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	25MHz
Connectivity	UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	16
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 7x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°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/pic18f1320-e-ss

TABLE 1-2: PIC18F1220/1320 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PDIP/ SOIC	SSOP	QFN			
						PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN4/INT0	8	9	9			
RB0				I/O	TTL	Digital I/O.
AN4				I	Analog	Analog input 4.
INT0				I	ST	External interrupt 0.
RB1/AN5/TX/CK/INT1	9	10	10			
RB1				I/O	TTL	Digital I/O.
AN5				I	Analog	Analog input 5.
TX				O	—	EUSART asynchronous transmit.
CK				I/O	ST	EUSART synchronous clock (see related RX/DT).
INT1				I	ST	External interrupt 1.
RB2/P1B/INT2	17	19	23			
RB2				I/O	TTL	Digital I/O.
P1B				O	—	Enhanced CCP1/PWM output.
INT2				I	ST	External interrupt 2.
RB3/CCP1/P1A	18	20	24			
RB3				I/O	TTL	Digital I/O.
CCP1				I/O	ST	Capture 1 input/Compare 1 output/PWM 1 output.
P1A				O	—	Enhanced CCP1/PWM output.
RB4/AN6/RX/DT/KBI0	10	11	12			
RB4				I/O	TTL	Digital I/O.
AN6				I	Analog	Analog input 6.
RX				I	ST	EUSART asynchronous receive.
DT				I/O	ST	EUSART synchronous data (see related TX/CK).
KBI0				I	TTL	Interrupt-on-change pin.
RB5/PGM/KBI1	11	12	13			
RB5				I/O	TTL	Digital I/O.
PGM				I/O	ST	Low-Voltage ICSP™ Programming enable pin.
KBI1				I	TTL	Interrupt-on-change pin.
RB6/PGC/T1OSO/ T13CKI/P1C/KBI2	12	13	15			
RB6				I/O	TTL	Digital I/O.
PGC				I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
T1OSO				O	—	Timer1 oscillator output.
T13CKI				I	ST	Timer1/Timer3 external clock output.
P1C				O	—	Enhanced CCP1/PWM output.
KBI2				I	TTL	Interrupt-on-change pin.
RB7/PGD/T1OSI/ P1D/KBI3	13	14	16			
RB7				I/O	TTL	Digital I/O.
PGD				I/O	ST	In-Circuit Debugger and ICSP programming data pin.
T1OSI				I	CMOS	Timer1 oscillator input.
P1D				O	—	Enhanced CCP1/PWM output.
KBI3				I	TTL	Interrupt-on-change pin.
VSS	5	5, 6	3, 5	P	—	Ground reference for logic and I/O pins.
VDD	14	15, 16	17, 19	P	—	Positive supply for logic and I/O pins.
NC	—	—	18	—	—	No connect.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels; I = Input
O = Output; P = Power
OD = Open-drain (no P diode to VDD)

2.7.2 OSCILLATOR TRANSITIONS

The PIC18F1220/1320 devices contain circuitry to prevent clocking "glitches" when switching between clock sources. A short pause in the system clock occurs during the clock switch. The length of this pause is between eight and nine clock periods of the new clock source. This ensures that the new clock source is stable and that its pulse width will not be less than the shortest pulse width of the two clock sources.

Clock transitions are discussed in greater detail in **Section 3.1.2 "Entering Power Managed Modes"**.

2.8 Effects of Power Managed Modes on the Various Clock Sources

When the device executes a `SLEEP` instruction, the system is switched to one of the power managed modes, depending on the state of the IDLEN and SCS1:SCS0 bits of the OSCCON register. See **Section 3.0 "Power Managed Modes"** for details.

When PRI_IDLE mode is selected, the designated primary oscillator continues to run without interruption. For all other power managed modes, the oscillator using the OSC1 pin is disabled. The OSC1 pin (and OSC2 pin, if used by the oscillator) will stop oscillating.

In Secondary Clock modes (SEC_RUN and SEC_IDLE), the Timer1 oscillator is operating and providing the system clock. The Timer1 oscillator may also run in all power managed modes if required to clock Timer1 or Timer3.

In Internal Oscillator modes (RC_RUN and RC_IDLE), the internal oscillator block provides the system clock source. The INTRC output can be used directly to provide the system clock and may be enabled to support various special features, regardless of the power managed mode (see **Section 19.2 "Watchdog Timer (WDT)"** through **Section 19.4 "Fail-Safe Clock Monitor"**). The INTOSC output at 8 MHz may be used directly to clock the system, or may be divided down first. The INTOSC output is disabled if the system clock is provided directly from the INTRC output.

If the Sleep mode is selected, all clock sources are stopped. Since all the transistor switching currents have been stopped, Sleep mode achieves the lowest current consumption of the device (only leakage currents).

Enabling any on-chip feature that will operate during Sleep will increase the current consumed during Sleep. The INTRC is required to support WDT operation. The Timer1 oscillator may be operating to support a real-time clock. Other features may be operating that do not require a system clock source (i.e., INTn pins, A/D conversions and others).

2.9 Power-up Delays

Power-up delays are controlled by two timers, so that no external Reset circuitry is required for most applications. The delays ensure that the device is kept in Reset until the device power supply is stable under normal circumstances and the primary clock is operating and stable. For additional information on power-up delays, see Sections 4.1 through 4.5.

The first timer is the Power-up Timer (PWRT), which provides a fixed delay on power-up (parameter 33, Table 22-8) if enabled in Configuration Register 2L. The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable (LP, XT and HS modes). The OST does this by counting 1024 oscillator cycles before allowing the oscillator to clock the device.

When the HSPLL Oscillator mode is selected, the device is kept in Reset for an additional 2 ms following the HS mode OST delay, so the PLL can lock to the incoming clock frequency.

There is a delay of 5 to 10 μs following POR while the controller becomes ready to execute instructions. This delay runs concurrently with any other delays. This may be the only delay that occurs when any of the EC, RC or INTIO modes are used as the primary clock source.

TABLE 2-3: OSC1 AND OSC2 PIN STATES IN SLEEP MODE

Oscillator Mode	OSC1 Pin	OSC2 Pin
RC, INTIO1	Floating, external resistor should pull high	At logic low (clock/4 output)
RCIO, INTIO2	Floating, external resistor should pull high	Configured as PORTA, bit 6
ECIO	Floating, pulled by external clock	Configured as PORTA, bit 6
EC	Floating, pulled by external clock	At logic low (clock/4 output)
LP, XT and HS	Feedback inverter disabled at quiescent voltage level	Feedback inverter disabled at quiescent voltage level

Note: See Table 4-1 in **Section 4.0 "Reset"** for time-outs due to Sleep and $\overline{\text{MCLR}}$ Reset.

3.3.1 PRI_IDLE MODE

This mode is unique among the three Low-Power Idle modes, in that it does not disable the primary system clock. For timing sensitive applications, this allows for the fastest resumption of device operation with its more accurate primary clock source, since the clock source does not have to “warm up” or transition from another oscillator.

PRI_IDLE mode is entered by setting the IDLEN bit, clearing the SCS bits and executing a SLEEP instruction. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source specified in Configuration Register 1H. The OSTS bit remains set in PRI_IDLE mode (see Figure 3-3).

When a wake event occurs, the CPU is clocked from the primary clock source. A delay of approximately 10 μs is required between the wake event and code execution starts. This is required to allow the CPU to become ready to execute instructions. After the wake-up, the OSTS bit remains set. The IDLEN and SCS bits are not affected by the wake-up (see Figure 3-4).

FIGURE 3-3: TRANSITION TIMING TO PRI_IDLE MODE

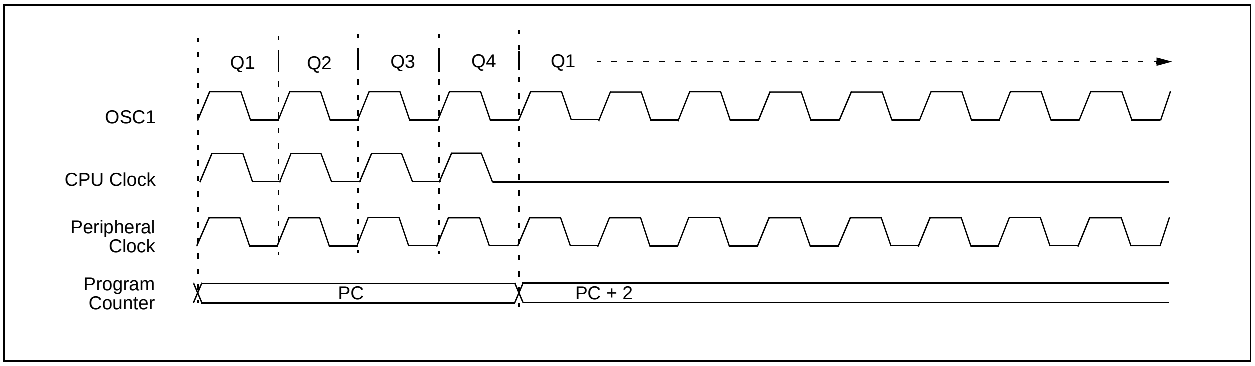
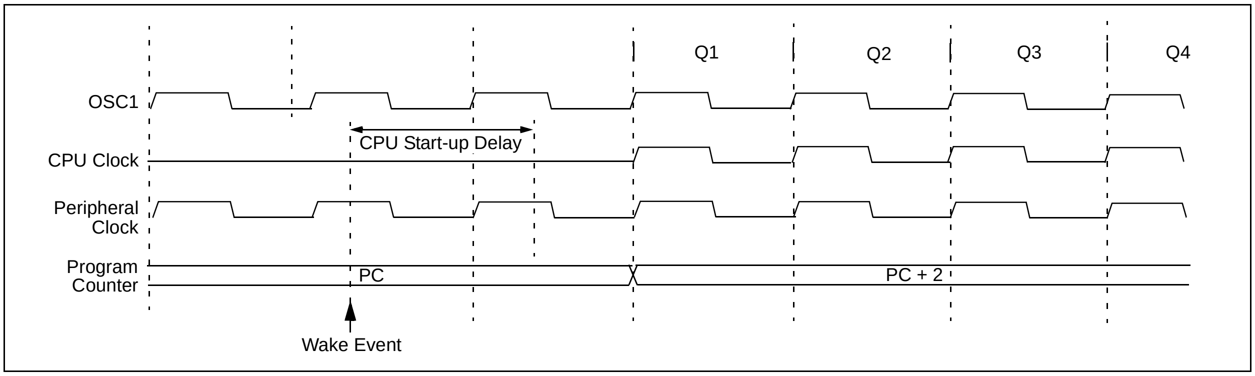


FIGURE 3-4: TRANSITION TIMING FOR WAKE FROM PRI_IDLE MODE



3.4 Run Modes

If the IDLEN bit is clear when a SLEEP instruction is executed, the CPU and peripherals are both clocked from the source selected using the SCS1:SCS0 bits. While these operating modes may not afford the power conservation of Idle or Sleep modes, they do allow the device to continue executing instructions by using a lower frequency clock source. RC_RUN mode also offers the possibility of executing code at a frequency greater than the primary clock.

Wake-up from a power managed Run mode can be triggered by an interrupt, or any Reset, to return to full-power operation. As the CPU is executing code in Run modes, several additional exits from Run modes are possible. They include exit to Sleep mode, exit to a corresponding Idle mode and exit by executing a RESET instruction. While the device is in any of the power managed Run modes, a WDT time-out will result in a WDT Reset.

3.4.1 PRI_RUN MODE

The PRI_RUN mode is the normal full-power execution mode. If the SLEEP instruction is never executed, the microcontroller operates in this mode (a SLEEP instruction is executed to enter all other power managed modes). All other power managed modes exit to PRI_RUN mode when an interrupt or WDT time-out occur.

There is no entry to PRI_RUN mode. The OSTS bit is set. The IOFS bit may be set if the internal oscillator block is the primary clock source (see **Section 2.7.1 "Oscillator Control Register"**).

3.4.2 SEC_RUN MODE

The SEC_RUN mode is the compatible mode to the "clock switching" feature offered in other PIC18 devices. In this mode, the CPU and peripherals are clocked from the Timer1 oscillator. This gives users the option of lower power consumption while still using a high accuracy clock source.

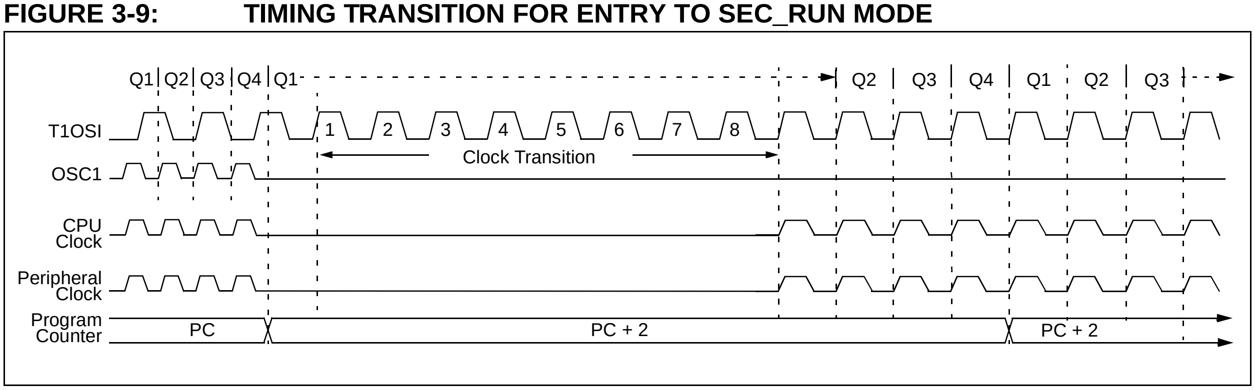
SEC_RUN mode is entered by clearing the IDLEN bit, setting SCS1:SCS0 = 01 and executing a SLEEP instruction. The system clock source is switched to the Timer1 oscillator (see Figure 3-9), the primary oscillator is shut down, the T1RUN bit (T1CON<6>) is set and the OSTS bit is cleared.

> **Note:** The Timer1 oscillator should already be running prior to entering SEC_RUN mode. If the T1OSCEN bit is not set when the SLEEP instruction is executed, the SLEEP instruction will be ignored and entry to SEC_RUN mode will not occur. If the Timer1 oscillator is enabled, but not yet running, system clocks will be delayed until the oscillator has started; in such situations, initial oscillator operation is far from stable and unpredictable operation may result.

When a wake event occurs, the peripherals and CPU continue to be clocked from the Timer1 oscillator while the primary clock is started. When the primary clock becomes ready, a clock switchback to the primary clock occurs (see Figure 3-6). When the clock switch is complete, the T1RUN bit is cleared, the OSTS bit is set and the primary clock is providing the system clock. The IDLEN and SCS bits are not affected by the wake-up. The Timer1 oscillator continues to run.

Firmware can force an exit from SEC_RUN mode. By clearing the T1OSCEN bit (T1CON<3>), an exit from SEC_RUN back to normal full-power operation is triggered. The Timer1 oscillator will continue to run and provide the system clock, even though the T1OSCEN bit is cleared. The primary clock is started. When the primary clock becomes ready, a clock switchback to the primary clock occurs (see Figure 3-6). When the clock switch is complete, the Timer1 oscillator is disabled, the T1RUN bit is cleared, the OSTS bit is set and the primary clock is providing the system clock. The IDLEN and SCS bits are not affected by the wake-up.

FIGURE 3-9: TIMING TRANSITION FOR ENTRY TO SEC_RUN MODE



3.4.4 EXIT TO IDLE MODE

An exit from a power managed Run mode to its corresponding Idle mode is executed by setting the IDLEN bit and executing a SLEEP instruction. The CPU is halted at the beginning of the instruction following the SLEEP instruction. There are no changes to any of the clock source status bits (OSTS, IOFS or T1RUN). While the CPU is halted, the peripherals continue to be clocked from the previously selected clock source.

3.4.5 EXIT TO SLEEP MODE

An exit from a power managed Run mode to Sleep mode is executed by clearing the IDLEN and SCS1:SCS0 bits and executing a SLEEP instruction. The code is no different than the method used to invoke Sleep mode from the normal operating (full-power) mode.

The primary clock and internal oscillator block are disabled. The INTRC will continue to operate if the WDT is enabled. The Timer1 oscillator will continue to run, if enabled in the T1CON register (Register 12-1). All clock source Status bits are cleared (OSTS, IOFS and T1RUN).

3.5 Wake from Power Managed Modes

An exit from any of the power managed modes is triggered by an interrupt, a Reset or a WDT time-out. This section discusses the triggers that cause exits from power managed modes. The clocking subsystem actions are discussed in each of the power managed modes (see Sections 3.2 through 3.4).

Note:	If application code is timing sensitive, it should wait for the OSTS bit to become set before continuing. Use the interval during the low-power exit sequence (before OSTS is set) to perform timing insensitive “housekeeping” tasks.

Device behavior during Low-Power mode exits is summarized in Table 3-3.

3.5.1 EXIT BY INTERRUPT

Any of the available interrupt sources can cause the device to exit a power managed mode and resume full-power operation. To enable this functionality, an interrupt source must be enabled by setting its enable bit in one of the INTCON or PIE registers. The exit sequence is initiated when the corresponding interrupt flag bit is set. On all exits from Low-Power mode by interrupt, code execution branches to the interrupt vector if the GIE/GIEH bit (INTCON<7>) is set. Otherwise, code execution continues or resumes without branching (see **Section 9.0 “Interrupts”**).

3.6.1 EXAMPLE – EUSART

An adjustment may be indicated when the EUSART begins to generate framing errors, or receives data with errors while in Asynchronous mode. Framing errors indicate that the system clock frequency is too high – try decrementing the value in the OSCTUNE register to reduce the system clock frequency. Errors in data may suggest that the system clock speed is too low – increment OSCTUNE.

3.6.2 EXAMPLE – TIMERS

This technique compares system 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 Timer1 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 greater than expected, then the internal oscillator block is running too fast – decrement OSCTUNE.

3.6.3 EXAMPLE – CCP 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 – decrement OSCTUNE. If the measured time is much less than the calculated time, the internal oscillator block is running too slow–increment OSCTUNE.

REGISTER 6-1: EECON1: EEPROM CONTROL 1 REGISTER

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR(1)	WREN	WR	RD
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
S = Bit can only be set	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	HC = Bit is cleared by hardware

bit 7 **EEPGD:** Flash Program or Data EEPROM Memory Select bit
1 = Access program Flash memory
0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
1 = Accesses Configuration, User ID and Device ID Registers
0 = Accesses Flash Program or data EEPROM Memory

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **FREE:** Flash Row Erase Enable bit
1 = Erase the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation – TBLPTR<5:0> are ignored)
0 = Perform write only

bit 3 **WRERR:** EEPROM Error Flag bit(1)
1 = A write operation was prematurely terminated (any Reset during self-timed programming)
0 = The write operation completed normally

bit 2 **WREN:** Program/Erase Enable bit
1 = Allows program/erase cycles
0 = Inhibits programming/erasing of program Flash and data EEPROM

bit 1 **WR:** Write Control bit
1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle. (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
0 = Write cycle completed

bit 0 **RD:** Read Control bit
1 = Initiates a memory read (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1.)
0 = Read completed

Note 1: When a WRERR occurs, the EEPGD and CFGS bits are not cleared. This allows tracing of the error condition.

6.4 Erasing Flash Program Memory

The minimum erase block size is 32 words or 64 bytes under firmware control. Only through the use of an external programmer, or through ICSP control, can larger blocks of program memory be bulk erased. Word erase in Flash memory is not supported.

When initiating an erase sequence from the micro-controller itself, a block of 64 bytes of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. TBLPTR<5:0> are ignored.

The EECON1 register commands the erase operation. The EEPGD bit must be set to point to the Flash program memory. The CFGS bit must be clear to access program Flash and data EEPROM memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation. The WR bit is set as part of the required instruction sequence (as shown in Example 6-2) and starts the actual erase operation. It is not necessary to load the TABLAT register with any data as it is ignored.

For protection, the write initiate sequence using EECON2 must be used.

A long write is necessary for erasing the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

6.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

The sequence of events for erasing a block of internal program memory location is:

1. Load Table Pointer with address of row being erased.
2. Set the EECON1 register for the erase operation:
   - set EEPGD bit to point to program memory;
   - clear the CFGS bit to access program memory;
   - set WREN bit to enable writes;
   - set FREE bit to enable the erase.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write AAh to EECON2.
6. Set the WR bit. This will begin the row erase cycle.
7. The CPU will stall for duration of the erase (about 2 ms using internal timer).
8. Execute a NOP.
9. Re-enable interrupts.

EXAMPLE 6-2: ERASING A FLASH PROGRAM MEMORY ROW

```
                    MOVLW   CODE_ADDR_UPPER     ; load TBLPTR with the base
                    MOVWF   TBLPTRU             ; address of the memory block
                    MOVLW   CODE_ADDR_HIGH
                    MOVWF   TBLPTRH
                    MOVLW   CODE_ADDR_LOW
                    MOVWF   TBLPTRL
        ERASE_ROW
                    BSF     EECON1, EEPGD       ; point to FLASH program memory
                    BSF     EECON1, WREN        ; enable write to memory
                    BSF     EECON1, FREE        ; enable Row Erase operation
                    BCF     INTCON, GIE         ; disable interrupts
                    MOVLW   55h
                    MOVWF   EECON2              ; write 55H
    Required        MOVLW   AAh
    Sequence        MOVWF   EECON2              ; write AAH
                    BSF     EECON1, WR          ; start erase (CPU stall)
                    NOP
                    BSF     INTCON, GIE         ; re-enable interrupts
```

7.7 Operation During Code-Protect

Data EEPROM memory has its own code-protect bits in Configuration Words. External read and write operations are disabled if either of these mechanisms are enabled.

The microcontroller itself can both read and write to the internal data EEPROM, regardless of the state of the code-protect Configuration bit. Refer to **Section 19.0 "Special Features of the CPU"** for additional information.

7.8 Using the Data EEPROM

The data EEPROM is a high endurance, byte addressable array that has been optimized for the storage of frequently changing information (e.g., program variables or other data that are updated often). Frequently changing values will typically be updated more often than specification D124. If this is not the case, an array refresh must be performed. For this reason, variables that change infrequently (such as constants, IDs, calibration, etc.) should be stored in Flash program memory.

A simple data EEPROM refresh routine is shown in Example 7-3.

Note:	If data EEPROM is only used to store constants and/or data that changes rarely, an array refresh is likely not required. See specification D124.

EXAMPLE 7-3: DATA EEPROM REFRESH ROUTINE

```
        CLRF    EEADR              ; Start at address 0
        BCF     EECON1, CFGS       ; Set for memory
        BCF     EECON1, EEPGD      ; Set for Data EEPROM
        BCF     INTCON, GIE        ; Disable interrupts
        BSF     EECON1, WREN       ; Enable writes
Loop                               ; Loop to refresh array
        BSF     EECON1, RD         ; Read current address
        MOVLW   55h                ;
        MOVWF   EECON2             ; Write 55h
        MOVLW   AAh                ;
        MOVWF   EECON2             ; Write AAh
        BSF     EECON1, WR         ; Set WR bit to begin write
        BTFSC   EECON1, WR         ; Wait for write to complete
        BRA     $-2
        INCFSZ  EEADR, F           ; Increment address
        BRA     Loop               ; Not zero, do it again

        BCF     EECON1, WREN       ; Disable writes
        BSF     INTCON, GIE        ; Enable interrupts
```

TABLE 7-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
EEADR	EEPROM Address Register								0000 0000	0000 0000
EEDATA	EEPROM Data Register								0000 0000	0000 0000
EECON2	EEPROM Control Register 2 (not a physical register)								—	—
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	uu-0 u000
IPR2	OSCFIP	—	—	EEIP	—	LVDIP	TMR3IP	—	1--1 -11-	1--1 -11-
PIR2	OSCFIF	—	—	EEIF	—	LVDIF	TMR3IF	—	0--0 -00-	0--0 -00-
PIE2	OSCFIE	—	—	EEIE	—	LVDIE	TMR3IE	—	0--0 -00-	0--0 -00-

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used during Flash/EEPROM access.

REGISTER 9-5: PIR2: PERIPHERAL INTERRUPT REQUEST REGISTER 2

R/W-0/0	U-0	U-0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	U-0
OSCFIF	—	—	EEIF	—	LVDIF	TMR3IF	—
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7 **OSCFIF:** Oscillator Fail Interrupt Flag bit
1 = System oscillator failed, clock input has changed to INTOSC (must be cleared in software)
0 = System clock operating

bit 6-5 **Unimplemented:** Read as ‘0’

bit 4 **EEIF:** Data EEPROM/Flash Write Operation Interrupt Flag bit
1 = The write operation is complete (must be cleared in software)
0 = The write operation is not complete or has not been started

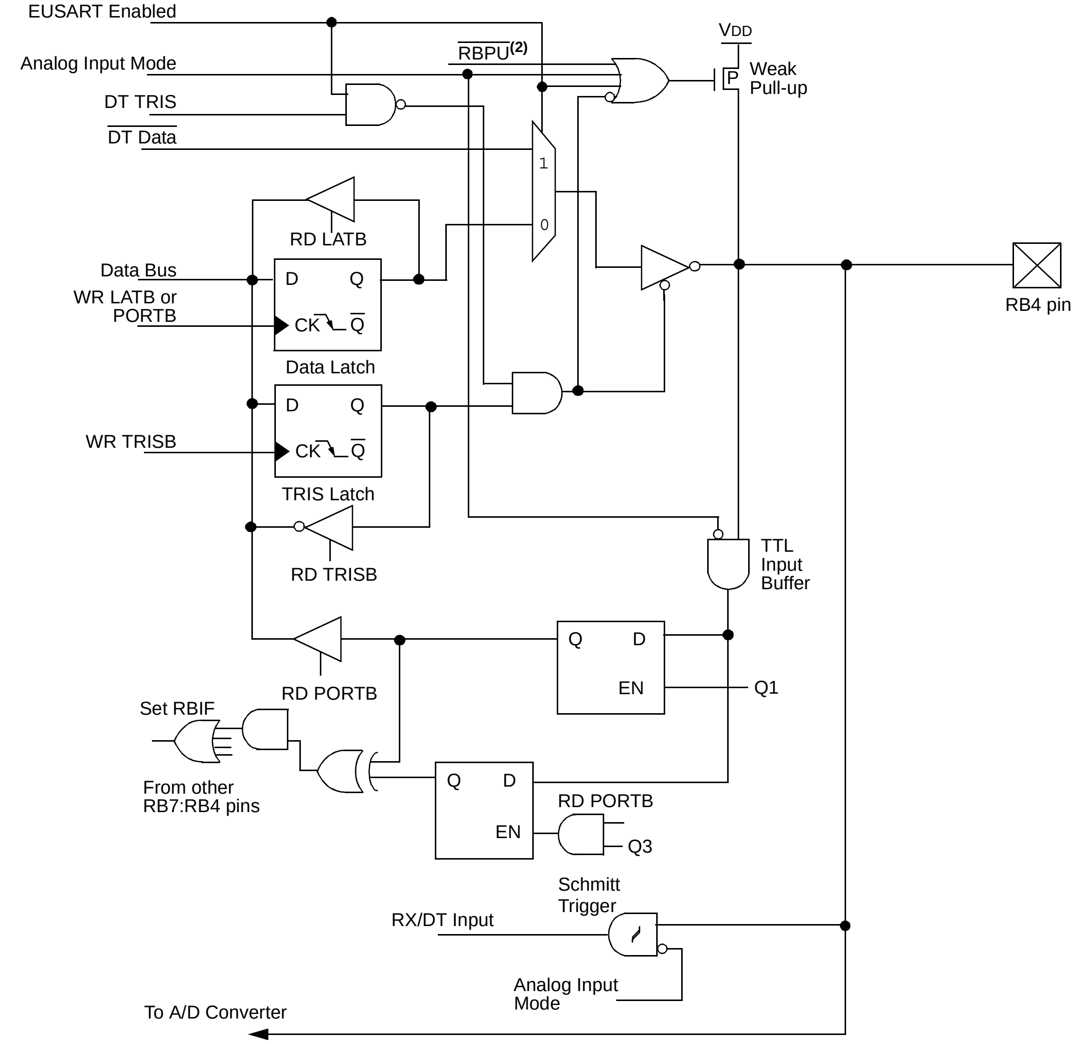
bit 3 **Unimplemented:** Read as ‘0’

bit 2 **LVDIF:** Low-Voltage Detect Interrupt Flag bit
1 = A low-voltage condition occurred (must be cleared in software)
0 = The device voltage is above the Low-Voltage Detect trip point

bit 1 **TMR3IF:** TMR3 Overflow Interrupt Flag bit
1 = TMR3 register overflowed (must be cleared in software)
0 = TMR3 register did not overflow

bit 0 **Unimplemented:** Read as ‘0’

FIGURE 10-11: BLOCK DIAGRAM OF RB4/AN6/RX/DT/KBI0 PIN



Note 1: I/O pins have diode protection to VDD and VSS.

2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (INTCON2<7>).

12.7 Using Timer1 as a Real-Time Clock

Adding an external LP oscillator to Timer1 (such as the one described in **Section 12.2 "Timer1 Oscillator"**, above), gives users the option to include RTC functionality to their applications. This is accomplished with an inexpensive watch crystal to provide an accurate time base and several lines of application code to calculate the time. When operating in Sleep mode and using a battery or supercapacitor as a power source, it can completely eliminate the need for a separate RTC device and battery backup.

The application code routine, `RTCisr`, shown in Example 12-1, demonstrates a simple method to increment a counter at one-second intervals using an Interrupt Service Routine. Incrementing the TMR1 register pair to overflow, triggers the interrupt and calls the routine, which increments the seconds counter by one; additional counters for minutes and hours are incremented as the previous counter overflow.

Since the register pair is 16 bits wide, counting up to overflow the register directly from a 32.768 kHz clock would take two seconds. To force the overflow at the required one-second intervals, it is necessary to preload it; the simplest method is to set the MSb of TMR1H with a BSF instruction. Note that the TMR1L register is never preloaded or altered; doing so may introduce cumulative error over many cycles.

For this method to be accurate, Timer1 must operate in Asynchronous mode and the Timer1 overflow interrupt must be enabled (PIE1<0> = 1), as shown in the routine, `RTCinit`. The Timer1 oscillator must also be enabled and running at all times.

EXAMPLE 12-1: IMPLEMENTING A REAL-TIME CLOCK USING A TIMER1 INTERRUPT SERVICE

```
RTCinit
           MOVLW   0x80                ; Preload TMR1 register pair
           MOVWF   TMR1H               ; for 1 second overflow
           CLRF    TMR1L
           MOVLW   b'00001111'         ; Configure for external clock,
           MOVWF   T1OSC               ; Asynchronous operation, external oscillator
           CLRF    secs                ; Initialize timekeeping registers
           CLRF    mins                ;
           MOVLW   .12
           MOVWF   hours
           BSF     PIE1, TMR1IE        ; Enable Timer1 interrupt
           RETURN
RTCisr
           BSF     TMR1H, 7            ; Preload for 1 sec overflow
           BCF     PIR1, TMR1IF        ; Clear interrupt flag
           INCF    secs, F             ; Increment seconds
           MOVLW   .59                 ; 60 seconds elapsed?
           CPFSGT  secs
           RETURN                      ; No, done
           CLRF    secs                ; Clear seconds
           INCF    mins, F             ; Increment minutes
           MOVLW   .59                 ; 60 minutes elapsed?
           CPFSGT  mins
           RETURN                      ; No, done
           CLRF    mins                ; clear minutes
           INCF    hours, F            ; Increment hours
           MOVLW   .23                 ; 24 hours elapsed?
           CPFSGT  hours
           RETURN                      ; No, done
           MOVLW   .01                 ; Reset hours to 1
           MOVWF   hours
           RETURN                      ; Done
```

TABLE 15-5: REGISTERS ASSOCIATED WITH ENHANCED PWM AND TIMER2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
RCON	IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	0--1 11qq	0--q qquu
PIR1	—	ADIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF	-000 -000	-000 -000
PIE1	—	ADIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE	-000 -000	-000 -000
IPR1	—	ADIP	RCIP	TXIP	—	CCP1IP	TMR2IP	TMR1IP	-111 -111	-111 -111
TMR2	Timer2 Module Register								0000 0000	0000 0000
PR2	Timer2 Module Period Register								1111 1111	1111 1111
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
TRISB	PORTB Data Direction Register								1111 1111	1111 1111
CCPR1H	Enhanced Capture/Compare/PWM Register 1 High Byte								xxxx xxxx	uuuu uuuu
CCPR1L	Enhanced Capture/Compare/PWM Register 1 Low Byte								xxxx xxxx	uuuu uuuu
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000	0000 0000
ECCPAS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	0000 0000	0000 0000
PWM1CON	PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0	0000 0000	uuuu uuuu
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0	0000 qq00	0000 qq00

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0'.
Shaded cells are not used by the ECCP module in Enhanced PWM mode.

16.3.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the EUSART are suspended. Because of this, the Baud Rate Generator is inactive and a proper byte reception cannot be performed. The auto-wake-up feature allows the controller to wake-up due to activity on the RX/DT line while the EUSART is operating in Asynchronous mode.

The auto-wake-up feature is enabled by setting the WUE bit (BAUDCTL<1>). Once set, the typical receive sequence on RX/DT is disabled and the EUSART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RX/DT line. (This coincides with the start of a Sync Break or a Wake-up Signal character for the LIN protocol.)

Following a wake-up event, the module generates an RCIF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes (Figure 16-7) and asynchronously if the device is in Sleep mode (Figure 16-8). The interrupt condition is cleared by reading the RCREG register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RX line, following the wake-up event. At this point, the EUSART module is in Idle mode and returns to normal operation. This signals to the user that the Sync Break event is over.

16.3.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RX/DT, information with any state changes before the Stop bit may signal a false end-of-character and cause data or framing errors. To work properly, therefore, the initial character in the transmission must be all '0's. This can be 00h (8 bytes) for standard RS-232 devices, or 000h (12 bits) for LIN bus.

Oscillator start-up time must also be considered, especially in applications using oscillators with longer start-up intervals (i.e., LP, XT or HS/PLL mode). The Sync Break (or Wake-up Signal) character must be of sufficient length and be followed by a sufficient period, to allow enough time for the selected oscillator to start and provide proper initialization of the EUSART.

16.3.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the EUSART in an Idle mode. The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared after this when a rising edge is seen on RX/DT. The interrupt condition is then cleared by reading the RCREG register. Ordinarily, the data in RCREG will be dummy data and should be discarded.

The fact that the WUE bit has been cleared (or is still set) and the RCIF flag is set should not be used as an indicator of the integrity of the data in RCREG. Users should consider implementing a parallel method in firmware to verify received data integrity.

To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

FIGURE 16-7: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION

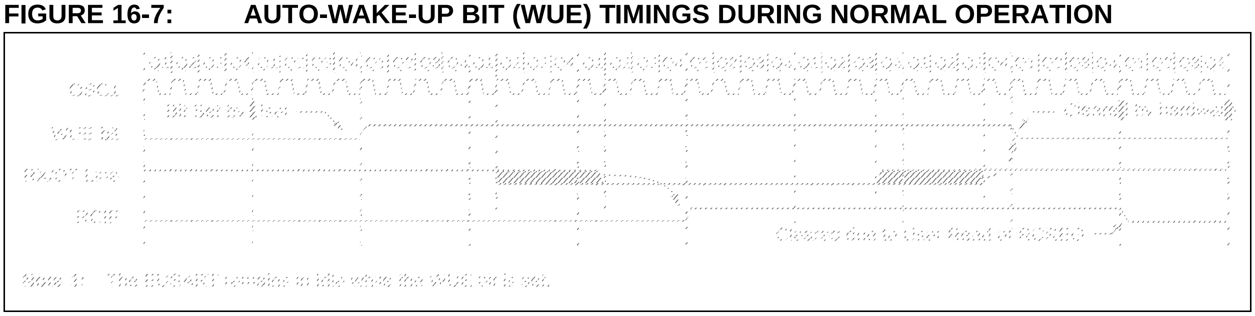
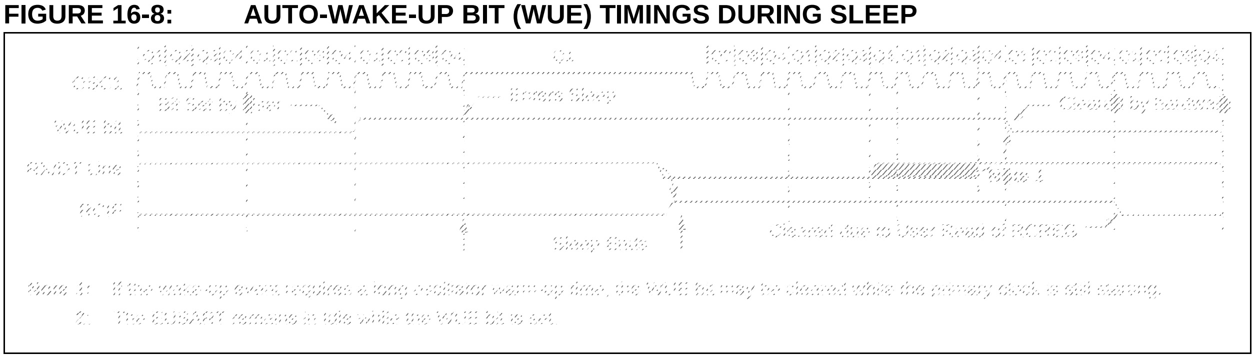


FIGURE 16-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP



16.5.2 EUSART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of Sleep, or any Idle mode and bit SREN, which is a "don't care" in Slave mode.

If receive is enabled by setting the CREN bit prior to entering Sleep or any Idle mode, then a word may be received while in this low-power mode. Once the word is received, the RSR register will transfer the data to the RCREG register; if the RCIE enable bit is set, the interrupt generated will wake the chip from low-power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. If interrupts are desired, set enable bit RCIE.
3. If 9-bit reception is desired, set bit RX9.
4. To enable reception, set enable bit CREN.
5. Flag bit, RCIF, will be set when reception is complete. An interrupt will be generated if enable bit, RCIE, was set.
6. Read the RCSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing bit CREN.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

TABLE 16-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	—	ADIF	RCIF	TXIF	—	CCP1IF	TMR2IF	TMR1IF	-000 -000	-000 -000
PIE1	—	ADIE	RCIE	TXIE	—	CCP1IE	TMR2IE	TMR1IE	-000 -000	-000 -000
IPR1	—	ADIP	RCIP	TXIP	—	CCP1IP	TMR2IP	TMR1IP	-111 -111	-111 -111
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
RCREG	EUSART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010
BAUDCTL	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	-1-1 0-00	-1-1 0-00
SPBRGH	Baud Rate Generator Register High Byte								0000 0000	0000 0000
SPBRG	Baud Rate Generator Register Low Byte								0000 0000	0000 0000

Legend: x = unknown, – = unimplemented, read as '0'. Shaded cells are not used for synchronous slave reception.

REGISTER 17-3: ADCON2: A/D CONTROL REGISTER 2

R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

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
000 = 0 T_{AD}(1)
001 = 2 T_{AD}
010 = 4 T_{AD}
011 = 6 T_{AD}
100 = 8 T_{AD}
101 = 12 T_{AD}
110 = 16 T_{AD}
111 = 20 T_{AD})

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

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

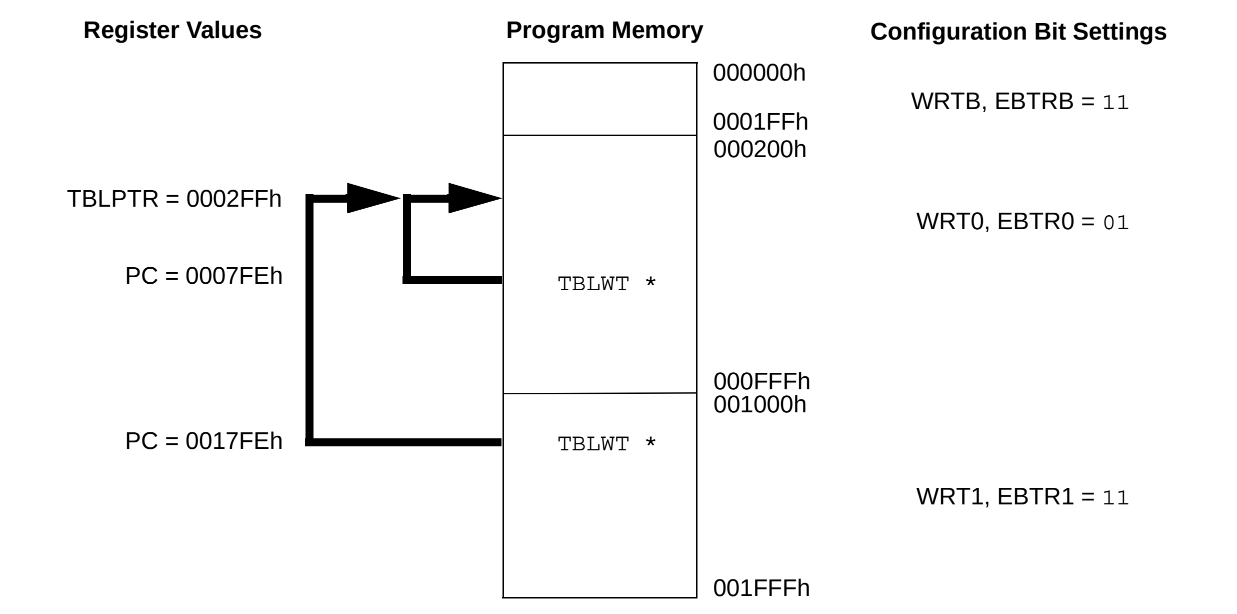
19.5.1 PROGRAM MEMORY CODE PROTECTION

The program memory may be read to, or written from, any location using the table read and table write instructions. The device ID may be read with table reads. The Configuration registers may be read and written with the table read and table write instructions.

In normal execution mode, the CPn bits have no direct effect. CPn bits inhibit external reads and writes. A block of user memory may be protected from table writes if the WRTn Configuration bit is '0'. The EBTRn bits control table reads. For a block of user memory with the EBTRn bit set to '0', a table read instruction that executes from within that block is allowed to read. A table read instruction that executes from a location outside of that block is not allowed to read and will result in reading '0's. Figures 19-6 through 19-8 illustrate table write and table read protection.

> **Note:** Code protection bits may only be written to a '0' from a '1' state. It is not possible to write a '1' to a bit in the '0' state. Code protection bits are only set to '1' by a full Chip Erase or Block Erase function. The full Chip Erase and Block Erase functions can only be initiated via ICSP or an external programmer.

FIGURE 19-6: TABLE WRITE (WRTn) DISALLOWED: PIC18F1320



Results: All table writes disabled to Blockn whenever WRTn = 0.

21.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers (MCU) and dsPIC® digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB XC Compiler
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB X SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
  - MPLAB ICD 3
  - PICkit™ 3
- Device Programmers
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits and Starter Kits
- Third-party development tools

21.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

22.3 DC Characteristics: PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (Continued)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended			
Param No.	**Symbol**	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
	VOL	**Output Low Voltage**				
D080		I/O ports	—	0.6	V	IOL = 8.5 mA, VDD = 4.5V, -40°C to +85°C
D083		OSC2/CLKO (RC mode)	—	0.6	V	IOL = 1.6 mA, VDD = 4.5V, -40°C to +85°C
	VOH	**Output High Voltage(3)**				
D090		I/O ports	VDD – 0.7	—	V	IOH = -3.0 mA, VDD = 4.5V, -40°C to +85°C
D092		OSC2/CLKO (RC mode)	VDD – 0.7	—	V	IOH = -1.3 mA, VDD = 4.5V, -40°C to +85°C
D150	VOD	**Open-Drain High Voltage**	—	8.5	V	RA4 pin
		Capacitive Loading Specs on Output Pins				
D100(4)	COSC2	OSC2 pin	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1
D101	CIO	All I/O pins and OSC2 (in RC mode)	—	50	pF	To meet the AC timing specifications
D102	CB	SCL, SDA	—	400	pF	In I²C mode

Note 1: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PIC® device be driven with an external clock while in RC mode.

2: The leakage current on the $\overline{\text{MCLR}}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as current sourced by the pin.

4: Parameter is characterized but not tested.

FIGURE 23-23: TOTAL IPD, -40°C TO +125°C SLEEP MODE, ALL PERIPHERALS DISABLED

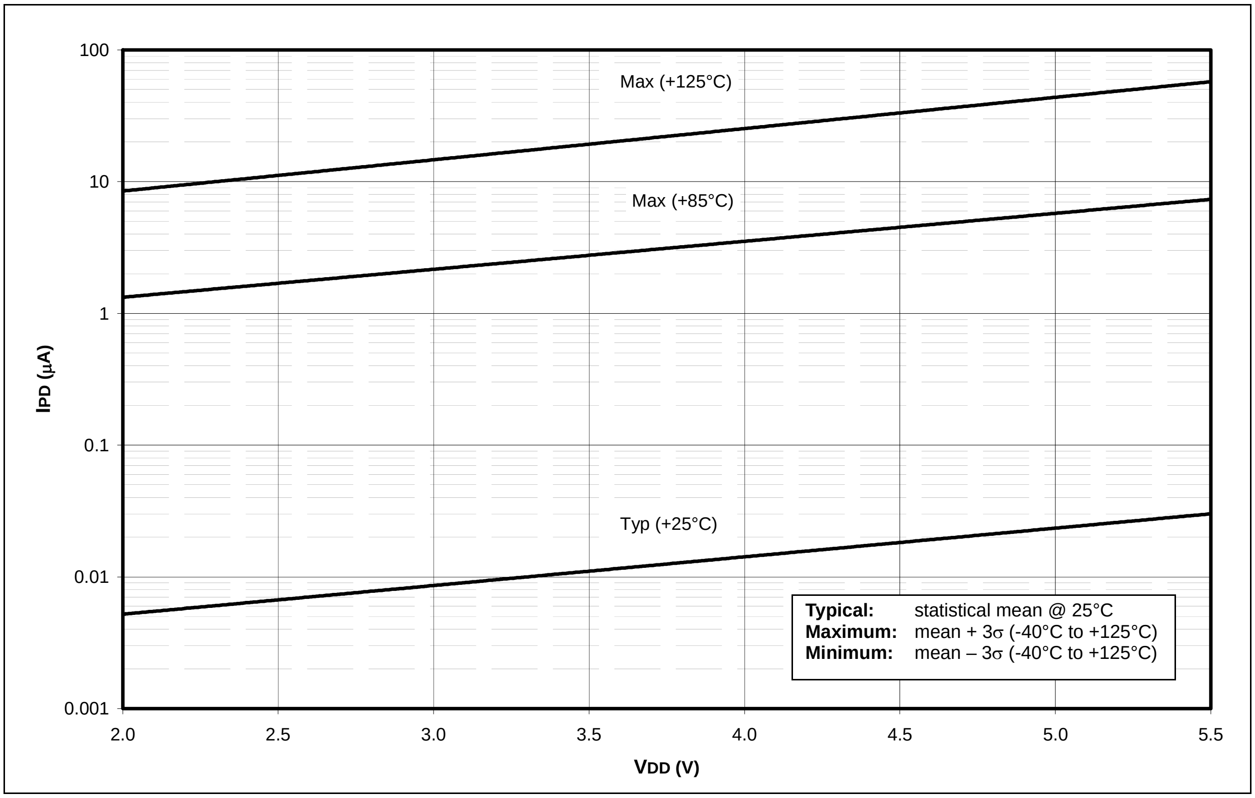


FIGURE 23-24: VOH VS. IOH OVER TEMPERATURE (-40°C TO +125°C), VDD = 3.0V