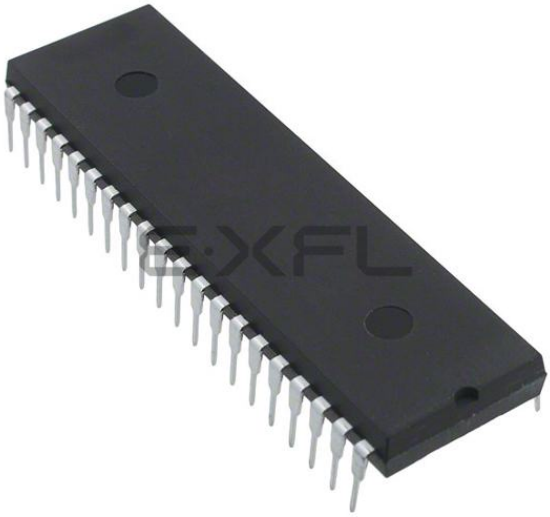


**Welcome to E-XFL.COM**

## 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 | CANbus, I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, HLVD, POR, PWM, WDT |
| Number of I/O | 36 |
| Program Memory Size | 96KB (48K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 1K x 8 |
| RAM Size | 3.25K x 8 |
| Voltage - Supply (Vcc/Vdd) | 4.2V ~ 5.5V |
| Data Converters | A/D 11x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Through Hole |
| Package / Case | 40-DIP (0.600", 15.24mm) |
| Supplier Device Package | 40-PDIP |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic18f4685-i-p |

**TABLE 1-3: PIC18F4682/4685 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | | | Pin Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| | PDIP | QFN | TQFP | | | |
| | | | | | | PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. |
| RB0/INT0/FLT0/AN10 | 33 | 9 | 8 | | | |
| RB0 | | | | I/O | TTL | Digital I/O. |
| INT0 | | | | I | ST | External interrupt 0. |
| FLT0 | | | | I | ST | Enhanced PWM Fault input (ECCP1 module). |
| AN10 | | | | I | Analog | Analog input 10. |
| RB1/INT1/AN8 | 34 | 10 | 9 | | | |
| RB1 | | | | I/O | TTL | Digital I/O. |
| INT1 | | | | I | ST | External interrupt 1. |
| AN8 | | | | I | Analog | Analog input 8. |
| RB2/INT2/CANTX | 35 | 11 | 10 | | | |
| RB2 | | | | I/O | TTL | Digital I/O. |
| INT2 | | | | I | ST | External interrupt 2. |
| CANTX | | | | O | TTL | CAN bus TX. |
| RB3/CANRX | 36 | 12 | 11 | | | |
| RB3 | | | | I/O | TTL | Digital I/O. |
| CANRX | | | | I | TTL | CAN bus RX. |
| RB4/KBI0/AN9 | 37 | 14 | 14 | | | |
| RB4 | | | | I/O | TTL | Digital I/O. |
| KBI0 | | | | I | TTL | Interrupt-on-change pin. |
| AN9 | | | | I | Analog | Analog input 9. |
| RB5/KBI1/PGM | 38 | 15 | 15 | | | |
| RB5 | | | | I/O | TTL | Digital I/O. |
| KBI1 | | | | I | TTL | Interrupt-on-change pin. |
| PGM | | | | I/O | ST | Low-Voltage ICSP™ Programming enable pin. |
| RB6/KBI2/PGC | 39 | 16 | 16 | | | |
| RB6 | | | | I/O | TTL | Digital I/O. |
| KBI2 | | | | I | TTL | Interrupt-on-change pin. |
| PGC | | | | I/O | ST | In-Circuit Debugger and ICSP programming clock pin. |
| RB7/KBI3/PGD | 40 | 17 | 17 | | | |
| RB7 | | | | I/O | TTL | Digital I/O. |
| KBI3 | | | | I | TTL | Interrupt-on-change pin. |
| PGD | | | | I/O | ST | In-Circuit Debugger and ICSP programming data pin. |

**Legend:** TTL = TTL compatible input
CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output
P = Power

## 2.4 RC Oscillator

For timing insensitive applications, the “RC” and “RCIO” device options offer additional cost savings. The actual oscillator frequency is a function of several factors:

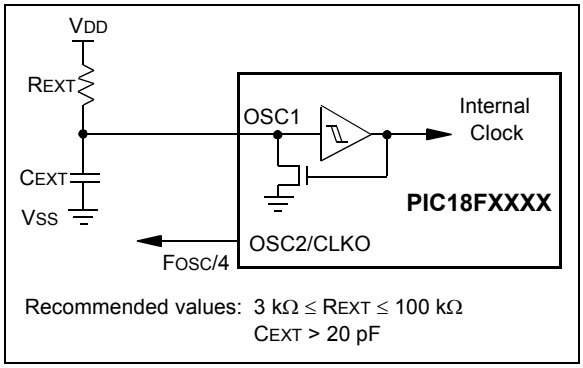
- supply voltage
- values of the external resistor (REXT) and capacitor (CEXT)
- operating temperature

Given the same device, operating voltage and temperature and component values, there will also be unit-to-unit frequency variations. These are due to factors such as:

- normal manufacturing variation
- difference in lead frame capacitance between package types (especially for low CEXT values)
- variations within the tolerance of limits of REXT and CEXT

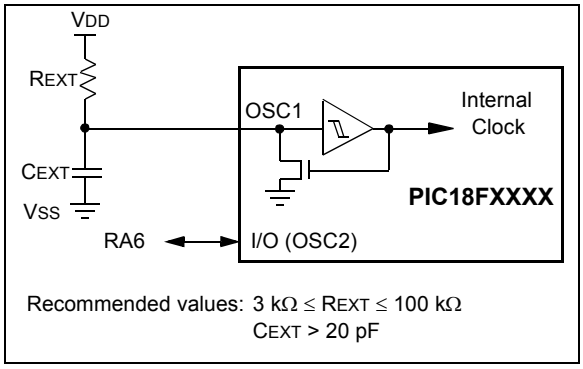
In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-5 shows how the R/C combination is connected.

**FIGURE 2-5: RC OSCILLATOR MODE**




The RCIO Oscillator mode (Figure 2-6) functions like the RC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

**FIGURE 2-6: RCIO OSCILLATOR MODE**




## 2.5 PLL Frequency Multiplier

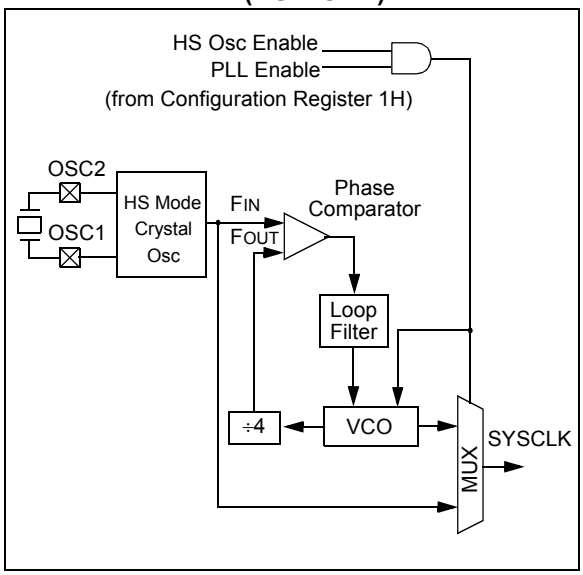
A Phase Locked Loop (PLL) circuit is provided as an option for users who wish to use a lower frequency oscillator circuit or to clock the device up to its highest rated frequency from a crystal oscillator. This may be useful for customers who are concerned with EMI due to high-frequency crystals or users who require higher clock speeds from an internal oscillator.

### 2.5.1 HSPLL OSCILLATOR MODE

The HSPLL mode makes use of the HS mode oscillator for frequencies up to 10 MHz. A PLL then multiplies the oscillator output frequency by 4 to produce an internal clock frequency up to 40 MHz.

The PLL is only available to the crystal oscillator when the FOSC3:FOSC0 Configuration bits are programmed for HSPLL mode (= 0110).

**FIGURE 2-7: PLL BLOCK DIAGRAM (HS MODE)**




### 2.5.2 PLL AND INTOSC

The PLL is also available to the internal oscillator block in selected oscillator modes. In this configuration, the PLL is enabled in software and generates a clock output of up to 32 MHz. The operation of INTOSC with the PLL is described in **Section 2.6.4 “PLL in INTOSC Modes”**.

**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
                MOVWF   TBLPTRL
READ_BLOCK
                TBLRD*+                         ; read into TABLAT, and inc
                MOVF    TABLAT, W               ; get data
                MOVWF   POSTINC0                ; store data
                DECFSZ  COUNTER                 ; done?
                BRA     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
                MOVWF   POSTINC0
                MOVLW   NEW_DATA_HIGH
                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
                MOVWF   TBLPTRL
                BSF     EECON1, EEPGD           ; point to Flash program memory
                BCF     EECON1, CFGS            ; access 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        MOVWF   EECON2                  ; write 55h
Sequence        MOVLW   0AAh
                MOVWF   EECON2                  ; write 0AAh
                BSF     EECON1, WR              ; start erase (CPU stall)
                BSF     INTCON, GIE             ; re-enable interrupts
                TBLRD*-                         ; dummy read decrement
                MOVLW   BUFFER_ADDR_HIGH        ; point to buffer
                MOVWF   FSR0H
                MOVLW   BUFFER_ADDR_LOW
                MOVWF   FSR0L
WRITE_BUFFER_BACK
                MOVLW   D’64                    ; number of bytes in holding register
                MOVWF   COUNTER
WRITE_BYTE_TO_HREGS
                MOVF    POSTINC0, W             ; get low byte of buffer data
                MOVWF   TABLAT                  ; present data to table latch
                TBLWT+*                         ; write data, perform a short write
                                                ; to internal TBLWT holding register.
                DECFSZ  COUNTER                 ; loop until buffers are full
                BRA     WRITE_BYTE_TO_HREGS
```

## 7.3 Reading the Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADRH:EEADR register pair, clear the EEPGD control bit (EECON1<7>) and then set control bit, RD (EECON1<0>). The data is available on the very next instruction cycle; therefore, the EEDATA register can be read by the next instruction. EEDATA will hold this value until another read operation, or until it is written to by the user (during a write operation).

The basic process is shown in Example 7-1.

## 7.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADRH:EEADR register pair and the data written to the EEDATA register. The sequence in Example 7-2 must be followed to initiate the write cycle.

The write will not begin if this sequence is not exactly followed (write 55h to EECON2, write 0AAh to EECON2, then set WR bit) for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code execution (i.e., runaway programs). The WREN bit should be kept clear at all times, except when updating the EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, EECON1, EEADRH:EEADR and EEDATA cannot be modified. The WR bit will be inhibited from being set unless the WREN bit is set. The WREN bit must be set on a previous instruction. Both WR and WREN cannot be set with the same instruction.

At the completion of the write cycle, the WR bit is cleared in hardware and the EEPROM Interrupt Flag bit (EEIF) is set. The user may either enable this interrupt, or poll this bit. EEIF must be cleared by software.

## 7.5 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

**EXAMPLE 7-1: DATA EEPROM READ**

```
        MOVLW   DATA_EE_ADDRH      ;
        MOVWF   EEADRH             ; Upper bits of Data Memory Address to read
        MOVLW   DATA_EE_ADDR       ;
        MOVWF   EEADR              ; Lower bits of Data Memory Address to read
        BCF     EECON1, EEPGD      ; Point to DATA memory
        BCF     EECON1, CFGS       ; Access EEPROM
        BSF     EECON1, RD         ; EEPROM Read
        MOVF    EEDATA, W          ; W = EEDATA
```

**EXAMPLE 7-2: DATA EEPROM WRITE**

```
                MOVLW   DATA_EE_ADDRH      ;
                MOVWF   EEADRH             ; Upper bits of Data Memory Address to write
                MOVLW   DATA_EE_ADDR       ;
                MOVWF   EEADR              ; Lower bits of Data Memory Address to write
                MOVLW   DATA_EE_DATA       ;
                MOVWF   EEDATA             ; Data Memory Value to write
                BCF     EECON1, EPGD       ; Point to DATA memory
                BCF     EECON1, CFGS       ; Access EEPROM
                BSF     EECON1, WREN       ; Enable writes

                BCF     INTCON, GIE        ; Disable Interrupts
                MOVLW   55h                ;
  Required      MOVWF   EECON2             ; Write 55h
  Sequence      MOVLW   0AAh               ;
                MOVWF   EECON2             ; Write 0AAh
                BSF     EECON1, WR         ; Set WR bit to begin write
                BSF     INTCON, GIE        ; Enable Interrupts

                                           ; User code execution
                BCF     EECON1, WREN       ; Disable writes on write complete (EEIF set)
```

## 9.0 INTERRUPTS

The PIC18F2682/2685/4682/4685 devices have multiple interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high priority level or a low priority level. The high priority interrupt vector is at 000008h and the low priority interrupt vector is at 000018h. High priority interrupt events will interrupt any low priority interrupts that may be in progress.

There are ten registers which are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3

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

Each interrupt source has three bits to control its operation. The functions of these bits are:

- Flag bit to indicate that an interrupt event occurred
- Enable bit that allows program execution to branch to the interrupt vector address when the flag bit is set
- Priority bit to select high priority or low priority

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When interrupt priority is enabled, there are two bits which enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts that have the priority bit set (high priority). Setting the GIEL bit (INTCON<6>) enables all interrupts that have the priority bit cleared (low priority). When the interrupt flag, enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 000008h or 000018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit, which enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit, which enables/disables all interrupt sources. All interrupts branch to address 000008h in Compatibility mode.

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

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (000008h or 000018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

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

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

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

**REGISTER 9-8: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2**

| R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| OSCFIE | CMIE(1) | — | EEIE | BCLIE | HLVDIE | TMR3IE | ECCP1IE(1) |
| 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 **OSCFIE:** Oscillator Fail Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **CMIE:** Comparator Interrupt Enable bit(1)
1 = Enabled
0 = Disabled

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

bit 4 **EEIE:** Data EEPROM/Flash Write Operation Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **BCLIE:** Bus Collision Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **HLVDIE:** High/Low-Voltage Detect Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR3IE:** TMR3 Overflow Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **ECCP1IE:** ECCP1 Interrupt Enable bit(1)
1 = Enabled
0 = Disabled

**Note 1:** These bits are available on PIC18F4682/4685 devices only.

## 12.2 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 12-2). When the RD16 control bit (T1CON<7>) is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L will load the contents of the high byte of Timer1 into the Timer1 High Byte Buffer register. This provides the user with the ability to accurately read all 16 bits of Timer1 without having to determine whether a read of the high byte, followed by a read of the low byte, has become invalid due to a rollover between reads.

A write to the high byte of Timer1 must also take place through the TMR1H Buffer register. The Timer1 high byte is updated with the contents of TMR1H when a write occurs to TMR1L. This allows a user to write all 16 bits to both the high and low bytes of Timer1 at once.

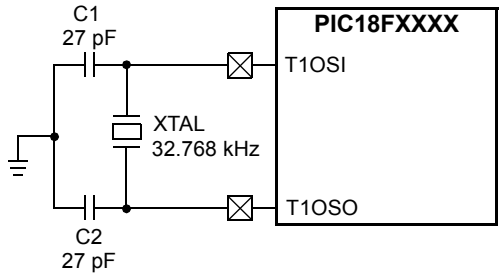
The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

## 12.3 Timer1 Oscillator

An on-chip crystal oscillator circuit is incorporated between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting the Timer1 Oscillator Enable bit, T1OSCEN (T1CON<3>). The oscillator is a low-power circuit rated for 32 kHz crystals. It will continue to run during all power-managed modes. The circuit for a typical LP oscillator is shown in Figure 12-3. Table 12-1 shows the capacitor selection for the Timer1 oscillator.

The user must provide a software time delay to ensure proper start-up of the Timer1 oscillator.

**FIGURE 12-3: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR**




**Note:** See the Notes with Table 12-1 for additional information about capacitor selection.

**TABLE 12-1: CAPACITOR SELECTION FOR THE TIMER1 OSCILLATOR(1,2,3,4)**

| Osc Type | Freq | C1 | C2 |
|---|---|---|---|
| LP | 32.768 kHz | 27 pF | 27 pF |

**Note 1:** Microchip suggests these values as a starting point in validating the oscillator circuit.

**2:** Higher capacitance increases the stability of the oscillator but also increases the start-up time.

**3:** Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

**4:** Capacitor values are for design guidance only.

### 12.3.1 USING TIMER1 AS A CLOCK SOURCE

The Timer1 oscillator is also available as a clock source in power-managed modes. By setting the Clock Select bits, SCS1:SCS0 (OSCCON<1:0>), to '01', the device switches to SEC_RUN mode; both the CPU and peripherals are clocked from the Timer1 oscillator. If the IDLEN bit (OSCCON<7>) is cleared and a SLEEP instruction is executed, the device enters SEC_IDLE mode. Additional details are available in **Section 3.0 "Power-Managed Modes"**.

Whenever the Timer1 oscillator is providing the clock source, the Timer1 system clock status flag, T1RUN (T1CON<6>), is set. This can be used to determine the controller's current clocking mode. It can also indicate the clock source being currently used by the Fail-Safe Clock Monitor. If the Clock Monitor is enabled and the Timer1 oscillator fails while providing the clock, polling the T1RUN bit will indicate whether the clock is being provided by the Timer1 oscillator or another source.

### 12.3.2 LOW-POWER TIMER1 OPTION

The Timer1 oscillator can operate at two distinct levels of power consumption based on device configuration. When the LPT1OSC Configuration bit is set, the Timer1 oscillator operates in a low-power mode. When LPT1OSC is not set, Timer1 operates at a higher power level. Power consumption for a particular mode is relatively constant, regardless of the device's operating mode. The default Timer1 configuration is the higher power mode.

As the low-power Timer1 mode tends to be more sensitive to interference, high noise environments may cause some oscillator instability. The low-power option is, therefore, best suited for low noise applications where power conservation is an important design consideration.

**FIGURE 17-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)**

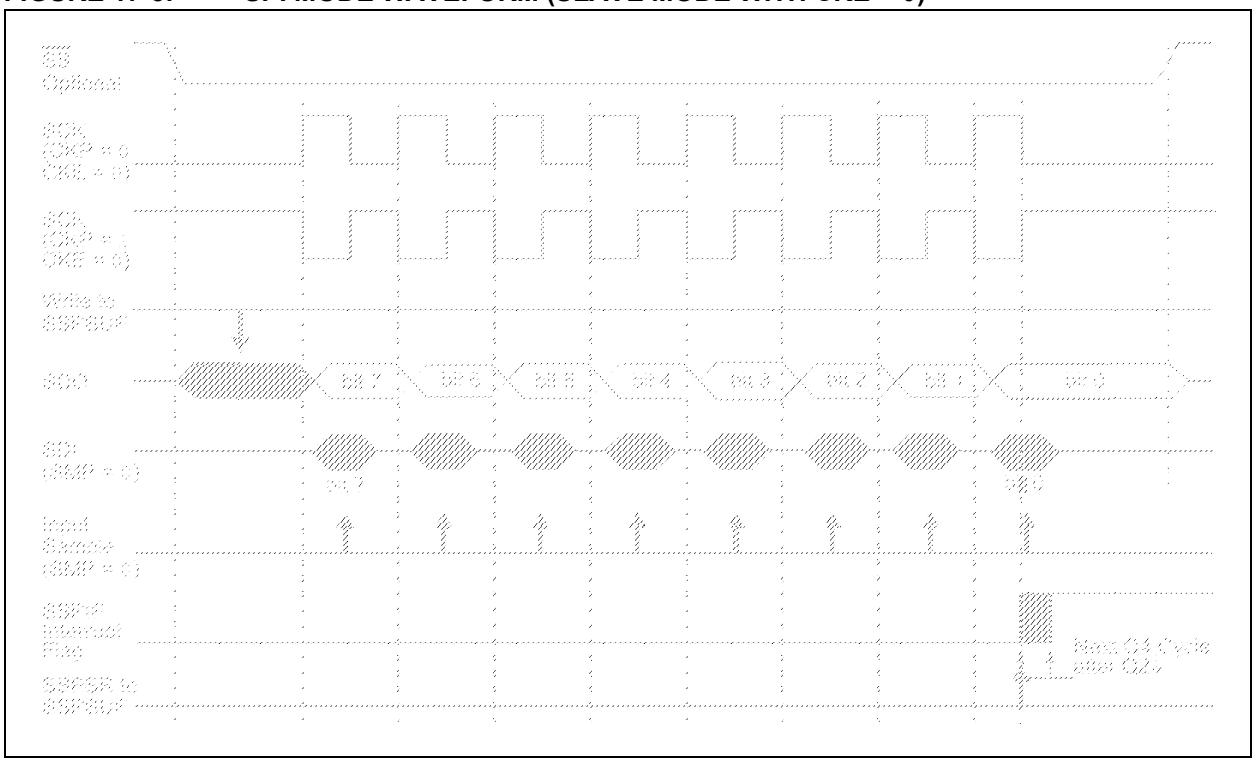


**FIGURE 17-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)**

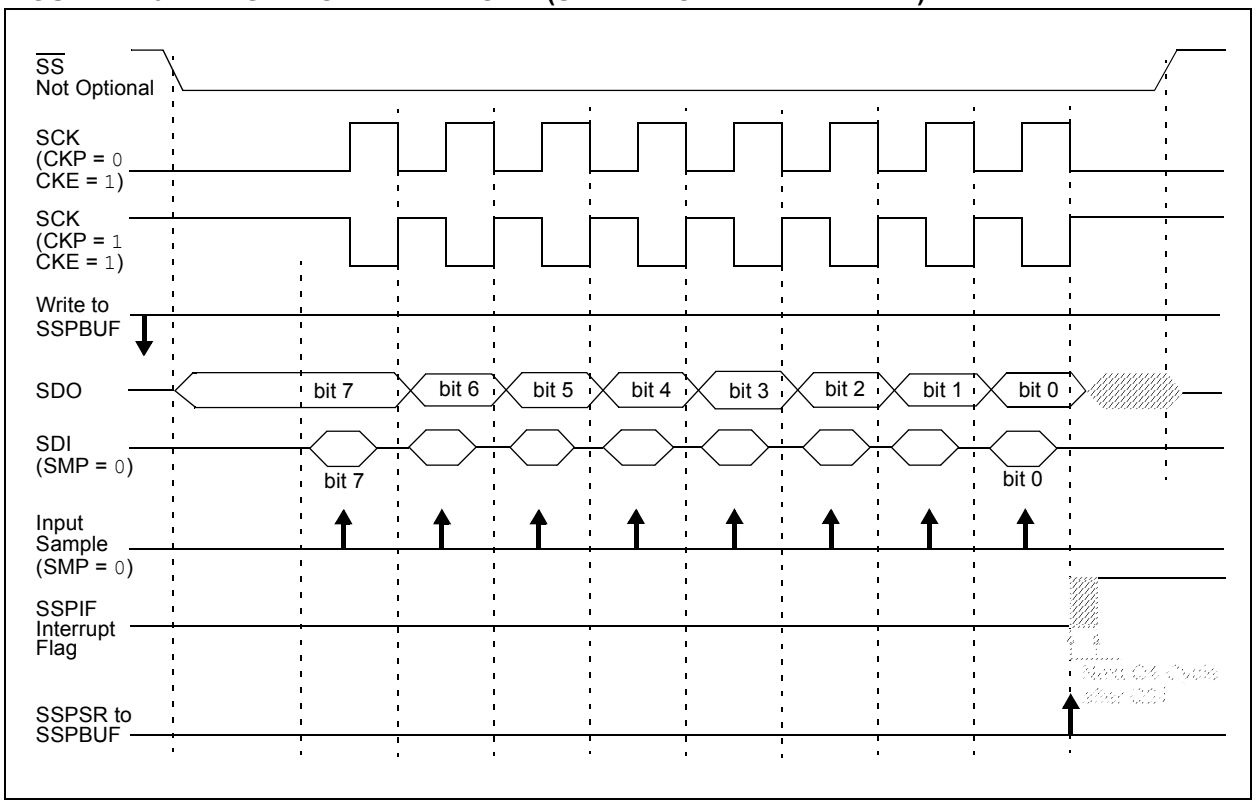


**FIGURE 17-10: I²C™ SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)**

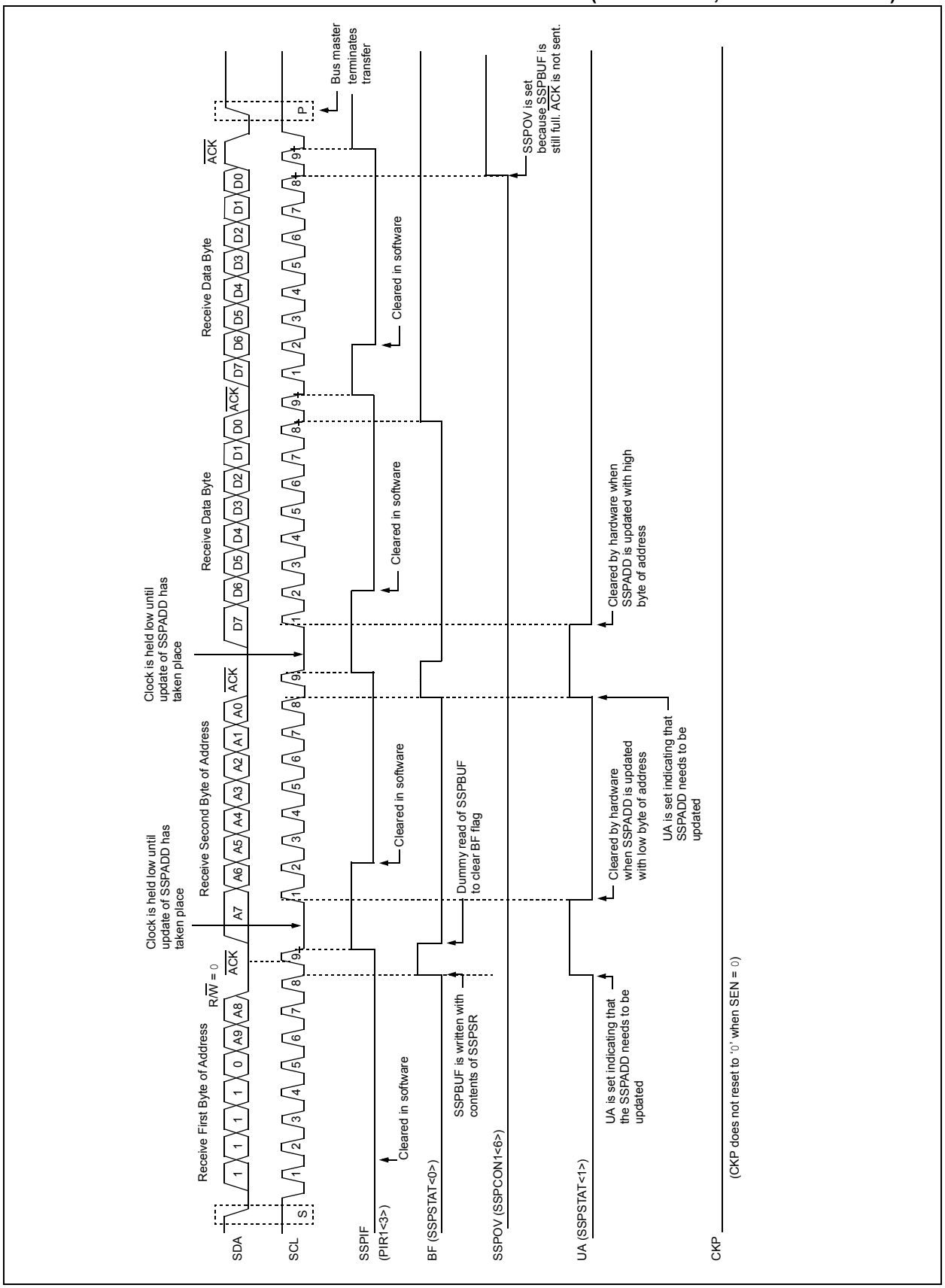


**FIGURE 18-4: ASYNCHRONOUS TRANSMISSION**

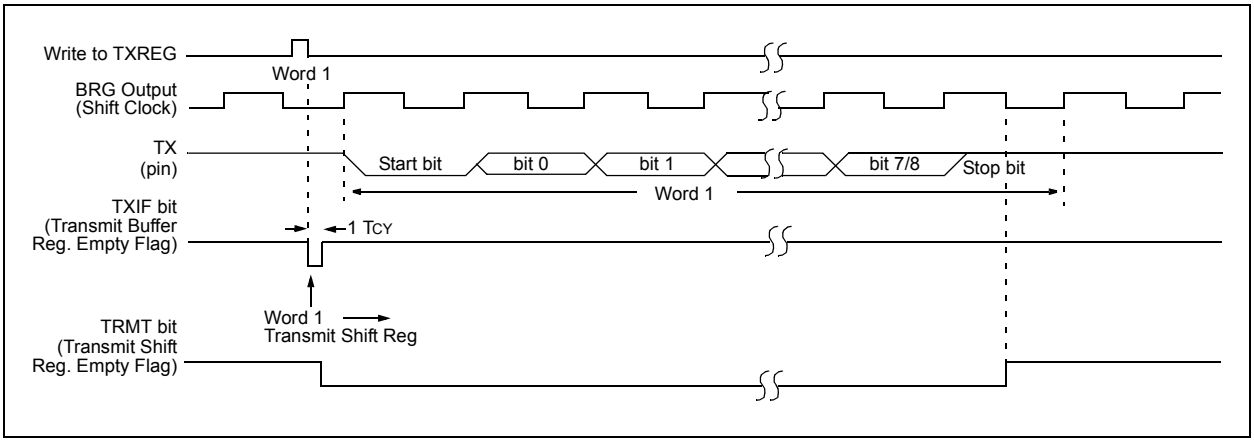
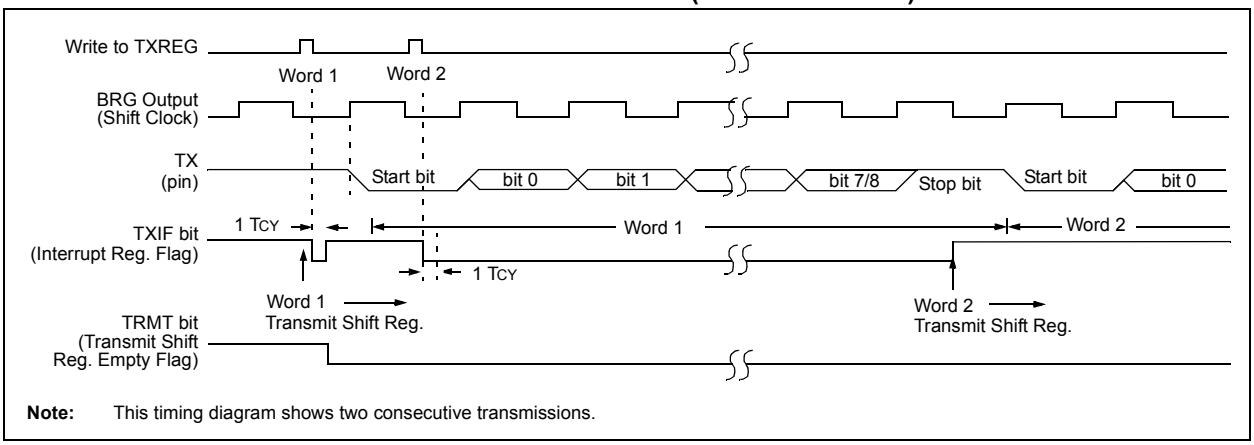



**FIGURE 18-5: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)**




**Note:** This timing diagram shows two consecutive transmissions.

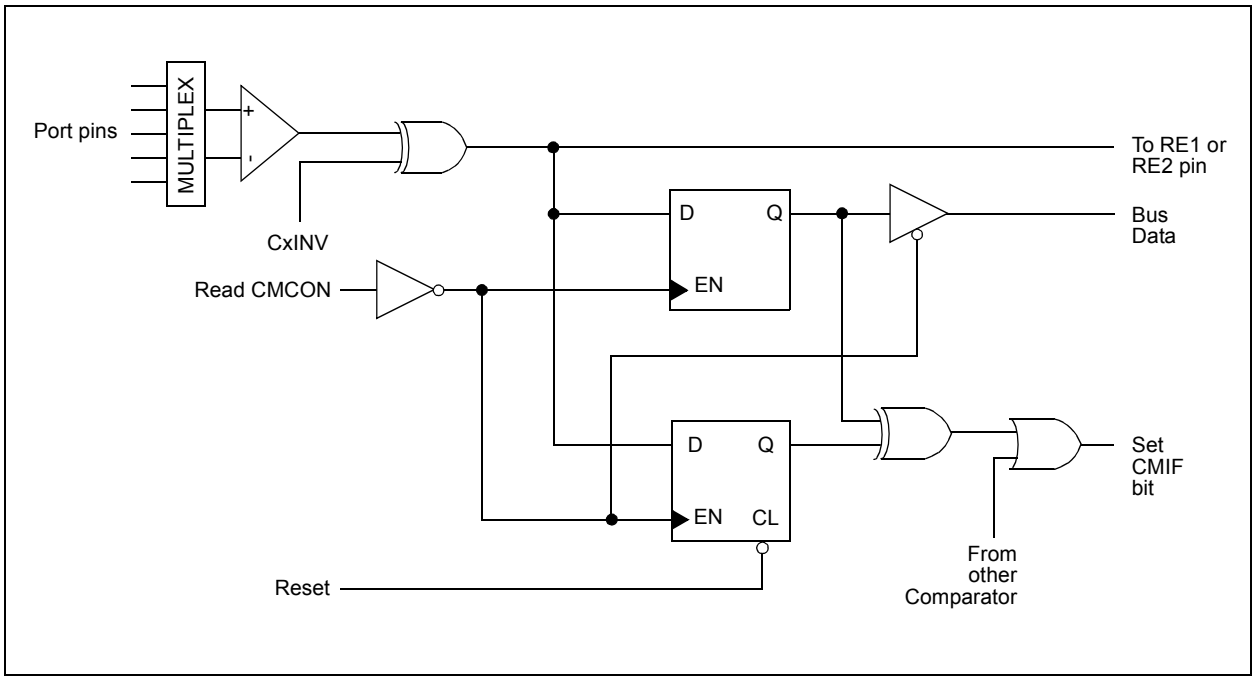
**TABLE 18-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION**

| 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 | 51 |
| PIR1 | PSPIF(1) | ADIF | RCIF | TXIF | SSPIF | CCP1IF | TMR2IF | TMR1IF | 54 |
| PIE1 | PSPIE(1) | ADIE | RCIE | TXIE | SSPIE | CCP1IE | TMR2IE | TMR1IE | 54 |
| IPR1 | PSPIP(1) | ADIP | RCIP | TXIP | SSPIP | CCP1IP | TMR2IP | TMR1IP | 54 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 53 |
| TXREG | EUSART Transmit Register | | | | | | | | 53 |
| TXSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 53 |
| BAUDCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 53 |
| SPBRGH | EUSART Baud Rate Generator Register High Byte | | | | | | | | 53 |
| SPBRG | EUSART Baud Rate Generator Register Low Byte | | | | | | | | 53 |

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

**Note 1:** Reserved in PIC18F2682/2685 devices; always maintain these bits clear.

**FIGURE 20-3: COMPARATOR OUTPUT BLOCK DIAGRAM**




## 20.6 Comparator Interrupts

The comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMCON<7:6>, to determine the actual change that occurred. The CMIF bit (PIR2<6>) is the Comparator Interrupt Flag. The CMIF bit must be reset by clearing it. Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated.

Both the CMIE bit (PIE2<6>) and the PEIE bit (INTCON<6>) must be set to enable the interrupt. In addition, the GIE bit (INTCON<7>) must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMIF bit will still be set if an interrupt condition occurs.

| **Note:** | If a change in the CMCON register (C1OUT or C2OUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CMIF (PIR registers) interrupt flag may not get set. |
|---|---|

The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of CMCON will end the mismatch condition.

b) Clear flag bit CMIF.

A mismatch condition will continue to set flag bit CMIF. Reading CMCON will end the mismatch condition and allow flag bit CMIF to be cleared.

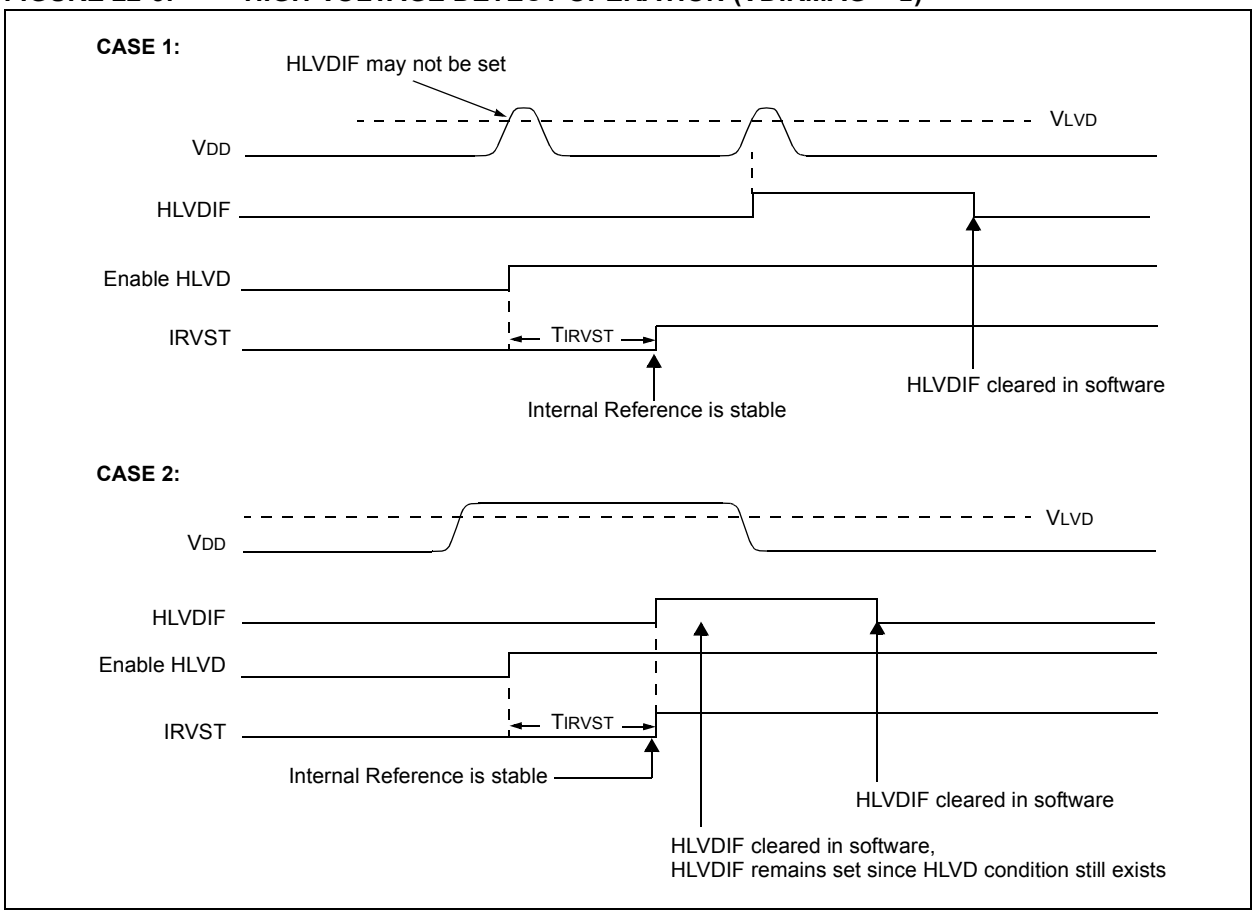
## 20.7 Comparator Operation During Sleep

When a comparator is active and the device is placed in Sleep mode, the comparator remains active and the interrupt is functional if enabled. This interrupt will wake-up the device from Sleep mode, when enabled. While the comparator is powered up, higher Sleep currents than shown in the power-down current specification will occur. Each operational comparator will consume additional current, as shown in the comparator specifications. To minimize power consumption while in Sleep mode, turn off the comparators (CM2:CM0 = 111) before entering Sleep. If the device wakes up from Sleep, the contents of the CMCON register are not affected.

## 20.8 Effects of a Reset

A device Reset forces the CMCON register to its Reset state, causing the comparator module to be in the Comparator Reset mode (CM2:CM0 = 000). This ensures that all potential inputs are analog inputs. Device current is minimized when analog inputs are present at Reset time. The comparators are powered down during the Reset interval.

**FIGURE 22-3: HIGH-VOLTAGE DETECT OPERATION (VDIRMAG = 1)**

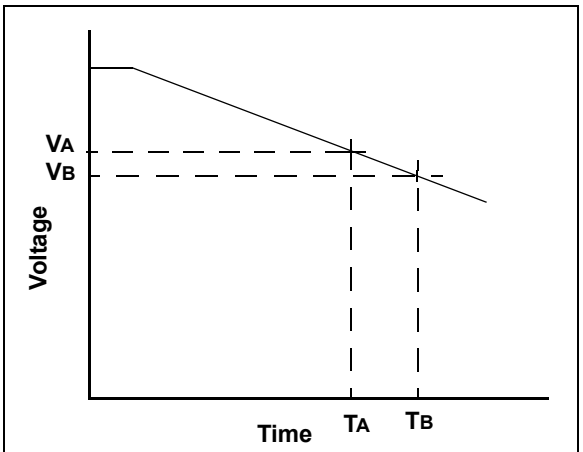



## 22.5 Applications

In many applications, the ability to detect a drop below, or rise above a particular threshold is desirable. For example, the HLVD module could be periodically enabled to detect Universal Serial Bus (USB) attach or detach. This assumes the device is powered by a lower voltage source than the USB when detached. An attach would indicate a high-voltage detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature could save a design a few extra components and an attach signal (input pin).

For general battery applications, Figure 22-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage VA, the HLVD logic generates an interrupt at time TA. The interrupt could cause the execution of an ISR, which would allow the application to perform "house-keeping tasks" and perform a controlled shutdown before the device voltage exits the valid operating range at TB. The HLVD, thus, would give the application a time window, represented by the difference between TA and TB, to safely exit.

**FIGURE 22-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION**




**Legend:** VA = HLVD trip point
VB = Minimum valid device operating voltage

**REGISTER 23-3: ECANCON: ENHANCED CAN CONTROL REGISTER**

| R/W-0 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| MDSEL1(1) | MDSEL0(1) | FIFOWM(2) | EWIN4 | EWIN3 | EWIN2 | EWIN1 | EWIN0 |
| 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 7-6 **MDSEL1:MDSEL0:** Mode Select bits(1)
00 = Legacy mode (Mode 0, default)
01 = Enhanced Legacy mode (Mode 1)
10 = Enhanced FIFO mode (Mode 2)
11 = Reserved

bit 5 **FIFOWM:** FIFO High Water Mark bit(2)
1 = Will cause FIFO interrupt when one receive buffer remains(3)
0 = Will cause FIFO interrupt when four receive buffers remain

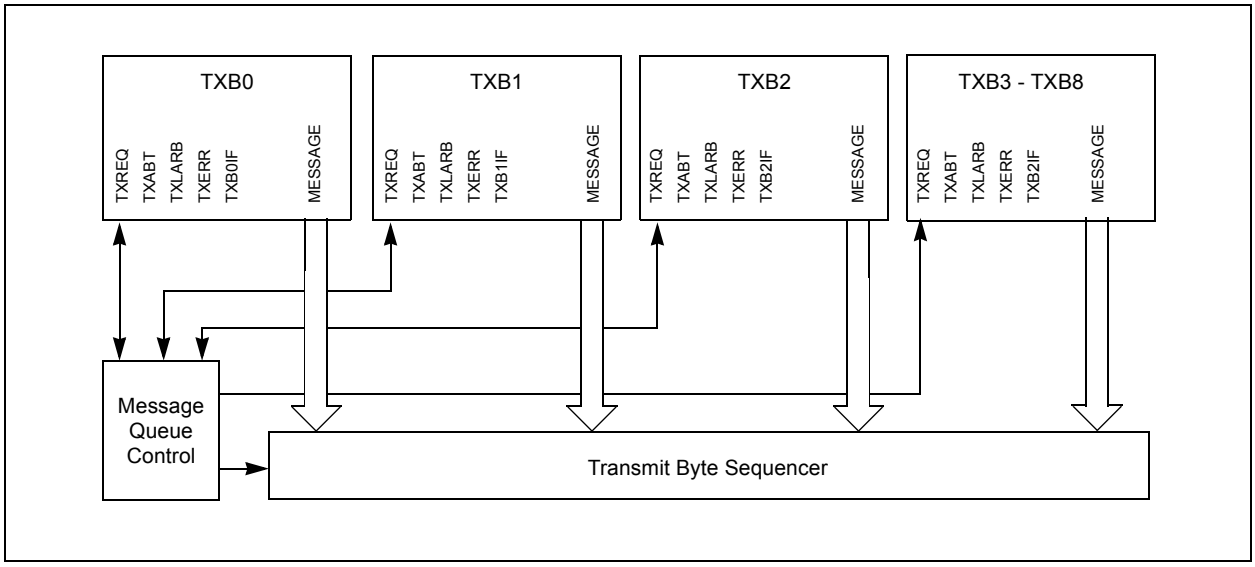
bit 4-0 **EWIN4:EWIN0:** Enhanced Window Address bits
These bits map the group of 16 banked CAN SFRs into access bank addresses 0F60-0F6Dh. Exact group of registers to map is determined by binary value of these bits.
Mode 0:
**Unimplemented:** Read as '0'
Mode 1, 2:
00000 = Acceptance Filters 0, 1, 2 and BRGCON2, 3
00001 = Acceptance Filters 3, 4, 5 and BRGCON1, CIOCON
00010 = Acceptance Filter Masks, Error and Interrupt Control
00011 = Transmit Buffer 0
00100 = Transmit Buffer 1
00101 = Transmit Buffer 2
00110 = Acceptance Filters 6, 7, 8
00111 = Acceptance Filters 9, 10, 11
01000 = Acceptance Filters 12, 13, 14
01001 = Acceptance Filters 15
01010-01110 = Reserved
01111 = RXINT0, RXINT1
10000 = Receive Buffer 0
10001 = Receive Buffer 1
10010 = TX/RX Buffer 0
10011 = TX/RX Buffer 1
10100 = TX/RX Buffer 2
10101 = TX/RX Buffer 3
10110 = TX/RX Buffer 4
10111 = TX/RX Buffer 5
11000-11111 = Reserved

**Note 1:** These bits can only be changed in Configuration mode. See Register 23-1 to change to Configuration mode.
**2:** This bit is used in Mode 2 only.
**3:** FIFO length of 4 or less will cause this bit to be set.

### 23.6.3 TRANSMIT PRIORITY

Transmit priority is a prioritization within the PIC18F2682/2685/4682/4685 devices of the pending transmittable messages. This is independent from and not related to any prioritization implicit in the message arbitration scheme built into the CAN protocol. Prior to sending the SOF, the priority of all buffers that are queued for transmission is compared. The transmit buffer with the highest priority will be sent first. If two buffers have the same priority setting, the buffer with the highest buffer number will be sent first. There are four levels of transmit priority. If TXP bits for a particular message buffer are set to '11', that buffer has the highest possible priority. If TXP bits for a particular message buffer are set to '00', that buffer has the lowest possible priority.

**FIGURE 23-2: TRANSMIT BUFFERS**



## 24.3 Two-Speed Start-up

The Two-Speed Start-up feature helps to minimize the latency period from oscillator start-up to code execution by allowing the microcontroller to use the INTRC oscillator as a clock source until the primary clock source is available. It is enabled by setting the IESO Configuration bit.

Two-Speed Start-up should be enabled only if the primary oscillator mode is LP, XT, HS or HSPLL (Crystal-based modes). Other sources do not require an OST start-up delay; for these, Two-Speed Start-up should be disabled.

When enabled, Resets and wake-ups from Sleep mode cause the device to configure itself to run from the internal oscillator block as the clock source, following the time-out of the Power-up Timer after a Power-on Reset is enabled. This allows almost immediate code execution while the primary oscillator starts and the OST is running. Once the OST times out, the device automatically switches to PRI_RUN mode.

Because the OSCCON register is cleared on Reset events, the INTOSC (or postscaler) clock source is not initially available after a Reset event; the INTRC clock is used directly at its base frequency. To use a higher clock speed on wake-up, the INTOSC or postscaler clock sources can be selected to provide a higher clock speed by setting bits, IRCF2:IRCF0, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF2:IRCF0 bits prior to entering Sleep mode.

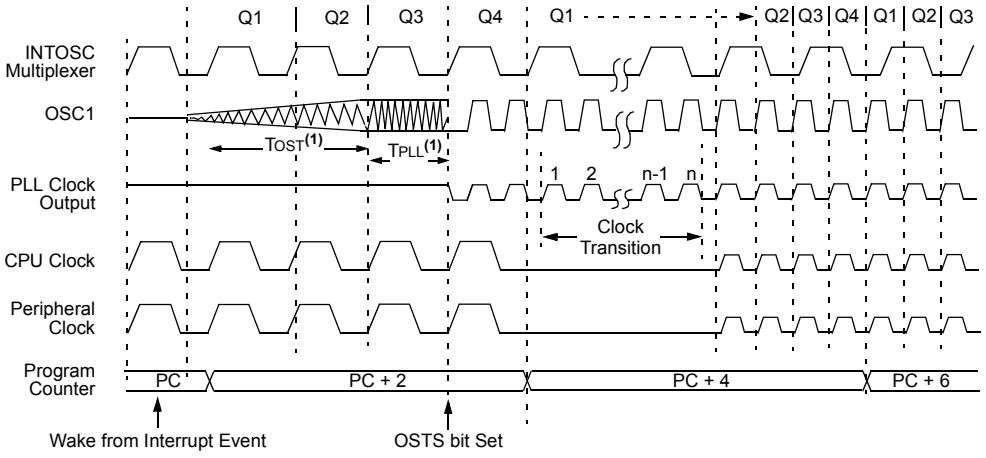
In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

### 24.3.1 SPECIAL CONSIDERATIONS FOR USING TWO-SPEED START-UP

While using the INTRC oscillator in Two-Speed Start-up, the device still obeys the normal command sequences for entering power-managed modes, including serial SLEEP instructions (refer to **Section 3.1.4 "Multiple Sleep Commands"**). In practice, this means that user code can change the SCS1:SCS0 bit settings or issue SLEEP instructions before the OST times out. This would allow an application to briefly wake-up, perform routine "housekeeping" tasks and return to Sleep before the device starts to operate from the primary oscillator.

User code can also check if the primary clock source is currently providing the device clocking by checking the status of the OSTS bit (OSCCON<3>). If the bit is set, the primary oscillator is providing the clock. Otherwise, the internal oscillator block is providing the clock during wake-up from Reset or Sleep mode.

**FIGURE 24-2: TIMING TRANSITION FOR TWO-SPEED START-UP (INTOSC TO HSPLL)**




**Note 1:** TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

**BNC** **Branch if Not Carry**

Syntax: BNC n

Operands: -128 ≤ n ≤ 127

Operation: if Carry bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0011 | nnnn | nnnn |

Description: If the Carry bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNC Jump

Before Instruction
PC = address (HERE)
After Instruction
If Carry = 0;
PC = address (Jump)
If Carry = 1;
PC = address (HERE + 2)

**BNN** **Branch if Not Negative**

Syntax: BNN n

Operands: -128 ≤ n ≤ 127

Operation: if Negative bit is '0'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0111 | nnnn | nnnn |

Description: If the Negative bit is '0', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNN Jump

Before Instruction
PC = address (HERE)
After Instruction
If Negative = 0;
PC = address (Jump)
If Negative = 1;
PC = address (HERE + 2)

**BTFSC** **Bit Test File, Skip if Clear**

Syntax: BTFSC f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

| 1011 | bbba | ffff | ffff |
|---|---|---|---|

Description: If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh).
See **Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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    BTFSC   FLAG, 1, 0
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)
After Instruction
If FLAG<1> = 0;
PC = address (TRUE)
If FLAG<1> = 1;
PC = address (FALSE)

**BTFSS** **Bit Test File, Skip if Set**

Syntax: BTFSS f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

| 1010 | bbba | ffff | ffff |
|---|---|---|---|

Description: If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction.
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh).
See **Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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    BTFSS   FLAG, 1, 0
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)
After Instruction
If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)

**INCFSZ** **Increment f, Skip if 0**

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

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

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

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0011 | 11da | ffff | ffff |

Description: The contents of register 'f' are incremented. 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 two-cycle instruction.

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

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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    INCFSZ   CNT, 1, 0
NZERO   :
ZERO    :
```

Before Instruction
PC = Address (HERE)
After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT ≠ 0;
PC = Address (NZERO)

**INFSNZ** **Increment f, Skip if Not 0**

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

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

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

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0100 | 10da | ffff | ffff |

Description: The contents of register 'f' are incremented. 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 two-cycle instruction.

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

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

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    INFSNZ  REG, 1, 0
ZERO
NZERO
```

Before Instruction
PC = Address (HERE)
After Instruction
REG = REG + 1
If REG ≠ 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)

**LFSR** **Load FSR**

Syntax: LFSR f, k

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

Operation: $k \rightarrow FSRf$

Status Affected: None

Encoding:

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

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

Words: 2

Cycles: 2

Q Cycle Activity:

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

Example: `LFSR 2, 3ABh`

After Instruction
FSR2H = 03h
FSR2L = ABh

**MOVF** **Move f**

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

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

Operation: $f \rightarrow dest$

Status Affected: N, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 00da | ffff | ffff |

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

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

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 25.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write W |

Example: `MOVF REG, 0, 0`

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

## 27.2 DC Characteristics: Power-Down and Supply Current PIC18F2682/2685/4682/4685 (Industrial) PIC18LF2682/2685/4682/4685 (Industrial) (Continued)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **PIC18LF2682/2685/4682/4685** (Industrial) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | | |
| **PIC18F2682/2685/4682/4685** (Industrial, Extended) | | **Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | |
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | PIC18LF268X/468X | 250 | 600 | µA | -40°C | VDD = 2.0V | FOSC = 1 MHz (**PRI_RUN**, EC oscillator) |
| | | 250 | 600 | µA | +25°C | | |
| | | 250 | 600 | µA | +85°C | | |
| | PIC18LF268X/468X | 550 | 1.2 | mA | -40°C | VDD = 3.0V | |
| | | 480 | 1.2 | mA | +25°C | | |
| | | 460 | 1.2 | mA | +85°C | | |
| | All devices | 1.2 | 3 | mA | -40°C | VDD = 5.0V | |
| | | 1.1 | 3 | mA | +25°C | | |
| | | 1 | 3 | mA | +85°C | | |
| | Extended devices only | 1 | 3 | mA | +125°C | | |
| | PIC18LF268X/468X | 0.72 | 2.2 | mA | -40°C | VDD = 2.0V | FOSC = 4 MHz (**PRI_RUN**, EC oscillator) |
| | | 0.74 | 2.2 | mA | +25°C | | |
| | | 0.74 | 2.2 | mA | +85°C | | |
| | PIC18LF268X/468X | 1.3 | 3.3 | mA | -40°C | VDD = 3.0V | |
| | | 1.3 | 3.3 | mA | +25°C | | |
| | | 1.3 | 3.3 | mA | +85°C | | |
| | All devices | 2.7 | 6.6 | mA | -40°C | VDD = 5.0V | |
| | | 2.6 | 6.6 | mA | +25°C | | |
| | | 2.5 | 6.6 | mA | +85°C | | |
| | Extended devices only | 2.6 | 6.6 | mA | +125°C | | |
| | Extended devices only | 8.4 | 21 | mA | +125°C | VDD = 4.2V | FOSC = 25 MHz (**PRI_RUN**, EC oscillator) |
| | | 11 | 28 | mA | +125°C | VDD = 5.0V | |
| | All devices | 15 | 38 | mA | -40°C | VDD = 4.2V | FOSC = 40 MHz (**PRI_RUN**, EC oscillator) |
| | | 16 | 38 | mA | +25°C | | |
| | | 16 | 38 | mA | +85°C | | |
| | All devices | 21 | 44 | mA | -40°C | VDD = 5.0V | |
| | | 21 | 44 | mA | +25°C | | |
| | | 21 | 44 | mA | +85°C | | |

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

**Note 1:** The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in high-impedance state and tied to VDD or VSS 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 IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

**3:** For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT 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.