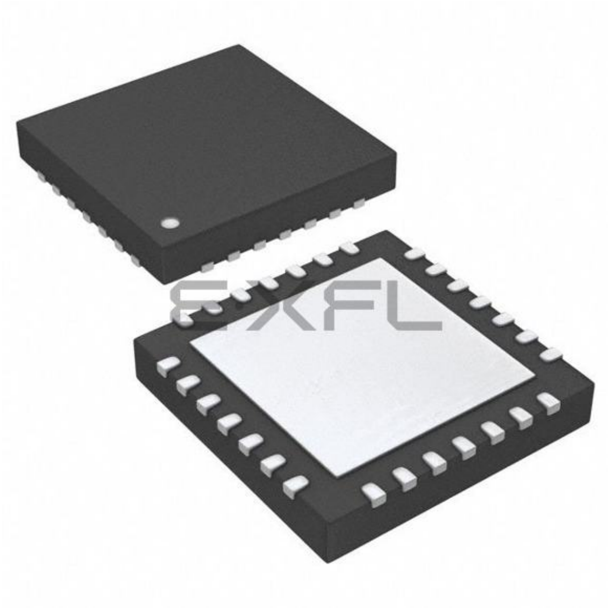


Welcome to E-XFL.COM

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	Obsolete
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	28-VQFN Exposed Pad
Supplier Device Package	28-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f1320t-e-ml

3.3.3 RC_IDLE MODE

In RC_IDLE mode, the CPU is disabled, but the peripherals continue to be clocked from the internal oscillator block using the INTOSC multiplexer. This mode allows for controllable power conservation during Idle periods.

This mode is entered by setting the IDLEN bit, setting SCS1 (SCS0 is ignored) and executing a SLEEP instruction. The INTOSC multiplexer may be used to select a higher clock frequency by modifying the IRCF bits before executing the SLEEP instruction. When the clock source is switched to the INTOSC multiplexer (see Figure 3-7), the primary oscillator is shut down and the OSTS bit is cleared.

If the IRCF bits are set to a non-zero value (thus, enabling the INTOSC output), the IOFS bit becomes set after the INTOSC output becomes stable, in about 1 ms. Clocks to the peripherals continue while the INTOSC source stabilizes. If the IRCF bits were previously at a non-zero value before the SLEEP instruction was executed and the INTOSC source was already stable, the IOFS bit will remain set. If the IRCF bits are all clear, the INTOSC output is not enabled and the IOFS bit will remain clear; there will be no indication of the current clock source.

When a wake event occurs, the peripherals continue to be clocked from the INTOSC multiplexer. After a 10 μs delay following the wake event, the CPU begins executing code, being clocked by the INTOSC multiplexer. The microcontroller operates in RC_RUN mode until the primary clock becomes ready. When the primary clock becomes ready, a clock switchback to the primary clock occurs (see Figure 3-8). When the clock switch is complete, the IOFS 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 INTRC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

FIGURE 3-7: TIMING TRANSITION TO RC_IDLE MODE

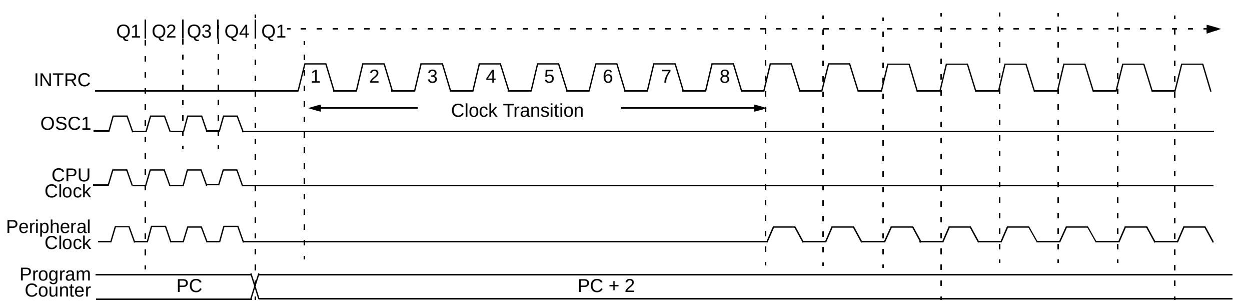
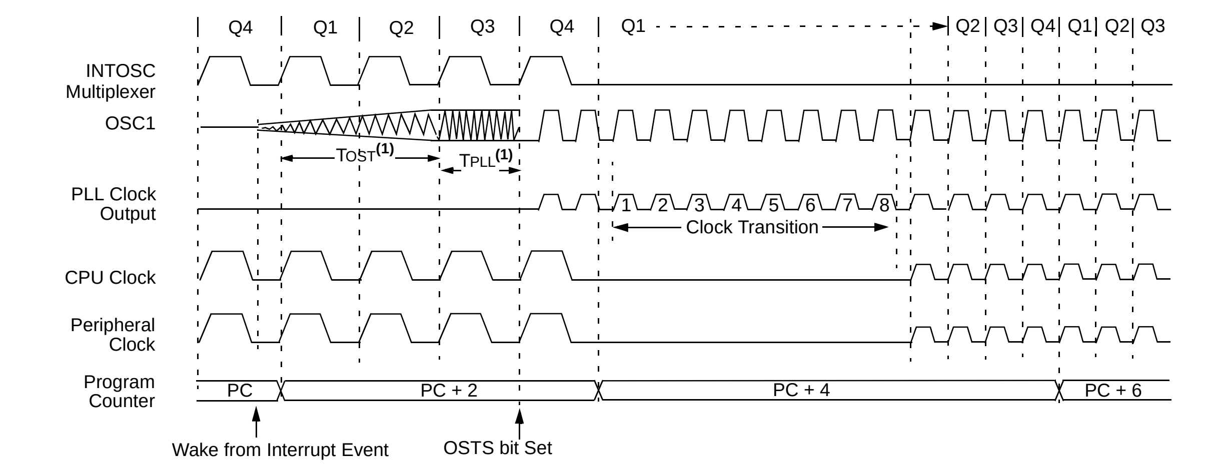


FIGURE 3-8: TIMING TRANSITION FOR WAKE FROM RC_RUN MODE (RC_RUN TO PRI_RUN)



Note 1: T_{OST} = 1024 T_{OSC}; T_{PLL} = 2 ms (approx). These intervals are not shown to scale.

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”**).

TABLE 4-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{MCLR}$ Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
BSR	1220	1320	---- 0000	---- 0000	---- uuuu
INDF2	1220	1320	N/A	N/A	N/A
POSTINC2	1220	1320	N/A	N/A	N/A
POSTDEC2	1220	1320	N/A	N/A	N/A
PREINC2	1220	1320	N/A	N/A	N/A
PLUSW2	1220	1320	N/A	N/A	N/A
FSR2H	1220	1320	---- 0000	---- 0000	---- uuuu
FSR2L	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	1220	1320	---x xxxx	---u uuuu	---u uuuu
TMR0H	1220	1320	0000 0000	0000 0000	uuuu uuuu
TMR0L	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	1220	1320	1111 1111	1111 1111	uuuu uuuu
OSCCON	1220	1320	0000 q000	0000 q000	uuuu qquu
LVDCON	1220	1320	--00 0101	--00 0101	--uu uuuu
WDTCON	1220	1320	---- ---0	---- ---0	---- ---u
RCON(4)	1220	1320	0--1 11q0	0--q qquu	u--u qquu
TMR1H	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	1220	1320	0000 0000	u0uu uuuu	uuuu uuuu
TMR2	1220	1320	0000 0000	0000 0000	uuuu uuuu
PR2	1220	1320	1111 1111	1111 1111	1111 1111
T2CON	1220	1320	-000 0000	-000 0000	-uuu uuuu
ADRESH	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	1220	1320	00-0 0000	00-0 0000	uu-u uuuu
ADCON1	1220	1320	-000 0000	-000 0000	-uuu uuuu
ADCON2	1220	1320	0-00 0000	0-00 0000	u-uu uuuu
CCPR1H	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	1220	1320	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	1220	1320	0000 0000	0000 0000	uuuu uuuu
PWM1CON	1220	1320	0000 0000	0000 0000	uuuu uuuu
ECCPAS	1220	1320	0000 0000	0000 0000	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 4-2 for Reset value for specific condition.

5: Bits 6 and 7 of PORTA, LATA and TRISA are enabled, depending on the Oscillator mode selected. When not enabled as PORTA pins, they are disabled and read '0'.

6: Bit 5 of PORTA is enabled if $\overline{MCLR}$ is disabled.

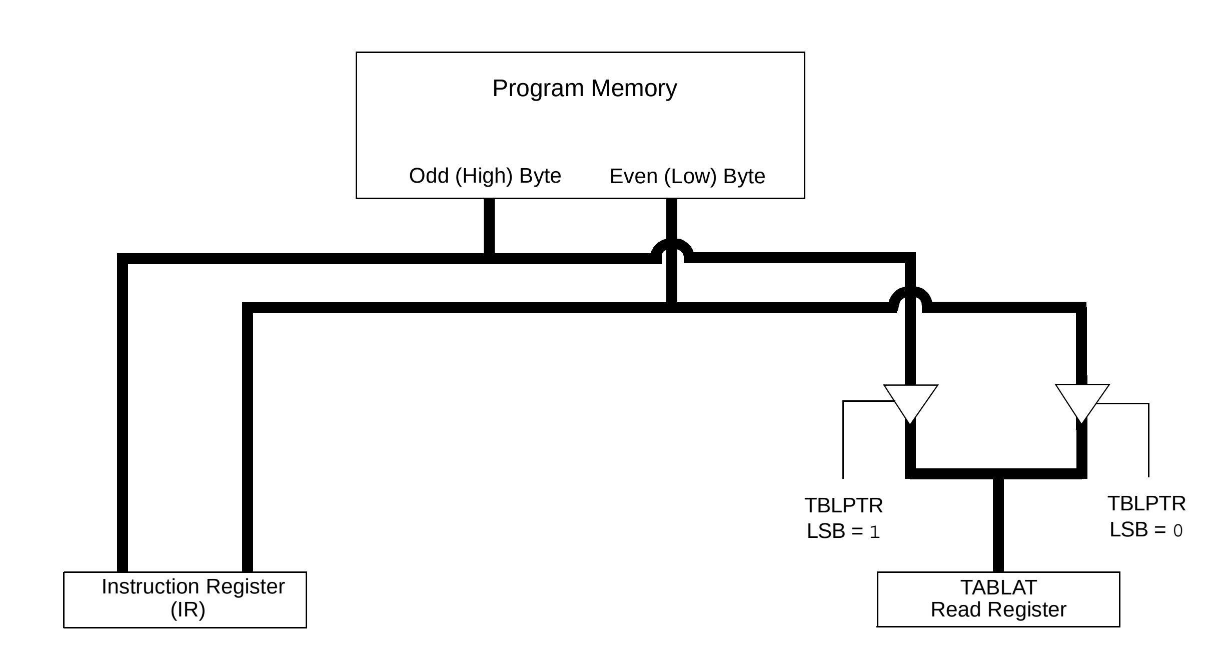
6.3 Reading the Flash Program Memory

The TBLRD instruction is used to retrieve data from program memory and place it into data RAM. Table reads from program memory are performed one byte at a time.

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

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

FIGURE 6-4: READS FROM FLASH PROGRAM MEMORY



EXAMPLE 6-1: READING A FLASH PROGRAM MEMORY WORD

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

EXAMPLE 6-3: WRITING TO FLASH PROGRAM MEMORY

```
            MOVLW   D'64                    ; number of bytes in erase block
            MOVWF   COUNTER
            MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
            MOVWF   FSR0H
            MOVLW   BUFFER_ADDR_LOW
            MOVWF   FSR0L
            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           ; 6 LSB = 0
            MOVWF   TBLPTRL
READ_BLOCK
            TBLRD*+                         ; read into TABLAT, and inc
            MOVF    TABLAT, W               ; get data
            MOVWF   POSTINC0                ; store data and increment FSR0
            DECFSZ  COUNTER                 ; done?
            GOTO    READ_BLOCK              ; repeat
MODIFY_WORD
            MOVLW   DATA_ADDR_HIGH          ; point to buffer
            MOVWF   FSR0H
            MOVLW   DATA_ADDR_LOW
            MOVWF   FSR0L
            MOVLW   NEW_DATA_LOW            ; update buffer word and increment FSR0
            MOVWF   POSTINC0
            MOVLW   NEW_DATA_HIGH           ; update buffer word
            MOVWF   INDF0
ERASE_BLOCK
            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           ; 6 LSB = 0
            MOVWF   TBLPTRL
            BCF     EECON1, CFGS            ; point to PROG/EEPROM memory
            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                     ; Required sequence
            MOVWF   EECON2                  ; write 55H
            MOVLW   AAh
            MOVWF   EECON2                  ; write AAH
            BSF     EECON1, WR              ; start erase (CPU stall)
            NOP
            BSF     INTCON, GIE             ; re-enable interrupts
WRITE_BUFFER_BACK
            MOVLW   8                       ; number of write buffer groups of 8 bytes
            MOVWF   COUNTER_HI
            MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
            MOVWF   FSR0H
            MOVLW   BUFFER_ADDR_LOW
            MOVWF   FSR0L
PROGRAM_LOOP
            MOVLW   8                       ; number of bytes in holding register
            MOVWF   COUNTER
```


9.5 RCON Register

The RCON register contains bits used to determine the cause of the last Reset or wake-up from a low-power mode. RCON also contains the bit that enables interrupt priorities (IPEN).

REGISTER 9-10: RCON: RESET CONTROL REGISTER

R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
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 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

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

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
For details of bit operation, see Register 5-3.

bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
For details of bit operation, see Register 5-3.

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
For details of bit operation, see Register 5-3.

bit 1 **$\overline{POR}$:** Power-on Reset Status bit
For details of bit operation, see Register 5-3.

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
For details of bit operation, see Register 5-3.

FIGURE 10-6: MCLR/VPP/RA5 PIN BLOCK DIAGRAM

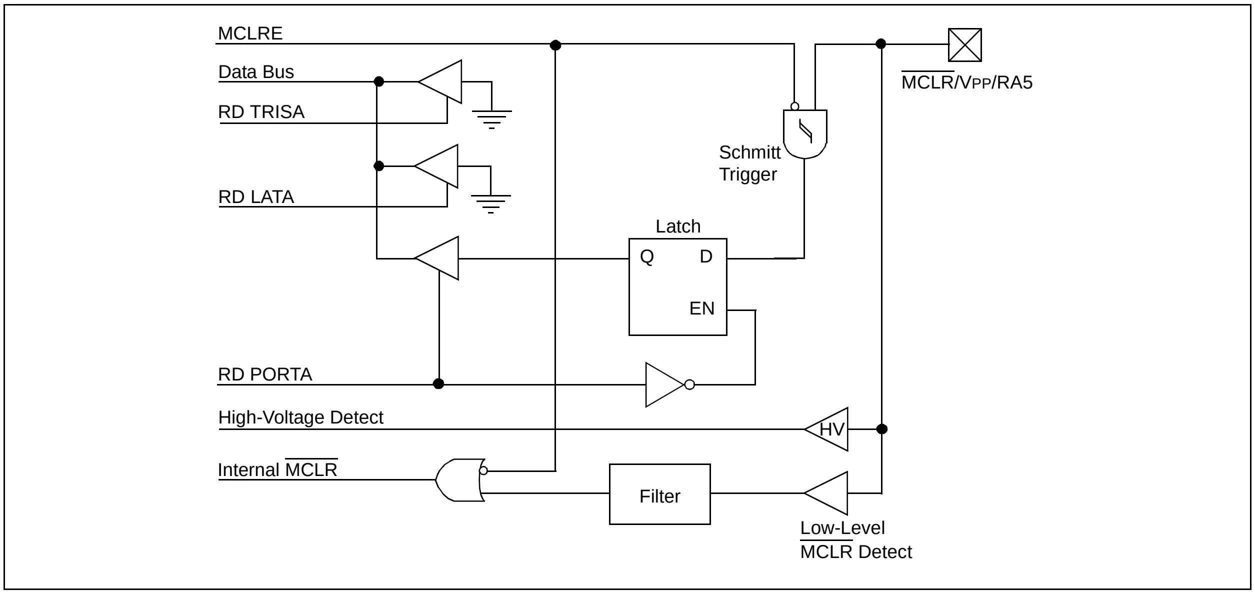


TABLE 10-1: PORTA FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit 0	ST	Input/output port pin or analog input.
RA1/AN1/LVDIN	bit 1	ST	Input/output port pin, analog input or Low-Voltage Detect input.
RA2/AN2/VREF-	bit 2	ST	Input/output port pin, analog input or VREF-.
RA3/AN3/VREF+	bit 3	ST	Input/output port pin, analog input or VREF+.
RA4/T0CKI	bit 4	ST	Input/output port pin or external clock input for Timer0. Output is open-drain type.
MCLR/VPP/RA5	bit 5	ST	Master Clear input or programming voltage input (if MCLR is enabled); input only port pin or programming voltage input (if MCLR is disabled).
OSC2/CLKO/RA6	bit 6	ST	OSC2, clock output or I/O pin.
OSC1/CLKI/RA7	bit 7	ST	OSC1, clock input or I/O pin.

Legend: TTL = TTL input, ST = Schmitt Trigger input

TABLE 10-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

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
PORTA	RA7(1)	RA6(1)	RA5(2)	RA4	RA3	RA2	RA1	RA0	xx0x 0000	uu0u 0000
LATA	LATA7(1)	LATA6(1)	—	LATA Data Output Register					xx-x xxxx	uu-u uuuu
TRISA	TRISA7(1)	TRISA6(1)	—	PORTA Data Direction Register					11-1 1111	11-1 1111
ADCON1	—	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	-000 0000	-000 0000

Legend: x = unknown, u = unchanged, – = unimplemented locations read as '0'. Shaded cells are not used by PORTA.

Note 1: RA7:RA6 and their associated latch and data direction bits are enabled as I/O pins based on oscillator configuration; otherwise, they are read as '0'.

2: RA5 is an input only if MCLR is disabled.

10.2 PORTB, TRISB and LATB Registers

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= `1`) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a High-impedance mode). Clearing a TRISB bit (= `0`) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATB) is also memory mapped. Read-modify-write operations on the LATB register read and write the latched output value for PORTB.

EXAMPLE 10-2: INITIALIZING PORTB

```
CLRF    PORTB   ; Initialize PORTB by
                ; clearing output
                ; data latches
CLRF    LATB    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0x70    ; Set RB0, RB1, RB4 as
MOVWF   ADCON1  ; digital I/O pins
MOVLW   0xCF    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISB   ; Set RB<3:0> as inputs
                ; RB<5:4> as outputs
                ; RB<7:6> as inputs
```

Pins RB0-RB2 are multiplexed with INT0-INT2; pins RB0, RB1 and RB4 are multiplexed with A/D inputs; pins RB1 and RB4 are multiplexed with EUSART; and pins RB2, RB3, RB6 and RB7 are multiplexed with ECCP.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit, RBPU (INTCON2<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Note:	On a Power-on Reset, RB4:RB0 are configured as analog inputs by default and read as '`0`'; RB7:RB5 are configured as digital inputs.

Four of the PORTB pins (RB7:RB4) have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB Port Change Interrupt with Flag bit, RBIF (INTCON<0>).

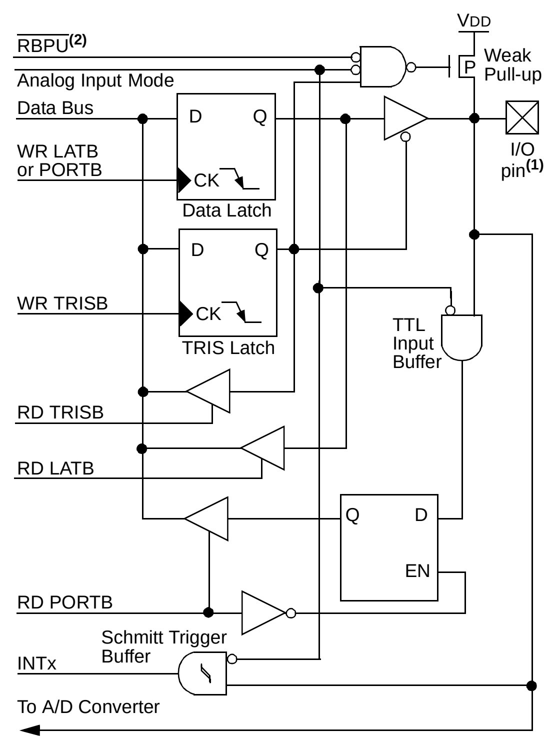
This interrupt can wake the device from Sleep. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB (except with the `MOVFF` instruction). This will end the mismatch condition.
b) Clear flag bit, RBIF.

A mismatch condition will continue to set flag bit, RBIF. Reading PORTB will end the mismatch condition and allow flag bit, RBIF, to be cleared.

The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

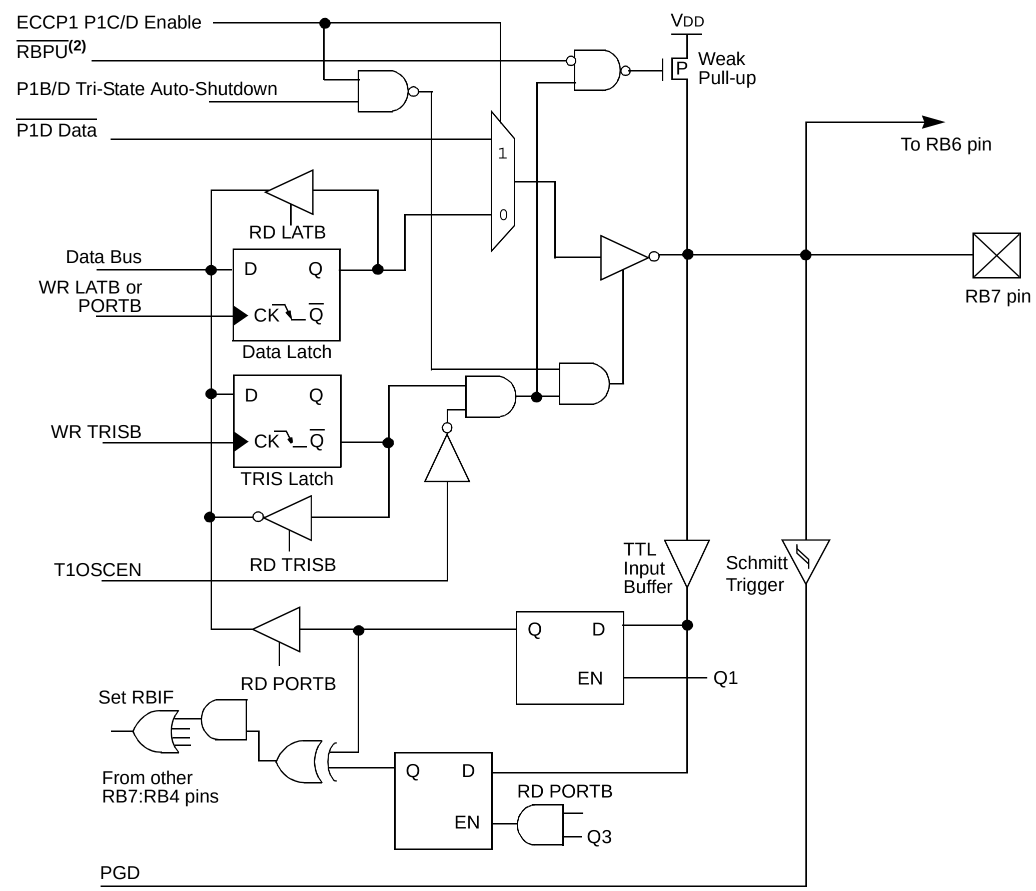
FIGURE 10-7: BLOCK DIAGRAM OF RB0/AN4/INT0 PIN



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>).

FIGURE 10-14: BLOCK DIAGRAM OF RB7/PGD/T1OSI/P1D/KBI3 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>).

14.2 Timer1 Oscillator

The Timer1 oscillator may be used as the clock source for Timer3. The Timer1 oscillator is enabled by setting the T1OSCEN (T1CON<3>) bit. The oscillator is a low-power oscillator rated for 32 kHz crystals. See **Section 12.2 “Timer1 Oscillator”** for further details.

14.3 Timer3 Interrupt

The TMR3 register pair (TMR3H:TMR3L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR3 interrupt, if enabled, is generated on overflow, which is latched in interrupt flag bit, TMR3IF (PIR2<1>). This interrupt can be enabled/disabled by setting/clearing TMR3 Interrupt Enable bit, TMR3IE (PIE2<1>).

14.4 Resetting Timer3 Using a CCP Trigger Output

If the CCP module is configured in Compare mode to generate a “special event trigger” (CCP1M3:CCP1M0 = `1011`), this signal will reset Timer3. See **Section 15.4.4 “Special Event Trigger”** for more information.

> **Note:** The special event triggers from the CCP module will not set interrupt flag bit, TMR3IF (PIR1<0>).

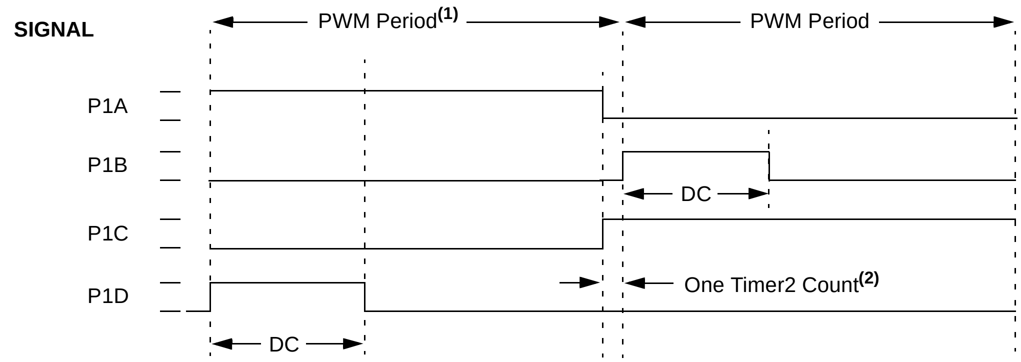
Timer3 must be configured for either Timer or Synchronized Counter mode to take advantage of this feature. If Timer3 is running in Asynchronous Counter mode, this Reset operation may not work. In the event that a write to Timer3 coincides with a special event trigger from CCP1, the write will take precedence. In this mode of operation, the CCPR1H:CCPR1L register pair effectively becomes the period register for Timer3.

TABLE 14-1: REGISTERS ASSOCIATED WITH TIMER3 AS A TIMER/COUNTER

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
PIR2	OSCFIF	—	—	EEIF	—	LVDIF	TMR3IF	—	0--0 -00-	0--0 -00-
PIE2	OSCFIE	—	—	EEIE	—	LVDIE	TMR3IE	—	0--0 -00-	0--0 -00-
IPR2	OSCFIP	—	—	EEIP	—	LVDIP	TMR3IP	—	1--1 -11-	1--1 -11-
TMR3L	Holding Register for the Least Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
TMR3H	Holding Register for the Most Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0000 0000	u0uu uuuu
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0-00 0000	u-uu uuuu

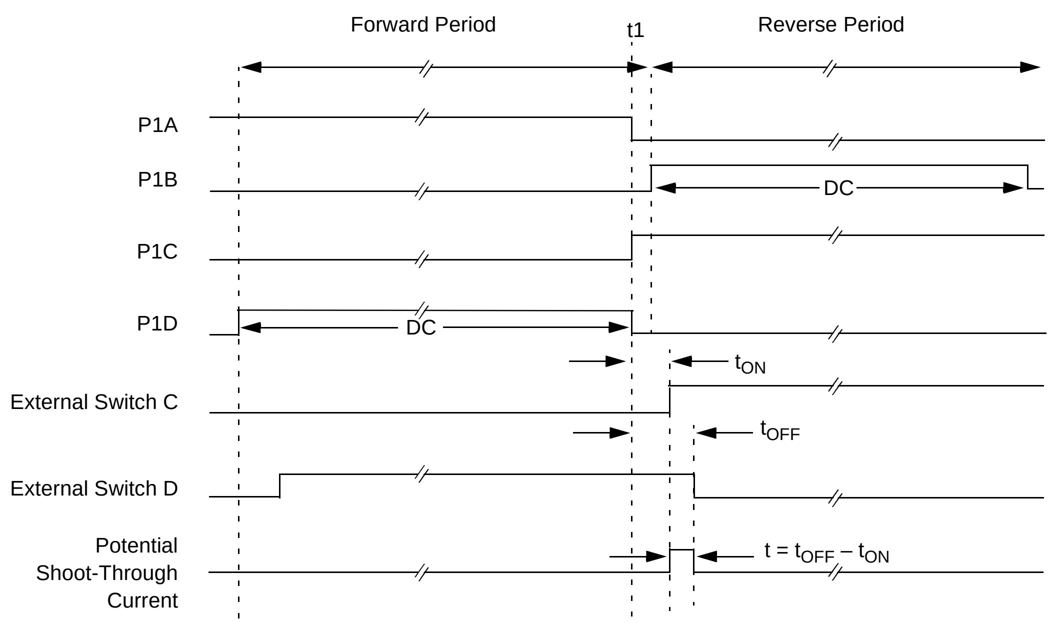
Legend: x = unknown, u = unchanged, – = unimplemented, read as ‘0’. Shaded cells are not used by the Timer3 module.

FIGURE 15-10: PWM DIRECTION CHANGE (ACTIVE-HIGH)



Note 1: The direction bit in the CCP1 Control register (CCP1CON<7>) is written any time during the PWM cycle.
2: When changing directions, the P1A and P1C toggle one Timer2 count before the end of the current PWM cycle. The modulated P1B and P1D signals are inactive at this time.

FIGURE 15-11: PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE (ACTIVE-HIGH)



Note 1: t_{ON} is the turn-on delay of power switch QC and its driver.
2: t_{OFF} is the turn-off delay of power switch QD and its driver.

TABLE 16-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0											
	FOSC = 40.000 MHz			FOSC = 20.000 MHz			FOSC = 10.000 MHz			FOSC = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
2.4	—	—	—	—	—	—	2.441	1.73	255	2403	-0.16	207
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0								
	FOSC = 4.000 MHz			FOSC = 2.000 MHz			FOSC = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	300	-0.16	207
1.2	1.202	0.16	207	1201	-0.16	103	1201	-0.16	51
2.4	2.404	0.16	103	2403	-0.16	51	2403	-0.16	25
9.6	9.615	0.16	25	9615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1											
	FOSC = 40.000 MHz			FOSC = 20.000 MHz			FOSC = 10.000 MHz			FOSC = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.00	8332	0.300	0.02	4165	0.300	0.02	2082	300	-0.04	1665
1.2	1.200	0.02	2082	1.200	-0.03	1041	1.200	-0.03	520	1201	-0.16	415
2.4	2.402	0.06	1040	2.399	-0.03	520	2.404	0.16	259	2403	-0.16	207
9.6	9.615	0.16	259	9.615	0.16	129	9.615	0.16	64	9615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

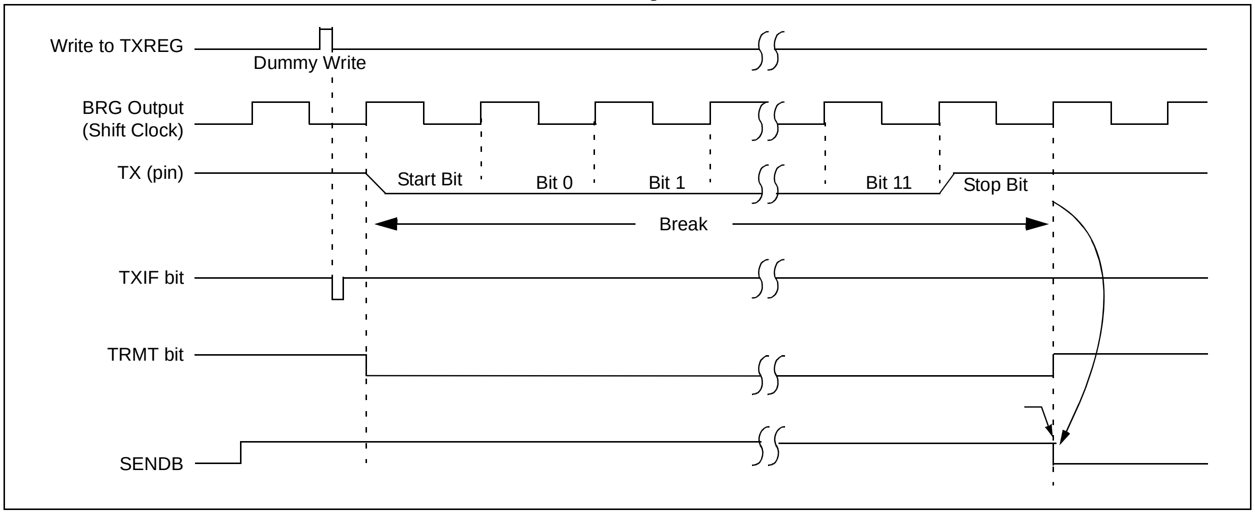
BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1								
	FOSC = 4.000 MHz			FOSC = 2.000 MHz			FOSC = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.04	832	300	-0.16	415	300	-0.16	207
1.2	1.202	0.16	207	1201	-0.16	103	1201	-0.16	51
2.4	2.404	0.16	103	2403	-0.16	51	2403	-0.16	25
9.6	9.615	0.16	25	9615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

16.3.6.2 Receiving a Break Sync

To receive a Break Sync:

1. Configure the EUSART for asynchronous transmit and receive. TXEN should remain clear. SPBRGH:SPBRG may be left as is.
2. Enable auto-wake-up. Set WUE.
3. Enable RXIF interrupts. Set RCIE, PEIE, GIE.
4. The controller may be placed in any power managed mode.
5. An RCIF will be generated at the beginning of the Break signal. When the interrupt is received, read RCREG to clear RCIF and discard. Allow the controller to return to PRI_RUN mode.
6. Wait for the RX line to go high at the end of the Break signal. Wait for any of the following: WUE to clear automatically (poll), RB4/RX to go high (poll) or for RBIF to be set (poll or interrupt). If RBIF is used, check to be sure that RB4/RX is high before continuing.
7. Enable Auto-Baud Rate Detect. Set ABDEN.
8. Return from the interrupt. Allow the primary clock to start and stabilize (PRI_RUN or PRI_IDLE).
9. When the next RCIF interrupt occurs, the received baud rate has been measured. Read RCREG to clear RCIF and discard. Check SPBRGH:SPBRG for a valid value. The EUSART is ready for normal communications. Return from the interrupt. Allow the primary clock to run (PRI_RUN or PRI_IDLE).
10. Process subsequent RCIF interrupts normally as in asynchronous reception. TXEN should now be set if transmissions are needed. TXIF and TXIE may be set if transmit interrupts are desired. Remain in PRI_RUN or PRI_IDLE until communications are complete. Clear TXEN and return to step 2.

FIGURE 16-9: SEND BREAK CHARACTER SEQUENCE



19.5.2 DATA EEPROM CODE PROTECTION

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

19.5.3 CONFIGURATION REGISTER PROTECTION

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

19.6 ID Locations

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

19.7 In-Circuit Serial Programming

PIC18F1220/1320 microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data and three other lines for power, ground and the programming voltage. This allows customers to manufacture boards with unprogrammed devices and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed (see Table 19-4).

Note: The Timer1 oscillator shares the T1OSI and T1OSO pins with the PGD and PGC pins used for programming and debugging.

When using the Timer1 oscillator, In-Circuit Serial Programming (ICSP) may not function correctly (high voltage or low voltage), or the In-Circuit Debugger (ICD) may not communicate with the controller. As a result of using either ICSP or ICD, the Timer1 crystal may be damaged.

If ICSP or ICD operations are required, the crystal should be disconnected from the circuit (disconnect either lead), or installed after programming. The oscillator loading capacitors may remain in-circuit during ICSP or ICD operation.

TABLE 19-4: ICSP/ICD CONNECTIONS

Signal	Pin	Notes
PGD	RB7/PGD/T1OSI/P1D/KBI3	Shared with T1OSC – protect crystal
PGC	RB6/PGC/T1OSO/T13CKI/P1C/KBI2	Shared with T1OSC – protect crystal
$\overline{MCLR}$	$\overline{MCLR}$/VPP/RA5	
VDD	VDD	
VSS	VSS	
PGM	RB5/PGM/KBI1	Optional – pull RB5 low is LVP enabled

19.8 In-Circuit Debugger

When the DEBUG bit in Configuration register, CONFIG4L, is programmed to a '0', the In-Circuit Debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® IDE. When the microcontroller has this feature enabled, some resources are not available for general use. Table 19-5 shows which resources are required by the background debugger.

TABLE 19-5: DEBUGGER RESOURCES

I/O pins:	RB6, RB7
Stack:	2 levels
Program Memory:	512 bytes
Data Memory:	10 bytes

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to $\overline{MCLR}$/VPP, VDD, VSS, RB7 and RB6. This will interface to the In-Circuit Debugger module available from Microchip, or one of the third party development tool companies (see the note following **Section 19.7 "In-Circuit Serial Programming"** for more information).

20.2 Instruction Set

ADDLW **ADD literal to W**

Syntax: [*label*] ADDLW k

Operands: $0 \le k \le 255$

Operation: (W) + k → W

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

Encoding:

0000	1111	kkkk	kkkk

Description: The contents of W are added to the 8-bit literal 'k' and the result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDLW 0x15`

Before Instruction

W = 0x10

After Instruction

W = 0x25

ADDWF **ADD W to f**

Syntax: [*label*] ADDWF f [,d [,a]]

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

Operation: (W) + (f) → dest

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

Encoding:

0010	01da	ffff	ffff

Description: Add W to register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank will be selected. If 'a' is '1', the BSR is used.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDWF REG, W`

Before Instruction

W = 0x17
REG = 0xC2

After Instruction

W = 0xD9
REG = 0xC2

DECFSZ — Decrement f, skip if 0

Syntax: [*label*] DECFSZ f [,d [,a]]

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

Operation: (f) – 1 → dest,
skip if result = `0`

Status Affected: None

Encoding:

0010	11da	ffff	ffff

Description: The contents of register ‘f’ are decremented. If ‘d’ is ‘`0`’, the result is placed in W. If ‘d’ is ‘`1`’, the result is placed back in register ‘f’ (default).
If the result is ‘`0`’, the next instruction, which is already fetched, is discarded and a `NOP` is executed instead, making it a 2-cycle instruction. If ‘a’ is ‘`0`’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = `1`, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

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

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE      DECFSZ   CNT
          GOTO     LOOP
CONTINUE
```

Before Instruction
PC = Address (`HERE`)

After Instruction
CNT = CNT – 1
If CNT = 0;
PC = Address (`CONTINUE`)
If CNT ≠ 0;
PC = Address (`HERE + 2`)

DCFSNZ — Decrement f, skip if not 0

Syntax: [*label*] DCFSNZ f [,d [,a]]

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

Operation: (f) – 1 → dest,
skip if result ≠ `0`

Status Affected: None

Encoding:

0100	11da	ffff	ffff

Description: The contents of register ‘f’ are decremented. If ‘d’ is ‘`0`’, the result is placed in W. If ‘d’ is ‘`1`’, the result is placed back in register ‘f’ (default).
If the result is not ‘`0`’, the next instruction, which is already fetched, is discarded and a `NOP` is executed instead, making it a 2-cycle instruction. If ‘a’ is ‘`0`’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = `1`, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

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

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    DCFSNZ TEMP
ZERO    :
NZERO   :
```

Before Instruction
TEMP = ?

After Instruction
TEMP = TEMP – 1,
If TEMP = 0;
PC = Address (`ZERO`)
If TEMP ≠ 0;
PC = Address (`NZERO`)

TBLWT **Table Write**

Syntax: [*label*] TBLWT (*; *+; *-; +*)

Operands: None

Operation: if TBLWT*,
(TABLAT) → Holding Register;
TBLPTR – No Change;
if TBLWT*+,
(TABLAT) → Holding Register;
(TBLPTR) + 1 → TBLPTR;
if TBLWT*-,
(TABLAT) → Holding Register;
(TBLPTR) – 1 → TBLPTR;
if TBLWT+*,
(TBLPTR) + 1 → TBLPTR;
(TABLAT) → Holding Register;

Status Affected: None

Encoding:

0000	0000	0000	11nn nn = 0* = 1*+ = 2*- = 3+*

Description: This instruction uses the 3 LSBs of TBLPTR to determine which of the eight holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 6.0 "Flash Program Memory"** for additional details on programming Flash memory.)
The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The `TBLWT` instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

TBLWT **Table Write (Continued)**

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)

Example 1: `TBLWT *+;`

Before Instruction

TABLAT	=	0x55
TBLPTR	=	0x00A356
HOLDING REGISTER (0x00A356)	=	0xFF

After Instructions (table write completion)

TABLAT	=	0x55
TBLPTR	=	0x00A357
HOLDING REGISTER (0x00A356)	=	0x55

Example 2: `TBLWT +*;`

Before Instruction

TABLAT	=	0x34
TBLPTR	=	0x01389A
HOLDING REGISTER (0x01389A)	=	0xFF
HOLDING REGISTER (0x01389B)	=	0xFF

After Instruction (table write completion)

TABLAT	=	0x34
TBLPTR	=	0x01389B
HOLDING REGISTER (0x01389A)	=	0xFF
HOLDING REGISTER (0x01389B)	=	0x34

22.2 DC Characteristics: Power-Down and Supply Current PIC18F1220/1320 (Industrial) PIC18LF1220/1320 (Industrial) (Continued)

PIC18LF1220/1320 (Industrial)	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial
PIC18F1220/1320 (Industrial, Extended)	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial -40°C ≤ T_A ≤ +125°C for extended

Param No.	Device	Typ.	Max.	Units	Conditions		
	Module Differential Currents (ΔI_{WDT}, ΔI_{BOR}, ΔI_{LVD}, ΔI_{OSCB}, ΔI_{AD})						
D022 (ΔI_{WDT})	**Watchdog Timer**	1.5	4.0	µA	-40°C	V_{DD} = 2.0V	
		2.2	4.0	µA	+25°C		
		3.1	5.0	µA	+85°C		
		2.5	6.0	µA	-40°C	V_{DD} = 3.0V	
		3.3	6.0	µA	+25°C		
		4.7	7.0	µA	+85°C		
		3.7	10.0	µA	-40°C	V_{DD} = 5.0V	
		4.5	10.0	µA	+25°C		
		6.1	13.0	µA	+85°C		
D022A (ΔI_{BOR})	**Brown-out Reset**	19	35.0	µA	-40°C to +85°C	V_{DD} = 3.0V	
		24	45.0	µA	-40°C to +85°C	V_{DD} = 5.0V	
D022B (ΔI_{LVD})	**Low-Voltage Detect**	8.5	25.0	µA	-40°C to +85°C	V_{DD} = 2.0V	
		16	35.0	µA	-40°C to +85°C	V_{DD} = 3.0V	
		20	45.0	µA	-40°C to +85°C	V_{DD} = 5.0V	
D025 (ΔI_{OSCB})	**Timer1 Oscillator**	1.7	3.5	µA	-40°C	V_{DD} = 2.0V	32 kHz on Timer1(4)
		1.8	3.5	µA	+25°C		
		2.1	4.5	µA	+85°C		
		2.2	4.5	µA	-40°C	V_{DD} = 3.0V	32 kHz on Timer1(4)
		2.6	4.5	µA	+25°C		
		2.8	5.5	µA	+85°C		
		3.0	6.0	µA	-40°C	V_{DD} = 5.0V	32 kHz on Timer1(4)
		3.3	6.0	µA	+25°C		
		3.6	7.0	µA	+85°C		
D026 (ΔI_{AD})	**A/D Converter**	1.0	3.0	µA	-40°C to +85°C	V_{DD} = 2.0V	A/D on, not converting
		1.0	4.0	µA	-40°C to +85°C	V_{DD} = 3.0V	
		2.0	10.0	µA	-40°C to +85°C	V_{DD} = 5.0V	
		1.0	8.0	µA	-40°C to +125°C	V_{DD} = 5.0V	

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

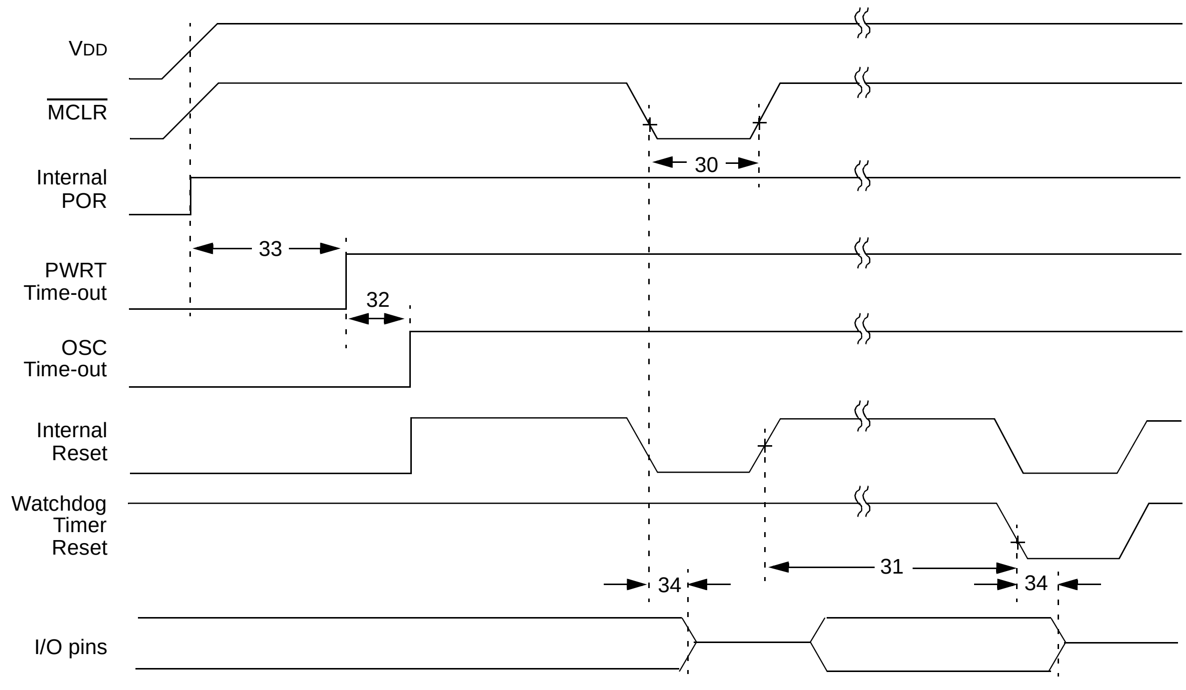
Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to V_{DD} or V_{SS} and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, 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 I_{DD} measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V_{DD};
$\overline{MCLR}$ = V_{DD}; WDT enabled/disabled as specified.

3: For RC oscillator configurations, current through R_{EXT} is not included. The current through the resistor can be estimated by the formula $Ir = V_{DD}/2R_{EXT}$ (mA) with R_{EXT} in kΩ.

4: 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.

FIGURE 22-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



Note: Refer to Figure 22-5 for load conditions.

FIGURE 22-9: BROWN-OUT RESET TIMING

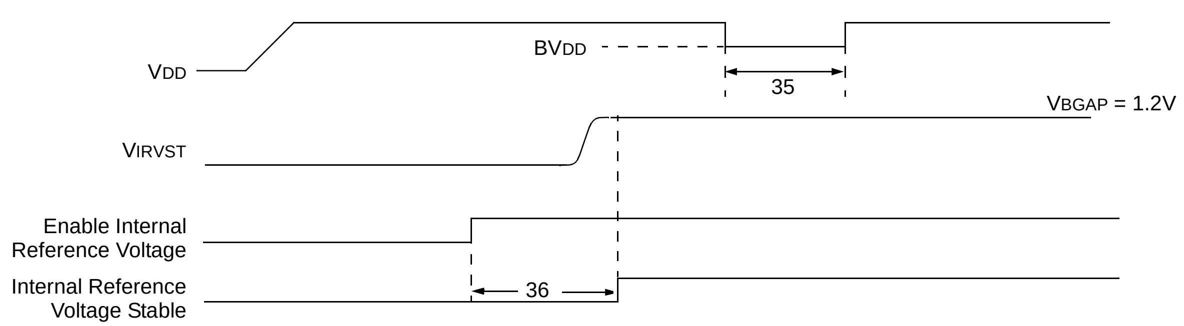


FIGURE 23-5: MAXIMUM IDD vs. FOSC OVER VDD PRI_RUN, EC MODE, -40°C TO +125°C

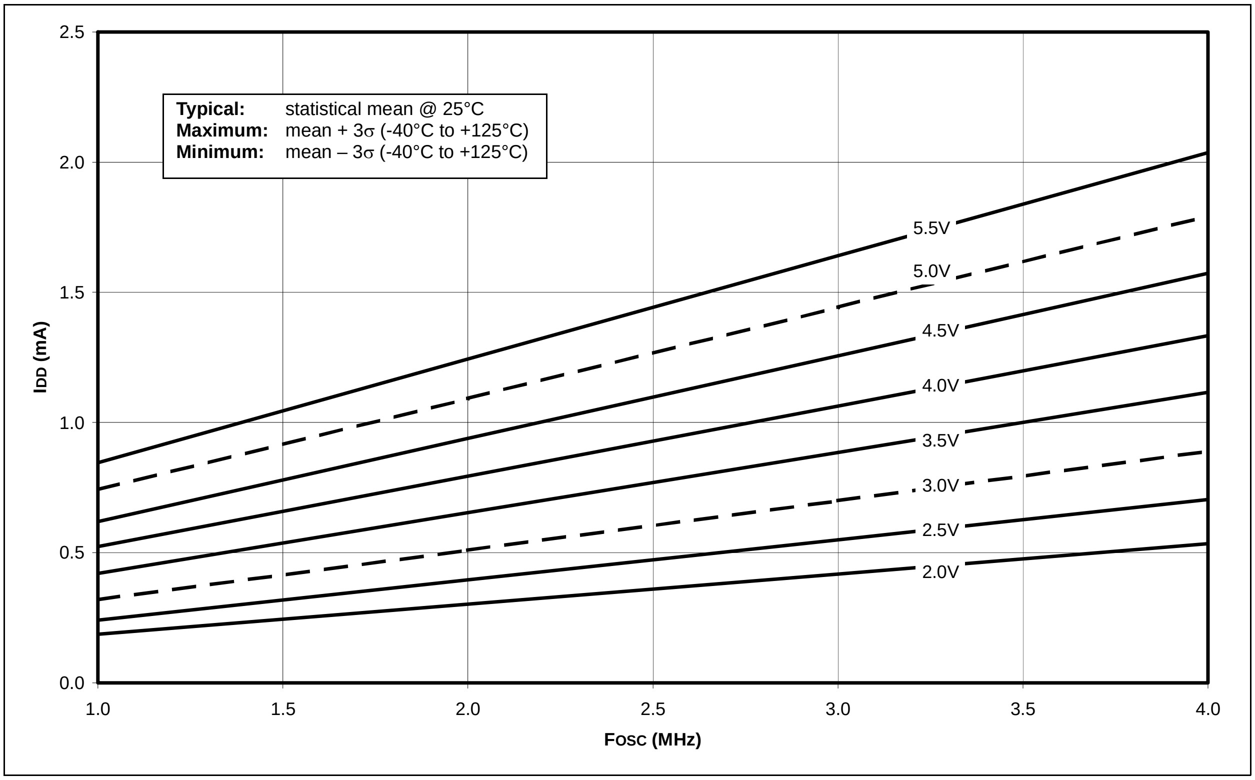


FIGURE 23-6: TYPICAL IDD vs. FOSC OVER VDD PRI_RUN, EC MODE, +25°C