator (INTOSC) Frequency Tuning bits
011111 = Maximum frequency
•
•
000001
000000 = Center frequency; fast RC oscillator is running at the calibrated frequency
111111
•
•
100000 = Minimum frequency

3.6.3 INTERNAL OSCILLATOR OUTPUT FREQUENCY AND TUNING

The internal oscillator block is calibrated at the factory to produce an INTOSC output frequency of 16 MHz. It can be adjusted in the user's application by writing to TUN<5:0> (OSCTUNE<5:0>) in the OSCTUNE register (Register 3-3).

When the OSCTUNE register is modified, the INTOSC (HF-INTOSC and MF-INTOSC) frequency will begin shifting to the new frequency. The oscillator will require some time to stabilize. Code execution continues during this shift and there is no indication that the shift has occurred.

The LF-INTOSC oscillator operates independently of the HF-INTOSC or the MF-INTOSC source. Any changes in the HF-INTOSC or the MF-INTOSC source, across voltage and temperature, are not necessarily reflected by changes in LF-INTOSC or vice versa. The frequency of LF-INTOSC is not affected by OSCTUNE.

3.6.4 INTOSC FREQUENCY DRIFT

The INTOSC frequency may drift as VDD or temperature changes and can affect the controller operation in a variety of ways. It is possible to adjust the INTOSC frequency by modifying the value in the OSCTUNE register. Depending on the device, this may have no effect on the LF-INTOSC clock source frequency.

Tuning INTOSC requires knowing when to make the adjustment, in which direction it should be made, and in some cases, how large a change is needed. Three compensation techniques are shown here.

3.6.4.1 Compensating with the EUSARTx

An adjustment may be required when the EUSARTx begins to generate framing errors or receives data with errors while in Asynchronous mode. Framing errors indicate that the device clock frequency is too high. To adjust for this, decrement the value in OSCTUNE to reduce the clock frequency. On the other hand, errors in data may suggest that the clock speed is too low. To compensate, increment OSCTUNE to increase the clock frequency.

3.6.4.2 Compensating with the Timers

This technique compares device clock speed to some reference clock. Two timers may be used; one timer is clocked by the peripheral clock, while the other is clocked by a fixed reference source, such as the SOSC oscillator.

Both timers are cleared, but the timer clocked by the reference generates interrupts. When an interrupt occurs, the internally clocked timer is read and both timers are cleared. If the internally clocked timer value is much greater than expected, then the internal oscillator block is running too fast. To adjust for this, decrement the OSCTUNE register.

3.6.4.3 Compensating with the CCP Module in Capture Mode

A CCP module can use free-running Timer1 (or Timer3), clocked by the internal oscillator block and an external event with a known period (i.e., AC power frequency). The time of the first event is captured in the CCPRxH:CCPRxL registers and is recorded for use later. When the second event causes a capture, the time of the first event is subtracted from the time of the second event. Since the period of the external event is known, the time difference between events can be calculated.

If the measured time is much greater than the calculated time, the internal oscillator block is running too fast. To compensate, decrement the OSCTUNE register. If the measured time is much less than the calculated time, the internal oscillator block is running too slow. To compensate, increment the OSCTUNE register.

3.7 Reference Clock Output

In addition to the FOSC/4 clock output, in certain oscillator modes, the device clock in the PIC18F66K80 family can also be configured to provide a reference clock output signal to a port pin. This feature is available in all oscillator configurations and allows the user to select a greater range of clock submultiples to drive external devices in the application.

This reference clock output is controlled by the REFOCON register (Register 3-4). Setting the ROON bit (REFOCON<7>) makes the clock signal available on the REFO (RC3) pin. The RODIV<3:0> bits enable the selection of 16 different clock divider options.

The ROSSLP and ROSEL bits (REFOCON<5:4>) control the availability of the reference output during Sleep mode. The ROSEL bit determines if the oscillator on OSC1 and OSC2, or the current system clock source, is used for the reference clock output. The ROSSLP bit determines if the reference source is available on RE3 when the device is in Sleep mode.

To use the reference clock output in Sleep mode, both the ROSSLP and ROSEL bits must be set. The device clock must also be configured for an EC or HS mode. If not, the oscillator on OSC1 and OSC2 will be powered down when the device enters Sleep mode. Clearing the ROSEL bit allows the reference output frequency to change as the system clock changes during any clock switches.

FIGURE 5-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1

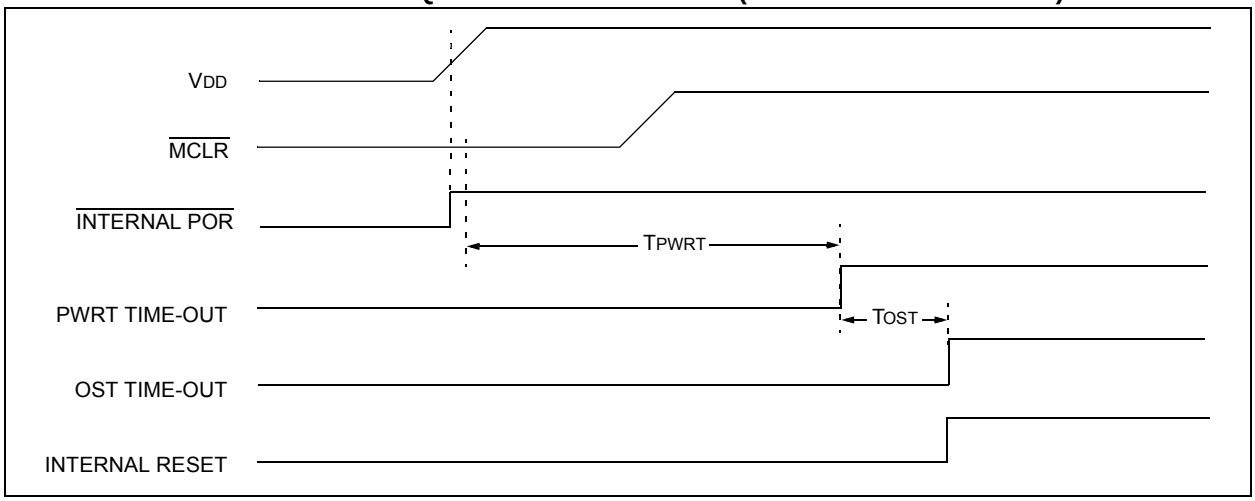


FIGURE 5-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2

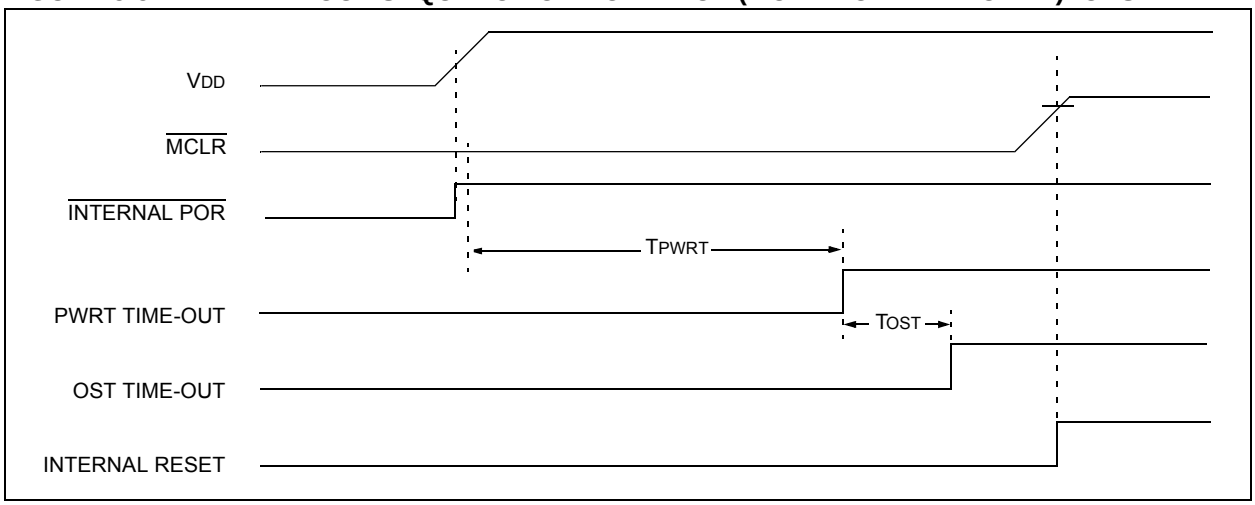
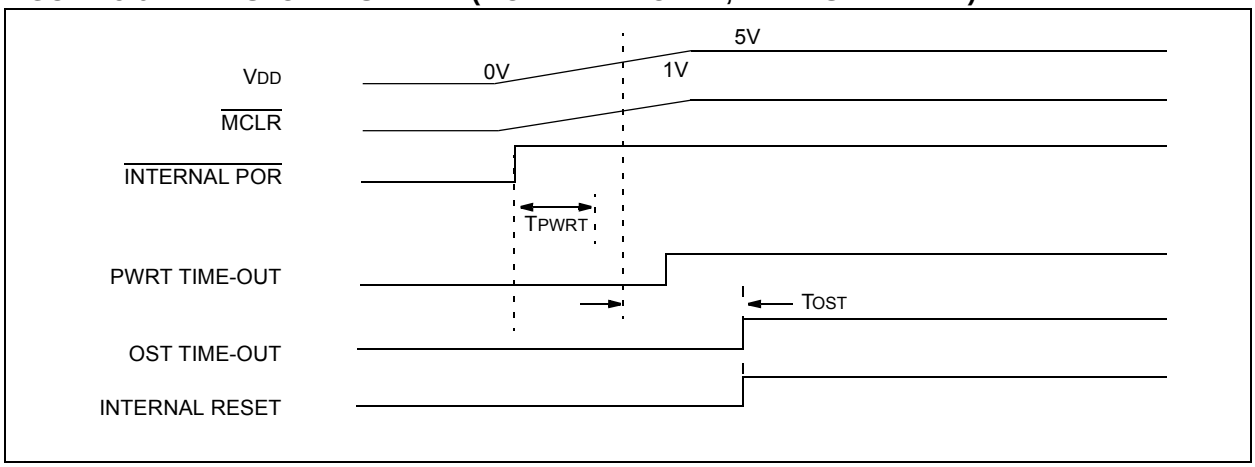


FIGURE 5-6: SLOW RISE TIME (MCLR TIED TO VDD, VDD RISE > TPWRT)



6.6 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = 1) significantly changes certain aspects of data memory and its addressing. Using the Access Bank for many of the core PIC18 instructions introduces a new addressing mode for the data memory space. This mode also alters the behavior of Indirect Addressing using FSR2 and its associated operands.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode. Inherent and literal instructions do not change at all. Indirect Addressing with FSR0 and FSR1 also remains unchanged.

6.6.1 INDEXED ADDRESSING WITH LITERAL OFFSET

Enabling the PIC18 extended instruction set changes the behavior of Indirect Addressing using the FSR2 register pair and its associated file operands. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented instructions – can invoke a form of Indexed Addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset or the Indexed Literal Offset mode.

When using the extended instruction set, this addressing mode requires the following:

- Use of the Access Bank (‘a’ = 0)
- A file address argument that is less than or equal to 5Fh

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in Direct Addressing) or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

6.6.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

Any of the core PIC18 instructions that can use Direct Addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit = 1), or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is shown in Figure 6-9.

Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in **Section 29.2.1 “Extended Instruction Syntax”**.

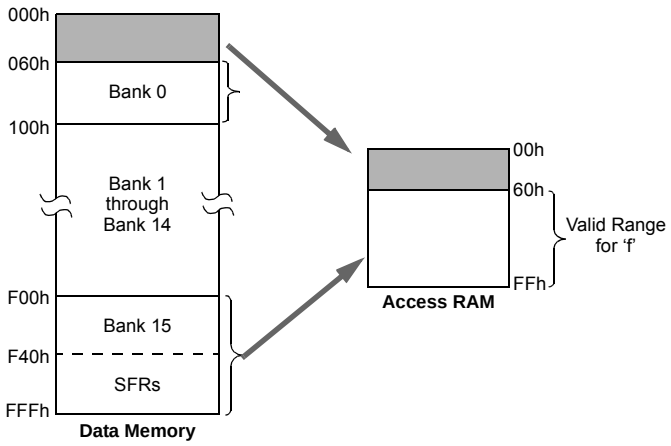
FIGURE 6-9: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

EXAMPLE INSTRUCTION: `ADDWF, f, d, a` (Opcode: `0010 01da ffff ffff`)

When a = 0 and f ≥ 60h:

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and FFFh. This is the same as locations, F60h to FFFh, (Bank 15) of data memory.

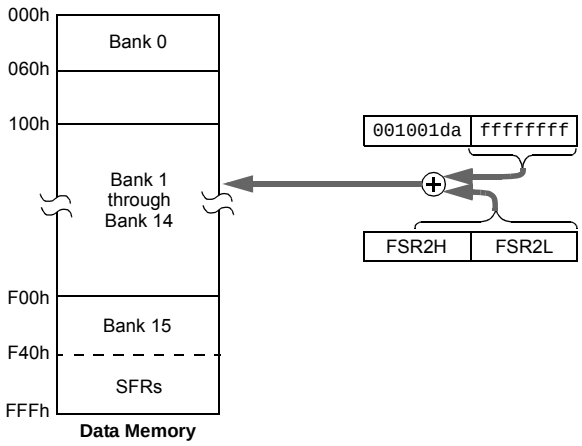
Locations below 060h are not available in this addressing mode.



When a = 0 and f ≤ 5Fh:

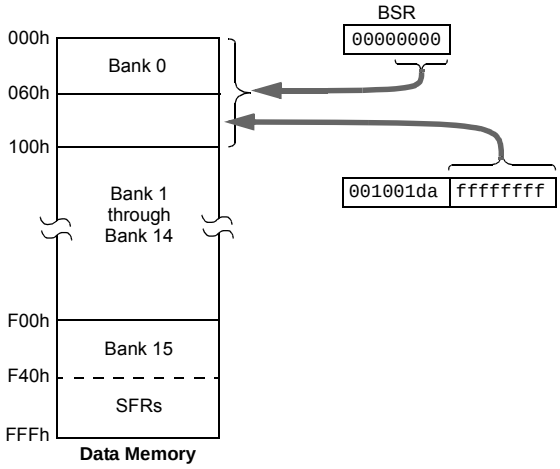
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

Note that in this mode, the correct syntax is now:
`ADDWF [k], d`
where 'k' is the same as 'f'.



When a = 1 (all values of f):

The instruction executes in Direct mode (also known as Direct Long mode). 'f' is interpreted as a location in one of the 16 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.



10.0 INTERRUPTS

Members of the PIC18F66K80 family of devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high-priority level or a low-priority level. The high-priority interrupt vector is at 0008h and the low-priority interrupt vector is at 0018h. High-priority interrupt events will interrupt any low-priority interrupts that may be in progress.

The registers for controlling interrupt operation are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3, PIR4 and PIR5
- PIE1, PIE2, PIE3, PIE4 and PIE5
- IPR1, IPR2, IPR3, IPR4 and IPR5

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/compiler to automatically take care of the placement of these bits within the specified register.

In general, interrupt sources have three bits to control their operation. They are:

- **Flag bit** – Indicating that an interrupt event occurred
- **Enable bit** – Enabling program execution to branch to the interrupt vector address when the flag bit is set
- **Priority bit** – Specifying high priority or low priority

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When interrupt priority is enabled, there are two bits that enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts that have the priority bit set (high priority). Setting the GIEL bit (INTCON<6>) enables all interrupts that have the priority bit cleared (low priority). When the interrupt flag, enable bit and appropriate Global Interrupt Enable bit are set, the interrupt will vector immediately to address 0008h or 0018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit that enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit that enables/disables all interrupt sources. All interrupts branch to address 0008h in Compatibility mode.

When an interrupt is responded to, the Global Interrupt Enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High-priority interrupt sources can interrupt a low-priority interrupt. Low-priority interrupts are not processed while high-priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (0008h or 0018h). Once in the Interrupt Service Routine (ISR), the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

The “return from interrupt” instruction, `RETFIE`, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used) that re-enables interrupts.

For external interrupt events, such as the INTx pins or the PORTB input change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set regardless of the status of their corresponding enable bit or the GIE bit.

Note:	Do not use the `MOVFF` instruction to modify any of the Interrupt Control registers while **any** interrupt is enabled. Doing so may cause erratic microcontroller behavior.

REGISTER 10-13: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRXIE	WAKIE	ERRIE	TXB2IE	TXB1IE	TXB0IE	RXB1IE	RXB0IE/ FIFOFIE
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 **IRXIE:** Invalid Message Received Interrupt Flag bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 6 **WAKIE:** Bus Wake-up Activity Interrupt Flag bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 5 **ERRIE:** Error Interrupt Flag bit (multiple sources in the COMSTAT register)
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 4 **TXB2IE:** Transmit Buffer 2 Interrupt Flag bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 3 **TXB1IE:** Transmit Buffer 1 Interrupt Flag bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 2 **TXB0IE:** Transmit Buffer 0 Interrupt Flag bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 1 **RXB1IE:** Receive Buffer 1 Interrupt Flag bit
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 0 Bit operation is dependent on the selected mode:

Mode 0:
RXB0IE: Receive Buffer 0 Interrupt Flag bit
1 = Interrupt is enabled
0 = Interrupt is disabled

Mode 1:
Unimplemented: Read as '0'

Mode 2:
FIFOFIE: FIFO Full Interrupt Flag bit
1 = Interrupt is enabled
0 = Interrupt is disabled

NOTES:

18.9 Measuring Temperature with the CTMU Module

The CTMU, along with an internal diode, can be used to measure the temperature. The A/D can be connected to the internal diode and the CTMU module can source the current to the diode. The A/D reading will reflect the temperature. With the increase, the A/D readings will go low. This can be used for low-cost temperature measurement applications.

EXAMPLE 18-6: ROUTINE FOR TEMPERATURE MEASUREMENT USING INTERNAL DIODE

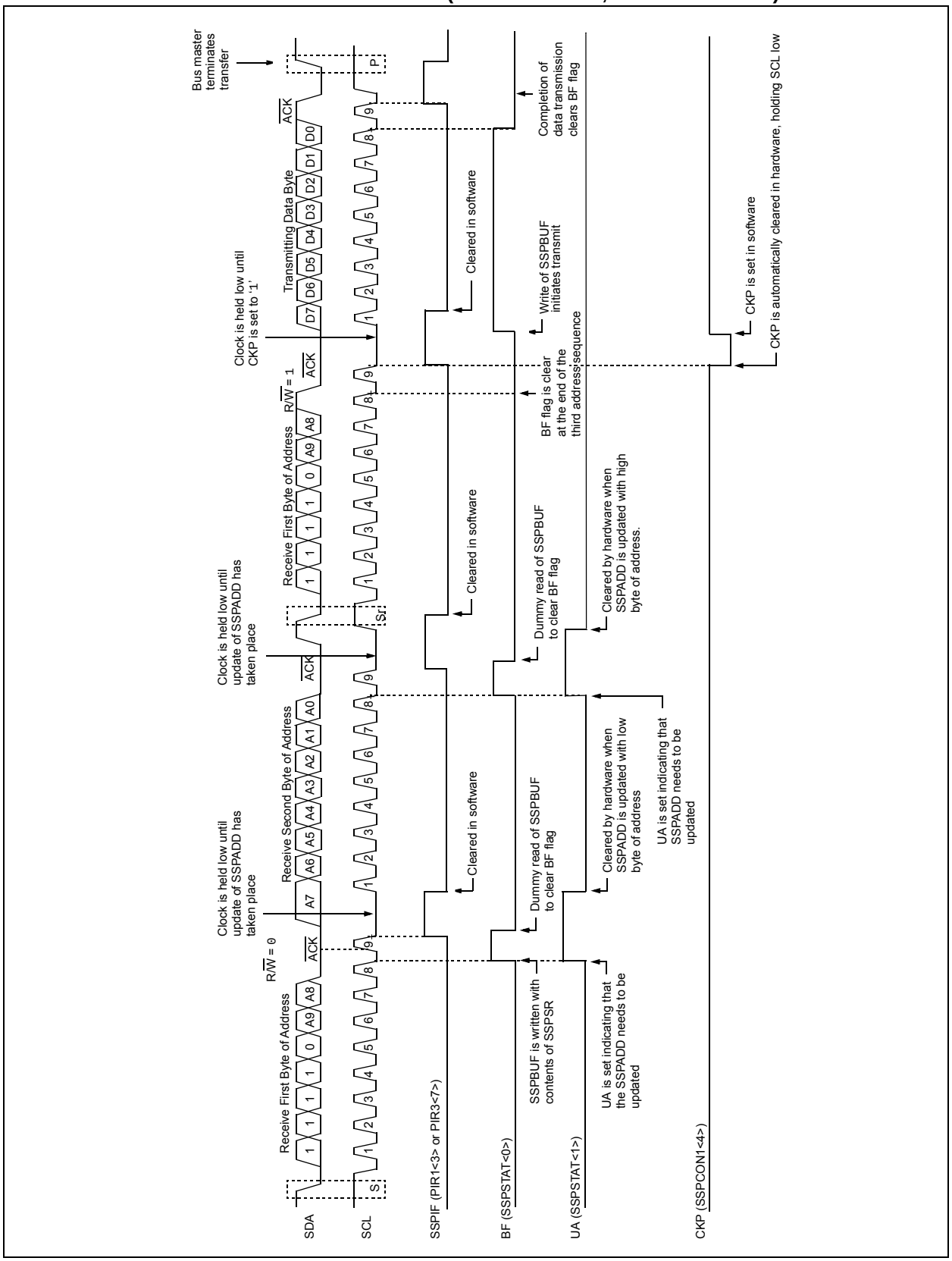
```
// Initialize CTMU
CTMUICON = 0x03;
CTMUCONHbits.CTMUEN = 1;
CTMUCONLbits.EDG1STAT = 1;

// Initialize ADC
ADCON0 = 0xE5;                          // Enable ADC and connect to Internal diode
ADCON1 = 0x00;
ADCON2 = 0xBE;                          //Right Justified

ADCON0bits.GO = 1;                      // Start conversion
while(ADCON0bits.G0);
Temp = ADRES;                           // Read ADC results (inversely proportional to temperature)
```

Note: The temperature diode is not calibrated or standardized; the user must calibrate the diode to their application.

FIGURE 21-13: I²C™ SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)



REGISTER 23-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
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 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

bit 6 **Unimplemented:** Read as '0'

bit 5-3 **ACQT<2:0>:** A/D Acquisition Time Select bits
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

bit 2-0 **ADCS<2:0>:** A/D Conversion Clock Select bits
111 = FRC (clock derived from A/D RC oscillator)(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock derived from A/D RC oscillator)(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

Note 1: If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

23.3 A/D Acquisition Requirements

For the A/D Converter to meet its specified accuracy, the Charge Holding (C_{HOLD}) capacitor must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 23-5. The source impedance (R_S) and the internal sampling switch (R_{SS}) impedance directly affect the time required to charge the capacitor C_{HOLD}. The sampling switch (R_{SS}) impedance varies over the device voltage (V_{DD}).

The source impedance affects the offset voltage at the analog input (due to pin leakage current). **The maximum recommended impedance for analog sources is 2.5 kΩ.** After the analog input channel is selected or changed, the channel must be sampled for at least the minimum acquisition time before starting a conversion.

Note:	When the conversion is started, the holding capacitor is disconnected from the input pin.

To calculate the minimum acquisition time, Equation 23-1 can be used. This equation assumes that 1/2 LSb error is used (1,024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Equation 23-3 shows the calculation of the minimum required acquisition time, T_{ACQ}. This calculation is based on the following application system assumptions:

C_{HOLD}	=	25 pF
Rs	=	2.5 kΩ
Conversion Error	≤	1/2 LSb
V_{DD}	=	3V → Rss = 2 kΩ
Temperature	=	85°C (system max.)

EQUATION 23-1: ACQUISITION TIME

$$T_{ACQ} = \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$

EQUATION 23-2: A/D MINIMUM CHARGING TIME

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \bullet (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$
or
$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ln(1/2048)$$

EQUATION 23-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$

$T_{AMP} = 0.2\ \mu s$

$T_{COFF} = (\text{Temp} - 25°C)(0.02\ \mu s/°C)$
$(85°C - 25°C)(0.02\ \mu s/°C)$
$1.2\ \mu s$

Temperature coefficient is only required for temperatures > 25°C. Below 25°C, T_{COFF} = 0 ms.

$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ln(1/2048)\ \mu s$
$-(25\ pF)(1\ k\Omega + 2\ k\Omega + 2.5\ k\Omega)\ln(0.0004883)\ \mu s$
$1.05\ \mu s$

$T_{ACQ} = 0.2\ \mu s + 1.05\ \mu s + 1.2\ \mu s$
$2.45\ \mu s$

REGISTER 27-2: CANSTAT: CAN STATUS REGISTER

Mode 0	R-1	R-0	R-0	R-0	R-0	R-0	R-0	U-0
	OPMODE2(1)	OPMODE1(1)	OPMODE0(1)	—	ICODE2	ICODE1	ICODE0	—

Mode 1,2	R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	OPMODE2(1)	OPMODE1(1)	OPMODE0(1)	EICODE4	EICODE3	EICODE2	EICODE1	EICODE0
	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 7-5 **OPMODE<2:0>:** Operation Mode Status bits(1)
111 = Reserved
110 = Reserved
101 = Reserved
100 = Configuration mode
011 = Listen Only mode
010 = Loopback mode
001 = Disable/Sleep mode
000 = Normal mode

bit 4 Mode 0:
Unimplemented: Read as '0'

bit 3-1,4-0 Mode 0:
ICODE<2:0>: Interrupt Code bits
When an interrupt occurs, a prioritized coded interrupt value will be present in these bits. This code indicates the source of the interrupt. By copying ICODE<3:1> to WIN<3:0> (Mode 0) or EICODE<4:0> to EWIN<4:0> (Mode 1 and 2), it is possible to select the correct buffer to map into the Access Bank area. See Example 27-2 for a code example. To simplify the description, the following table lists all five bits.

	Mode 0	Mode 1	Mode 2
No interrupt	00000	00000	00000
CAN bus error interrupt	00010	00010	00010
TXB2 interrupt	00100	00100	00100
TXB1 interrupt	00110	00110	00110
TXB0 interrupt	01000	01000	01000
RXB1 interrupt	01010	10001	-----
RXB0 interrupt	01100	10000	10000
Wake-up interrupt	00010	01110	01110
RXB0 interrupt	-----	10000	10000
RXB1 interrupt	-----	10001	10000
RX/TX B0 interrupt	-----	10010	10010(2)
RX/TX B1 interrupt	-----	10011	10011(2)
RX/TX B2 interrupt	-----	10100	10100(2)
RX/TX B3 interrupt	-----	10101	10101(2)
RX/TX B4 interrupt	-----	10110	10110(2)
RX/TX B5 interrupt	-----	10111	10111(2)

bit 0 Mode 0:
Unimplemented: Read as '0'

bit 4-0 Mode 1, 2:
EICODE<4:0>: Interrupt Code bits
See ICODE<3:1> above.

Note 1: To achieve maximum power saving and/or able to wake-up on CAN bus activity, switch the CAN module in Disable/Sleep mode before putting the device to Sleep.

2: If the buffer is configured as a receiver, the EICODE bits will contain '10000' upon interrupt.

REGISTER 27-26: BnSIDL: TX/RX BUFFER 'n' STANDARD IDENTIFIER REGISTERS, LOW BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0](1)

R-x	R-x	R-x	R-x	R-x	U-0	R-x	R-x
SID2	SID1	SID0	SRR	EXIDE	—	EID17	EID16
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 7-5 **SID<2:0>:** Standard Identifier bits (if EXID = 0)
Extended Identifier bits, EID<20:18> (if EXID = 1).

bit 4 **SRR:** Substitute Remote Transmission Request bit
This bit is always '1' when EXID = 1 or equal to the value of RXRTRRO (BnCON<5>) when EXID = 0.

bit 3 **EXIDE:** Extended Identifier Enable bit
1 = Received message is an extended identifier frame (SID<10:0> are EID<28:18>)
0 = Received message is a standard identifier frame

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

bit 1-0 **EID<17:16>:** Extended Identifier bits

Note 1: These registers are available in Mode 1 and 2 only.

REGISTER 27-27: BnSIDL: TX/RX BUFFER 'n' STANDARD IDENTIFIER REGISTERS, LOW BYTE IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1](1)

R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDE	—	EID17	EID16
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 7-5 **SID<2:0>:** Standard Identifier bits (if EXIDE (TXBnSIDL<3>) = 0)
Extended Identifier bits, EID<20:18> (if EXIDE = 1).

bit 4 **Unimplemented:** Read as '0'

bit 3 **EXIDE:** Extended Identifier Enable bit
1 = Message will transmit extended ID, SID<10:0> bits become EID<28:18>
0 = Received will transmit standard ID, EID<17:0> are ignored

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

bit 1-0 **EID<17:16>:** Extended Identifier bits

Note 1: These registers are available in Mode 1 and 2 only.

REGISTER 27-35: BnDLC: TX/RX BUFFER 'n' DATA LENGTH CODE REGISTERS IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 1](1)

U-0	R/W-x	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0
bit 7							bit 0

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

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

bit 6 **TXRTR:** Transmitter Remote Transmission Request bit
1 = Transmitted message will have the RTR bit set
0 = Transmitted message will have the RTR bit cleared

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **DLC<3:0>:** Data Length Code bits
1111-1001 = Reserved
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 byte
0000 = Data length = 0 bytes

Note 1: These registers are available in Mode 1 and 2 only.

REGISTER 27-36: BSEL0: BUFFER SELECT REGISTER 0(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
B5TXEN	B4TXEN	B3TXEN	B2TXEN	B1TXEN	B0TXEN	—	—
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 7-2 **B<5:0>TXEN:** Buffer 5 to Buffer 0 Transmit Enable bits
1 = Buffer is configured in Transmit mode
0 = Buffer is configured in Receive mode

bit 1-0 **Unimplemented:** Read as '0'

Note 1: These registers are available in Mode 1 and 2 only.

LFSR **Load FSR**

Syntax: LFSR f, k

Operands: $0 \le f \le 2$
$0 \le k \le 4095$

Operation: $k \rightarrow FSRf$

Status Affected: None

Encoding:

1110 1111	1110 0000	00ff $k_7$kkk	k_{11}kkk kkkk

Description: The 12-bit literal 'k' is loaded into the file select register pointed to by 'f'.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k' MSB	Process Data	Write literal 'k' MSB to FSRfH
Decode	Read literal 'k' LSB	Process Data	Write literal 'k' to FSRfL

Example: `LFSR 2, 3ABh`

After Instruction
FSR2H = 03h
FSR2L = ABh

MOVF **Move f**

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

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

Operation: $f \rightarrow dest$

Status Affected: N, Z

Encoding:

0101	00da	ffff	ffff

Description: The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256-byte bank.

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

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 29.2.3 "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 W

Example: `MOVF REG, 0, 0`

Before Instruction
REG = 22h
W = FFh
After Instruction
REG = 22h
W = 22h

31.2 DC Characteristics: Power-Down and Supply Current PIC18F66K80 Family (Industrial/Extended) (Continued)

PIC18F66K80 Family (Industrial/Extended)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended					
Param No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	Supply Current (IDD) Cont.(2,3)						
	PIC18**LF**XXK80	270	600	µA	-40°C	VDD = 1.8V(4) Regulator Disabled	FOSC = 4 MHz (**PRI_RUN** mode, EC oscillator)
		270	600	µA	+25°C		
		270	600	µA	+60°C		
		300	700	µA	+85°C		
		320	850	µA	+125°C		
	PIC18**LF**XXK80	540	1000	µA	-40°C	VDD = 3.3V(4) Regulator Disabled	
		540	1000	µA	+25°C		
		540	1000	µA	+60°C		
		550	1100	µA	+85°C		
		560	1200	µA	+125°C		
	PIC18**F**XXK80	566	1020	µA	-40°C	VDD = 3.3V(5) Regulator Enabled	
		585	1020	µA	+25°C		
		590	1020	µA	+60°C		
		595	1120	µA	+85°C		
		600	1220	µA	+125°C		
	PIC18**F**XXK80	630	2000	µA	-40°C	VDD = 5V(5) Regulator Enabled	
		630	2000	µA	+25°C		
		630	2000	µA	+60°C		
		640	2000	µA	+85°C		
		650	2000	µA	+125°C		

Legend: Shading of rows is to assist in readability of the table.

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, SOSC oscillator, BOR, etc.).

2: 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.

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

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;

$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

3: Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

4: For LF devices, $\overline{RETEN}$ (CONFIG1L<0>) = 1.

5: For F devices, SRETEN (WDTCON<4>) = 1 and $\overline{RETEN}$ (CONFIG1L<0>) = 0.

31.6.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 31-4: EXTERNAL CLOCK TIMING

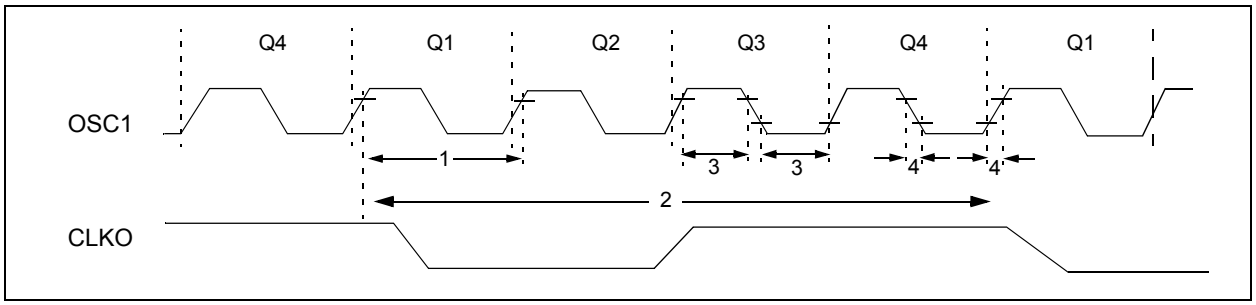


TABLE 31-6: EXTERNAL CLOCK TIMING REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	FOSC	External CLKIN Frequency(1)	DC	64	MHz	EC, ECIO Oscillator mode
		Oscillator Frequency(1)	DC	4	MHz	RC Oscillator mode
			0.1	4	MHz	XT Oscillator mode
			4	16	MHz	HS Oscillator mode
			4	16	MHz	HS + PLL Oscillator mode
			5	33	kHz	LP Oscillator mode
1	TOSC	External CLKIN Period(1)	15.6	—	ns	EC, ECIO Oscillator mode
		Oscillator Period(1)	250	—	ns	RC Oscillator mode
			250	10,000	ns	XT Oscillator mode
			40	250	ns	HS Oscillator mode
			62.5	250	ns	HS + PLL Oscillator mode
			5	200	µs	LP Oscillator mode
2	TCY	Instruction Cycle Time(1)	62.5	—	ns	TCY = 4/FOSC
3	TOSL, TOSH	External Clock in (OSC1) High or Low Time	30	—	ns	XT Oscillator mode
			2.5	—	µs	LP Oscillator mode
			10	—	ns	HS Oscillator mode
4	TOSR, TOSF	External Clock in (OSC1) Rise or Fall Time	—	20	ns	XT Oscillator mode
			—	50	ns	LP Oscillator mode
			—	7.5	ns	HS 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/CLKIN pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.

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

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

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

Trademarks

ISBN: 978-1-62076-074-1

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949

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