

Welcome to [E-XFL.COM](http://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	64MHz
Connectivity	I <sup>2</sup> C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 35x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/pic18lf45k40-e-pt">https://www.e-xfl.com/product-detail/microchip-technology/pic18lf45k40-e-pt</a>

## Digital Peripherals (Continued)

- Programmable CRC with Memory Scan:
  - Reliable data/program memory monitoring for Fail-Safe operation (e.g., Class B)
  - Calculate CRC over any portion of Flash or EEPROM
  - High-speed or background operation
- Hardware Limit Timer (TMR2/4/6+HLT):
  - Hardware monitoring and Fault detection
- Peripheral Pin Select (PPS):
  - Enables pin mapping of digital I/O
- Data Signal Modulator (DSM)

## Analog Peripherals

- 10-Bit Analog-to-Digital Converter with Computation (ADC<sup>2</sup>):
  - 35 external channels
  - Conversion available during Sleep
  - Four internal analog channels
  - Internal and external trigger options
  - Automated math functions on input signals:
    - averaging, filter calculations, oversampling and threshold comparison
- Hardware Capacitive Voltage Divider (CVD) Support:
  - 8-bit precharge timer
  - Adjustable sample and hold capacitor array
  - Guard ring digital output drive
- Zero-Cross Detect (ZCD):
  - Detect when AC signal on pin crosses ground
- 5-Bit Digital-to-Analog Converter (DAC):
  - Output available externally
  - Programmable 5-bit voltage (% of V<sub>DD</sub>)
  - Internal connections to comparators, Fixed Voltage Reference and ADC
- Two Comparators (CMP):
  - Four external inputs
  - External output via PPS
- Fixed Voltage Reference (FVR) module:
  - 1.024V, 2.048V and 4.096V output levels

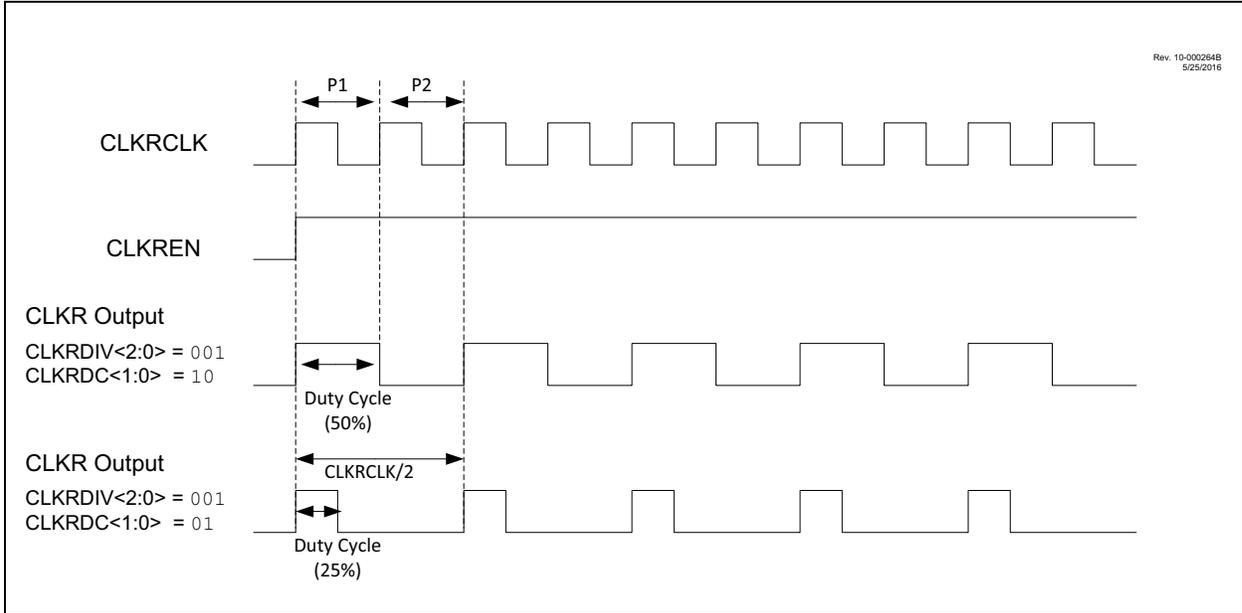
## Clocking Structure

- High-Precision Internal Oscillator Block (HFINTOSC):
  - Selectable frequency range up to 64 MHz
  - ±1% at calibration
- 32 kHz Low-Power Internal Oscillator (LFINTOSC)
- External 32 kHz Crystal Oscillator (SOSC)
- External Oscillator Block:
  - Three crystal/resonator modes
  - 4x PLL with external sources
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if peripheral clock stops
- Oscillator Start-up Timer (OST)

## Programming/Debug Features

- In-Circuit Debug Integrated On-Chip
- In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) with Three Breakpoints via Two Pins

**FIGURE 5-2: CLOCK REFERENCE TIMING**



## 5.1 Clock Source

The input to the reference clock output can be selected using the CLKRCLK register.

### 5.1.1 CLOCK SYNCHRONIZATION

Once the reference clock enable (EN) is set, the module is ensured to be glitch-free at start-up.

When the reference clock output is disabled, the output signal will be disabled immediately.

Clock dividers and clock duty cycles can be changed while the module is enabled, but glitches may occur on the output. To avoid possible glitches, clock dividers and clock duty cycles should be changed only when the CLKREN is clear.

## 5.2 Programmable Clock Divider

The module takes the clock input and divides it based on the value of the DIV<2:0> bits of the CLKRCON register (Register 5-1).

The following configurations can be made based on the DIV<2:0> bits:

- Base FOSC value
- FOSC divided by 2
- FOSC divided by 4
- FOSC divided by 8
- FOSC divided by 16
- FOSC divided by 32
- FOSC divided by 64
- FOSC divided by 128

The clock divider values can be changed while the module is enabled; however, in order to prevent glitches on the output, the DIV<2:0> bits should only be changed when the module is disabled (EN = 0).

## 5.3 Selectable Duty Cycle

The DC<1:0> bits of the CLKRCON register can be used to modify the duty cycle of the output clock. A duty cycle of 25%, 50%, or 75% can be selected for all clock rates, with the exception of the undivided base Fosc value.

The duty cycle can be changed while the module is enabled; however, in order to prevent glitches on the output, the DC<1:0> bits should only be changed when the module is disabled (EN = 0).

**Note:** The DC1 bit is reset to '1'. This makes the default duty cycle 50% and not 0%.

## 5.4 Operation in Sleep Mode

The reference clock output module clock is based on the system clock. When the device goes to Sleep, the module outputs will remain in their current state. This will have a direct effect on peripherals using the reference clock output as an input signal. No change should occur in the module from entering or exiting from Sleep.

## 5.5 Register Definitions: Reference Clock

Long bit name prefixes for the Reference Clock peripherals are shown in Table 5-1. Refer to **Section 1.4.2.2 “Long Bit Names”** for more information.

**TABLE 5-1:**

Peripheral	Bit Name Prefix
CLKR	CLKR

**REGISTER 5-1: CLKRCON: REFERENCE CLOCK CONTROL REGISTER**

R/W-0/0	U-0	U-0	R/W-1/1	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	
EN	—	—	DC<1:0>	DIV<2:0>				
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      **EN:** Reference Clock Module Enable bit  
 1 = Reference clock module enabled  
 0 = Reference clock module is disabled

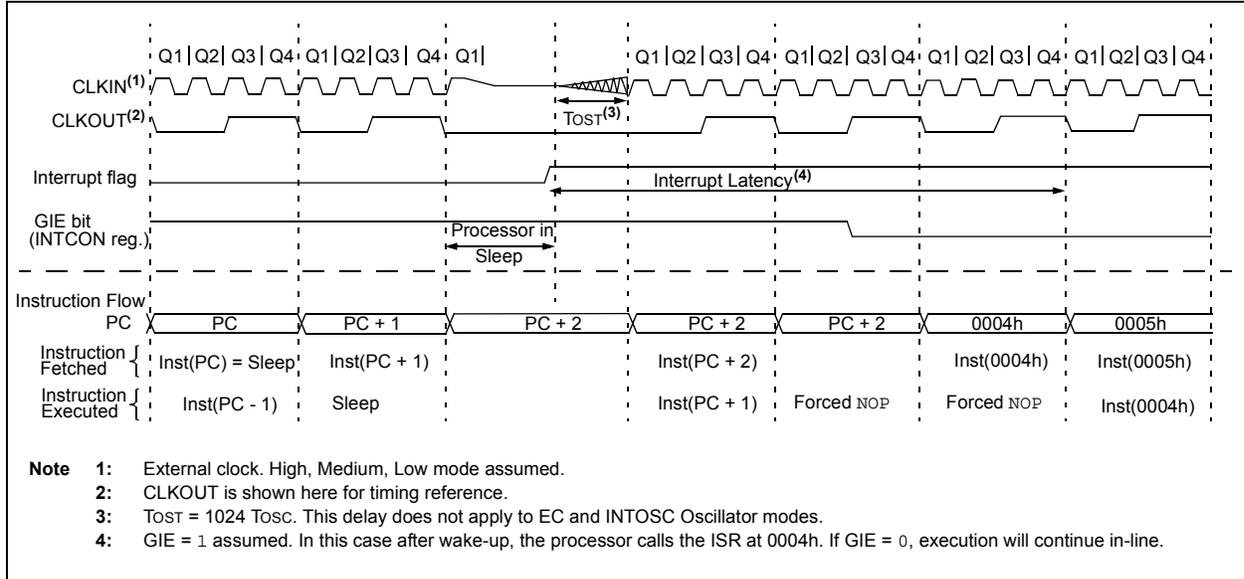
bit 6-5      **Unimplemented:** Read as ‘0’

bit 4-3      **DC<1:0>:** Reference Clock Duty Cycle bits<sup>(1)</sup>  
 11 = Clock outputs duty cycle of 75%  
 10 = Clock outputs duty cycle of 50%  
 01 = Clock outputs duty cycle of 25%  
 00 = Clock outputs duty cycle of 0%

bit 2-0      **DIV<2:0>:** Reference Clock Divider bits  
 111 = Base clock value divided by 128  
 110 = Base clock value divided by 64  
 101 = Base clock value divided by 32  
 100 = Base clock value divided by 16  
 011 = Base clock value divided by 8  
 010 = Base clock value divided by 4  
 001 = Base clock value divided by 2  
 000 = Base clock value

**Note 1:** Bits are valid for reference clock divider values of two or larger, the base clock cannot be further divided.

**FIGURE 6-2: WAKE-UP FROM SLEEP THROUGH INTERRUPT**



## 6.2.3 LOW-POWER SLEEP MODE

The PIC18F2x/4xK40 device family contains an internal Low Dropout (LDO) voltage regulator, which allows the device I/O pins to operate at voltages up to 5.5V while the internal device logic operates at a lower voltage. The LDO and its associated reference circuitry must remain active when the device is in Sleep mode.

The PIC18F2x/4xK40 devices allows the user to optimize the operating current in Sleep, depending on the application requirements.

Low-Power Sleep mode can be selected by setting the VREGPM bit of the VREGCON register.

### 6.2.3.1 Sleep Current vs. Wake-up Time

In the default operating mode, the LDO and reference circuitry remain in the normal configuration while in Sleep. The device is able to exit Sleep mode quickly since all circuits remain active. In Low-Power Sleep mode, when waking-up from Sleep, an extra delay time is required for these circuits to return to the normal configuration and stabilize.

The Low-Power Sleep mode is beneficial for applications that stay in Sleep mode for long periods of time. The Normal mode is beneficial for applications that need to wake from Sleep quickly and frequently.

# PIC18LF26/45/46K40

## REGISTER 9-3: WDTPSL: WWDT PRESCALE SELECT LOW BYTE REGISTER (READ-ONLY)

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
PSCNT<7:0>							
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 7-0 **PSCNT<7:0>**: Prescale Select Low Byte bits<sup>(1)</sup>

**Note 1:** The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should be read during normal operation.

## REGISTER 9-4: WDTPSH: WWDT PRESCALE SELECT HIGH BYTE REGISTER (READ-ONLY)

R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0	R-0/0
PSCNT<15:8>							
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 7-0 **PSCNT<15:8>**: Prescale Select High Byte bits<sup>(1)</sup>

**Note 1:** The 18-bit WDT prescale value, PSCNT<17:0> includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT<17:0> is intended for debug operations and should be read during normal operation.

## 10.3.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as either two bytes or four bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSb = 0). To maintain alignment with instruction boundaries, the PC increments in steps of two and the LSb will always read '0' (see **Section 10.1.1 "Program Counter"**).

Figure 10-3 shows an example of how instruction words are stored in the program memory.

The **CALL** and **GOTO** instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in Figure 10-3 shows how the instruction **GOTO 0006h** is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 35.0 "Instruction Set Summary"** provides further details of the instruction set.

## 10.3.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four two-word instructions: **CALL**, **MOVFF**, **GOTO** and **LFSR**. In all cases, the second word of the instruction always has '1111' as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

The use of '1111' in the 4 MSBs of an instruction specifies a special form of **NOP**. If the instruction is executed in proper sequence – immediately after the first word – the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason and the second word is executed by itself, a **NOP** is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC. Example 10-4 shows how this works.

**Note:** See **Section 10.8 "PIC18 Instruction Execution and the Extended Instruction Set"** for information on two-word instructions in the extended instruction set.

**FIGURE 10-3: INSTRUCTIONS IN PROGRAM MEMORY**

Program Memory Byte Locations →			Word Address		
			LSB = 1	LSB = 0	
				000000h	
				000002h	
				000004h	
				000006h	
Instruction 1:	MOVLW	055h	0Fh	55h	000008h
Instruction 2:	GOTO	0006h	EFh	03h	00000Ah
			F0h	00h	00000Ch
Instruction 3:	MOVFF	123h, 456h	C1h	23h	00000Eh
			F4h	56h	000010h
					000012h
					000014h

**EXAMPLE 10-4: TWO-WORD INSTRUCTIONS**

CASE 1:		
Object Code	Source Code	
0110 0110 0000 0000	TSTFSZ	REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2 ; No, skip this word
1111 0100 0101 0110		; Execute this word as a NOP
0010 0100 0000 0000	ADDWF	REG3 ; continue code
CASE 2:		
Object Code	Source Code	
0110 0110 0000 0000	TSTFSZ	REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2 ; Yes, execute this word
1111 0100 0101 0110		; 2nd word of instruction
0010 0100 0000 0000	ADDWF	REG3 ; continue code

## 11.1 Program Flash Memory

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

A read from program memory is executed one byte at a time. A write to program memory or program memory erase is executed on blocks of  $n$  bytes at a time. Refer to Table 11-3 for write and erase block sizes. A Bulk Erase operation cannot be issued from user code.

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

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

It is important to understand the PFM memory structure for erase and programming operations. Program memory word size is 16 bits wide. PFM is arranged in

rows. A row is the minimum size that can be erased by user software. Refer to Table 11-3 for the row sizes for the these devices.

After a row has been erased, all or a portion of this row can be programmed. Data to be written into the program memory row is written to 6-bit wide data write latches. These latches are not directly accessible, but may be loaded via sequential writes to the TABLAT register.

**Note:** To modify only a portion of a previously programmed row, then the contents of the entire row must be read and saved in RAM prior to the erase. Then, the new data and retained data can be written into the write latches to reprogram the row of PFM. However, any unprogrammed locations can be written without first erasing the row. In this case, it is not necessary to save and rewrite the other previously programmed locations

**TABLE 11-2: FLASH MEMORY ORGANIZATION BY DEVICE**

Device	Row Erase Size (Words)	Write Latches (Bytes)	Program Flash Memory (Words)	Data Memory (Bytes)
PIC18(L)F45K40	32	64	16384	256
PIC18(L)F26K40			32768	1024
PIC18(L)F46K40				
PIC18(L)F27K40	64	128	65536	
PIC18(L)F47K40				

## EXAMPLE 11-4: WRITING TO PROGRAM FLASH 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   BUFFER_ADDR_HIGH     ; point to buffer
        MOVWF   FSR0H
        MOVLW   BUFFER_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
        BCF    NVMCON1, NVMREG0      ; point to Program Flash Memory
        BSF    NVMCON1, NVMREG1      ; point to Program Flash Memory
        BSF    NVMCON1, WREN         ; enable write to memory
        BSF    NVMCON1, FREE        ; enable Erase operation
        BCF    INTCON, GIE          ; disable interrupts
        MOVLW   55h
        MOVWF   NVMCON2              ; write 55h
        MOVLW   AAh
        MOVWF   NVMCON2              ; write 0AAh
        BSF    NVMCON1, 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   BlockSize            ; number of bytes in holding register
        MOVWF   COUNTER
        MOVLW   D'64'/BlockSize     ; number of write blocks in 64 bytes
        MOVWF   COUNTER2
    
```

## 14.4 INTCON Registers

The INTCON registers are readable and writable registers, which contain various enable and priority bits.

## 14.5 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are eight Peripheral Interrupt Request Flag registers (PIR0, PIR1, PIR2, PIR3, PIR4, PIR5, PIR6 and PIR7).

## 14.6 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are eight Peripheral Interrupt Enable registers (PIE0, PIE1, PIE2, PIE3, PIE4, PIE5, PIE6 and PIE7). When IPEN = 0, the PEIE/GIEL bit must be set to enable any of these peripheral interrupts.

## 14.7 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are eight Peripheral Interrupt Priority registers (IPR0, IPR1, IPR2, IPR3, IPR4 and IPR5, IPR6 and IPR7). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

## 24.7 Rising Edge and Reverse Dead Band

In Half-Bridge mode, the rising edge dead band delays the turn-on of the CWG1A output after the rising edge of the CWG data input. In Full-Bridge mode, the reverse dead-band delay is only inserted when changing directions from Forward mode to Reverse mode, and only the modulated output CWG1B is affected.

The CWG1DBR register determines the duration of the dead-band interval on the rising edge of the input source signal. This duration is from 0 to 64 periods of the CWG clock.

Dead band is always initiated on the edge of the input source signal. A count of zero indicates that no dead band is present.

If the input source signal reverses polarity before the dead-band count is completed, then no signal will be seen on the respective output.

The CWG1DBR register value is double-buffered. When EN = 0 (Register 24-1), the buffer is loaded when CWG1DBR is written. If EN = 1, then the buffer will be loaded at the rising edge following the first falling edge of the data input, after the LD bit (Register 24-1) is set. Refer to Figure 24-12 for an example.

## 24.8 Falling Edge and Forward Dead Band

In Half-Bridge mode, the falling edge dead band delays the turn-on of the CWG1B output at the falling edge of the CWG data input. In Full-Bridge mode, the forward dead-band delay is only inserted when changing directions from Reverse mode to Forward mode, and only the modulated output CWG1D is affected.

The CWG1DBF register determines the duration of the dead-band interval on the falling edge of the input source signal. This duration is from zero to 64 periods of CWG clock.

Dead-band delay is always initiated on the edge of the input source signal. A count of zero indicates that no dead band is present.

If the input source signal reverses polarity before the dead-band count is completed, then no signal will be seen on the respective output.

The CWG1DBF register value is double-buffered. When EN = 0 (Register 24-1), the buffer is loaded when CWG1DBF is written. If EN = 1, then the buffer will be loaded at the rising edge following the first falling edge of the data input after the LD (Register 24-1) is set. Refer to Figure 24-13 for an example.

# PIC18LF26/45/46K40

## REGISTER 25-2: MDCON1: MODULATION CONTROL REGISTER 1

U-0	U-0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	CHPOL	CHSYNC	—	—	CLPOL	CLSYNC
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 7-6     **Unimplemented:** Read as '0'
- bit 5     **CHPOL:** Modulator High Carrier Polarity Select bit  
           1 = Selected high carrier signal is inverted  
           0 = Selected high carrier signal is not inverted
- bit 4     **CHSYNC:** Modulator High Carrier Synchronization Enable bit  
           1 = Modulator waits for a falling edge on the high time carrier signal before allowing a switch to the low time carrier  
           0 = Modulator output is not synchronized to the high time carrier signal<sup>(1)</sup>
- bit 3-2    **Unimplemented:** Read as '0'
- bit 1     **CLPOL:** Modulator Low Carrier Polarity Select bit  
           1 = Selected low carrier signal is inverted  
           0 = Selected low carrier signal is not inverted
- bit 0     **CLSYNC:** Modulator Low Carrier Synchronization Enable bit  
           1 = Modulator waits for a falling edge on the low time carrier signal before allowing a switch to the high time carrier  
           0 = Modulator output is not synchronized to the low time carrier signal<sup>(1)</sup>

**Note 1:** Narrowed carrier pulse widths or spurs may occur in the signal stream if the carrier is not synchronized.

## 26.6.2 ARBITRATION

Each master device must monitor the bus for Start and Stop bits. If the device detects that the bus is busy, it cannot begin a new message until the bus returns to an Idle state.

However, two master devices may try to initiate a transmission on or about the same time. When this occurs, the process of arbitration begins. Each transmitter checks the level of the SDA data line and compares it to the level that it expects to find. The first transmitter to observe that the two levels do not match, loses arbitration, and must stop transmitting on the SDA line.

For example, if one transmitter holds the SDA line to a logical one (lets it float) and a second transmitter holds it to a logical zero (pulls it low), the result is that the SDA line will be low. The first transmitter then observes that the level of the line is different than expected and concludes that another transmitter is communicating.

The first transmitter to notice this difference is the one that loses arbitration and must stop driving the SDA line. If this transmitter is also a master device, it also must stop driving the SCL line. It then can monitor the lines for a Stop condition before trying to reissue its transmission. In the meantime, the other device that has not noticed any difference between the expected and actual levels on the SDA line continues with its original transmission. It can do so without any complications, because so far, the transmission appears exactly as expected with no other transmitter disturbing the message.

Slave Transmit mode can also be arbitrated, when a master addresses multiple slaves, but this is less common.

If two master devices are sending a message to two different slave devices at the address stage, the master sending the lower slave address always wins arbitration. When two master devices send messages to the same slave address, and addresses can sometimes refer to multiple slaves, the arbitration process must continue into the data stage.

Arbitration usually occurs very rarely, but it is a necessary process for proper multi-master support.

## 26.7 Register Definitions: I<sup>2</sup>C Mode

The MSSPx module has seven registers for I<sup>2</sup>C operation.

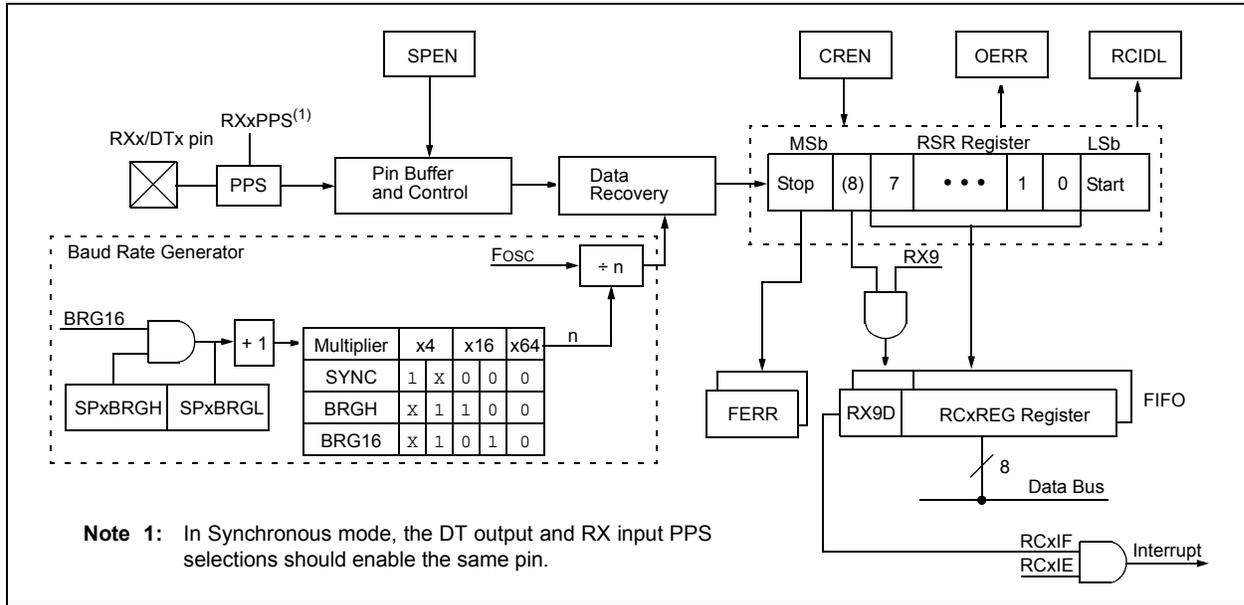
These are:

- MSSP Status Register (SSPxSTAT)
- MSSP Control Register 1 (SSPxCON1)
- MSSP Control Register 2 (SSPxCON2)
- MSSP Control Register 3 (SSPxCON3)
- Serial Receive/Transmit Buffer Register (SSPxBUF)
- MSSP Address Register (SSPxADD)
- I<sup>2</sup>C Slave Address Mask Register (SSPxMSK)
- MSSP Shift Register (SSPSR) – not directly accessible

SSPxCON1, SSPxCON2, SSPxCON3 and SSPxSTAT are the Control and Status registers in I<sup>2</sup>C mode operation. The SSPxCON1, SSPxCON2, and SSPxCON3 registers are readable and writable. The lower six bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write. SSPSR is the Shift register used for shifting data in or out. SSPxBUF is the buffer register to which data bytes are written to or read from. SSPxADD contains the slave device address when the MSSP is configured in I<sup>2</sup>C Slave mode. When the MSSP is configured in Master mode, the lower seven bits of SSPxADD act as the Baud Rate Generator reload value.

SSPxMSK holds the slave address mask value when the module is configured for 7-Bit Address Masking mode. While it is a separate register, it shares the same SFR address as SSPxADD; it is only accessible when the SSPM<3:0> bits are specifically set to permit access. In receive operations, SSPSR and SSPxBUF together, create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set. During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPSR.

**FIGURE 27-2: EUSART RECEIVE BLOCK DIAGRAM**



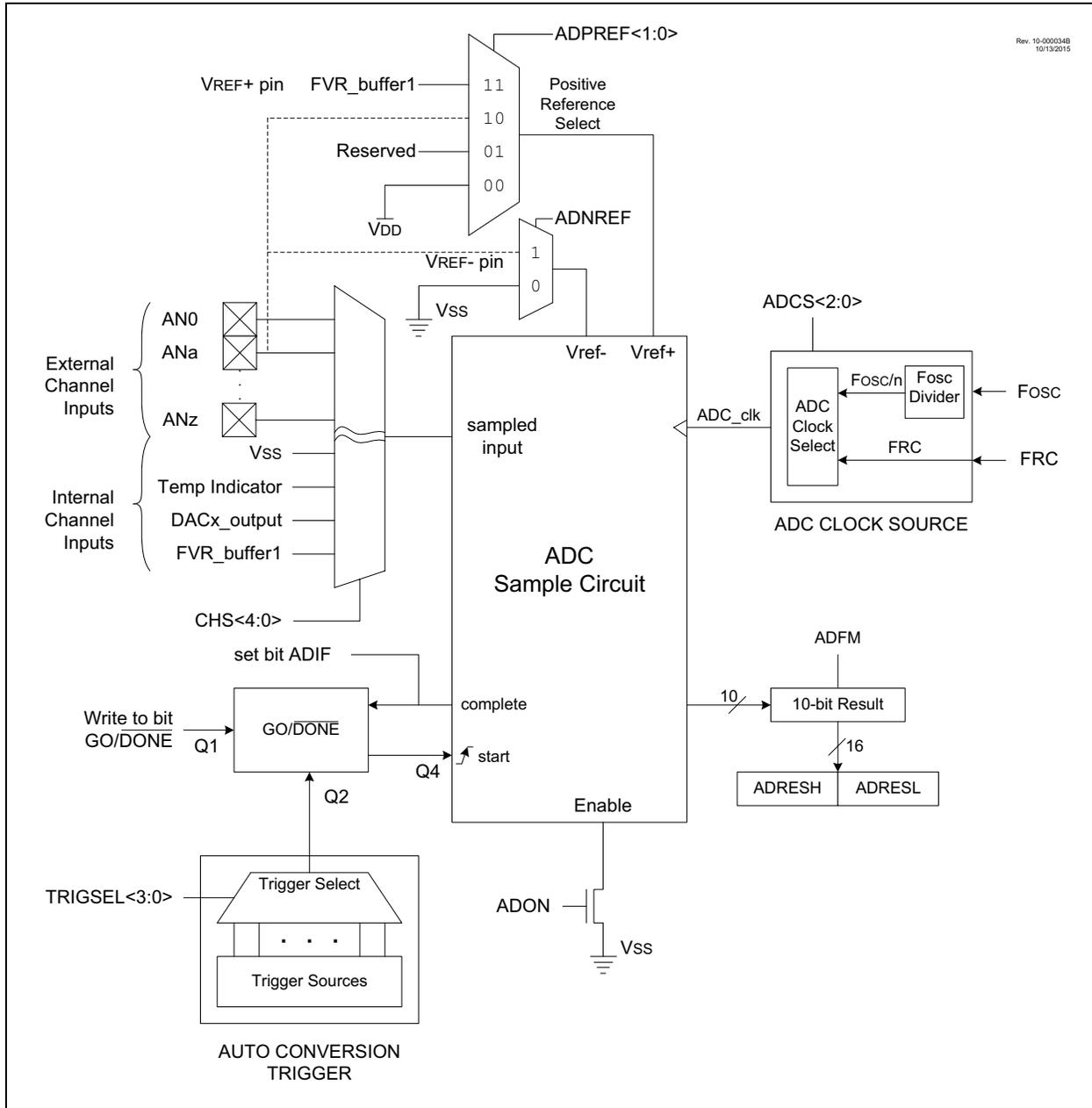
The operation of the EUSART module is controlled through three registers:

- Transmit Status and Control (TXxSTA)
- Receive Status and Control (RCxSTA)
- Baud Rate Control (BAUDxCON)

These registers are detailed in Register 27-1, Register 27-2 and Register 27-3, respectively.

The RXx/DTx and TXx/CKx input pins are selected with the RXxPPS and TXxPPS registers, respectively. TXx, CKx, and DTx output pins are selected with each pin's RxyPPS register. Since the RX input is coupled with the DT output in Synchronous mode, it is the user's responsibility to select the same pin for both of these functions when operating in Synchronous mode. The EUSART control logic will control the data direction drivers automatically.

**FIGURE 31-1: ADC<sup>2</sup> BLOCK DIAGRAM**



## 32.9 Comparator Response Time

The comparator output is indeterminate for a period of time after the change of an input source or the selection of a new reference voltage. This period is referred to as the response time. The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response time to a comparator input change. See the Comparator and Voltage Reference Specifications in Table 37-15 and Table 37-17 for more details.

## 32.10 Analog Input Connection Considerations

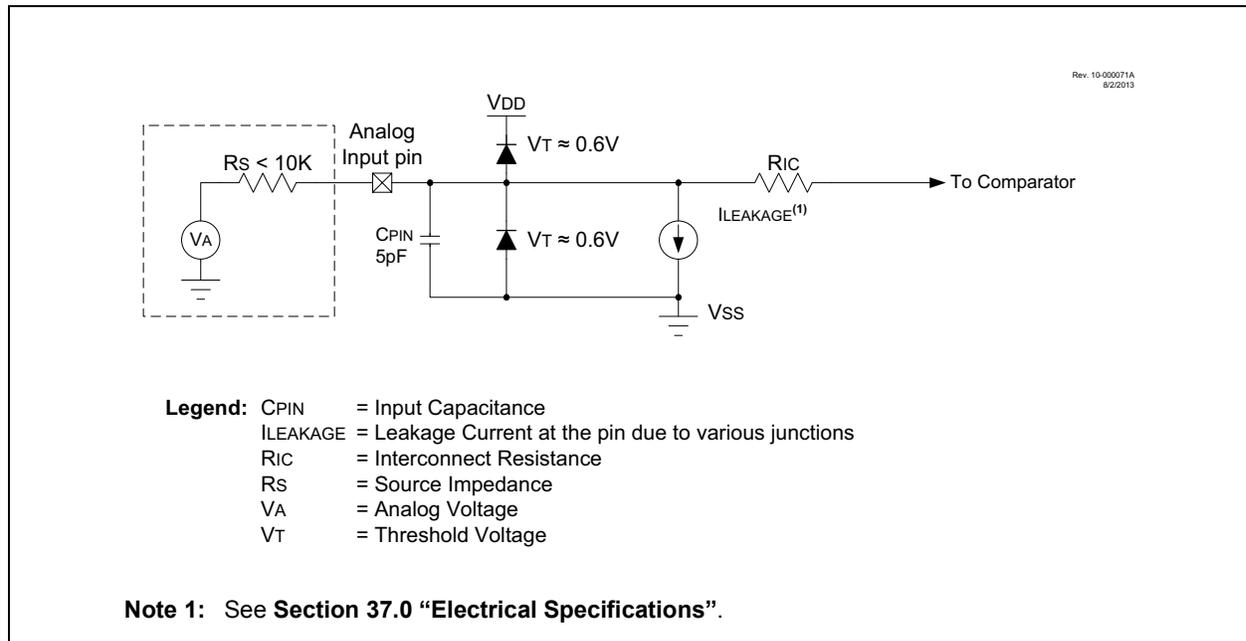
A simplified circuit for an analog input is shown in Figure 32-3. Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to  $V_{DD}$  and  $V_{SS}$ . The analog input, therefore, must be between  $V_{SS}$  and  $V_{DD}$ . If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up may occur.

A maximum source impedance of 10 k $\Omega$  is recommended for the analog sources. Also, any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current to minimize inaccuracies introduced.

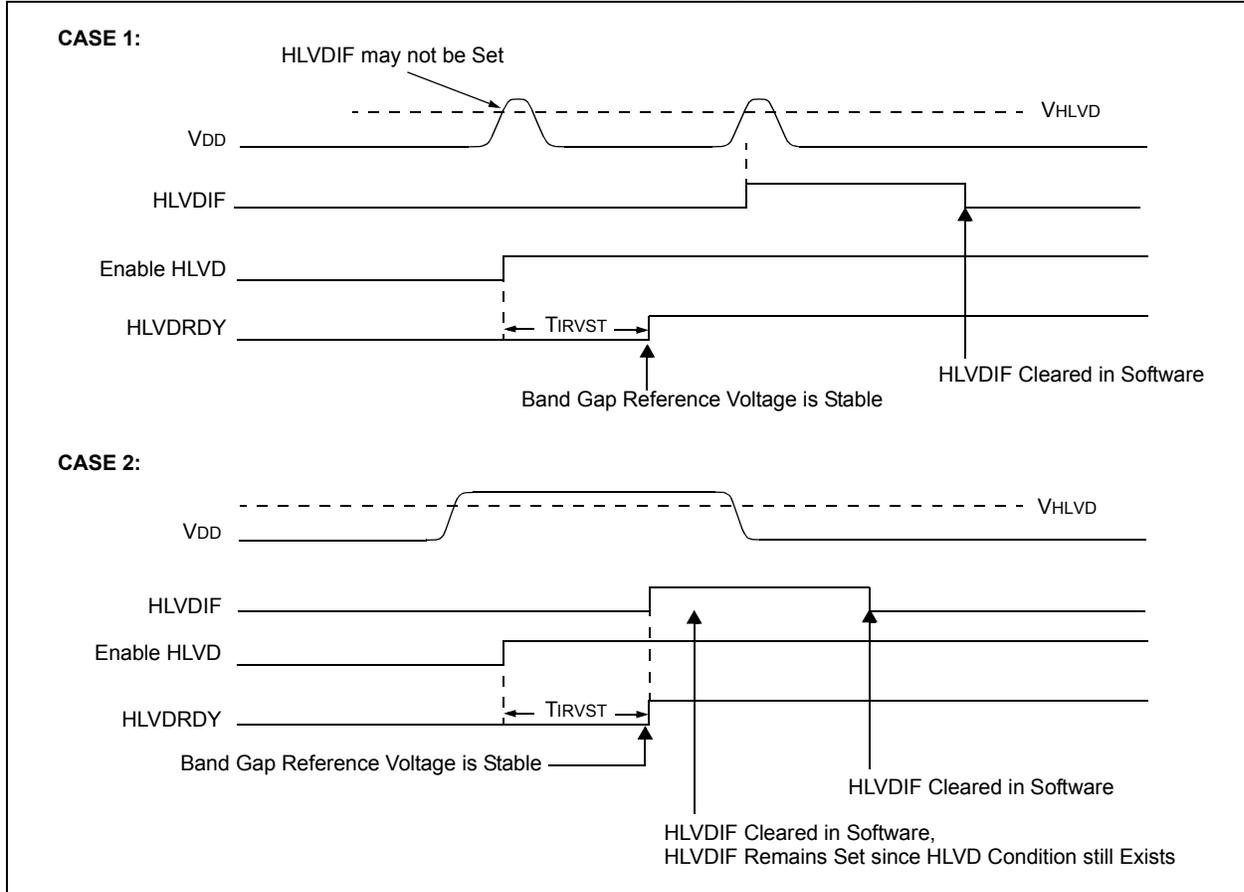
**Note 1:** When reading a PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert as an analog input, according to the input specification.

**2:** Analog levels on any pin defined as a digital input, may cause the input buffer to consume more current than is specified.

**FIGURE 32-3: ANALOG INPUT MODEL**



**FIGURE 33-3: HIGH-VOLTAGE DETECT OPERATION (HLVDINTH = 1)**

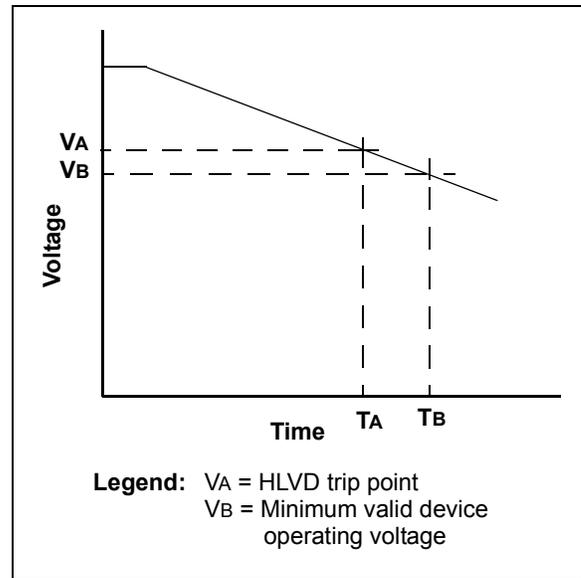


### 33.5 Applications

In many applications, it is desirable to detect a drop below, or rise above, a particular voltage threshold. 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 33-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage, V<sub>A</sub>, the HLVD logic generates an interrupt at time, T<sub>A</sub>. The interrupt could cause the execution of an Interrupt Service Routine (ISR), which would allow the application to perform "housekeeping tasks" and a controlled shutdown before the device voltage exits the valid operating range at T<sub>B</sub>. This would give the application a time window, represented by the difference between T<sub>A</sub> and T<sub>B</sub>, to safely exit.

**FIGURE 33-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION**



## NEGF

## Negate f

**Syntax:** NEGF f{,a}

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

**Operation:**  $(\bar{f}) + 1 \rightarrow f$

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

**Encoding:**

0110	110a	ffff	ffff
------	------	------	------

**Description:** Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \leq 95$  (5Fh). See **Section 35.2.3 "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 register 'f'

**Example:** NEGF REG, 1

Before Instruction

REG = 0011 1010 [3Ah]

After Instruction

REG = 1100 0110 [C6h]

## NOP

## No Operation

**Syntax:** NOP

**Operands:** None

**Operation:** No operation

**Status Affected:** None

**Encoding:**

0000	0000	0000	0000
1111	xxxx	xxxx	xxxx

**Description:** No operation.

**Words:** 1

**Cycles:** 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

**Example:**

None.

# PIC18(L)F26/45/46K40

## TBLRD Table Read

**Syntax:** TBLRD (\*; \*+; \*-; +\*)

**Operands:** None

**Operation:** if TBLRD \*,  
(Prog Mem (TBLPTR)) → TABLAT;  
TBLPTR – No Change;  
if TBLRD \*+,  
(Prog Mem (TBLPTR)) → TABLAT;  
(TBLPTR) + 1 → TBLPTR;  
if TBLRD \*-,  
(Prog Mem (TBLPTR)) → TABLAT;  
(TBLPTR) – 1 → TBLPTR;  
if TBLRD +\*,  
(TBLPTR) + 1 → TBLPTR;  
(Prog Mem (TBLPTR)) → TABLAT;

**Status Affected:** None

Encoding:	0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*
-----------	------	------	------	---

**Description:** This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used. The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

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

The TBLRD instruction can modify the value of TBLPTR as follows:

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

**Words:** 1

**Cycles:** 2

**Q Cycle Activity:**

	Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation	No operation
No operation	No operation (Read Program Memory)	No operation	No operation	No operation (Write TABLAT)

## TBLRD Table Read (Continued)

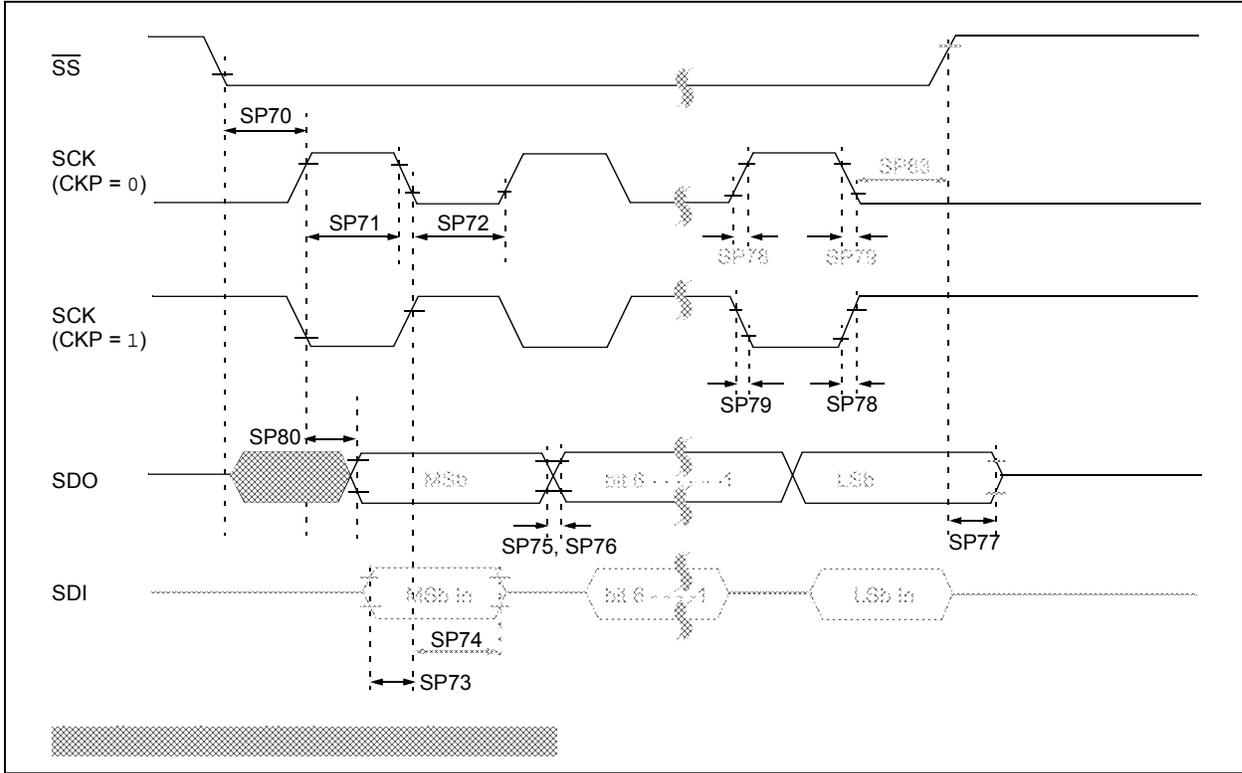
**Example1:** TBLRD \*+ ;

Before Instruction  
TABLAT = 55h  
TBLPTR = 00A356h  
MEMORY (00A356h) = 34h  
After Instruction  
TABLAT = 34h  
TBLPTR = 00A357h

**Example2:** TBLRD +\* ;

Before Instruction  
TABLAT = AAh  
TBLPTR = 01A357h  
MEMORY (01A357h) = 12h  
MEMORY (01A358h) = 34h  
After Instruction  
TABLAT = 34h  
TBLPTR = 01A358h

**FIGURE 37-18: SPI SLAVE MODE TIMING (CKE = 0)**



**FIGURE 37-19: SPI SLAVE MODE TIMING (CKE = 1)**

