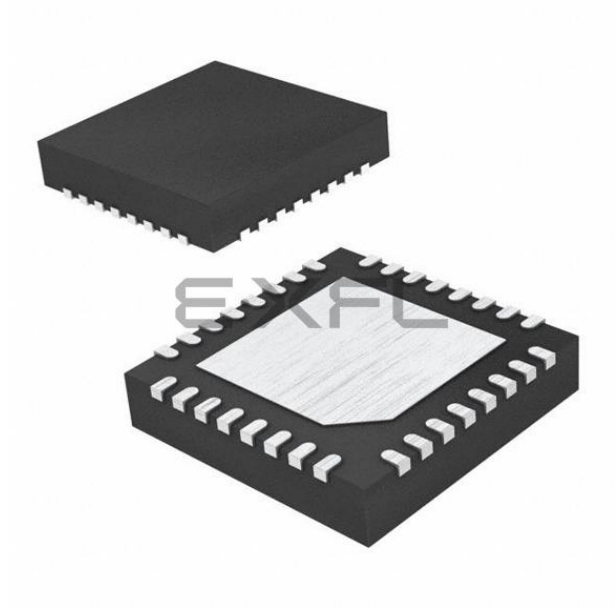
Microchip Technology - PIC18LF2331T-I/MM Datasheet



Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

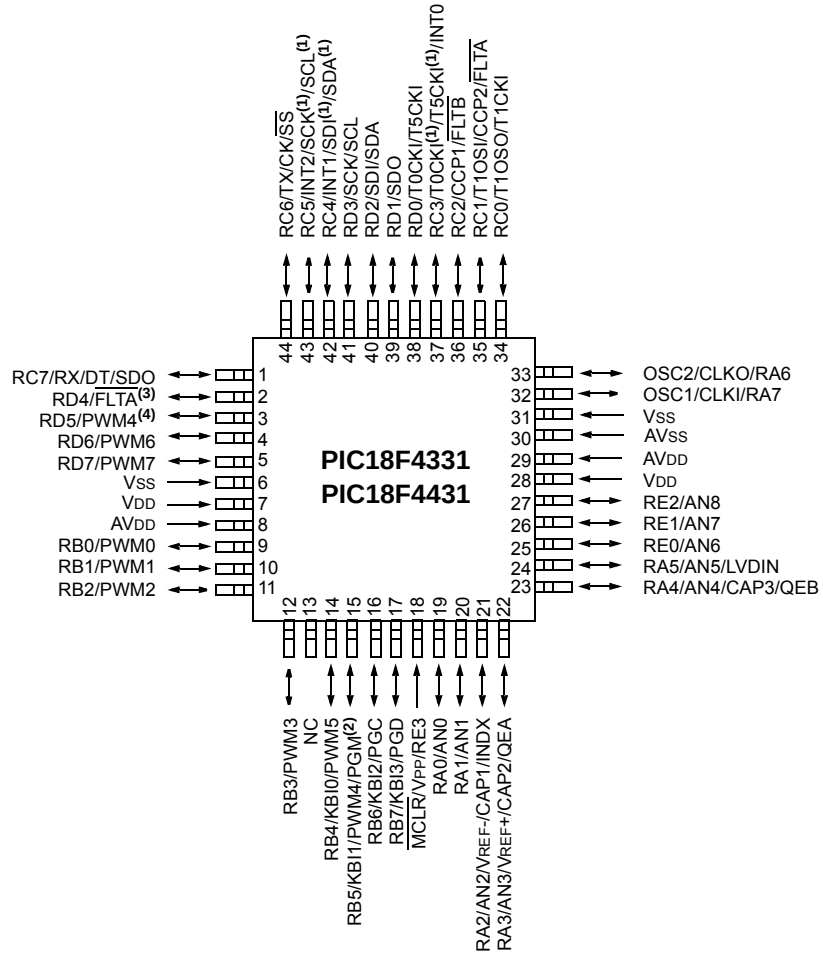
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	40MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, Power Control PWM, QEI, POR, PWM, WDT
Number of I/O	24
Program Memory Size	8KB (4K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 5x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN-S (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf2331t-i-mm

Pin Diagrams (Continued)

44-Pin QFN(2)



Note 1: RC3 is the alternate pin for T0CKI/T5CKI; RC4 is the alternate pin for SDI/SDA; RC5 is the alternate pin for SCK/SCL.

2: For the QFN package, it is recommended that the bottom pad be connected to VSS.

3: RD4 is the alternate pin for FLTA.

4: RD5 is the alternate pin for PWM4.

3.0 OSCILLATOR CONFIGURATIONS

3.1 Oscillator Types

The PIC18F2331/2431/4331/4431 devices can be operated in 10 different oscillator modes. The user can program the Configuration bits, FOSC<3:0>, in Configuration Register 1H to select one of these 10 modes:

1. LP Low-Power Crystal
2. XT Crystal/Resonator
3. HS High-Speed Crystal/Resonator
4. HSPLL High-Speed Crystal/Resonator with PLL Enabled
5. RC External Resistor/Capacitor with FOSC/4 Output on RA6
6. RCIO External Resistor/Capacitor with I/O on RA6
7. INTIO1 Internal Oscillator with FOSC/4 Output on RA6 and I/O on RA7
8. INTIO2 Internal Oscillator with I/O on RA6 and RA7
9. EC External Clock with FOSC/4 Output
10. ECIO External Clock with I/O on RA6

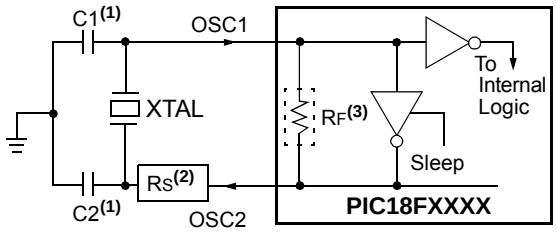
3.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HSPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 3-1 shows the pin connections.

The oscillator design requires the use of a parallel resonant crystal.

Note:	Use of a series resonant crystal may give a frequency out of the crystal manufacturers' specifications.

FIGURE 3-1: CRYSTAL/CERAMIC RESONATOR OPERATION (XT, LP, HS OR HSPLL CONFIGURATION)



Note 1: See Table 3-1 and Table 3-2 for initial values of C1 and C2.

2: A series resistor (RS) may be required for AT strip resonant crystals.

3: RF varies with the oscillator mode chosen.

TABLE 3-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS

Typical Capacitor Values Used:			
Mode	**Freq**	**OSC1**	**OSC2**
XT	455 kHz 2.0 MHz 4.0 MHz	56 pF 47 pF 33 pF	56 pF 47 pF 33 pF
HS	8.0 MHz 16.0 MHz	27 pF 22 pF	27 pF 22 pF

Capacitor values are for design guidance only.

These capacitors were tested with the resonators listed below for basic start-up and operation. **These values are not optimized**.

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

See the notes following Table 3-2 for additional information.

Resonators Used:	
455 kHz	4.0 MHz
2.0 MHz	8.0 MHz
16.0 MHz	

3.6 Internal Oscillator Block

The PIC18F2331/2431/4331/4431 devices include an internal oscillator block, which generates two different clock signals; either can be used as the system's clock source. This can eliminate the need for external oscillator circuits on the OSC1 and/or OSC2 pins.

The main output (INTOSC) is an 8 MHz clock source, which can be used to directly drive the system clock. It also drives a postscaler, which can provide a range of clock frequencies from 125 kHz to 4 MHz. The INTOSC output is enabled when a system clock frequency from 125 kHz to 8 MHz is selected.

The other clock source is the internal RC oscillator (INTRC), which provides a 31 kHz output. The INTRC oscillator is enabled by selecting the internal oscillator block as the system clock source, or when any of the following are enabled:

- Power-up Timer
- Fail-Safe Clock Monitor
- Watchdog Timer
- Two-Speed Start-up

These features are discussed in greater detail in **Section 23.0 "Special Features of the CPU"**.

The clock source frequency (INTOSC direct, INTRC direct or INTOSC postscaler) is selected by configuring the IRCF bits of the OSCCON register (Register 3-2).

3.6.1 INTIO MODES

Using the internal oscillator as the clock source can eliminate the need for up to two external oscillator pins, which can then be used for digital I/O. Two distinct configurations are available:

- In INTIO1 mode, the OSC2 pin outputs FOSC/4, while OSC1 functions as RA7 for digital input and output.
- In INTIO2 mode, OSC1 functions as RA7 and OSC2 functions as RA6, both for digital input and output.

3.6.2 INTRC OUTPUT FREQUENCY

The internal oscillator block is calibrated at the factory to produce an INTOSC output frequency of 8.0 MHz. This changes the frequency of the INTRC source from its nominal 31.25 kHz. Peripherals and features that depend on the INTRC source will be affected by this shift in frequency.

3.6.3 OSCTUNE REGISTER

The internal oscillator's output has been calibrated at the factory, but can be adjusted in the user's application. This is done by writing to the OSCTUNE register (Register 3-1). Each increment may adjust the FRC frequency by varying amounts and may not be monotonic. The next closest frequency may be multiple steps apart.

When the OSCTUNE register is modified, the INTOSC and INTRC frequencies will begin shifting to the new frequency. Code execution continues during this shift. There is no indication that the shift has occurred. Operation of features that depend on the INTRC clock source frequency, such as the WDT, Fail-Safe Clock Monitor and peripherals, will also be affected by the change in frequency.

3.6.4 INTOSC FREQUENCY DRIFT

The factory calibrates the internal oscillator block output (INTOSC) for 8 MHz. This frequency, however, may drift as the VDD or temperature changes, which can affect the controller operation in a variety of ways.

The INTOSC frequency can be adjusted by modifying the value in the OSCTUNE register. This has no effect on the INTRC clock source frequency.

Tuning the INTOSC source requires knowing when to make an adjustment, in which direction it should be made, and in some cases, how large a change is needed. Three compensation techniques are discussed in **Section 3.6.4.1 "Compensating with the EUSART"**, **Section 3.6.4.2 "Compensating with the Timers"** and **Section 3.6.4.3 "Compensating with the CCP Module in Capture Mode"**, but other techniques may be used.

5.0 RESET

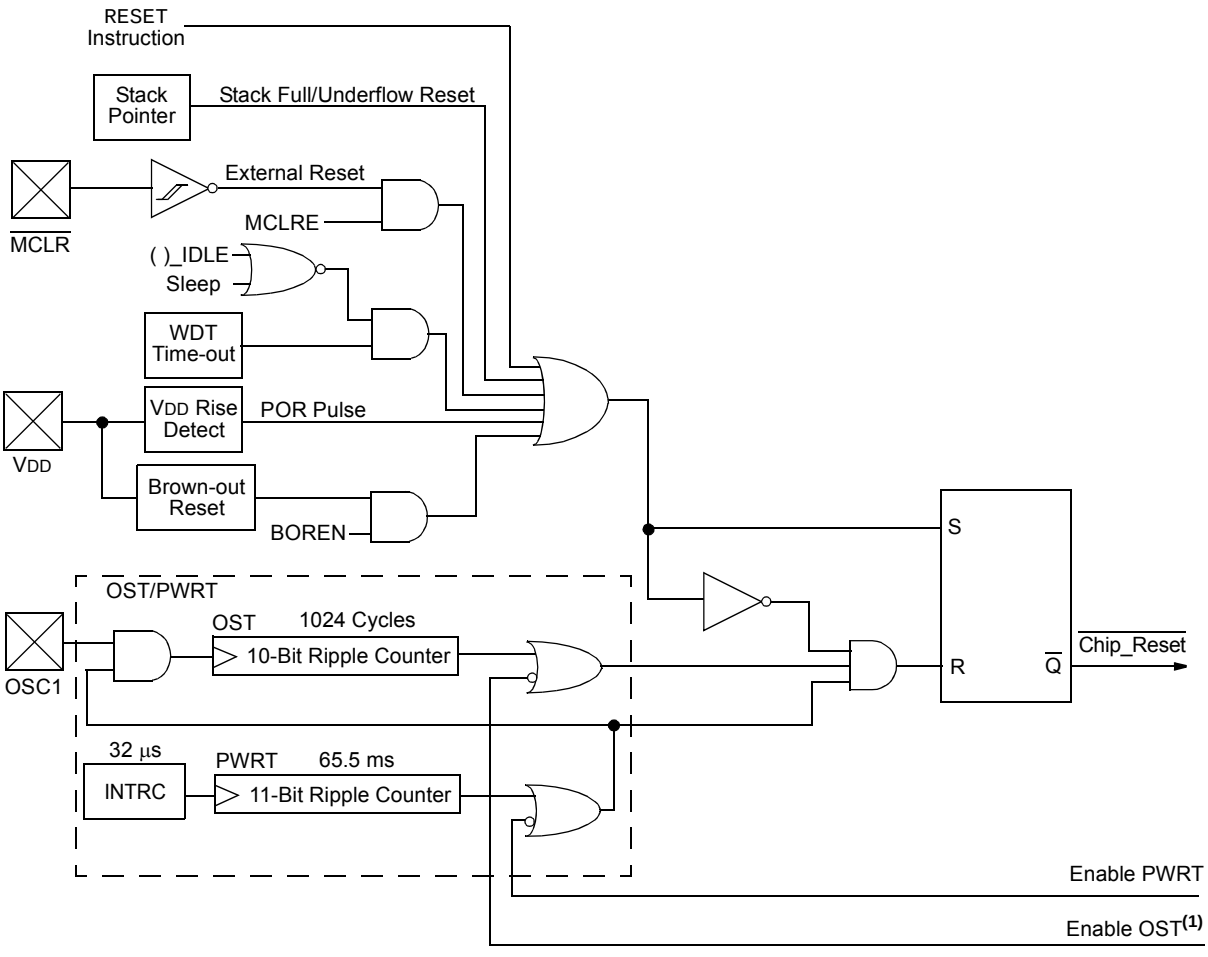
The PIC18F2331/2431/4331/4431 devices differentiate between various kinds of Reset:

a) Power-on Reset (POR)
b) $\overline{\text{MCLR}}$ Reset during normal operation
c) $\overline{\text{MCLR}}$ Reset during Sleep
d) Watchdog Timer (WDT) Reset (during execution)
e) Programmable Brown-out Reset (BOR)
f) RESET Instruction
g) Stack Full Reset
h) Stack Underflow Reset

This section discusses Resets generated by $\overline{\text{MCLR}}$, POR and BOR, and the operation of the various start-up timers. Stack Reset events are covered in **Section 6.1.2.4 "Stack Full/Underflow Resets"**. WDT Resets are covered in **Section 23.2 "Watchdog Timer (WDT)"**.

A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 5-1.

FIGURE 5-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



Note 1: See Table 5-1 for time-out situations.

6.1.2.3 PUSH and POP Instructions

Since the Top-of-Stack (TOS) 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 option. To push the current PC value onto the stack, a PUSH instruction can be executed. This will increment the Stack Pointer and load the current PC value onto the stack. TOSU, TOSH and TOSL can then 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.

6.1.2.4 Stack Full/Underflow Resets

These Resets are enabled by programming the STVREN bit in Configuration Register 4L. When the STVREN bit is cleared, a full or underflow condition will set the appropriate STKFUL or STKUNF bit, but not cause a device Reset. When the STVREN bit is set, a full or underflow condition will set the appropriate STKFUL or STKUNF bit and then cause a device Reset. The STKFUL or STKUNF bits are cleared by the user software or a Power-on Reset.

6.1.3 FAST REGISTER STACK

A Fast Register Stack is provided for the STATUS, WREG and BSR registers, to provide a “fast return” option for interrupts. The stack for each register is only one level deep and is neither readable nor writable. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the stack registers.

The values in the registers are then loaded back into their associated registers if the RETFIE, FAST instruction is used to return from the interrupt. If both low and high-priority interrupts are enabled, the stack registers cannot be used reliably to return from low-priority interrupts. If a high-priority interrupt occurs while servicing a low-priority interrupt, the stack register values stored by the low-priority interrupt will be overwritten. In these cases, users must save the key registers in software during a low-priority interrupt.

If interrupt priority is not used, all interrupts may use the Fast Register Stack for returns from interrupt. If no interrupts are used, the Fast Register Stack can be used to restore the STATUS, WREG and BSR registers at the end of a subroutine call. To use the Fast Register Stack for a subroutine call, a CALL label, FAST instruction must be executed to save the STATUS, WREG and BSR registers to the Fast Register Stack. A RETURN, FAST instruction is then executed to restore these registers from the Fast Register Stack.

Example 6-1 shows a source code example that uses the Fast Register Stack during a subroutine call and return.

EXAMPLE 6-1: FAST REGISTER STACK CODE EXAMPLE

```
CALL  SUB1, FAST    ;STATUS, WREG, BSR
                    ;SAVED IN FAST REGISTER
                    ;STACK
    •
    •

SUB1•
    •
    RETURN FAST     ;RESTORE VALUES SAVED
                    ;IN FAST REGISTER STACK
```

6.1.4 LOOK-UP TABLES IN PROGRAM MEMORY

There may be programming situations that require the creation of data structures, or look-up tables, in program memory. For PIC18 devices, look-up tables can be implemented two ways:

- Computed GOTO
- Table Reads

6.1.4.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in Example 6-2.

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW *nn* instructions. The W register is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW *nn* instructions that returns the value “*nn*” to the calling function.

The offset value (in WREG) specifies the number of bytes that the program counter should advance and should be multiples of 2 (LSb = 0).

In this method, only one data byte can be stored in each instruction location and room on the return address stack is required.

EXAMPLE 6-2: COMPUTED GOTO USING AN OFFSET VALUE

```
        MOVFW     OFFSET
        CALL      TABLE
ORG     0xnn00
TABLE   ADDWF     PCL
        RETLW     0xnn
        RETLW     0xnn
        RETLW     0xnn
          .
          .
          .
```

8.0 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 on one byte at a time. A write to program memory is executed on blocks of 8 bytes at a time. Program memory is erased in blocks of 64 bytes at a time. A bulk erase operation may not be issued from user code.

While writing or erasing program memory, instruction fetches cease 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.

8.1 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)

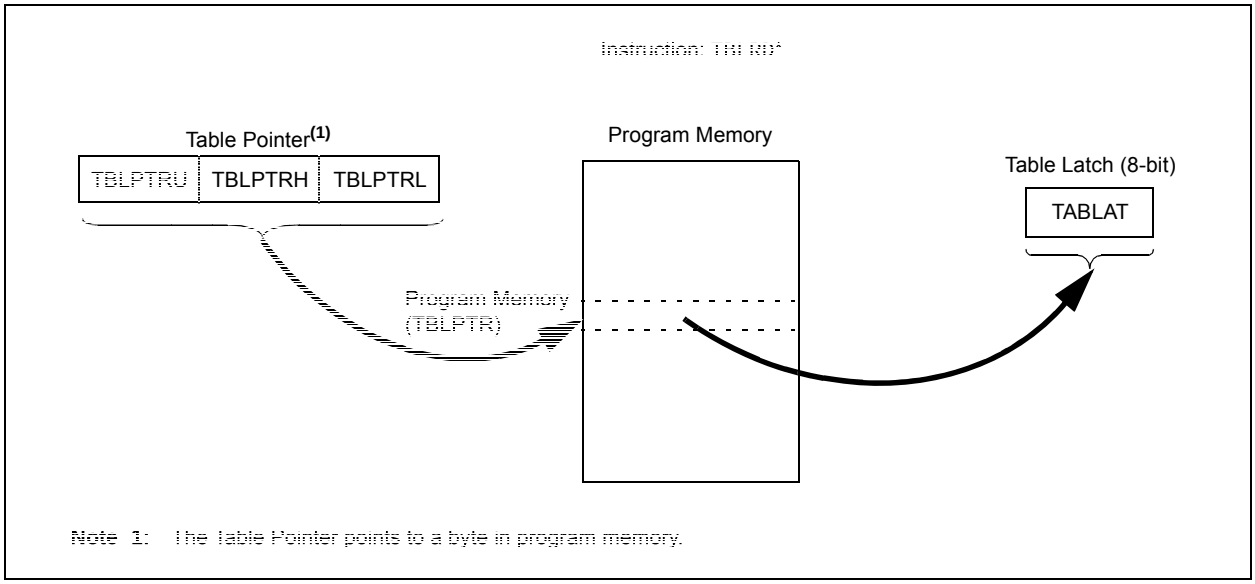
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into TABLAT in the data RAM space. Figure 8-1 shows the operation of a table read with program memory and data RAM.

Table write operations store data from TABLAT in the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in **Section 8.5 "Writing to Flash Program Memory"**. Figure 8-2 shows the operation of a table write with program memory and data RAM.

Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word-aligned. Therefore, a table block 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, (TBLPTRL<0> = 0).

FIGURE 8-1: TABLE READ OPERATION



8.5.1 FLASH PROGRAM MEMORY WRITE SEQUENCE

The sequence of events for programming an internal program memory location should be:

1. Read 64 bytes into RAM.
2. Update data values in RAM as necessary.
3. Load Table Pointer with address being erased.
4. Do the row erase procedure (see **Section 8.4.1 "Flash Program Memory Erase Sequence"**).
5. Load Table Pointer with the address of the first byte being written.
6. Write the first 8 bytes into the holding registers with auto-increment.
7. Set the EECON1 register for the write operation by doing the following:
   - Set the EEPGD bit to point to program memory
   - Clear the CFGS bit to access program memory
   - Set the WREN bit to enable byte writes
8. Disable interrupts.
9. Write 55h to EECON2.
10. Write 0AAh to EECON2.
11. Set the WR bit. This will begin the write cycle.
12. The CPU will stall for the duration of the write (about 2 ms using internal timer).
13. Execute a NOP.
14. Re-enable interrupts.
15. Repeat Steps 6-14 seven times to write 64 bytes.
16. Verify the memory (table read).

This procedure will require about 18 ms to update one row of 64 bytes of memory. An example of the required code is given in Example 8-3.

REGISTER 10-9: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	PTIE	IC3DRIE	IC2QEIE	IC1IE	TMR5IE
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 **PTIE:** PWM Time Base Interrupt Enable bit
1 = PTIF enabled
0 = PTIF disabled

bit 3 **IC3DRIE:** IC3 Interrupt Enable/Direction Change Interrupt Enable bit
IC3 Enabled (CAP3CON<3:0>):
1 = IC3 interrupt enabled
0 = IC3 interrupt disabled
QEI Enabled (QEIM<2:0>):
1 = Change of direction interrupt enabled
0 = Change of direction interrupt disabled

bit 2 **IC2QEIE:** IC2 Interrupt Flag/QEI Interrupt Flag Enable bit
IC2 Enabled (CAP2CON<3:0>):
1 = IC2 interrupt enabled)
0 = IC2 interrupt disabled
QEI Enabled (QEIM<2:0>):
1 = QEI interrupt enabled
0 = QEI interrupt disabled

bit 1 **IC1IE:** IC1 Interrupt Enable bit
1 = IC1 interrupt enabled
0 = IC1 interrupt disabled

bit 0 **TMR5IE:** Timer5 Interrupt Enable bit
1 = Timer5 interrupt enabled
0 = Timer5 interrupt disabled

14.2 Timer2 Interrupt

Timer2 can also generate an optional device interrupt. The Timer2 output signal (TMR2 to PR2 match) provides the input for the 4-bit output counter/postscaler. This counter generates the TMR2 match interrupt flag which is latched in TMR2IF (PIR1<1>).

The interrupt is enabled by setting the TMR2 Match Interrupt Enable bit, TMR2IE (PIE1<1>). A range of 16 postscale options (from 1:1 through 1:16 inclusive) can be selected with the postscaler control bits, T2OUTPS<3:0> (T2CON<6:3>).

14.3 Output of TMR2

The unscaled output of TMR2 is available primarily to the CCP modules, where it is used as a time base for operations in PWM mode. Timer2 can be optionally used as the shift clock source for the SSP module operating in SPI mode.

For additional information, see **Section 19.0 "Synchronous Serial Port (SSP) Module"**.

FIGURE 14-1: TIMER2 BLOCK DIAGRAM

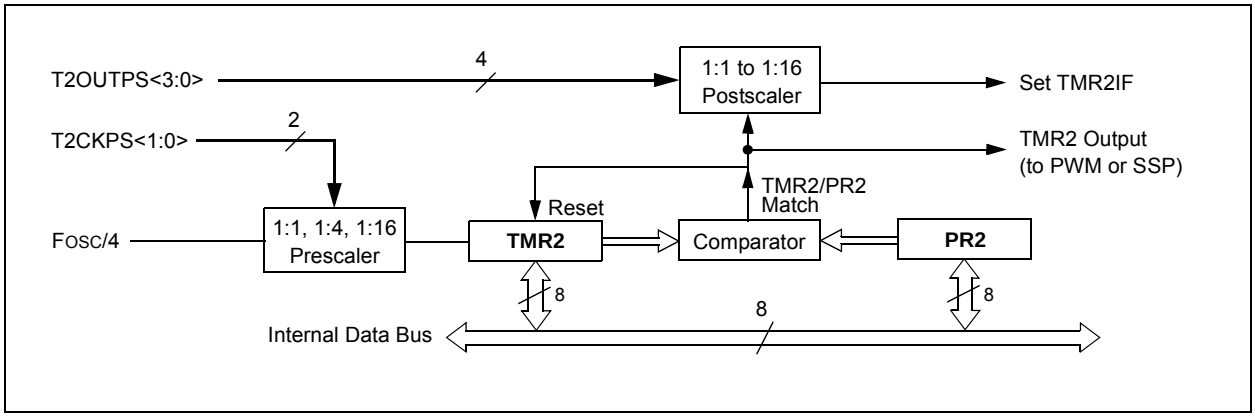


TABLE 14-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	54
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	57
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	57
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	57
TMR2	Timer2 Register								55
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	55
PR2	Timer2 Period Register								55

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.

16.3 Capture Mode

In Capture mode, CCPR1H:CCPR1L captures the 16-bit value of the TMR1 register when an event occurs on pin RC2/CCP1. An event is defined as one of the following:

- every falling edge
- every rising edge
- every 4th rising edge
- every 16th rising edge

The event is selected by control bits, CCP1M<3:0> (CCP1CON<3:0>). When a capture is made, the interrupt request flag bit, CCP1IF (PIR1<2>), is set; it must be cleared in software. If another capture occurs before the value in register CCPR1 is read, the old captured value is overwritten by the new captured value.

16.3.1 CCP PIN CONFIGURATION

In Capture mode, the RC2/CCP1 pin should be configured as an input by setting the TRISC<2> bit.

Note:	If the RC2/CCP1 pin is configured as an output, a write to the port can cause a capture condition.

16.3.2 TIMER1 MODE SELECTION

Timer1 must be running in Timer mode or Synchronized Counter mode to be used with the capture feature. In Asynchronous Counter mode, the capture operation may not work.

16.3.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep bit, CCP1IE (PIE1<2>), clear to avoid false interrupts and should clear the flag bit, CCP1IF, following any such change in operating mode.

16.3.4 CCP PRESCALER

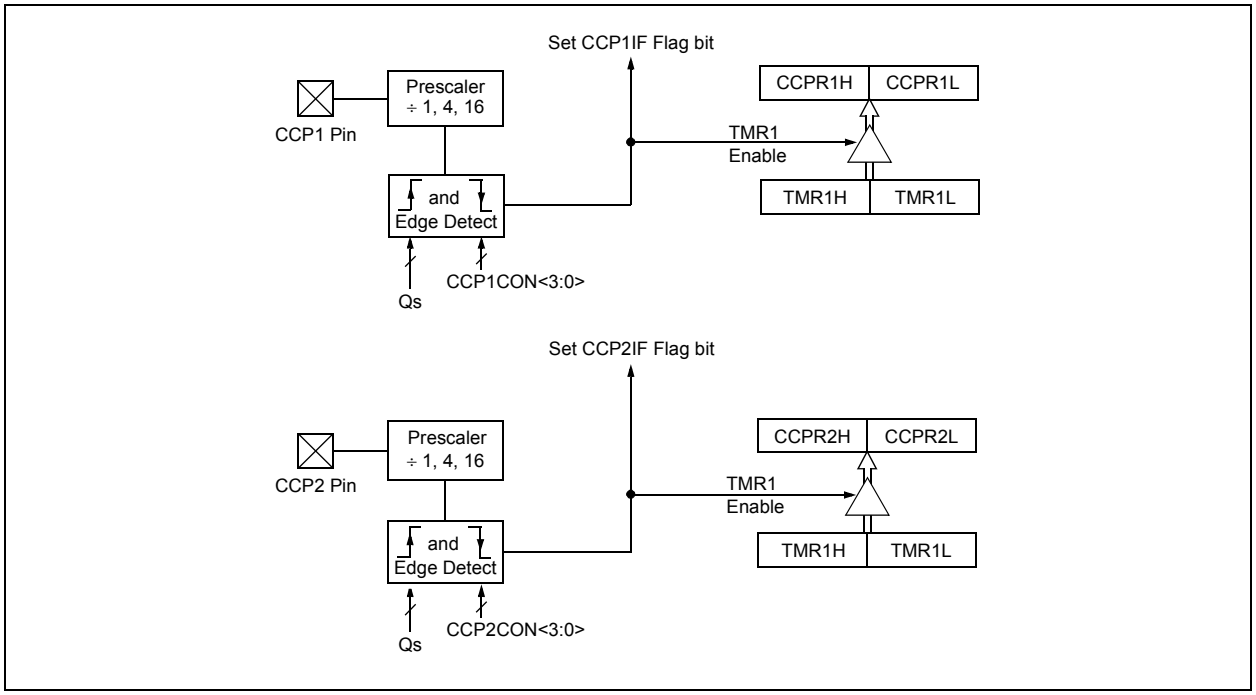
There are four prescaler settings specified by bits CCP1M<3:0>. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared, therefore, the first capture may be from a non-zero prescaler. Example 16-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

EXAMPLE 16-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
CLRF    CCP1CON         ; Turn CCP module off
MOVLW   NEW_CAPT_PS     ; Load WREG with the
                        ; new prescaler mode
                        ; value and CCP ON
MOVWF   CCP1CON         ; Load CCP1CON with
                        ; this value
```

FIGURE 16-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



When in Counter mode, the counter must be configured as the synchronous counter only ($\overline{T5SYNC}$ = 0). When configured in Asynchronous mode, the IC module will not work properly.

Note 1: Input capture prescalers are reset (cleared) when the input capture module is disabled (CAPxM = 0000).

2: When the Input Capture mode is changed, without first disabling the module and entering the new Input Capture mode, a false interrupt (or Special Event Trigger on IC1) may be generated. The user should either: (1) disable the input capture before entering another mode, or (2) disable IC interrupts to avoid false interrupts during IC mode changes.

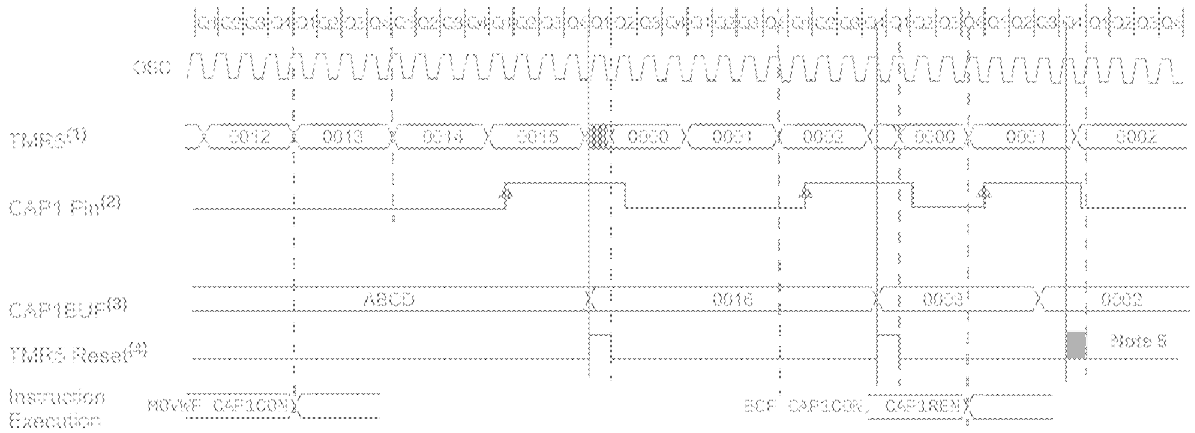
3: During IC mode changes, the prescaler count will not be cleared, therefore, the first capture in the new IC mode may be from the non-zero prescaler.

17.1.1 EDGE CAPTURE MODE

In this mode, the value of the time base is captured either on every rising edge, every falling edge, every 4th rising edge, or every 16th rising edge. The edge present on the input capture pin (CAP1, CAP2 or CAP3) is sampled by the synchronizing latch. The signal is used to load the Input Capture Buffer (ICxBUF register) on the following Q1 clock (see Figure 17-4). Consequently, Timer5 is either reset to '0' (Q1 immediately following the capture event) or left free running, depending on the setting of the Capture Reset Enable bit, CAPxREN, in the CAPxCON register.

Note: On the first capture edge following the setting of the Input Capture mode (i.e., MOVWF CAP1CON), Timer5 contents are always captured into the corresponding Input Capture Buffer (i.e., CAPxBUF). Timer5 can optionally be reset; however, this is dependent on the setting of the Capture Reset Enable bit, CAPxREN (see Figure 17-4).

FIGURE 17-4: EDGE CAPTURE MODE TIMING



Note 1: TMR5 is a synchronous time base input to the input capture; prescaler = 1:1. It increments on the Q1 rising edge.

2: IC1 is configured in Edge Capture mode (CAP1M<3:0> = 0010) with the time base reset upon edge capture (CAP1REN = 1) and no noise filter.

3: TMR5 value is latched by CAP1BUF on TCY. In the event that a write to TMR5 coincides with an input capture event, the write will always take precedence. All Input Capture Buffers, CAP1BUF, CAP2BUF and CAP3BUF, are updated with the incremented value of the time base on the next TCY clock edge when the capture event takes place (see Note 4 when Reset occurs).

4: TMR5 Reset is normally an asynchronous Reset signal to TMR5. When used with the input capture, it is active immediately after the time base value is captured.

5: TMR5 Reset pulse is disabled by clearing the CAP1REN bit (e.g., BCF CAP1CON, CAP1REN).

18.4.4 INTERRUPTS IN DOUBLE UPDATE MODE

This mode is available in Continuous Up/Down Count mode. In the Double Update mode (PTMOD<1:0> = 11), an interrupt event is generated each time the PTMR register is equal to zero and each time the PTMR matches with PTPER register. Figure 18-8 shows the interrupts in Continuous Up/Down Count mode with double updates.

The Double Update mode provides two additional functions to the user in Center-Aligned mode.

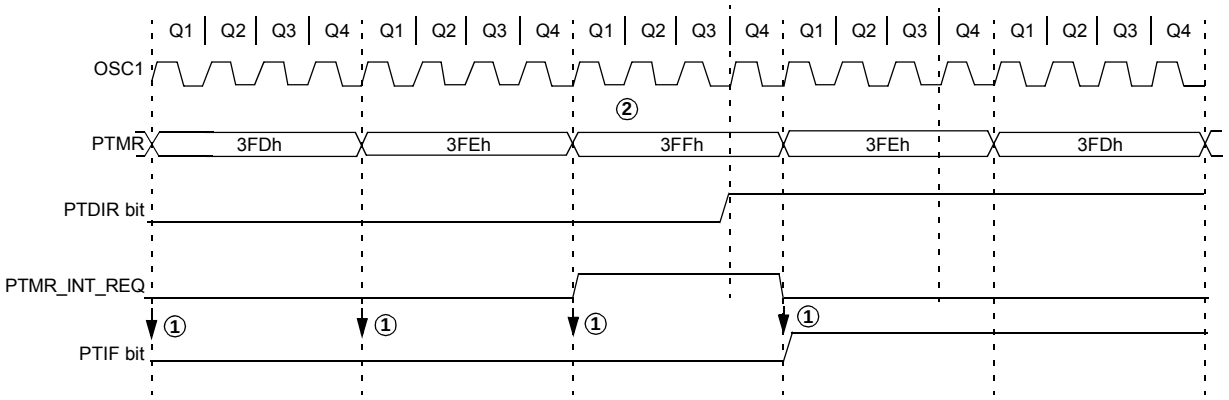
1. The control loop bandwidth is doubled because the PWM duty cycles can be updated twice per period.
2. Asymmetrical center-aligned PWM waveforms can be generated, which are useful for minimizing output waveform distortion in certain motor control applications.

Note:	Do not change the PTMOD bits while PTEN is active; it will yield unexpected results. To change the PWM Timer mode of operation, first clear the PTEN bit, load the PTMOD bits with the required data and then set PTEN.

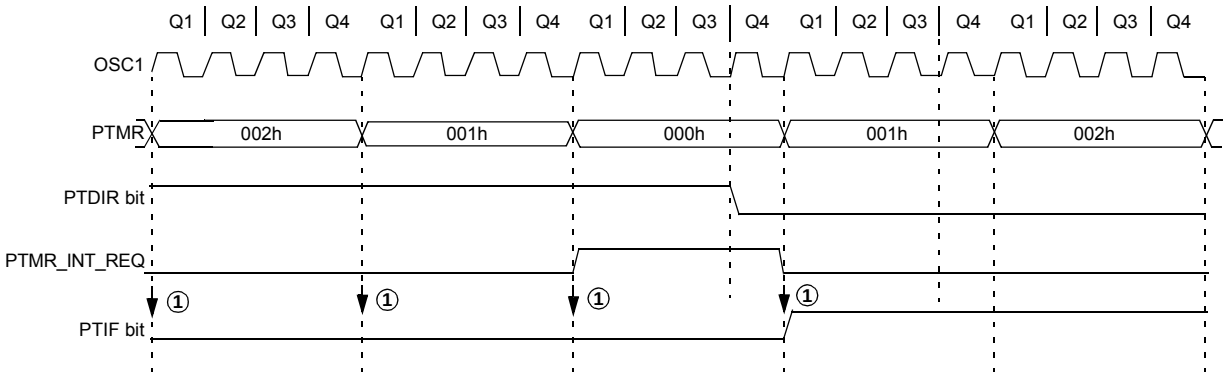
FIGURE 18-8: PWM TIME BASE INTERRUPT, CONTINUOUS UP/DOWN COUNT MODE WITH DOUBLE UPDATES

A: PRESCALER = 1:1

Case 1: PTMR Counting Upwards



Case 2: PTMR Counting Downwards



Note 1: Interrupt flag bit, PTIF, is sampled here (every Q1).

2: PWM Time Base Period register, PTPER, is loaded with the value, 3FFh, for this example.

TABLE 20-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	54
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	57
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	57
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	57
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	56
TXREG	EUSART Transmit Register								56
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	56
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	56
SPBRGH	EUSART Baud Rate Generator Register High Byte								56
SPBRG	EUSART Baud Rate Generator Register Low Byte								56

Legend: — = unimplemented, read as '0'. Shaded cells are not used for asynchronous transmission.

20.3.5 BREAK CHARACTER SEQUENCE

The Enhanced USART module has the capability of sending the special Break character sequences that are required by the LIN/J2602 bus standard. The Break character transmit consists of a Start bit, followed by twelve '0' bits and a Stop bit. The Frame Break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift register is loaded with data. Note that the value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN/J2602 specification).

Note that the data value written to the TXREG for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 20-9 for the timing of the Break character sequence.

20.3.5.1 Break and Sync Transmit Sequence

The following sequence will send a message frame header made up of a Break, followed by an Auto-Baud Sync byte. This sequence is typical of a LIN/J2602 bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to setup the Break character.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware. The Sync character now transmits in the preconfigured mode.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

20.3.6 RECEIVING A BREAK CHARACTER

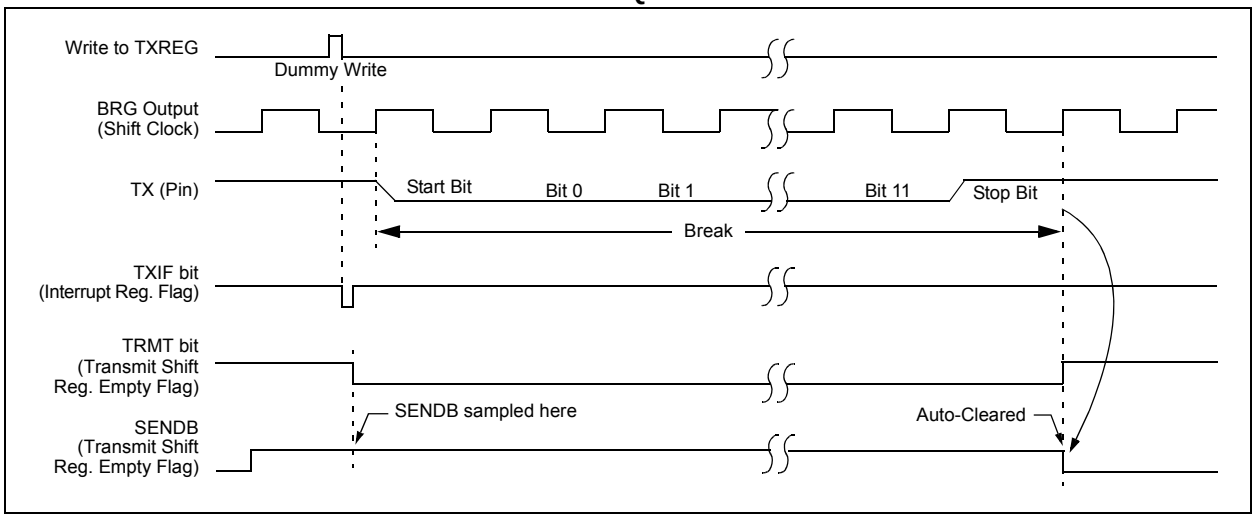
The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 of the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for Break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 20.3.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCIF interrupt and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Rate Detect feature. For both methods, the user can set the ABD bit before placing the EUSART in its Sleep mode.

FIGURE 20-9: SEND BREAK CHARACTER SEQUENCE



REGISTER 23-6: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)

R/P-1	U-0	U-0	U-0	U-0	R/P-1	U-0	R/P-1
$\overline{\text{DEBUG}}$	—	—	—	—	LVP	—	STVREN
bit 7							bit 0

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
-n = Value when device is unprogrammed		U = Unchanged from programmed state

bit 7 **$\overline{\text{DEBUG}}$:** Background Debugger Enable bit
1 = Background debugger is disabled; RB6 and RB7 are configured as general purpose I/O pins
0 = Background debugger is enabled; RB6 and RB7 are dedicated to In-Circuit Debug

bit 6-3 **Unimplemented:** Read as '0'

bit 2 **LVP:** Single-Supply ICSP™ Enable bit
1 = Single-Supply ICSP is enabled
0 = Single-Supply ICSP is disabled

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

bit 0 **STVREN:** Stack Full/Underflow Reset Enable bit
1 = Stack full/underflow will cause Reset
0 = Stack full/underflow will not cause Reset

23.2 Watchdog Timer (WDT)

For PIC18F2331/2431/4331/4431 devices, the WDT is driven by the INTRC source. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the INTRC oscillator.

The 4 ms period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexer, controlled by bits in Configuration Register 2H (see Register 23-3). Available periods range from 4 ms to 131.072 seconds (2.18 minutes). The WDT and postscaler are cleared when any of the following events occur: execute a SLEEP or CLRWDT instruction, the IRCF bits (OSCCON<6:4>) are changed or a clock failure has occurred (see **Section 23.4.1 "FSCM and the Watchdog Timer"**).

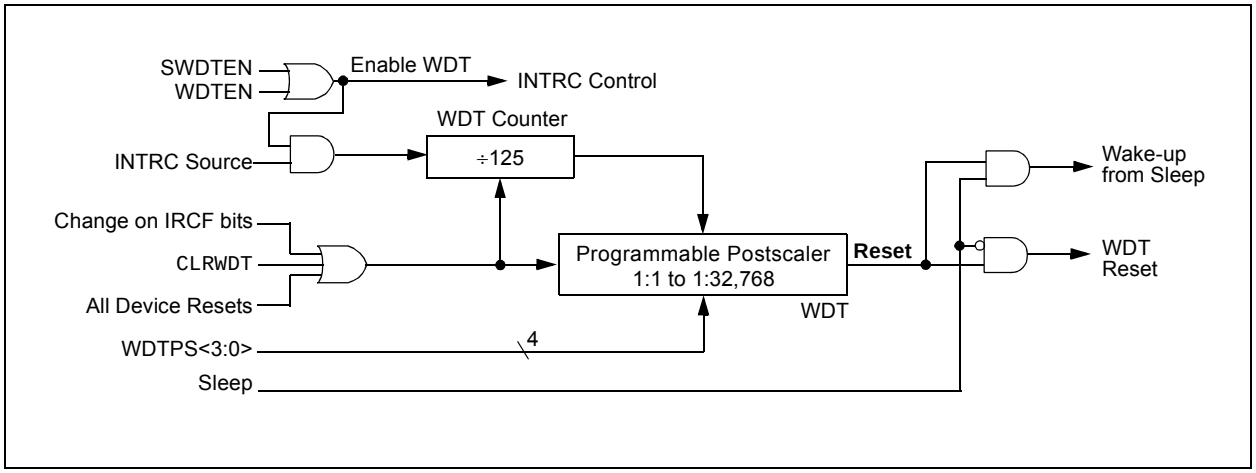
Adjustments to the internal oscillator clock period using the OSCTUNE register also affect the period of the WDT by the same factor. For example, if the INTRC period is increased by 3%, then the WDT period is increased by 3%.

Note 1:	The CLRWDT and SLEEP instructions clear the WDT and postscaler counts when executed.
2:	Changing the setting of the IRCF bits (OSCCON<6:4>) clears the WDT and postscaler counts.
3:	When a CLRWDT instruction is executed, the postscaler count will be cleared.
4:	If $\overline{\text{WINEN}}$ = 0, then CLRWDT must be executed only when WDTW = 1; otherwise, a device Reset will result.

23.2.1 CONTROL REGISTER

Register 23-15 shows the WDTCON register. This is a readable and writable register. The SWDTEN bit allows software to enable or disable the WDT, but only if the Configuration bit has disabled the WDT. The WDTW bit is a read-only bit that indicates when the WDT count is in the fourth quadrant (i.e., when the 8-bit WDT value is b'11000000' or greater).

FIGURE 23-1: WDT BLOCK DIAGRAM



COMF **Complement f**

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

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

Operation: $(\bar{f}) \rightarrow$ dest

Status Affected: N, Z

Encoding:

0001	11da	ffff	ffff

Description: The contents of register, 'f', are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register, 'f'. 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.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: COMF REG, W

Before Instruction
REG = 0x13
After Instruction
REG = 0x13
W = 0xEC

CPFSEQ **Compare f with W, Skip if f = W**

Syntax: [*label*] CPFSEQ f [,a]

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

Operation: (f) – (W),
skip if (f) = (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	001a	ffff	ffff

Description: Compares the contents of data memory location, 'f', to the contents of W by performing an unsigned subtraction. If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-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.

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	No operation

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    CPFSEQ REG
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

DAW **Decimal Adjust W Register**

Syntax: [*label*] DAW

Operands: None

Operation: If [W<3:0> > 9] or [DC = 1] then,
(W<3:0>) + 6 → W<3:0>;
else,
(W<3:0>) → W<3:0>;

If [W<7:4> 9] or [C = 1] then,
(W<7:4>) + 6 → W<7:4>;
else,
(W<7:4>) → W<7:4>

Status Affected: C, DC

Encoding:

0000	0000	0000	0111

Description: DAW adjusts the 8-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result. The Carry bit may be set by DAW regardless of its setting prior to the DAW instruction.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example 1: DAW

Before Instruction
W = 0xA5
C = 0
DC = 0

After Instruction
W = 0x05
C = 1
DC = 0

Example 2:

Before Instruction
W = 0xCE
C = 0
DC = 0

After Instruction
W = 0x34
C = 1
DC = 0

DECF **Decrement f**

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

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

Operation: (f) – 1 → dest

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

Encoding:

0000	01da	ffff	ffff

Description: Decrement register, ‘f’,. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register, ‘f’. 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.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: DECF CNT,

Before Instruction
CNT = 0x01
Z = 0

After Instruction
CNT = 0x00
Z = 1

MOVFF **Move f to f**

Syntax: [*label*] MOVFF f_s,f_d

Operands: $0 \le f_s \le 4095$
$0 \le f_d \le 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1st word (source)	1100	ffff	ffff	$ffff_s$
2nd word (destin.)	1111	ffff	ffff	$ffff_d$

Description: The contents of source register, 'f_s', are moved to destination register, 'f_d'. Location of source, 'f_s', can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination, 'f_d', can also be anywhere from 000h to FFFh.
Either source or destination can be W (a useful special situation).
MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).
The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.
The MOVFF instruction should not be used to modify interrupt settings while any interrupt is enabled (see the note on page 97).

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation No dummy read	No operation	Write register 'f' (dest)

Example: MOVFF REG1, REG2

Before Instruction
REG1 = 0x33
REG2 = 0x11
After Instruction
REG1 = 0x33
REG2 = 0x33

MOVLB **Move Literal to Low Nibble in BSR**

Syntax: [*label*] MOVLB k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow BSR$

Status Affected: None

Encoding:

0000	0001	0000	kkkk

Description: The 8-bit literal, 'k', is loaded into the Bank Select Register (BSR).

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: MOVLB 5

Before Instruction
BSR register = 0x02
After Instruction
BSR register = 0x05

26.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 26-5: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)

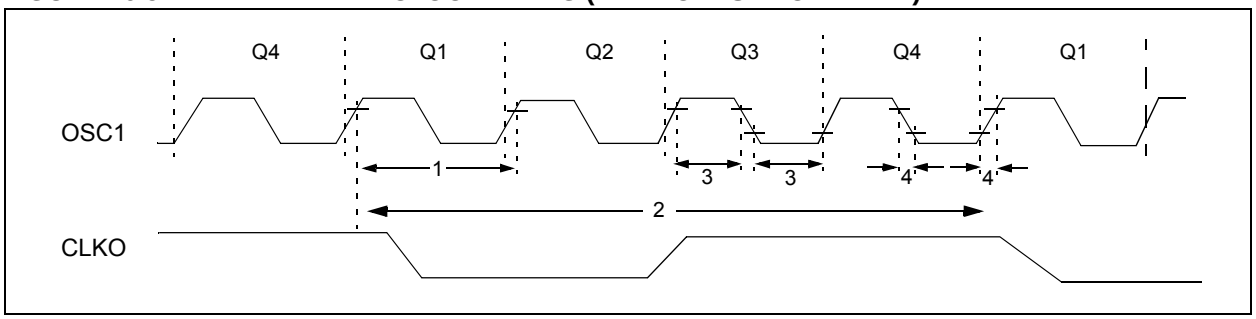


TABLE 26-4: EXTERNAL CLOCK TIMING REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	FOSC	External CLKI Frequency(1)	DC	40	MHz	EC, ECIO
		Oscillator Frequency(1)	DC	4	MHz	RC osc
			0.1	4	MHz	XT osc
			4	25	MHz	HS osc
			4	10	MHz	HS + PLL osc
			5	200	kHz	LP Osc mode
1	TOSC	External CLKI Period(1)	25	—	ns	EC, ECIO
		Oscillator Period(1)	250	—	ns	RC osc
			250	10,000	ns	XT osc
			25	250	ns	HS osc
			100	250	ns	HS + PLL osc
			25	—	µs	LP osc
2	TCY	Instruction Cycle Time(1)	100	—	ns	TCY = 4/FOSC
3	TosL, TosH	External Clock in (OSC1) High or Low Time	30	—	ns	XT osc
			2.5	—	µs	LP osc
			10	—	ns	HS osc
4	TosR, TosF	External Clock in (OSC1) Rise or Fall Time	—	20	ns	XT osc
			—	50	ns	LP osc
			—	7.5	ns	HS osc

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time base period for all configurations except PLL. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1/CLKI pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.