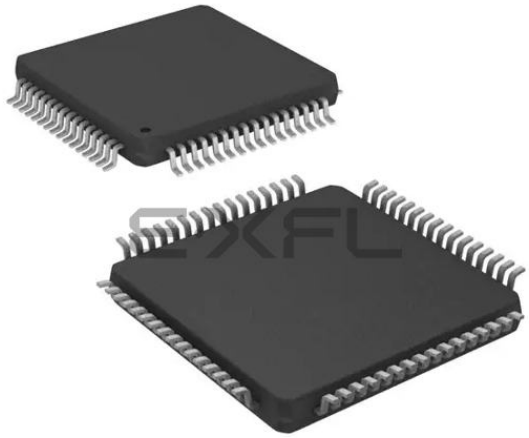


Welcome to E-XFL.COM

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, HLVD, POR, PWM, WDT
Number of I/O	54
Program Memory Size	96KB (48K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf6627t-i-pt

2.6 Internal Oscillator Block

The PIC18F8722 family of devices includes an internal oscillator block which generates two different clock signals; either can be used as the microcontroller's clock source. This may 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 device clock. It also drives a postscaler, which can provide a range of clock frequencies from 31 kHz to 4 MHz. The INTOSC output is enabled when a clock frequency from 125 kHz to 8 MHz is selected. The INTOSC output can also be enabled when 31 kHz is selected, depending on the INTSRC bit (OSCTUNE<7>).

The other clock source is the internal RC oscillator (INTRC), which provides a nominal 31 kHz output. INTRC is enabled if it is selected as the device clock source; it is also enabled automatically 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 25.0 "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 (page 39).

2.6.1 INTIO MODES

Using the internal oscillator as the clock source eliminates 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 (see Figure 2-8) for digital input and output.
- In INTIO2 mode, OSC1 functions as RA7 and OSC2 functions as RA6 (see Figure 2-9), both for digital input and output.

FIGURE 2-8: INTIO1 OSCILLATOR MODE

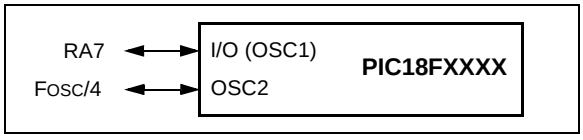
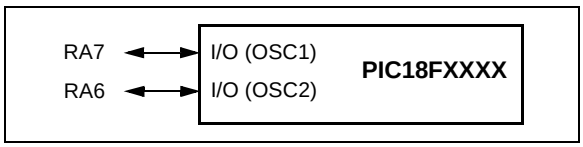


FIGURE 2-9: INTIO2 OSCILLATOR MODE



2.6.2 INTOSC OUTPUT FREQUENCY

The internal oscillator block is calibrated at the factory to produce an INTOSC output frequency of 8 MHz.

The INTRC oscillator operates independently of the INTOSC source. Any changes in INTOSC across voltage and temperature are not necessarily reflected by changes in INTRC or vice versa.

2.6.3 OSCTUNE REGISTER

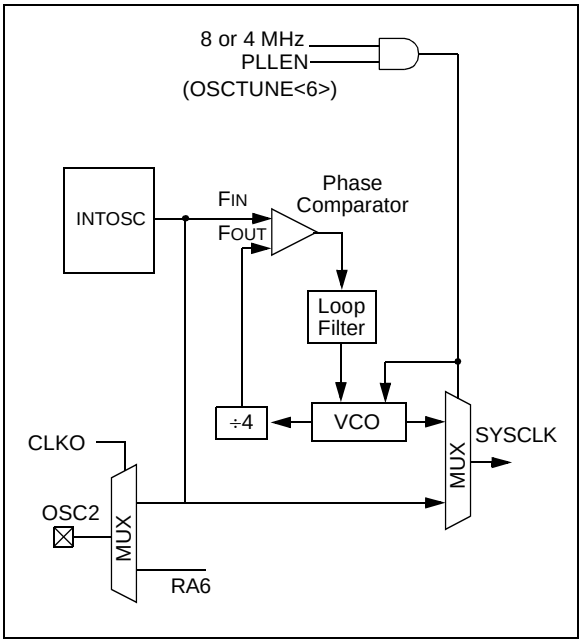
The INTOSC output has been calibrated at the factory but can be adjusted in the user's application. This is done by writing to TUN<4:0> (OSCTUNE<4:0>) in the OSCTUNE register (Register).

When the OSCTUNE register is modified, the INTOSC frequency will begin shifting to the new frequency. The INTOSC clock will stabilize within 1 ms. Code execution continues during this shift. There is no indication that the shift has occurred. The INTRC is not affected by OSCTUNE.

The OSCTUNE register also implements the INTSRC (OSCTUNE<7>) and PLLEN (OSCTUNE<6>) bits, which control certain features of the internal oscillator block. The INTSRC bit allows users to select which internal oscillator provides the clock source when the 31 kHz frequency option is selected. This is covered in greater detail in **Section 2.7.1 "Oscillator Control Register"**.

The PLLEN bit controls the operation of the Phase Locked Loop (PLL) in internal oscillator modes (see Figure 2-10).

FIGURE 2-10: INTOSC AND PLL BLOCK DIAGRAM



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
                MOVLWD  ATA_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
                MOVFF   POSTINC0, WREG          ; 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_WORD_TO_HREGS
```

7.5.4 16-BIT MODE TIMING

The presentation of control signals on the External Memory Bus is different for the various operating modes. Typical signal timing diagrams are shown in Figure 7-4 through Figure 7-6. All examples assume either 20-bit or 21-bit address widths.

FIGURE 7-4: EXTERNAL MEMORY BUS TIMING FOR TBLRD WITH A 1 TCY WAIT STATE (MICROPROCESSOR MODE)

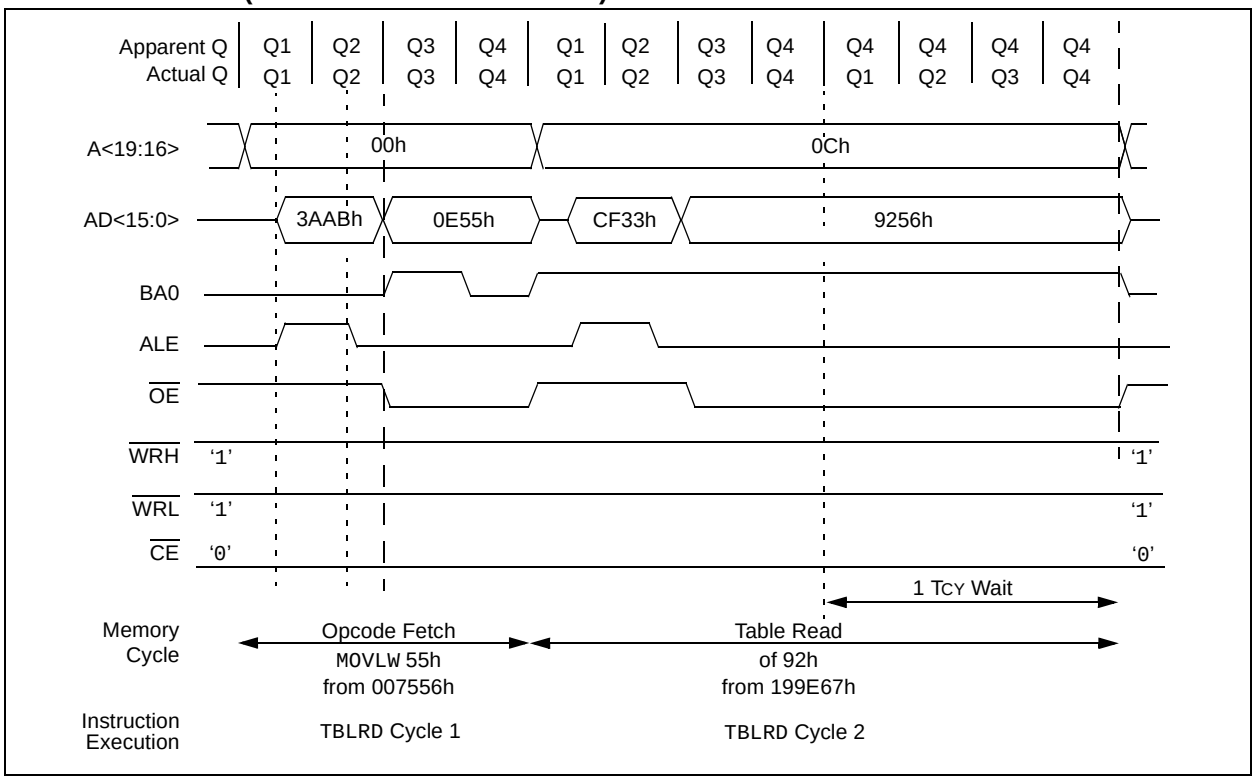
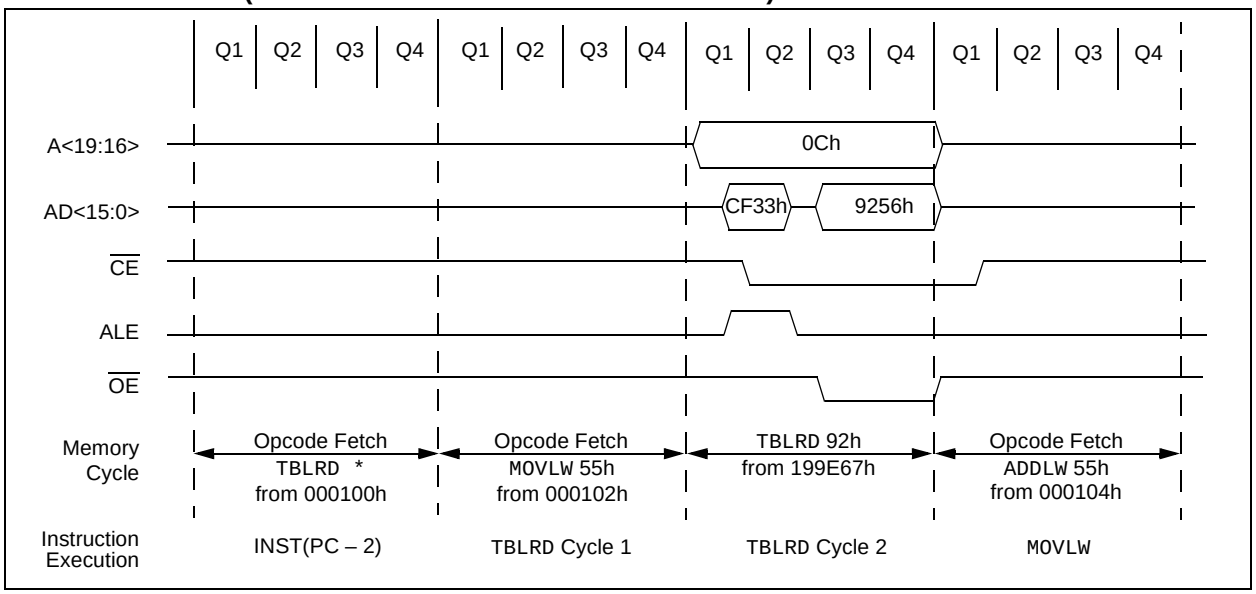


FIGURE 7-5: EXTERNAL MEMORY BUS TIMING FOR TBLRD (EXTENDED MICROCONTROLLER MODE)



10.6 INTx Pin Interrupts

External interrupts on the RB0/INT0, RB1/INT1, RB2/INT2 and RB3/INT3 pins are edge-triggered. If the corresponding INTEDGx bit in the INTCON2 register is set (= 1), the interrupt is triggered by a rising edge; if the bit is clear, the trigger is on the falling edge. When a valid edge appears on the RBx/INTx pin, the corresponding flag bit, INTxIF, is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxIE. Flag bit, INTxIF, must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt.

All external interrupts (INT0, INT1, INT2 and INT3) can wake-up the processor from the power-managed modes if bit INTxIE was set prior to going into power-managed modes. If the Global Interrupt Enable bit, GIE, is set, the processor will branch to the interrupt vector following wake-up.

Interrupt priority for INT1, INT2 and INT3 is determined by the value contained in the interrupt priority bits, INT1IP (INTCON3<6>), INT2IP (INTCON3<7>) and INT3IP (INTCON2<1>). There is no priority bit associated with INT0. It is always a high-priority interrupt source.

10.7 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow in the TMR0 register (FFh → 00h) will set flag bit, TMR0IF. In 16-bit mode, an overflow in the TMR0H:TMR0L register pair (FFFFh → 0000h) will set TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE (INTCON<5>). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP (INTCON2<2>). See **Section 12.0 "Timer0 Module"** for further details on the Timer0 module.

10.8 PORTB Interrupt-on-Change

An input change on PORTB<7:4> sets flag bit, RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit, RBIE (INTCON<3>). Interrupt priority for PORTB interrupt-on-change is determined by the value contained in the interrupt priority bit, RBIP (INTCON2<0>).

10.9 Context Saving During Interrupts

During interrupts, the return PC address is saved on the stack. Additionally, the WREG, STATUS and BSR registers are saved on the fast return stack. If a fast return from interrupt is not used (see **Section 5.3 "Data Memory Organization"**), the user may need to save the WREG, STATUS and BSR registers on entry to the Interrupt Service Routine. Depending on the user's application, other registers may also need to be saved. Example 10-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

EXAMPLE 10-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM

```
MOVWF   W_TEMP                      ; W_TEMP is in virtual bank
MOVFF   STATUS, STATUS_TEMP         ; STATUS_TEMP located anywhere
MOVFF   BSR, BSR_TEMP               ; BSR_TMEP located anywhere
;
; USER ISR CODE
;
MOVFF   BSR_TEMP, BSR               ; Restore BSR
MOVF    W_TEMP, W                   ; Restore WREG
MOVFF   STATUS_TEMP, STATUS         ; Restore STATUS
```

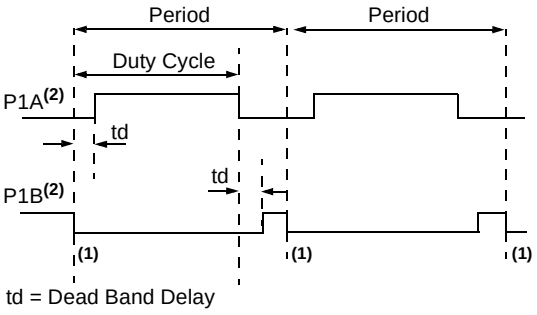
18.4.4 HALF-BRIDGE MODE

In the Half-Bridge Output mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the P1A pin, while the complementary PWM output signal is output on the P1B pin (Figure 18-4). This mode can be used for half-bridge applications, as shown in Figure 18-5, or for full-bridge applications, where four power switches are being modulated with two PWM signals.

In Half-Bridge Output mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of bits, P1DC<6:0> sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. See **Section 18.4.6 "Programmable Dead-Band Delay"** for more details on dead-band delay operations.

The P1A and P1B outputs are multiplexed with the PORTC<2> and PORTE<6> data latches. Alternatively, P1B can be assigned to PORTH<7> by programming the ECCPMX Configuration bit to '0'. See Table 18-1, Table 18-2 and Table 18-3 for more information. The associated TRIS bit must be cleared to configure P1A and P1B as outputs.

FIGURE 18-4: HALF-BRIDGE PWM OUTPUT



Note 1: At this time, the TMR2 register is equal to the PR2 register.

2: Output signals are shown as active-high.

FIGURE 18-5: EXAMPLES OF HALF-BRIDGE OUTPUT MODE APPLICATIONS

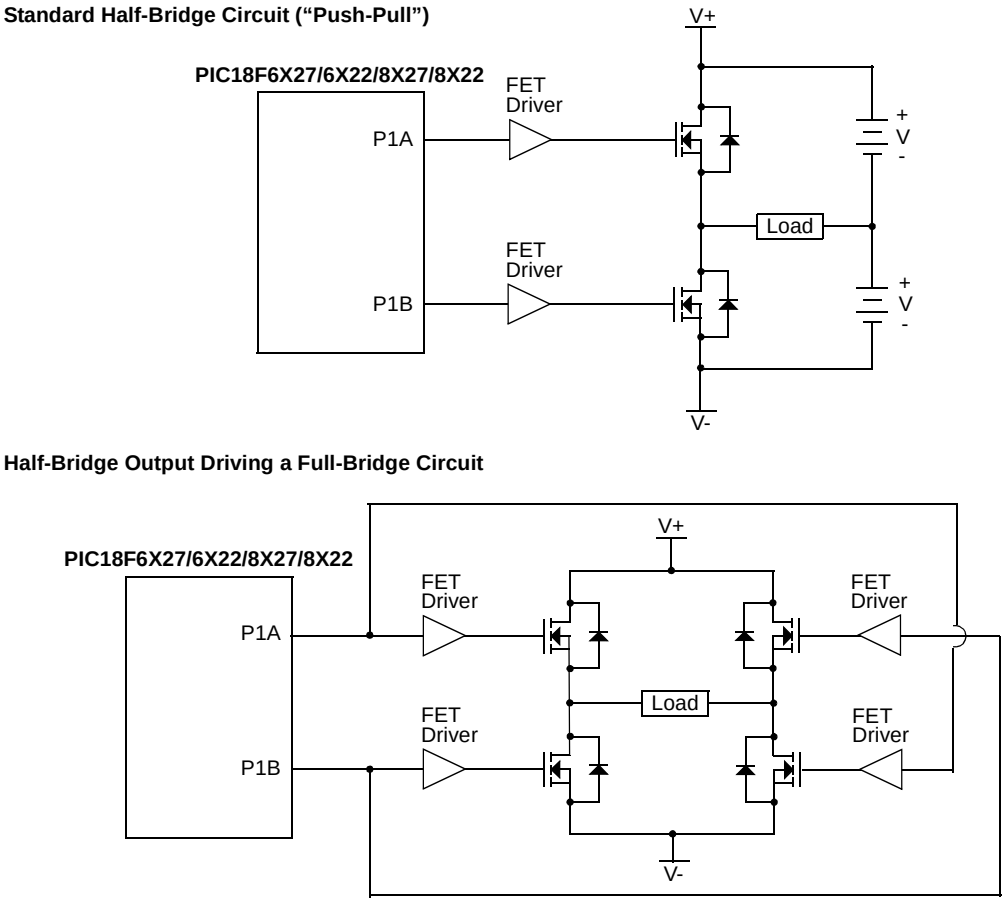
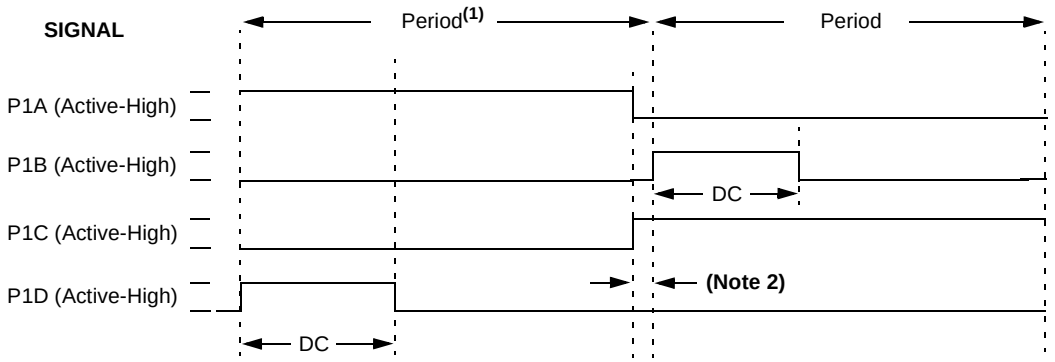
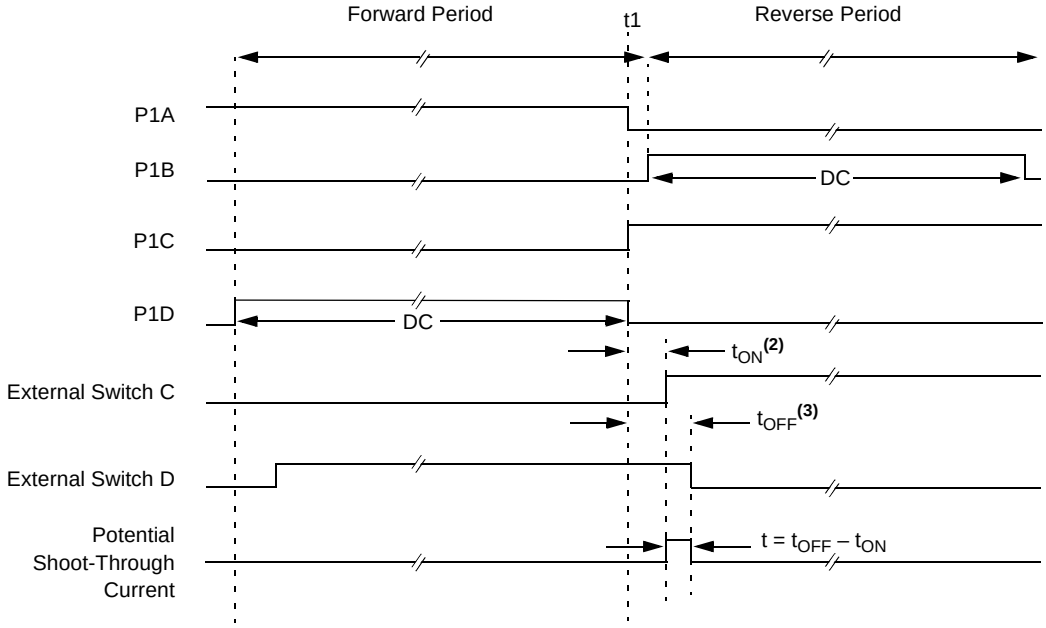


FIGURE 18-8: PWM DIRECTION CHANGE



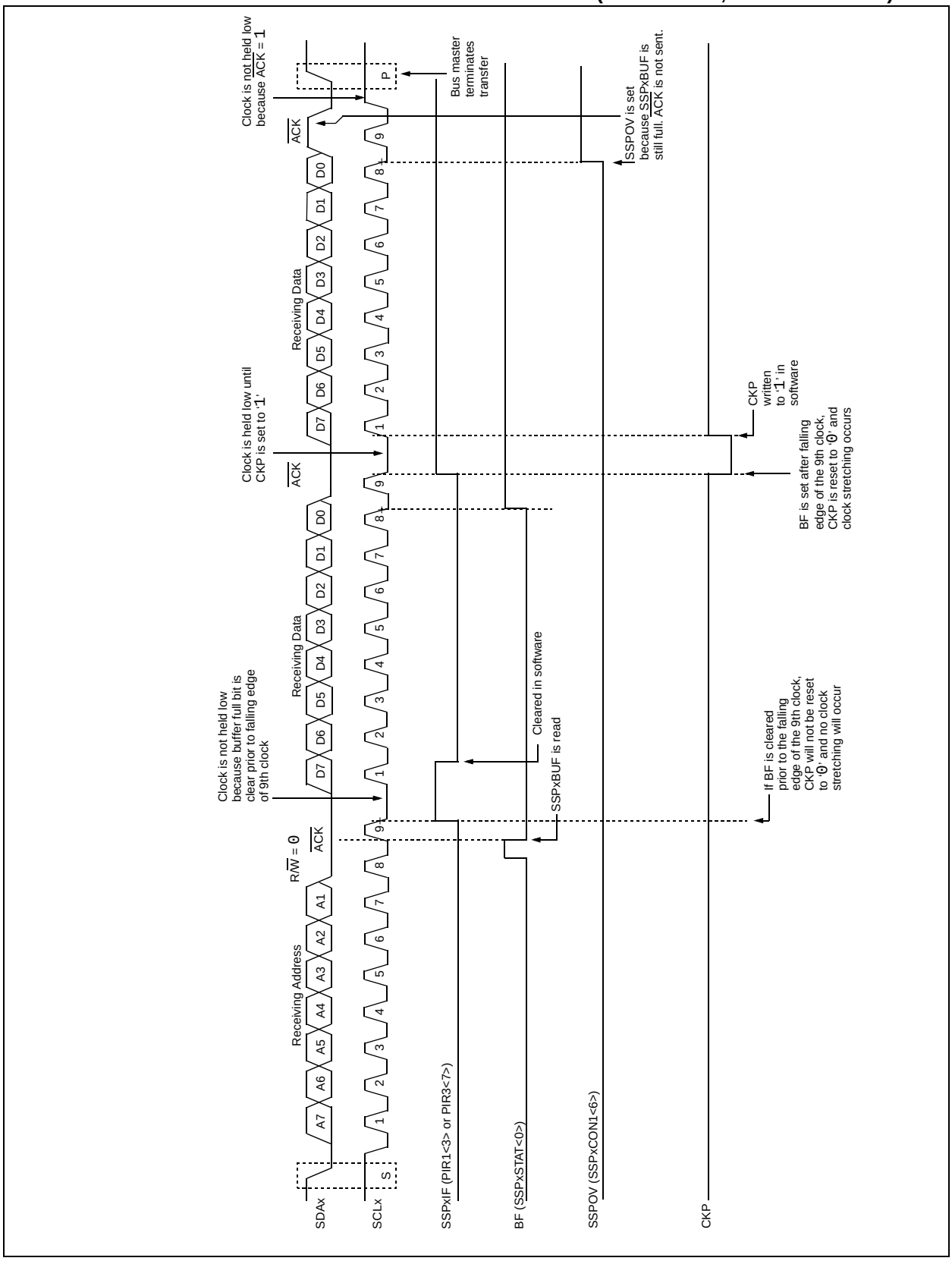
Note 1: The direction bit in the ECCP1 Control register (CCP1CON<7>) is written any time during the PWM cycle.
2: When changing directions, the P1A and P1C signals switch before the end of the current PWM cycle at intervals of 4 TOSC, 16 TOSC or 64 TOSC, depending on the Timer2 prescaler value. The modulated P1B and P1D signals are inactive at this time.

FIGURE 18-9: PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE(1)



Note 1: All signals are shown as active-high.
2: t_{ON} is the turn-on delay of power switch QC and its driver.
3: t_{OFF} is the turn-off delay of power switch QD and its driver.

FIGURE 19-13: I²C™ SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)



19.4.6.1 I²C Master Mode Operation

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDAx, while SCLx outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/$\overline{W}$) bit. In this case, the R/$\overline{W}$ bit will be logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address, followed by a '1' to indicate the receive bit. Serial data is received via SDAx, while SCLx outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

The Baud Rate Generator used for the SPI mode operation is used to set the SCLx clock frequency for either 100 kHz, 400 kHz or 1 MHz I²C operation. See **Section 19.4.7 "Baud Rate"** for more detail.

A typical transmit sequence would go as follows:

1. The user generates a Start condition by setting the Start Enable bit, SEN (SSPxCON2<0>).
2. SSPxIF is set. The MSSP module will wait the required start time before any other operation takes place.
3. The user loads the SSPxBUF with the slave address to transmit.
4. Address is shifted out the SDAx pin until all 8 bits are transmitted.
5. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPxCON2 register (SSPxCON2<6>).
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
7. The user loads the SSPxBUF with eight bits of data.
8. Data is shifted out the SDAx pin until all 8 bits are transmitted.
9. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPxCON2 register (SSPxCON2<6>).
10. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
11. The user generates a Stop condition by setting the Stop Enable bit, PEN (SSPxCON2<2>).
12. Interrupt is generated once the Stop condition is complete.

19.4.17.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

a) A low level is sampled on SDAx when SCLx goes from low level to high level.

b) SCLx goes low before SDAx is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user deasserts SDAx and the pin is allowed to float high, the BRG is loaded with SSPxADD<6:0> and counts down to '0'. The SCLx pin is then deasserted and when sampled high, the SDAx pin is sampled.

If SDAx is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', Figure 19-29). If SDAx is sampled high, the BRG is reloaded and begins counting. If SDAx goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDAx at exactly the same time.

If SCLx goes from high-to-low before the BRG times out and SDAx has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition (see Figure 19-30).

If, at the end of the BRG time-out, both SCLx and SDAx are still high, the SDAx pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCLx pin, the SCLx pin is driven low and the Repeated Start condition is complete.

FIGURE 19-29: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

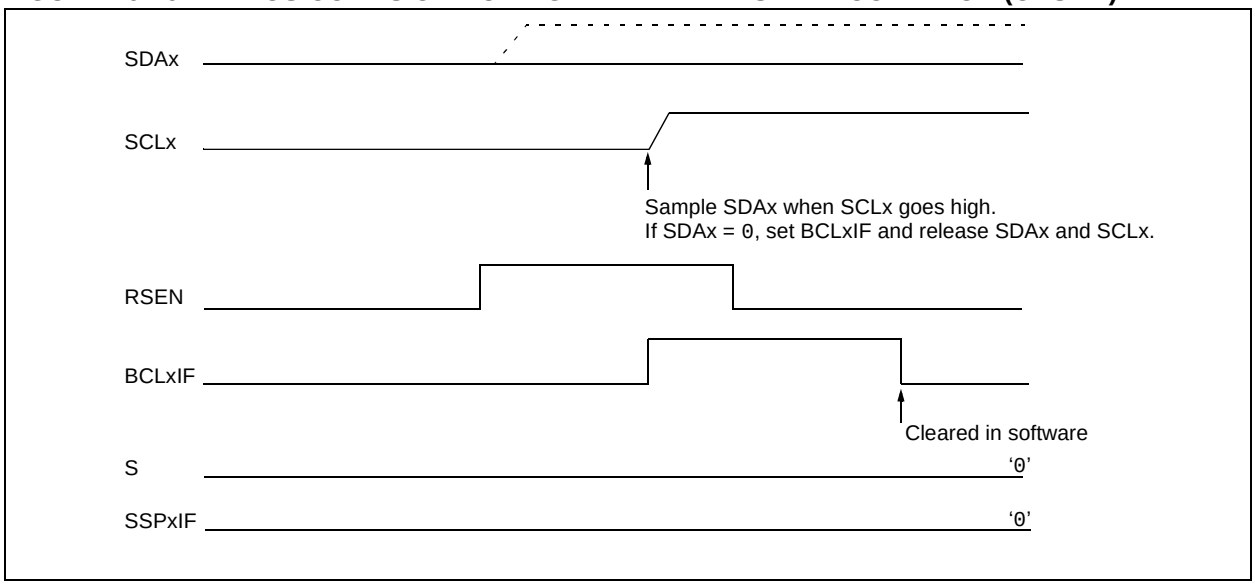
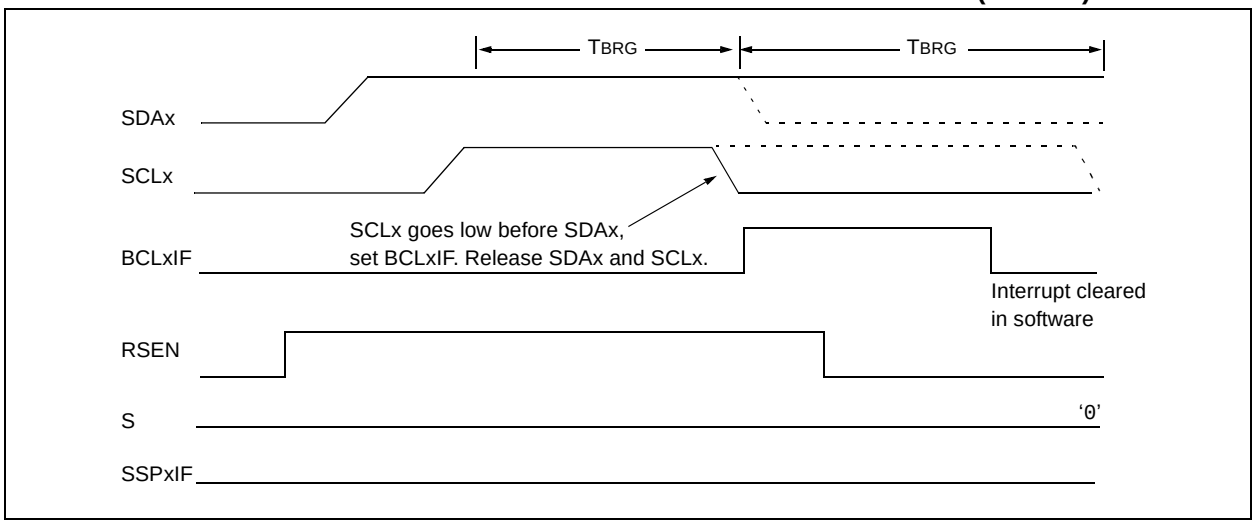


FIGURE 19-30: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



20.2.5 BREAK CHARACTER SEQUENCE

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN 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 (TXSTAx<3> and TXSTAx<5>) are set while the Transmit Shift register is loaded with data. Note that the value of data written to TXREGx 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 specification).

Note that the data value written to the TXREGx 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 20-10 for the timing of the Break character sequence.

20.2.5.1 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 bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to set up the Break character.
3. Load the TXREGx with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREGx 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 TXREGx becomes empty, as indicated by the TXxIF, the next data byte can be written to TXREGx.

20.2.6 RECEIVING A BREAK CHARACTER

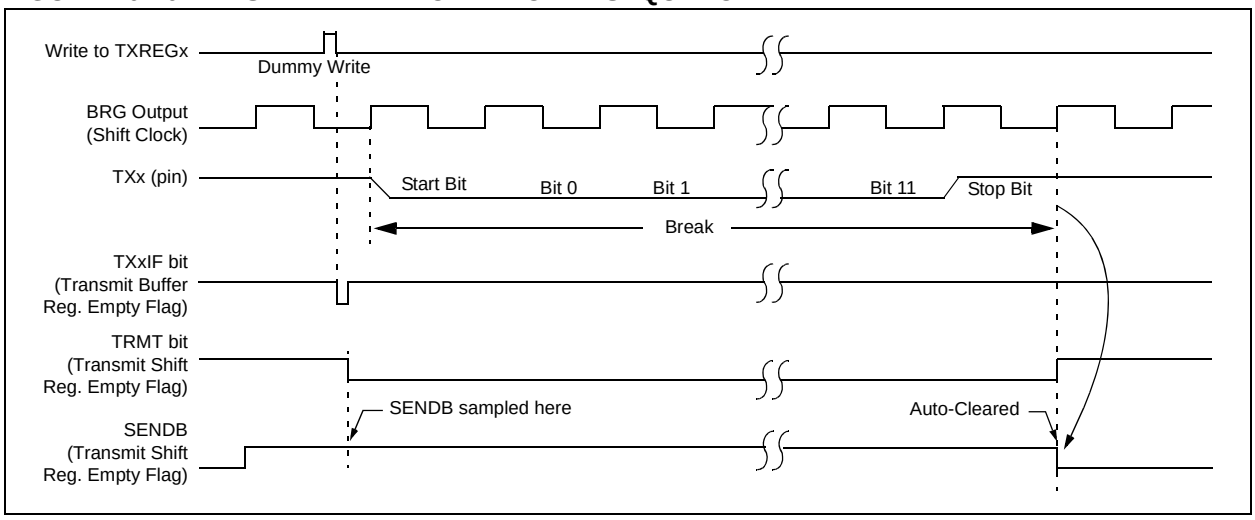
The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 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 20.2.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the EUSART will sample the next two transitions on RXx/DTx, cause an RCxIF 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 once the TXxIF interrupt is observed.

FIGURE 20-10: SEND BREAK CHARACTER SEQUENCE



20.3 EUSART Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTAx<7>). In this mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit SYNC (TXSTAx<4>). In addition, enable bit SPEN (RCSTAx<7>) is set in order to configure the TXx and RXx pins to CKx (clock) and DTx (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CKx line. Clock polarity is selected with the SCKP bit (BAUDCONx<4>); setting SCKP sets the Idle state on CKx as high, while clearing the bit sets the Idle state as low. This option is provided to support Microwire devices with this module.

20.3.1 EUSART SYNCHRONOUS MASTER TRANSMISSION

The EUSART transmitter block diagram is shown in Figure 20-3. The heart of the transmitter is the Transmit (Serial) Shift Register (TSRx). The Shift register obtains its data from the Read/Write Transmit Buffer register, TXREGx. The TXREGx register is loaded with data in software. The TSRx register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSRx is loaded with new data from the TXREGx (if available).

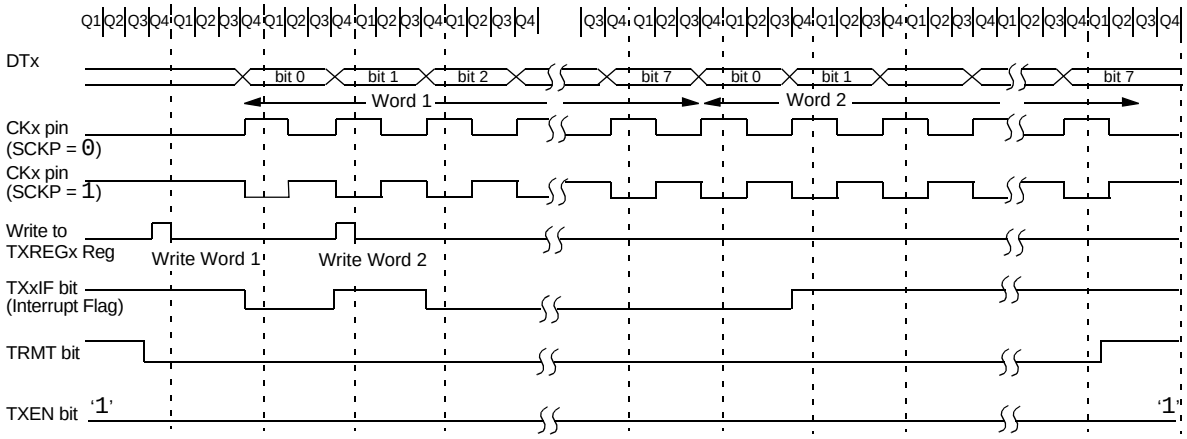
Once the TXREGx register transfers the data to the TSRx register (occurs in one TCY), the TXREGx is empty and the TXxIF flag bit is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXxIE. TXxIF is set regardless of the state of enable bit TXxIE; it cannot be cleared in software. It will reset only when new data is loaded into the TXREGx register.

While flag bit TXxIF indicates the status of the TXREGx register, another bit, TRMT (TXSTAx<1>), shows the status of the TSRx register. TRMT is a read-only bit which is set when the TSRx is empty. No interrupt logic is tied to this bit, so the user must poll this bit in order to determine if the TSRx register is empty. The TSRx is not mapped in data memory so it is not available to the user.

To set up a Synchronous Master Transmission:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit TXxIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREGx register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 20-11: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode, SPBRGx = 0, continuous transmission of two 8-bit words.

REGISTER 25-1: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)

R/P-0	R/P-0	U-0	U-0	R/P-0	R/P-1	R/P-1	R/P-1
IESO	FCMEN	—	—	FOSC3	FOSC2	FOSC1	FOSC0
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 **IESO:** Internal/External Oscillator Switchover bit
1 = Two-Speed Start-up enabled
0 = Two-Speed Start-up disabled

bit 6 **FCMEN:** Fail-Safe Clock Monitor Enable bit
1 = Fail-Safe Clock Monitor enabled
0 = Fail-Safe Clock Monitor disabled

bit 5-4 **Unimplemented:** Read as ‘0’

bit 3-0 **FOSC<3:0>:** Oscillator Selection bits
11xx = External RC oscillator, CLKO function on RA6
101x = External RC oscillator, CLKO function on RA6
1001 = Internal oscillator block, CLKO function on RA6, port function on RA7
1000 = Internal oscillator block, port function on RA6 and RA7
0111 = External RC oscillator, port function on RA6
0110 = HS oscillator, PLL enabled (Clock Frequency = 4 x FOSC1)
0101 = EC oscillator, port function on RA6
0100 = EC oscillator, CLKO function on RA6
0011 = External RC oscillator, CLKO function on RA6
0010 = HS oscillator
0001 = XT oscillator
0000 = LP oscillator

25.3 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 INTOSC 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.

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 IRCF<2:0> immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF2:0> bits prior to entering Sleep mode.

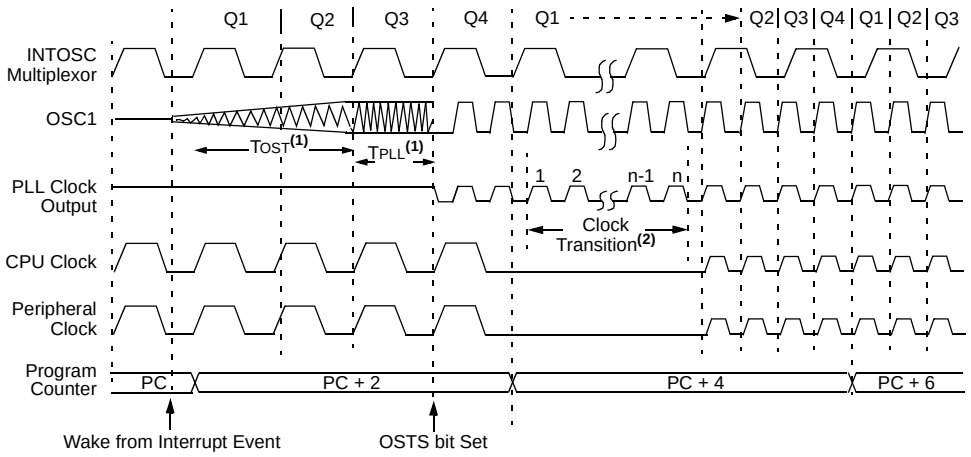
In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

25.3.1 SPECIAL CONSIDERATIONS FOR USING TWO-SPEED START-UP

While using the INTOSC oscillator in Two-Speed Start-up, the device still obeys the normal command sequences for entering power-managed modes, including multiple SLEEP instructions (refer to **Section 3.1.4 "Multiple Sleep Commands"**). In practice, this means that user code can change the SCS<1:0> 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 25-2: 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.

2: Clock transition typically occurs within 2-4 TOSC.

TABLE 26-2: PIC18FXXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BIT-ORIENTED OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	n, s	Call Subroutine 1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	n	Go to Address 1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	4
POP	—	Pop Top of Return Stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push Top of Return Stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software Device Reset	1	0000	0000	1111	1111	All	
RETFIE	s	Return from Interrupt Enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

GOTO **Unconditional Branch**

Syntax: GOTO k

Operands: $0 \le k \le 1048575$

Operation: $k \rightarrow PC<20:1>$

Status Affected: None

Encoding:

1st word (k<7:0>)	1110	1111	k7kkk	kkkk0
2nd word(k<19:8>)	1111	k19kkk	kkkk	kkkk8

Description: GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	No operation	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: GOTO THERE

After Instruction
PC = Address (THERE)

INCF **Increment f**

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

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

Operation: $(f) + 1 \rightarrow dest$

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

Encoding:

0010	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 '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 26.2.3 "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 to destination

Example: INCF CNT, 1, 0

Before Instruction
CNT = FFh
Z = 0
C = ?
DC = ?
After Instruction
CNT = 00h
Z = 1
C = 1
DC = 1

NEGF **Negate f**

Syntax: NEGF f {,a}

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

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

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 (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 26.2.3 “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 1111	0000 xxxx	0000 xxxx	0000 xxxx

Description: No operation.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

Example:

None.

26.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, the PIC18F8722 family of devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment Indirect and Indexed Addressing operations and the implementation of Indexed Literal Offset Addressing for many of the standard PIC18 instructions.

The additional features of the extended instruction set are enabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers, or use them for Indexed Addressing. Two of the instructions, ADDFSR and SUBFSR, each have an additional special instantiation for using FSR2. These versions (ADDULNK and SUBULNK) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 26-3. Detailed descriptions are provided in **Section 26.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 26-1 (page 322) apply to both the standard and extended PIC18 instruction sets.

Note:	The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

26.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of Indexed Addressing, it is enclosed in square brackets (“[ ]”). This is done to indicate that the argument is used as an index or offset. The MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 26.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**.

Note:	In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces (“{ }”).

TABLE 26-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected
				MSb			LSb	
ADDFSR	f, k	Add Literal to FSR	1	1110	1000	ffkk	kkkk	None
ADDULNK	k	Add Literal to FSR2 and Return	2	1110	1000	11kk	kkkk	None
CALLW		Call Subroutine using WREG	2	0000	0000	0001	0100	None
MOVSF	z_s, f_d	Move z_s (source) to 1st word	2	1110	1011	0zzz	zzzz	None
		f_d (destination) 2nd word		1111	ffff	ffff	ffff	
MOVSS	z_s, z_d	Move z_s (source) to 1st word	2	1110	1011	1zzz	zzzz	None
		z_d (destination) 2nd word		1111	xxxx	xzzz	zzzz	
PUSHL	k	Store Literal at FSR2, Decrement FSR2	1	1110	1010	kkkk	kkkk	None
SUBFSR	f, k	Subtract Literal from FSR	1	1110	1001	ffkk	kkkk	None
SUBULNK	k	Subtract Literal from FSR2 and Return	2	1110	1001	11kk	kkkk	None

28.4.2 TIMING CONDITIONS

The temperature and voltages specified in Table 28-5 apply to all timing specifications unless otherwise noted. Figure 28-5 specifies the load conditions for the timing specifications.

Note: Because of space limitations, the generic terms "PIC18FXXXX" and "PIC18LFXXXX" are used throughout this section to refer to the PIC18F6X27/6X22/8X27/8X22 and PIC18LF6X27/6X22/8X27/8X22 families of devices specifically and only those devices.

TABLE 28-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC

AC CHARACTERISTICS	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in the DC specifications in **Section 28.1** and **Section 28.3**. LF parts operate for industrial temperatures only.

FIGURE 28-5: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS

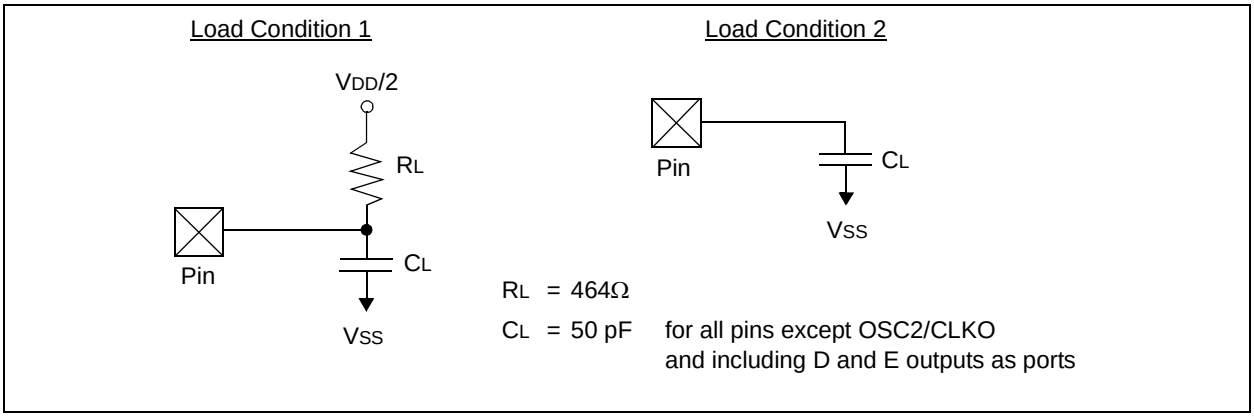
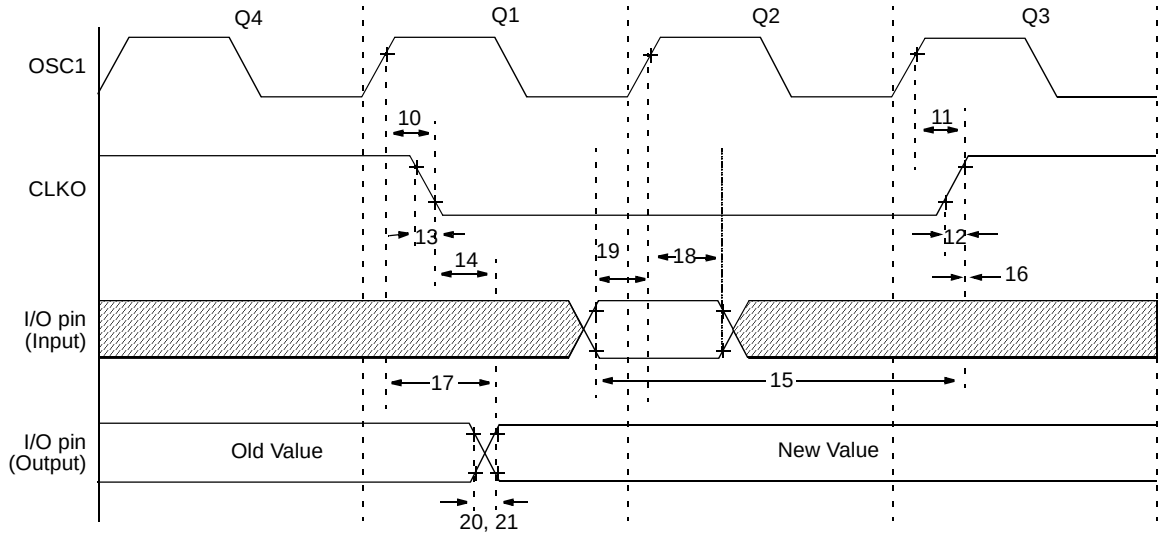


FIGURE 28-7: CLKO AND I/O TIMING



Note: Refer to Figure 28-5 for load conditions.

TABLE 28-9: CLKO AND I/O TIMING REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
10	TosH2ckL	OSC1 ↑ to CLKO ↓		—	75	200	ns	**(Note 1)**
11	TosH2ckH	OSC1 ↑ to CLKO ↑		—	75	200	ns	**(Note 1)**
12	TckR	CLKO Rise Time		—	35	100	ns	**(Note 1)**
13	TckF	CLKO Fall Time		—	35	100	ns	**(Note 1)**
14	TckL2ioV	CLKO ↓ to Port Out Valid		—	—	0.5 TCY + 20	ns	**(Note 1)**
15	TioV2ckH	Port In Valid before CLKO ↑		0.25 TCY + 25	—	—	ns	**(Note 1)**
16	TckH2ioI	Port In Hold after CLKO ↑		0	—	—	ns	**(Note 1)**
17	TosH2ioV	OSC1 ↑ (Q1 cycle) to Port Out Valid		—	50	150	ns	
18	TosH2ioI	OSC1 ↑ (Q2 cycle) to Port Input Invalid (I/O in hold time)	PIC18**F**XXXX	100	—	—	ns	
18A			PIC18**LF**XXXX	200	—	—	ns	VDD = 2.0V
19	TioV2osH	Port Input Valid to OSC1 ↑ (I/O in setup time)		0	—	—	ns	
20	TioR	Port Output Rise Time	PIC18**F**XXXX	—	10	25	ns	
20A			PIC18**LF**XXXX	—	—	60	ns	VDD = 2.0V
21	TioF	Port Output Fall Time	PIC18**F**XXXX	—	10	25	ns	
21A			PIC18**LF**XXXX	—	—	60	ns	VDD = 2.0V
22†	TinP	INTx pin High or Low Time		TCY	—	—	ns	
23†	TrbP	RB<7:4> Change INTx High or Low Time		TCY	—	—	ns	

† These parameters are asynchronous events not related to any internal clock edges.

Note 1: Measurements are taken in RC mode, where CLKO output is 4 x TOSC.

W

X