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	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	512 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-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f13k50-i-so

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@microchip.com**. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000000A is version A of document DS30000000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; **http://www.microchip.com**
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our web site at **www.microchip.com** to receive the most current information on all of our products.

3.1.1 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC<15:8> bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes PCL. Similarly, the upper two bytes of the program counter are transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see **Section 3.1.4.1 "Computed GOTO"**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit (LSb) of PCL is fixed to a value of '0'. The PC increments by two to address sequential instructions in the program memory.

The CALL, RCALL, GOTO and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

3.1.2 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC is pushed onto the stack when a CALL or RCALL instruction is executed or an interrupt is Acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the RETURN or CALL instructions.

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer, STKPTR. The stack space is not part of either program or data space. The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the Top-of-Stack (TOS) Special File Registers. Data can also be pushed to, or popped from the stack, using these registers.

A CALL type instruction causes a push onto the stack; the Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a pop from the stack; the contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

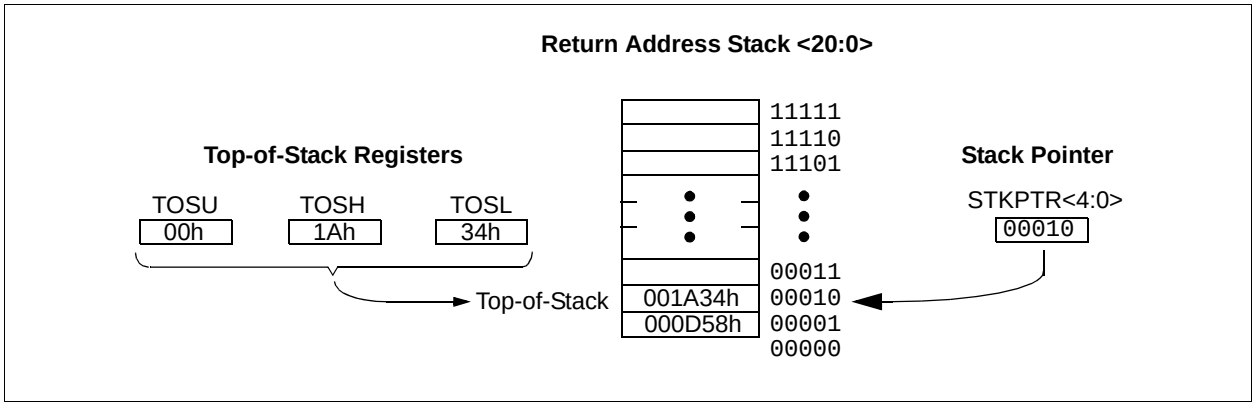
The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits indicate if the stack is full or has overflowed or has underflowed.

3.1.2.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, hold the contents of the stack location pointed to by the STKPTR register (Figure 3-2). This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the global interrupt enable bits while accessing the stack to prevent inadvertent stack corruption.

FIGURE 3-2: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



3.1.2.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 3-1) contains the Stack Pointer value, the STKFUL (stack full) bit and the STKUNF (Stack Underflow) bits. The value of the Stack Pointer can be 0 through 31. The Stack Pointer increments before values are pushed onto the stack and decrements after values are popped off the stack. On Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System (RTOS) for return stack maintenance.

After the PC is pushed onto the stack 31 times (without popping any values off the stack), the STKFUL bit is set. The STKFUL bit is cleared by software or by a POR.

The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (Refer to **Section 24.1 "Configuration Bits"** for a description of the device Configuration bits.) If STVREN is set (default), the 31st push will push the (PC + 2) value onto the stack, set the STKFUL bit and reset the device. The STKFUL bit will remain set and the Stack Pointer will be set to zero.

If STVREN is cleared, the STKFUL bit will be set on the 31st push and the Stack Pointer will increment to 31. Any additional pushes will not overwrite the 31st push and STKPTR will remain at 31.

When the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC and sets the STKUNF bit, while the Stack Pointer remains at zero. The STKUNF bit will remain set until cleared by software or until a POR occurs.

Note:	Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the Reset vector, where the stack conditions can be verified and appropriate actions can be taken. This is not the same as a Reset, as the contents of the SFRs are not affected.

3.1.2.3 PUSH and POP Instructions

Since the Top-of-Stack is readable and writable, the ability to push values onto the stack and pull values off the stack without disturbing normal program execution is a desirable feature. The PIC18 instruction set includes two instructions, PUSH and POP, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

The PUSH instruction places the current PC value onto the stack. This increments the Stack Pointer and loads the current PC value onto the stack.

The POP instruction discards the current TOS by decrementing the Stack Pointer. The previous value pushed onto the stack then becomes the TOS value.

REGISTER 3-1: STKPTR: STACK POINTER REGISTER

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKFUL(1)	STKUNF(1)	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

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

bit 7 **STKFUL:** Stack Full Flag bit(1)
- 1 = Stack became full or overflowed
- 0 = Stack has not become full or overflowed

bit 6 **STKUNF:** Stack Underflow Flag bit(1)
- 1 = Stack underflow occurred
- 0 = Stack underflow did not occur

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

bit 4-0 **SP<4:0>:** Stack Pointer Location bits

Note 1: Bit 7 and bit 6 are cleared by user software or by a POR.

4.0 FLASH PROGRAM MEMORY

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

A read from program memory is executed one byte at a time. A write to program memory is executed on blocks of 16 or 8 bytes at a time depending on the specific device (see Table 4-1). Program memory is erased in blocks of 64 bytes at a time. The difference between the write and erase block sizes requires from 1 to 8 block writes to restore the contents of a single block erase. A bulk erase operation can not be issued from user code.

TABLE 4-1: WRITE/ERASE BLOCK SIZES

Device	Write Block Size (bytes)	Erase Block Size (bytes)
PIC18F13K50	8	64
PIC18F14K50	16	64

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

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

4.1 Table Reads and Table Writes

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

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

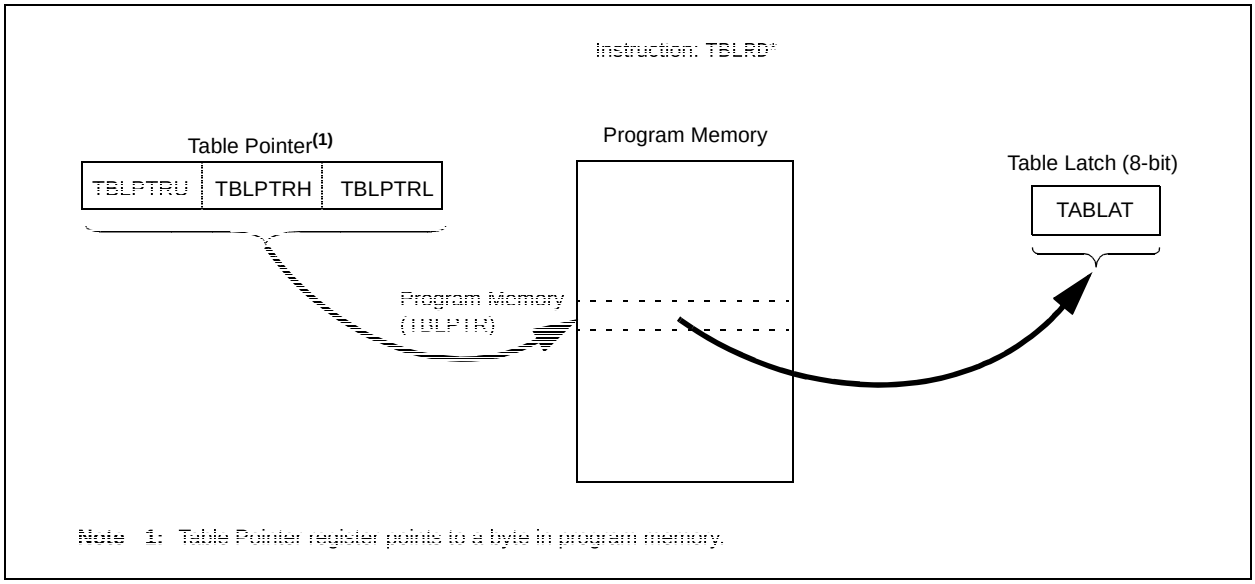
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

The table read operation retrieves one byte of data directly from program memory and places it into the TABLAT register. Figure 4-1 shows the operation of a table read.

The table write operation stores one byte of data from the TABLAT register into a write block holding register. The procedure to write the contents of the holding registers into program memory is detailed in **Section 4.5 “Writing to Flash Program Memory”**. Figure 4-2 shows the operation of a table write with program memory and data RAM.

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

FIGURE 4-1: TABLE READ OPERATION



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.

14.3 Compare Mode

In Compare mode, the 16-bit CCPR1 register value is constantly compared against either the TMR1 or TMR3 register pair value. When a match occurs, the CCP1 pin can be:

- driven high
- driven low
- toggled (high-to-low or low-to-high)
- remain unchanged (that is, reflects the state of the I/O latch)

The action on the pin is based on the value of the mode select bits (CCP1M<3:0>). At the same time, the interrupt flag bit, CCP1IF, is set.

14.3.1 CCP PIN CONFIGURATION

The user must configure the CCP1 pin as an output by clearing the appropriate TRIS bit.

Note:	Clearing the CCP1CON register will force the CCP1 compare output latch (depending on device configuration) to the default low level. This is not the PORTC I/O data latch.

14.3.2 TIMER1/TIMER3 MODE SELECTION

Timer1 and/or Timer3 must be running in Timer mode or Synchronized Counter mode if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation will not work reliably.

14.3.3 SOFTWARE INTERRUPT MODE

When the Generate Software Interrupt mode is chosen (CCP1M<3:0> = 1010), the CCP1 pin is not affected. Only the CCP1IF interrupt flag is affected.

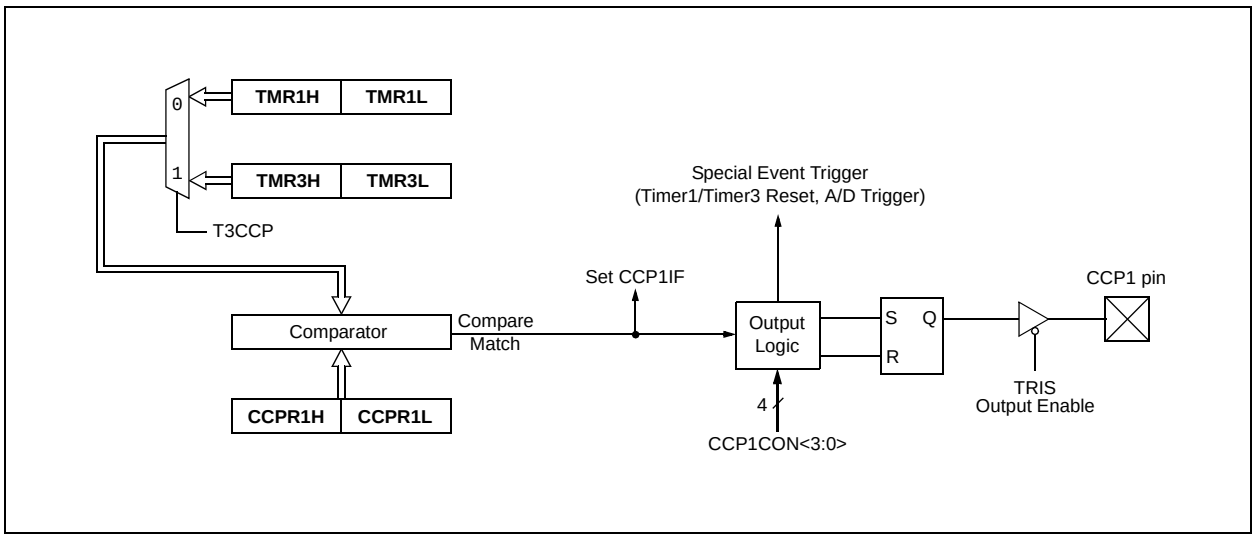
14.3.4 SPECIAL EVENT TRIGGER

The CCP module is equipped with a Special Event Trigger. This is an internal hardware signal generated in Compare mode to trigger actions by other modules. The Special Event Trigger is enabled by selecting the Compare Special Event Trigger mode (CCP1M<3:0> = 1011).

The Special Event Trigger resets the timer register pair for whichever timer resource is currently assigned as the module's time base. This allows the CCPR1 registers to serve as a programmable period register for either timer.

The Special Event Trigger can also start an A/D conversion. In order to do this, the A/D converter must already be enabled.

FIGURE 14-2: COMPARE MODE OPERATION BLOCK DIAGRAM



14.4.4 ENHANCED PWM AUTO-SHUTDOWN MODE

The PWM mode supports an Auto-Shutdown mode that will disable the PWM outputs when an external shutdown event occurs. Auto-Shutdown mode places the PWM output pins into a predetermined state. This mode is used to help prevent the PWM from damaging the application.

The auto-shutdown sources are selected using the ECCPAS<2:0> bits of the ECCPAS register. A shutdown event may be generated by:

- A logic '0' on the INT0 pin
- A logic '1' on a comparator (Cx) output

A shutdown condition is indicated by the ECCPASE (Auto-Shutdown Event Status) bit of the ECCPAS register. If the bit is a '0', the PWM pins are operating normally. If the bit is a '1', the PWM outputs are in the shutdown state.

When a shutdown event occurs, two things happen:

The ECCPASE bit is set to '1'. The ECCPASE will remain set until cleared in firmware or an auto-restart occurs (see **Section 14.4.5 "Auto-Restart Mode"**).

The enabled PWM pins are asynchronously placed in their shutdown states. The PWM output pins are grouped into pairs [P1A/P1C] and [P1B/P1D]. The state of each pin pair is determined by the PSSAC and PSSBD bits of the ECCPAS register. Each pin pair may be placed into one of three states:

- Drive logic '1'
- Drive logic '0'
- Tri-state (high-impedance)

REGISTER 14-2: ECCP1AS: ENHANCED CAPTURE/COMPARE/PWM AUTO-SHUTDOWN CONTROL 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
ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0
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 **ECCPASE:** ECCP Auto-Shutdown Event Status bit
1 = A shutdown event has occurred; ECCP outputs are in shutdown state
0 = ECCP outputs are operating

bit 6-4 **ECCPAS<2:0>:** ECCP Auto-shutdown Source Select bits
000 = Auto-Shutdown is disabled
001 = Comparator C1OUT output is high
010 = Comparator C2OUT output is high
011 = Either Comparator C1OUT or C2OUT is high
100 = VIL on INT0 pin
101 = VIL on INT0 pin or Comparator C1OUT output is high
110 = VIL on INT0 pin or Comparator C2OUT output is high
111 = VIL on INT0 pin or Comparator C1OUT or Comparator C2OUT is high

bit 3-2 **PSSACn:** Pins P1A and P1C Shutdown State Control bits
00 = Drive pins P1A and P1C to '0'
01 = Drive pins P1A and P1C to '1'
10 = Pins P1A and P1C tri-state
11 = Reserved, do not use

bit 1-0 **PSSBDn:** Pins P1B and P1D Shutdown State Control bits
00 = Drive pins P1B and P1D to '0'
01 = Drive pins P1B and P1D to '1'
10 = Pins P1B and P1D tri-state
11 = Reserved, do not use

15.2.1 REGISTERS

The MSSP module has four registers for SPI mode operation. These are:

- SSPCON1 – Control Register
- SSPSTAT – Status register
- SSPBUF – Serial Receive/Transmit Buffer
- SSPSR – Shift Register (Not directly accessible)

SSPCON1 and SSPSTAT are the control and status registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower six bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in and out. SSPBUF provides indirect access to the SSPSR register. SSPBUF is the buffer register to which data bytes are written, and from which data bytes are read.

In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

REGISTER 15-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
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 **SMP:** Sample bit
SPI Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode.

bit 6 **CKE:** SPI Clock Select bit(1)
1 = Transmit occurs on transition from active to Idle clock state
0 = Transmit occurs on transition from Idle to active clock state

bit 5 **D/$\overline{A}$:** Data/$\overline{Address}$ bit
Used in I^2C mode only.

bit 4 **P:** Stop bit
Used in I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3 **S:** Start bit
Used in I^2C mode only.

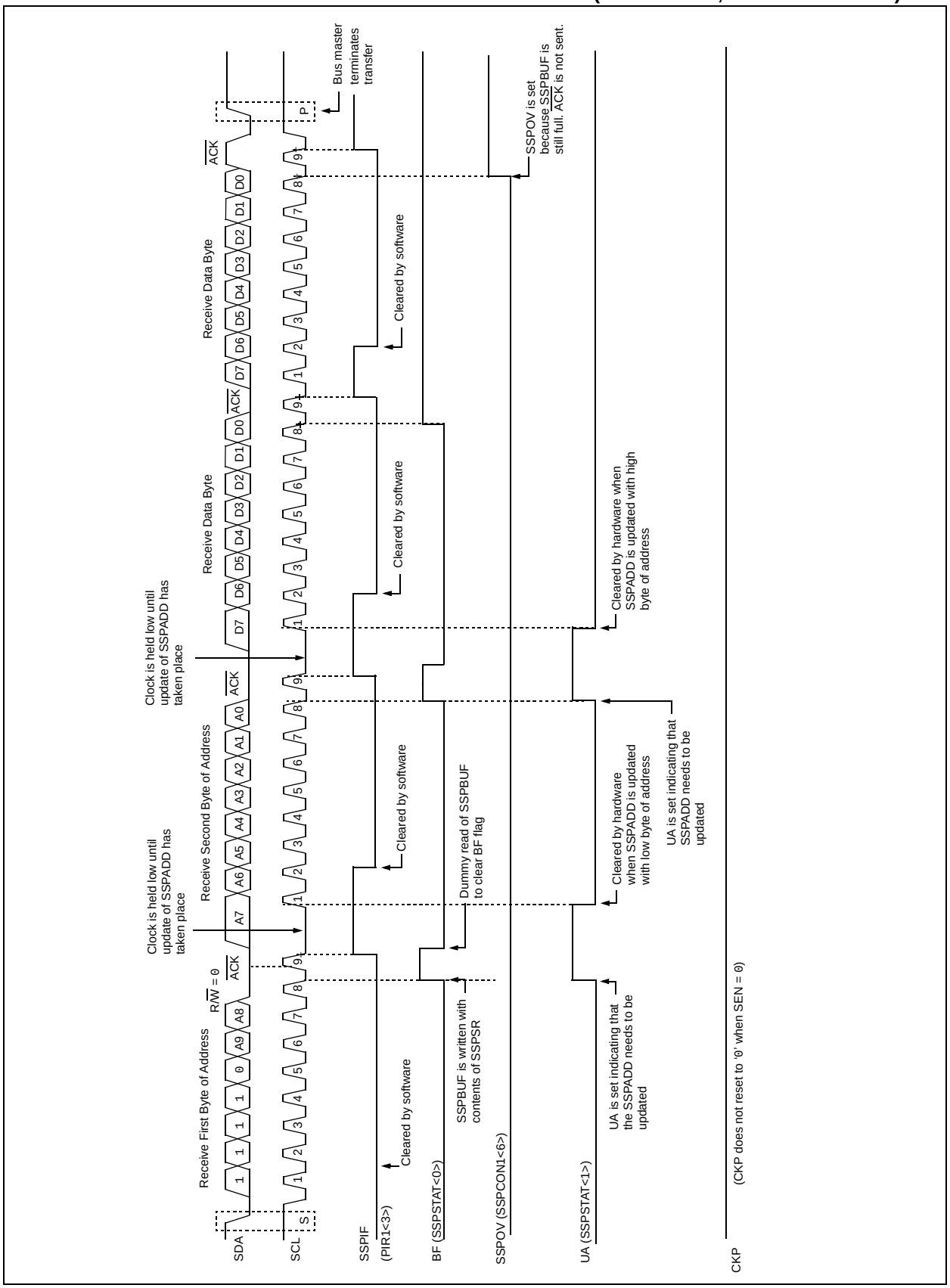
bit 2 **R/$\overline{W}$:** Read/$\overline{Write}$ Information bit
Used in I^2C mode only.

bit 1 **UA:** Update Address bit
Used in I^2C mode only.

bit 0 **BF:** Buffer Full Status bit (Receive mode only)
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty

Note 1: Polarity of clock state is set by the CKP bit of the SSPCON1 register.

FIGURE 15-10: I²C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)



15.3.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON1 and by setting the SSPEN bit. In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I^2C bus may be taken when the P bit is set, or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all I^2C bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

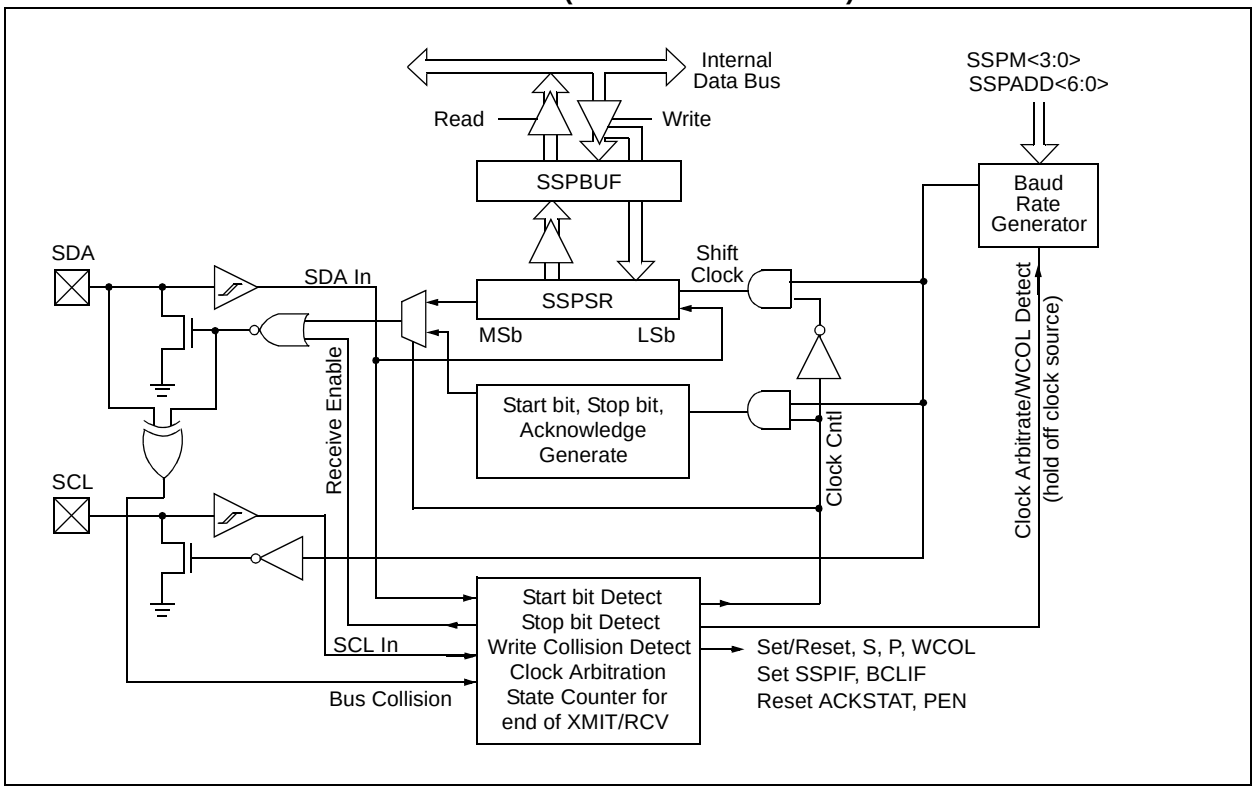
1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the I^2C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDA and SCL.

Note:	The MSSP module, when configured in I^2C Master mode, does not allow queuing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur.

The following events will cause the SSP Interrupt Flag bit, SSPIF, to be set (SSP interrupt, if enabled):

- Start condition
- Stop condition
- Data transfer byte transmitted/received
- Acknowledge transmit
- Repeated Start

FIGURE 15-16: MSSP BLOCK DIAGRAM (I^2C™ MASTER MODE)



15.3.10 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSPBUF register. This action will set the Buffer Full flag bit, BF and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted (see data hold time specification parameter SP106). SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high (see data setup time specification parameter SP107). When the SCL pin is released high, it is held that way for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an ACK bit during the ninth bit time if an address match occurred, or if data was received properly. The status of ACK is written into the ACKDT bit on the falling edge of the ninth clock. If the master receives an Acknowledge, the Acknowledge Status bit, ACKSTAT, is cleared. If not, the bit is set. After the ninth clock, the SSPIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPBUF, leaving SCL low and SDA unchanged (Figure 15-21).

After the write to the SSPBUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/W bit are completed. On the falling edge of the eighth clock, the master will deassert the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the ACK bit is loaded into the ACKSTAT Status bit of the SSPCON2 register. Following the falling edge of the ninth clock transmission of the address, the SSPIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.

15.3.10.1 BF Status Flag

In Transmit mode, the BF bit of the SSPSTAT register is set when the CPU writes to SSPBUF and is cleared when all eight bits are shifted out.

15.3.10.2 WCOL Status Flag

If the user writes the SSPBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL is set and the contents of the buffer are unchanged (the write does not occur).

WCOL must be cleared by software before the next transmission.

15.3.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit of the SSPCON2 register is cleared when the slave has sent an Acknowledge (ACK = 0) and is set when the slave does not Acknowledge (ACK = 1). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

15.3.11 I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN bit of the SSPCON2 register.

Note:	The MSSP module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable, ACKEN bit of the SSPCON2 register.

15.3.11.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when the SSPBUF register is read.

15.3.11.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when eight bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

15.3.11.3 WCOL Status Flag

If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

17.2 ADC Operation

17.2.1 STARTING A CONVERSION

To enable the ADC module, the ADON bit of the ADCON0 register must be set to a '1'. Setting the GO/DONE bit of the ADCON0 register to a '1' will, depending on the ACQT bits of the ADCON2 register, either immediately start the Analog-to-Digital conversion or start an acquisition delay followed by the Analog-to-Digital conversion.

Figure 17-3 shows the operation of the A/D converter after the GO bit has been set and the ACQT<2:0> bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 17-4 shows the operation of the A/D converter after the GO bit has been set and the ACQT<2:0> bits are set to '010' which selects a 4 TAD acquisition time before the conversion starts.

Note:	The GO/DONE bit should not be set in the same instruction that turns on the ADC. Refer to **Section 17.2.9 "A/D Conversion Procedure"**.

FIGURE 17-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)

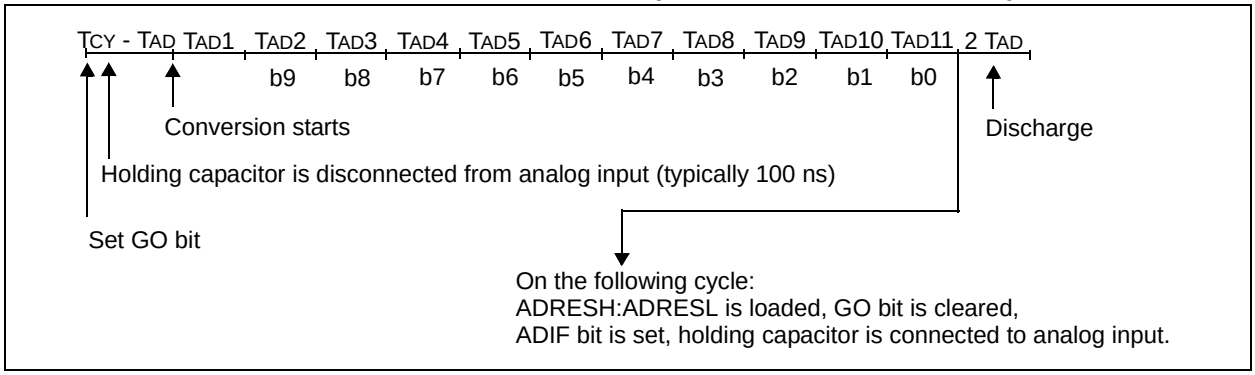
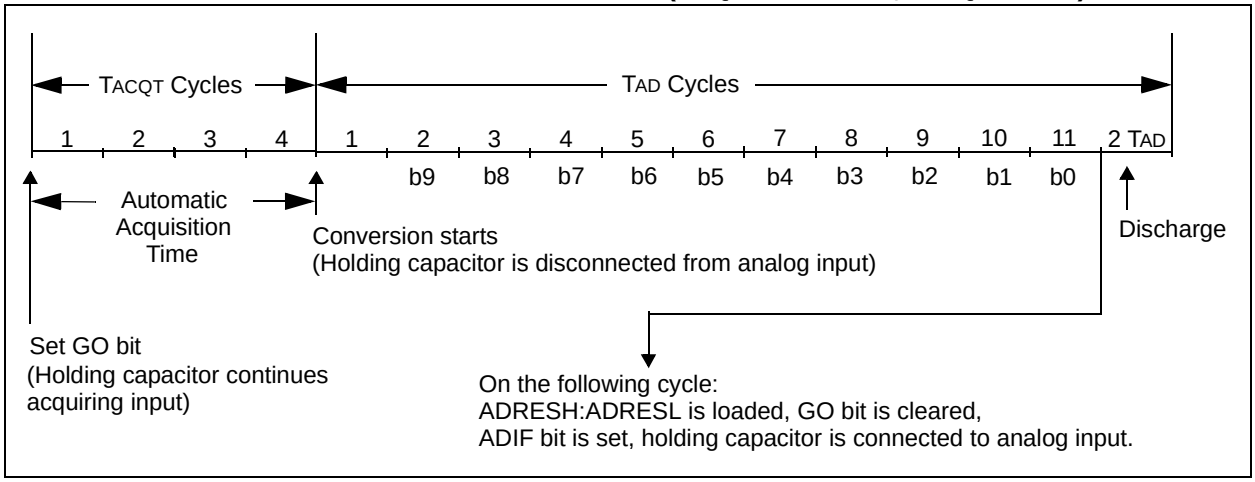


FIGURE 17-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)



REGISTER 18-3: CM2CON1: COMPARATOR 2 CONTROL REGISTER 1

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
MC1OUT	MC2OUT	C1RSEL	C2RSEL	C1HYS	C2HYS	C1SYNC	C2SYNC
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 **MC1OUT:** Mirror Copy of C1OUT bit

bit 6 **MC2OUT:** Mirror Copy of C2OUT bit

bit 5 **C1RSEL:** Comparator C1 Reference Select bit
1 = FVR routed to C1VREF input
0 = CVREF routed to C1VREF input

bit 4 **C2RSEL:** Comparator C2 Reference Select bit
1 = FVR routed to C2VREF input
0 = CVREF routed to C2VREF input

bit 3 **C1HYS:** Comparator C1 Hysteresis Enable bit
1 = Comparator C1 hysteresis enabled
0 = Comparator C1 hysteresis disabled

bit 2 **C2HYS:** Comparator C2 Hysteresis Enable bit
1 = Comparator C2 hysteresis enabled
0 = Comparator C2 hysteresis disabled

bit 1 **C1SYNC:** C1 Output Synchronous Mode bit
1 = C1 output is synchronous to rising edge to TMR1 clock
0 = C1 output is asynchronous

bit 0 **C2SYNC:** C2 Output Synchronous Mode bit
1 = C2 output is synchronous to rising edge to TMR1 clock
0 = C2 output is asynchronous

REGISTER 21-2: REFCON1: REFERENCE CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0
D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	---	D1NSS
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 **D1EN:** DAC 1 Enable bit
0 = DAC 1 is disabled
1 = DAC 1 is enabled

bit 6 **D1LPS:** DAC 1 Low-Power Voltage State Select bit
0 = V_{DAC} = DAC1 Negative reference source selected
1 = V_{DAC} = DAC1 Positive reference source selected

bit 5 **DAC1OE:** DAC 1 Voltage Output Enable bit
1 = DAC 1 voltage level is also outputed on the RC2/AN6/P1D/C12IN2-/CV_{REF}/INT2 pin
0 = DAC 1 voltage level is disconnected from RC2/AN6/P1D/C12IN2-/CV_{REF}/INT2 pin

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

bit 3-2 **D1PSS<1:0>:** DAC 1 Positive Source Select bits
00 = V_{DD}
01 = V_{REF}+
10 = FVR output
11 = Reserved, do not use

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

bit 0 **D1NSS:** DAC1 Negative Source Select bits
0 = V_{SS}
1 = V_{REF}-

REGISTER 21-3: REFCON2: REFERENCE CONTROL REGISTER 2

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
---	---	---	DAC1R4	DAC1R3	DAC1R2	DAC1R1	DAC1R0
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 **Unimplemented:** Read as '0'

bit 4-0 **DAC1R<4:0>:** DAC1 Voltage Output Select bits
V_{OUT} = ((V_{SOURCE}+) - (V_{SOURCE}-))*(DAC1R<4:0>/(2^5)) + V_{SOURCE}-

Note 1: The output select bits are always right justified to ensure that any number of bits can be used without affecting the register layout.

22.0 UNIVERSAL SERIAL BUS (USB)

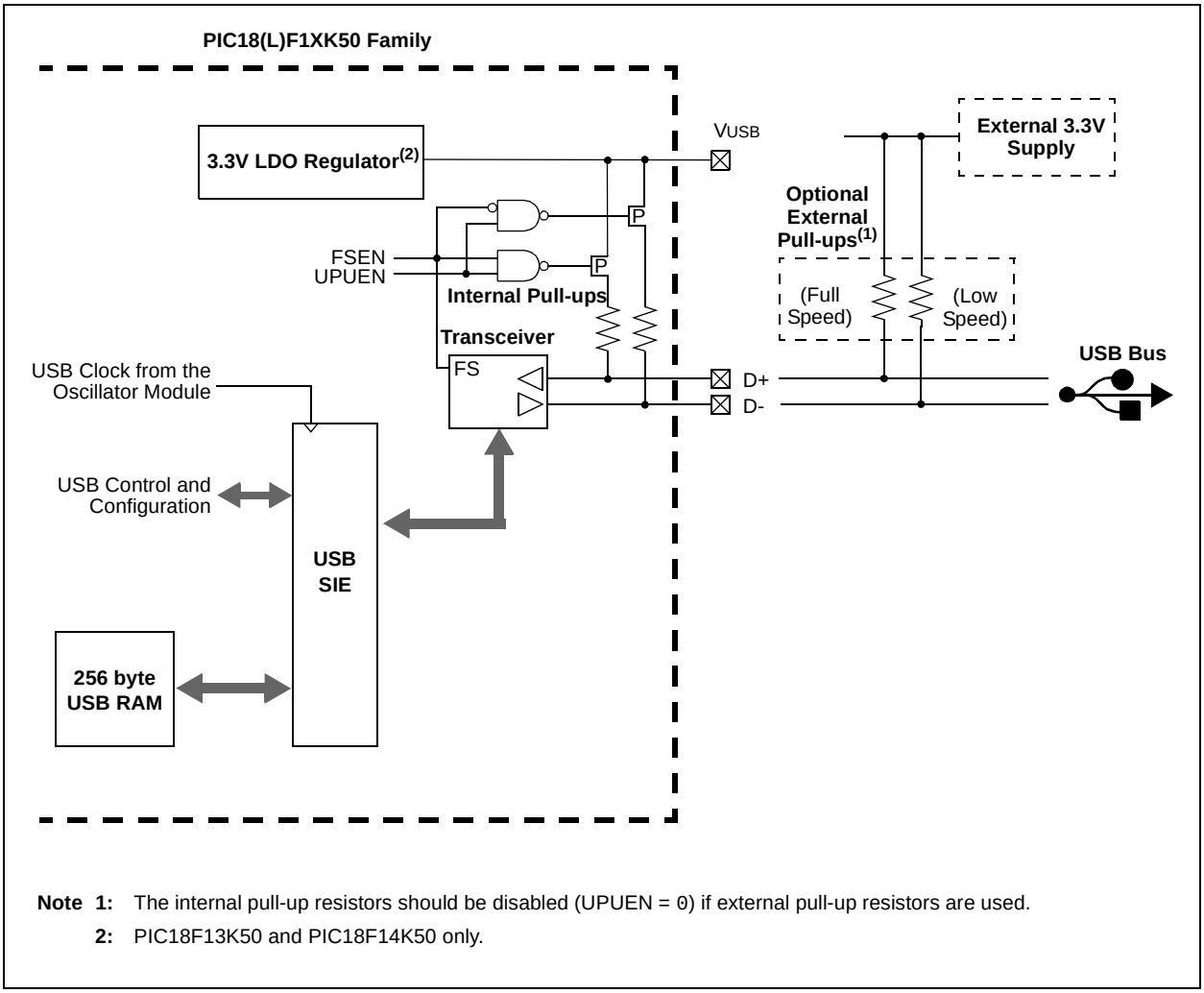
This section describes the details of the USB peripheral. Because of the very specific nature of the module, knowledge of USB is expected. Some high-level USB information is provided in **Section 22.10 "Overview of USB"** only for application design reference. Designers are encouraged to refer to the official specification published by the USB Implementers Forum (USB-IF) for the latest information. USB Specification Revision 2.0 is the most current specification at the time of publication of this document.

22.1 Overview of the USB Peripheral

PIC18(L)F1XK50 devices contain a full-speed and low-speed, compatible USB Serial Interface Engine (SIE) that allows fast communication between any USB host and the PIC microcontroller. The SIE can be interfaced directly to the USB by utilizing the internal transceiver.

Some special hardware features have been included to improve performance. Dual access port memory in the device's data memory space (USB RAM) has been supplied to share direct memory access between the microcontroller core and the SIE. Buffer descriptors are also provided, allowing users to freely program endpoint memory usage within the USB RAM space. Figure 22-1 presents a general overview of the USB peripheral and its features.

FIGURE 22-1: USB PERIPHERAL AND OPTIONS

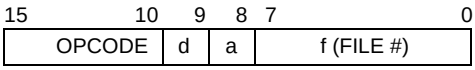


Note 1: The internal pull-up resistors should be disabled (UPUEN = 0) if external pull-up resistors are used.

2: PIC18F13K50 and PIC18F14K50 only.

FIGURE 25-1: GENERAL FORMAT FOR INSTRUCTIONS

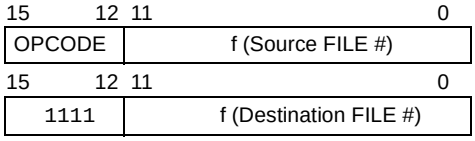
Byte-oriented file register operations — **Example Instruction**



`ADDWF MYREG, W, B`

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

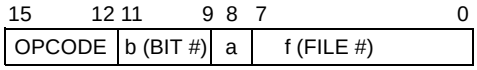
Byte to Byte move operations (2-word)



`MOVFF MYREG1, MYREG2`

f = 12-bit file register address

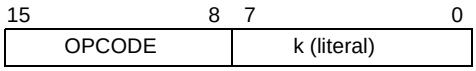
Bit-oriented file register operations



`BSF MYREG, bit, B`

b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

Literal operations

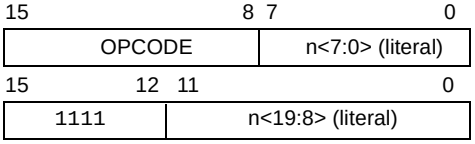


`MOVLW 7Fh`

k = 8-bit immediate value

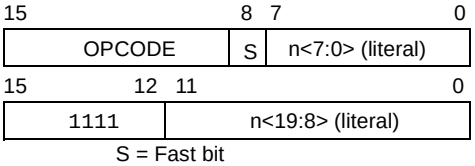
Control operations

CALL, GOTO and Branch operations



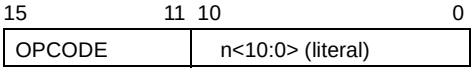
`GOTO Label`

n = 20-bit immediate value

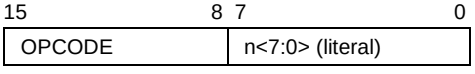


`CALL MYFUNC`

S = Fast bit



`BRA MYFUNC`



`BC MYFUNC`

XORWF **Exclusive OR W with f**

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

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

Operation: (W) .XOR. (f) → dest

Status Affected: N, Z

Encoding:

0001	10da	ffff	ffff

Description: Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the 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

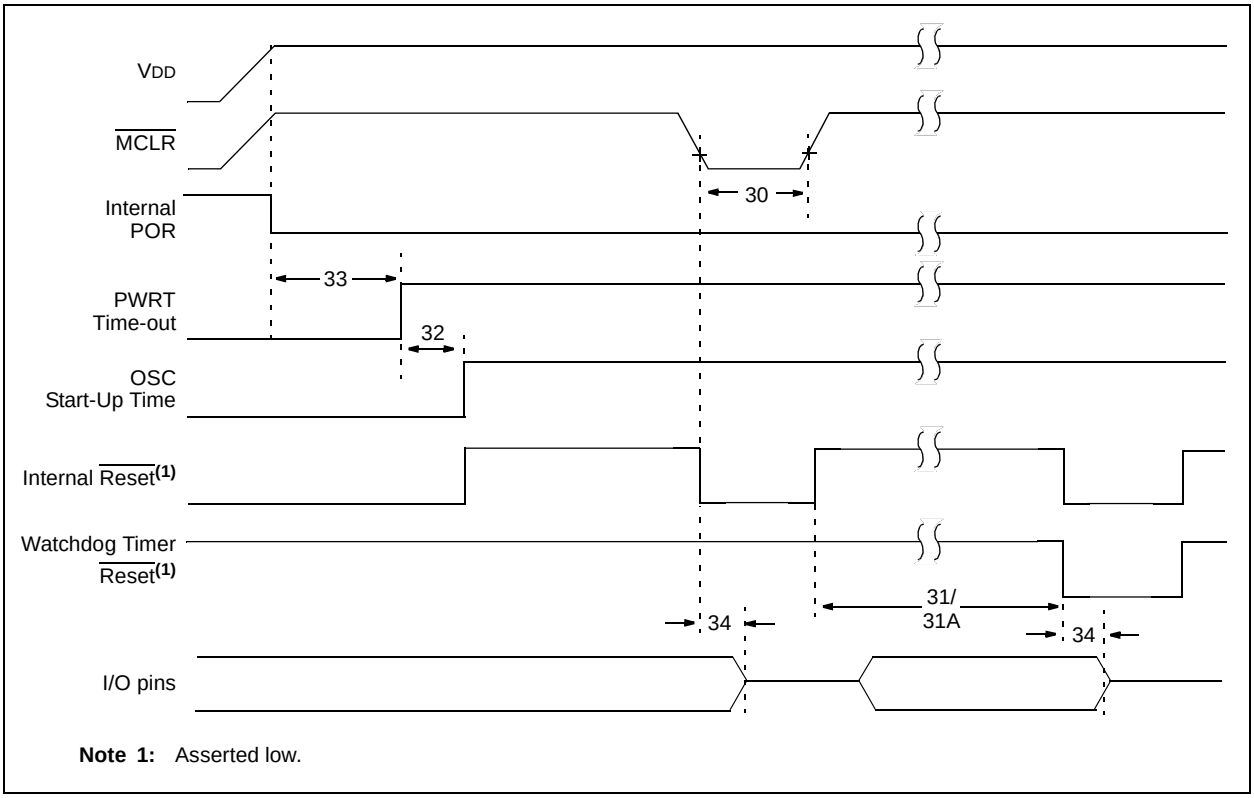
Q Cycle Activity:

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

Example: `XORWF REG, 1, 0`

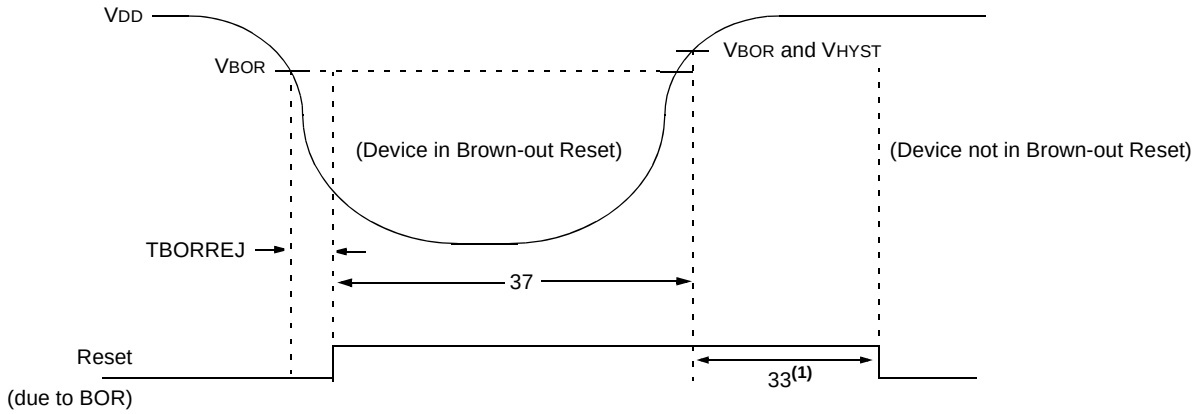
Before Instruction
REG = AFh
W = B5h
After Instruction
REG = 1Ah
W = B5h

FIGURE 27-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



Note 1: Asserted low.

FIGURE 27-9: BROWN-OUT RESET TIMING AND CHARACTERISTICS



Note 1: 64 ms delay only if $\overline{\text{PWRTE}}$ bit in the Configuration Word register is programmed to '0'. 2 ms delay if $\overline{\text{PWRTE}}$ = 0.

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

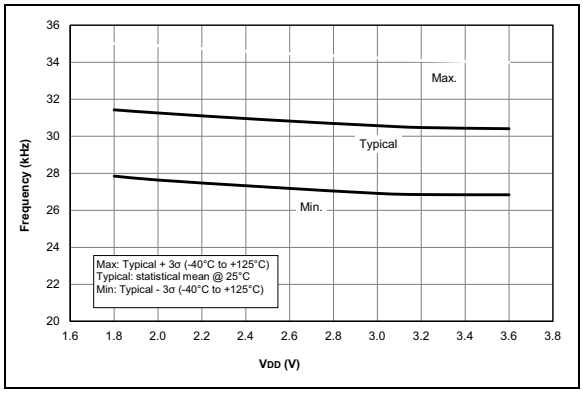


FIGURE 28-61: *LFINTOSC Frequency, Over V_{DD} and Temperature, PIC18LF1XK50 Only.*

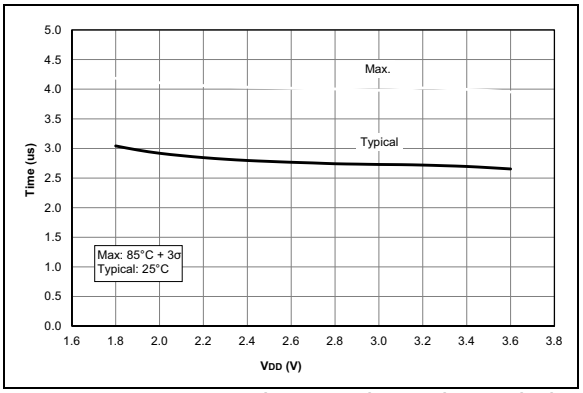


FIGURE 28-64: *Sleep Mode, Wake Period with HFINTOSC Source, PIC18LF1XK50 Only.*

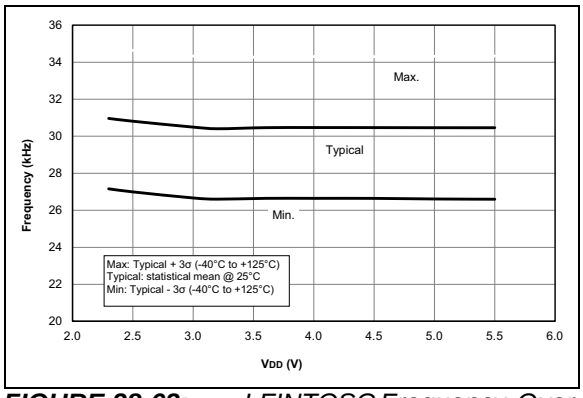


FIGURE 28-62: *LFINTOSC Frequency, Over V_{DD} and Temperature, PIC18F1XK50 Only.*

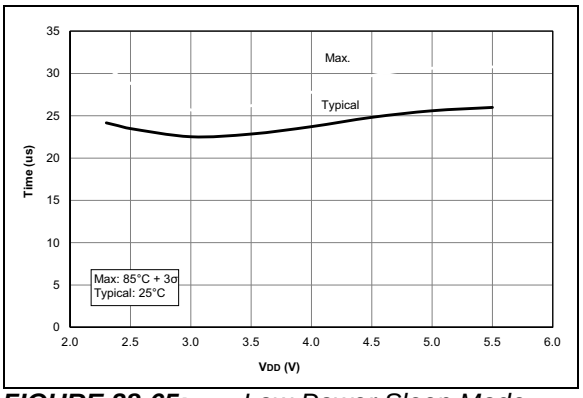


FIGURE 28-65: *Low-Power Sleep Mode, Wake Period with HFINTOSC Source, VREGPM = 1, PIC18F1XK50 Only.*

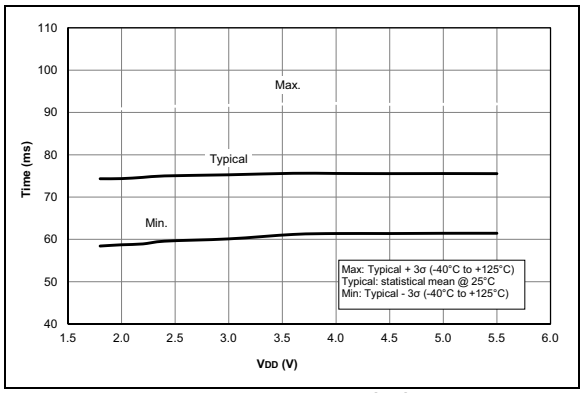


FIGURE 28-63: *PWRT Period.*

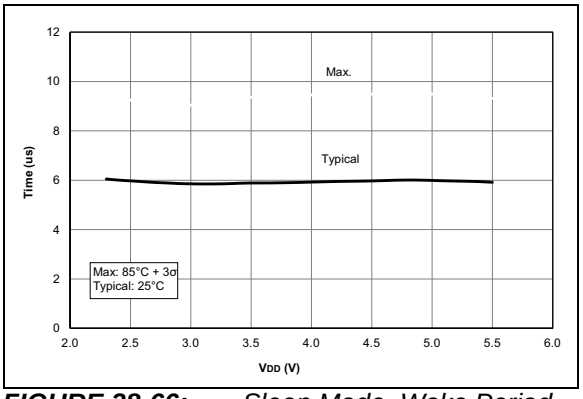


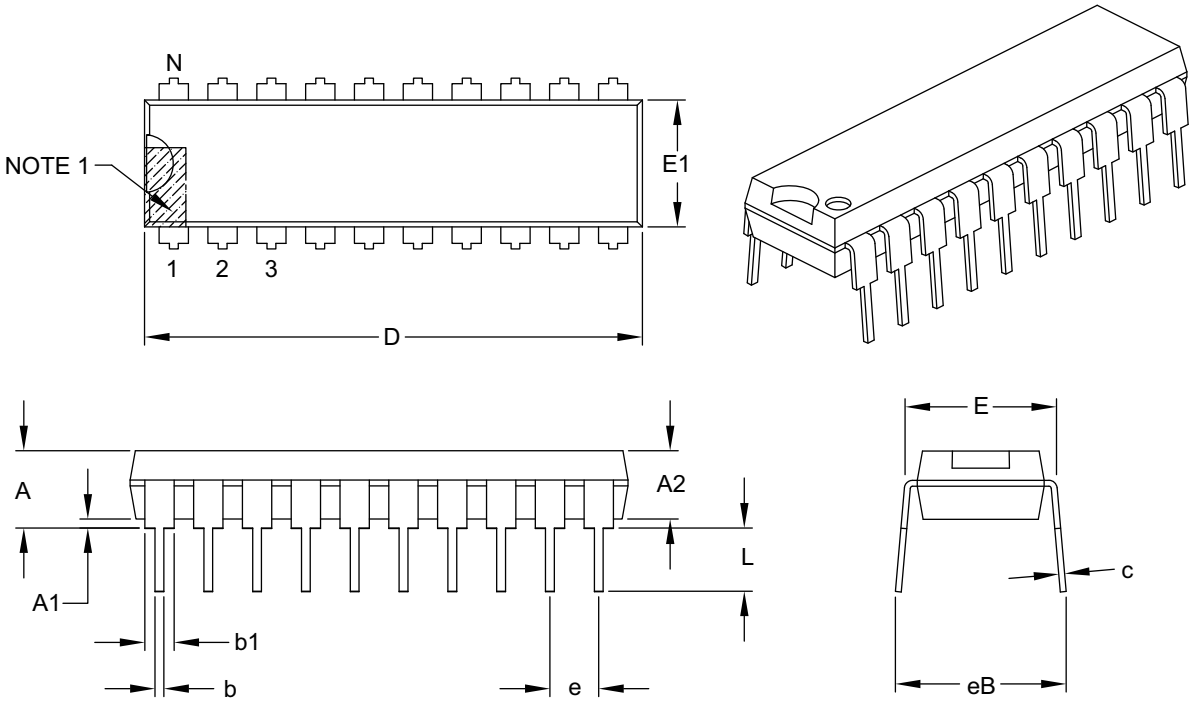
FIGURE 28-66: *Sleep Mode, Wake Period with HFINTOSC Source, VREGPM = 0, PIC18F1XK50 Only.*

29.2 Package Details

The following sections give the technical details of the packages.

20-Lead Plastic Dual In-Line (P) – 300 mil Body [PDIP]

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



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		20	
Pitch	e		.100 BSC	
Top to Seating Plane	A	–	–	.210
Molded Package Thickness	A2	.115	.130	.195
Base to Seating Plane	A1	.015	–	–
Shoulder to Shoulder Width	E	.300	.310	.325
Molded Package Width	E1	.240	.250	.280
Overall Length	D	.980	1.030	1.060
Tip to Seating Plane	L	.115	.130	.150
Lead Thickness	c	.008	.010	.015
Upper Lead Width	b1	.045	.060	.070
Lower Lead Width	b	.014	.018	.022
Overall Row Spacing §	eB	–	–	.430

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

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

Microchip Technology Drawing C04-019B