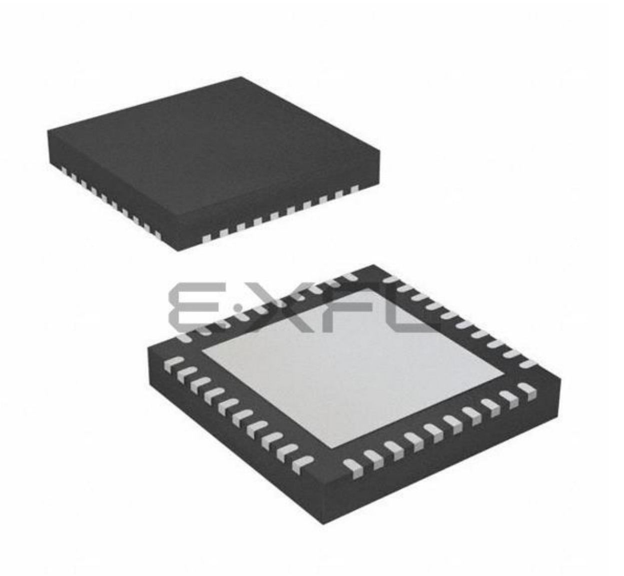


Welcome to E-XFL.COM

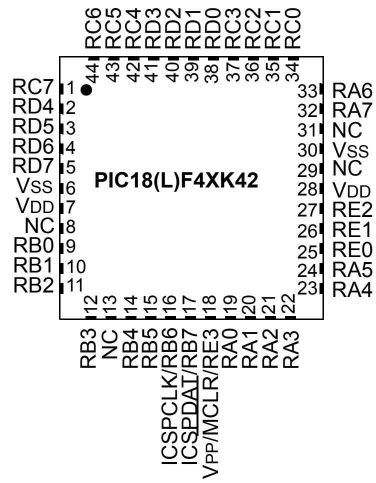
What is "Embedded - Microcontrollers"?

"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	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, HLVD, POR, PWM, WDT
Number of I/O	36
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 35x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	40-UFQFN Exposed Pad
Supplier Device Package	40-UQFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f46k42-i-mv

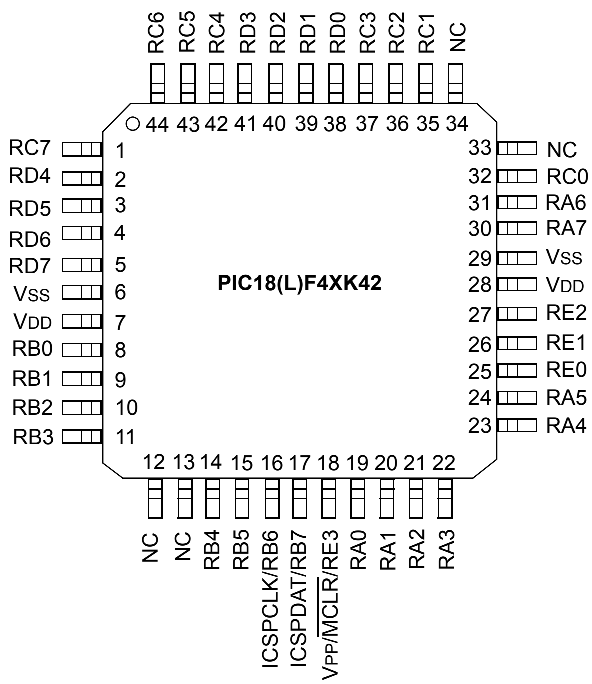
44-pin QFN (8x8x0.9mm)



Note 1: See Table 2 for location of all peripheral functions.

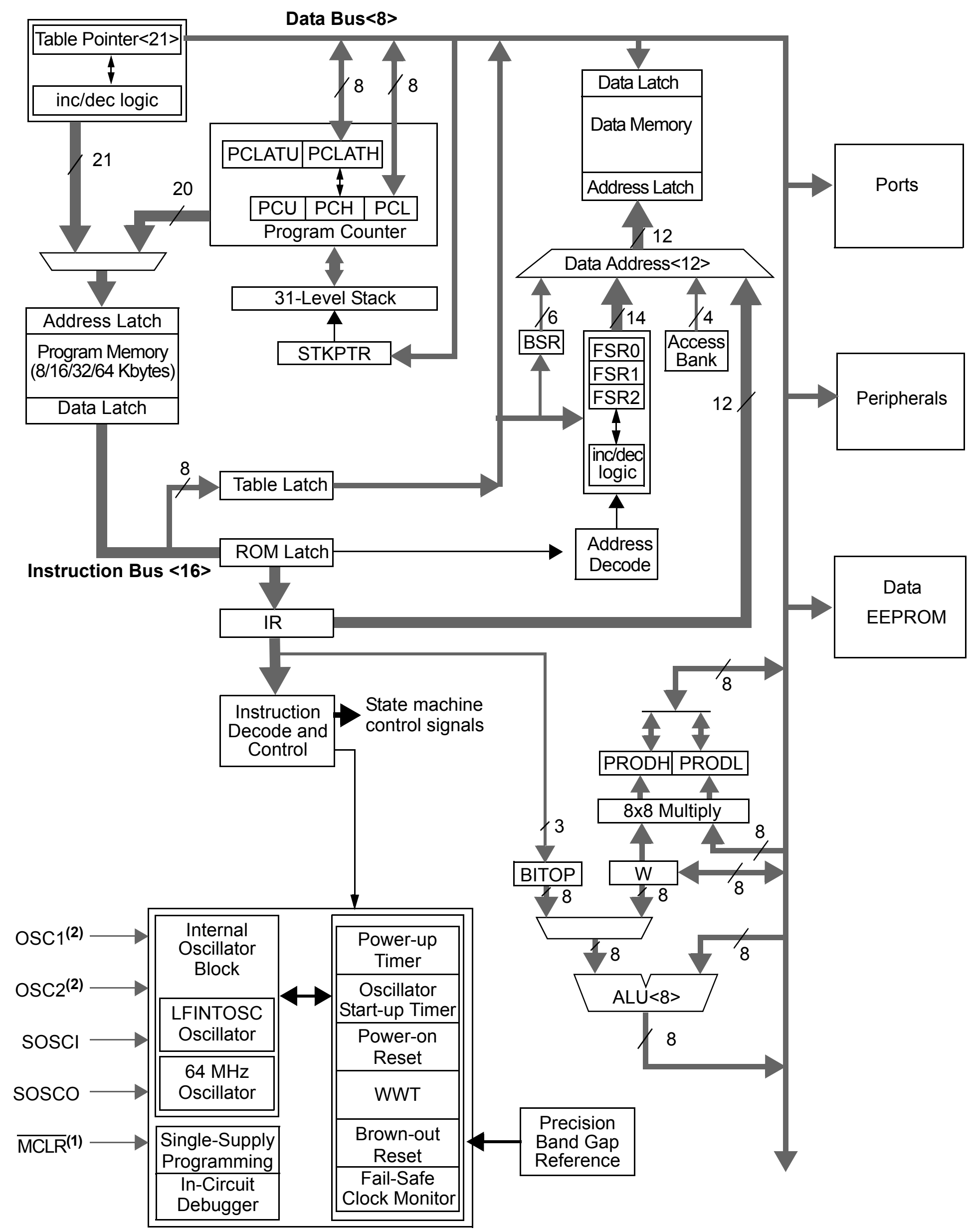
2: It is recommended that the exposed bottom pad be connected to VSS, however it must not be the only VSS connection to the device.

44-pin TQFP (10x10x1mm)



Note: See Table 2 for location of all peripheral functions.

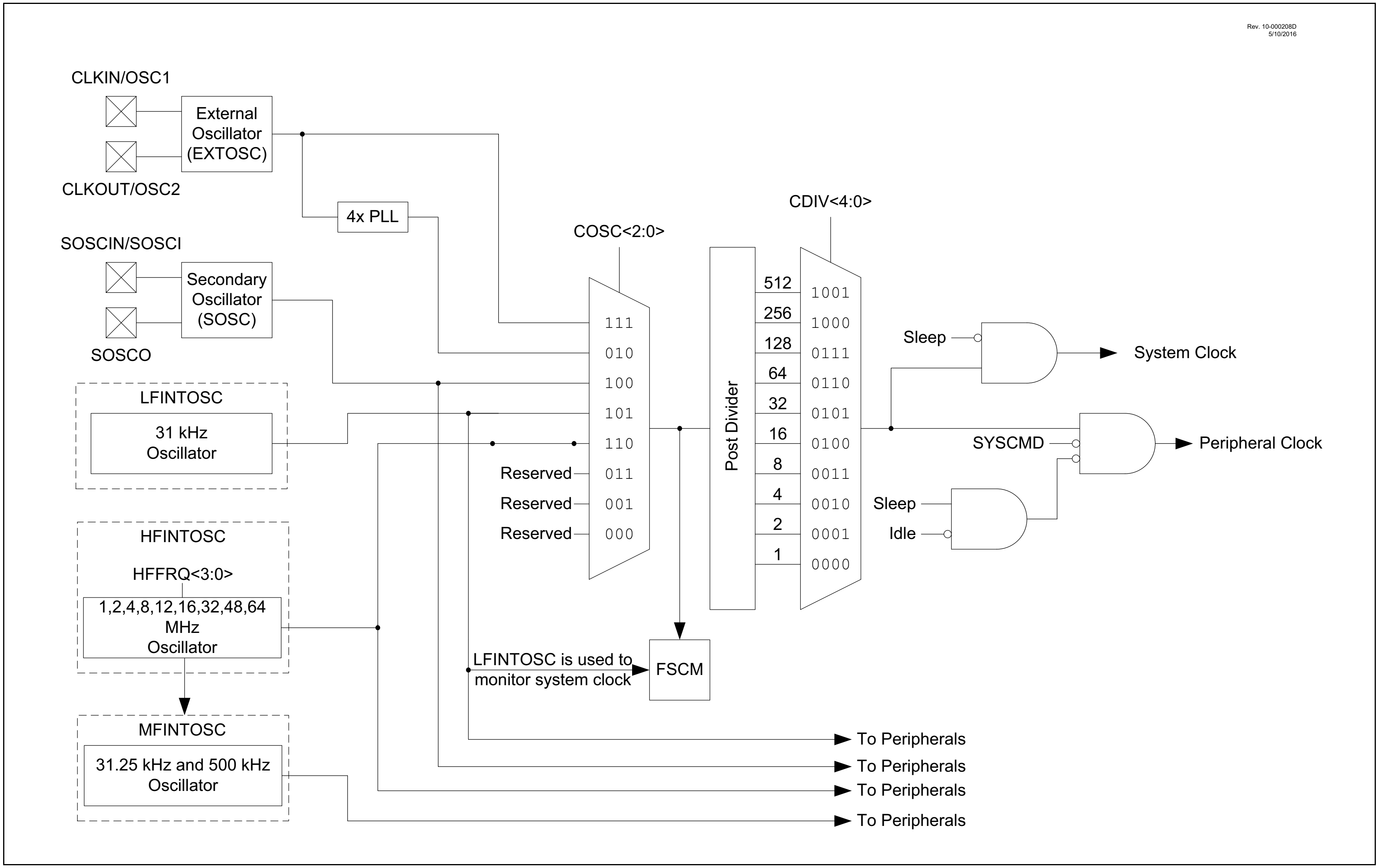
FIGURE 3-1: PIC18(L)F26/27/45/46/47/55/56/57K42 FAMILY BLOCK DIAGRAM



Note 1: RE3 is only available when $\overline{\text{MCLR}}$ functionality is disabled.

2: OSC1/CLKIN and OSC2/CLKOUT are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to Section 7.0, Oscillator Module (with Fail-Safe Clock Monitor) for additional information.

FIGURE 7-1: SIMPLIFIED PIC® MCU CLOCK SOURCE BLOCK DIAGRAM



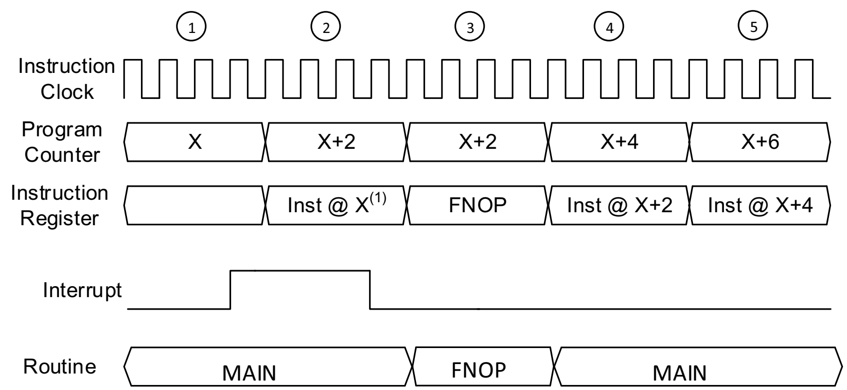
9.7.1 ABORTING INTERRUPTS

If the last instruction before the interrupt controller vectors to the ISR from main routine clears the GIE, PIE or PIR bit associated with the interrupt, the controller executes one force `NOP` cycle before it returns to the main routine.

Figure 9-10 illustrates the sequence of events when a peripheral interrupt is asserted and then cleared on the last executed instruction cycle.

If the GIE, PIE or PIR bit associated with the interrupt is cleared prior to vectoring to the ISR, then the controller continues executing the main routine.

FIGURE 9-10: INTERRUPT TIMING DIAGRAM - ABORTING INTERRUPTS



Note 1: Inst @ X clears the interrupt flag, Example BCF INTCON0, GIE.

REGISTER 9-5: PIR2: PERIPHERAL INTERRUPT REGISTER 2(1)

R-0/0	R-0/0	R-0/0	R-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0	R/W/HS-0/0
I2C1RXIF(2)	SPI1IF(3)	SPI1TXIF(4)	SPI1RXIF(4)	DMA1AIF	DMA1ORIF	DMA1DCNTIF	DMA1SCNTIF
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	HS = Hardware set

bit 7 **I2C1RXIF:** I^2C1 Receive Interrupt Flag bit(2)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 6 **SPI1IF:** SPI1 Interrupt Flag bit(3)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 5 **SPI1TXIF:** SPI1 Transmit Interrupt Flag bit(4)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 4 **SPI1RXIF:** SPI1 Receive Interrupt Flag bit(4)
1 = Interrupt has occurred
0 = Interrupt event has not occurred

bit 3 **DMA1AIF:** DMA1 Abort Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 2 **DMA1ORIF:** DMA1 Overrun Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 1 **DMA1DCNTIF:** DMA1 Destination Count Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

bit 0 **DMA1SCNTIF:** DMA1 Source Count Interrupt Flag bit
1 = Interrupt has occurred (must be cleared by software)
0 = Interrupt event has not occurred

Note 1: Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

2: I2CxTXIF and I2CxRXIF are read-only bits. To clear the interrupt condition, the CLRBF bit in I2CxSTAT1 register must be set.

3: SPIxIF is a read-only bit. To clear the interrupt condition, all bits in the SPIxINTF register must be cleared.

4: SPIxTXIF and SPIxRXIF are read-only bits and cannot be set/cleared by the software.

REGISTER 9-29: IPR4: PERIPHERAL INTERRUPT PRIORITY REGISTER 4

R/W-1/1	R/W-1/1	R/W-1/1	U-0	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
CLC1IP	CWG1IP	NCO1IP	—	CCP1IP	TMR2IP	TMR1GIP	TMR1IP
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 **CLC1IP:** CLC1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **CWG1IP:** CWG1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **NCO1IP:** NCO1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **Unimplemented:** Read as '0'

bit 3 **CCP1IP:** CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **TMR2IP:** TMR2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR1GIP:** TMR1 Gate Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP:** TMR1 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-34: IPR9: PERIPHERAL INTERRUPT PRIORITY REGISTER 9

U-0	U-0	U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
—	—	—	—	CLC3IP	CWG3IP	CCP3IP	TMR6IP
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 7-4 **Unimplemented:** Read as ‘0’

bit 3 **CLC3IP:** CLC3 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CWG3IP:** CWG3 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **CCP3IP:** CCP3 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR6IP:** TMR6 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-35: IPR10: PERIPHERAL INTERRUPT PRIORITY REGISTER 10

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	—	—	—	CLC4IP	CCP4IP
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 7-2 **Unimplemented:** Read as ‘0’

bit 1 **CLC4IP:** CLC4 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **CCP4IP:** CCP4 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 14-11: SCANCON0: SCANNER ACCESS CONTROL REGISTER 0

R/W-0/0	R/W-0/0	R/W/HC-0/0	U-0	U-0	R/W-0/0	R/W-0/0	R-0/0
EN	TRIGEN	SGO	—	—	MREG	BURSTMD	BUSY
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	HC = Bit is cleared by hardware

bit 7 **EN:** Scanner Enable bit(1)
1 = Scanner is enabled
0 = Scanner is disabled

bit 6 **TRIGEN:** Scanner Trigger Enable bit(2)
1 = Scanner trigger is enabled
0 = Scanner trigger is disabled
Refer Table 14-1.

bit 5 **SGO:** Scanner GO bit(3, 4)
1 = When the CRC is ready, the Memory region set by the MREG bit will be accessed and data is passed to the CRC peripheral.
0 = Scanner operations will not occur

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **MREG:** Scanner Memory Region Select bit(2)
1 = Scanner address points to Data EEPROM
0 = Scanner address points to Program Flash Memory

bit 1 **BURSTMD:** Scanner Burst Mode bit
1 = Memory access request to the CPU Arbiter is always true
0 = Memory access request to the CPU Arbiter is dependent on the CRC request and Trigger
Refer Table 14-1.

bit 0 **BUSY:** Scanner Busy Indicator bit
1 = Scanner cycle is in process
0 = Scanner cycle is compete (or never started)

Note 1: Setting EN = 1 (SCANCON0 register) does not affect any other register content.
2: Scanner trigger selection can be set using the SCANTRIG register.
3: This bit can be cleared in software. It is cleared in hardware when LADR>HADR (and a data cycle is not occurring) or when CRCGO = 0 (CRCCON0 register).
4: CRCEN and CRCGO bits (CRCCON0 register) must be set before setting the SGO bit.

16.2.1 DATA REGISTER

PORTx is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISx (Register 16-2). Setting a TRISx bit ('1') will make the corresponding PORTA pin an input (i.e., disable the output driver). Clearing a TRISx bit ('0') will make the corresponding PORTx pin an output (i.e., it enables output driver and puts the contents of the output latch on the selected pin). Example 16-1 shows how to initialize PORTx.

Reading the PORTx register (Register 16-1) reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATx).

The PORT data latch LATx (Register 16-3) holds the output port data and contains the latest value of a LATx or PORTx write.

EXAMPLE 16-1: INITIALIZING PORTA

```
; This code example illustrates
; initializing the PORTA register. The
; other ports are initialized in the same
; manner.

BANKSEL PORTA        ;
CLRF    PORTA        ;Init PORTA
BANKSEL LATA         ;Data Latch
CLRF    LATA         ;
BANKSEL ANSELA       ;
CLRF    ANSELA       ;digital I/O
BANKSEL TRISA        ;
MOVLW   B'11111000'  ;Set RA<7:3> as inputs
MOVWF   TRISA        ;and set RA<2:0> as
                     ;outputs
```

16.2.2 DIRECTION CONTROL

The TRISx register (Register 16-2) controls the PORTx pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISx register are maintained set when using them as analog inputs. I/O pins configured as analog inputs always read '0'.

16.2.3 ANALOG CONTROL

The ANSELx register (Register 16-4) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELx bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

The state of the ANSELx bits has no effect on digital output functions. A pin with TRIS clear and ANSEL set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing read-modify-write instructions on the affected port.

Note:	The ANSELx bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be initialized to '0' by user software.

16.2.4 OPEN-DRAIN CONTROL

The ODCONx register (Register 16-6) controls the open-drain feature of the port. Open-drain operation is independently selected for each pin. When an ODCONx bit is set, the corresponding port output becomes an open-drain driver capable of sinking current only. When an ODCONx bit is cleared, the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

Note:	It is necessary to set open-drain control when using the pin for I^2C.

16.2.5 SLEW RATE CONTROL

The SLRCONx register (Register 16-7) controls the slew rate option for each port pin. Slew rate for each port pin can be controlled independently. When an SLRCONx bit is set, the corresponding port pin drive is slew rate limited. When an SLRCONx bit is cleared, The corresponding port pin drive slews at the maximum rate possible.

22.0 TIMER2/4/6 MODULE

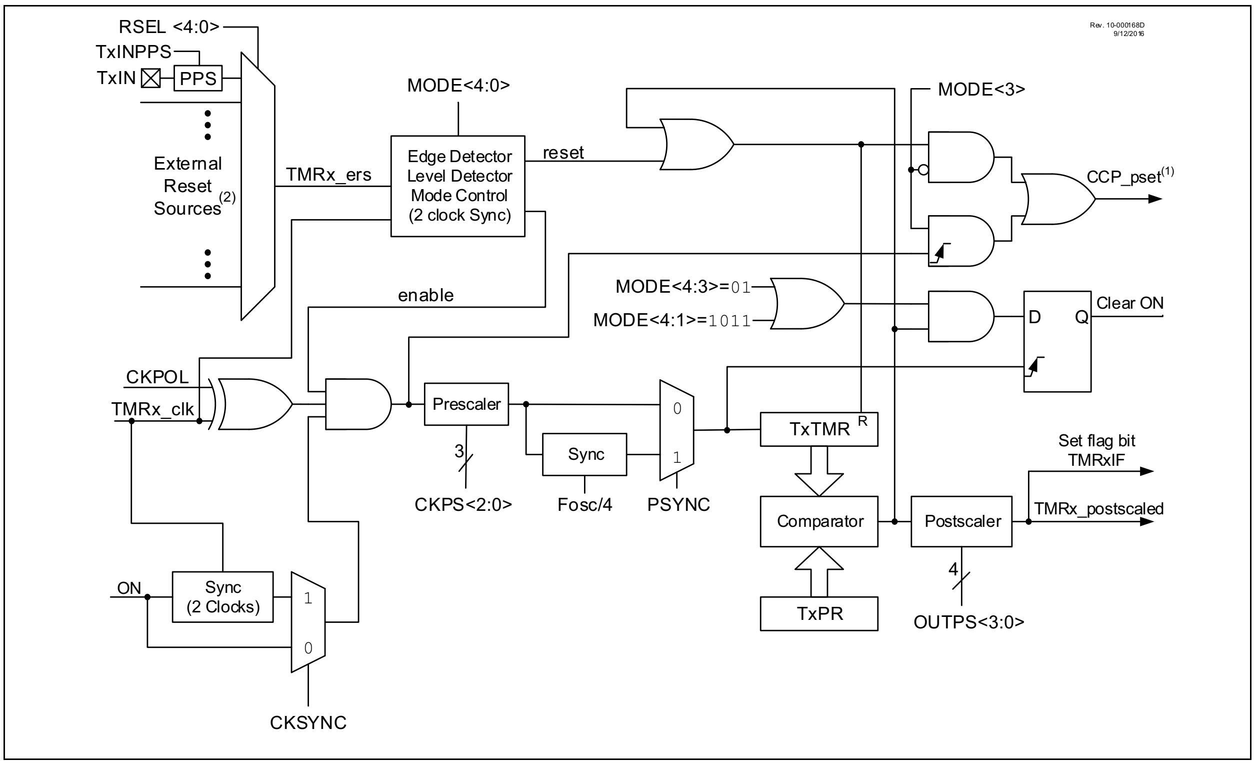
The Timer2/4/6 modules are 8-bit timers that can operate as free-running period counters or in conjunction with external signals that control start, run, freeze, and reset operation in One-Shot and Monostable modes of operation. Sophisticated waveform control such as pulse density modulation are possible by combining the operation of these timers with other internal peripherals such as the comparators and CCP modules. Features of the timer include:

- 8-bit timer register
- 8-bit period register
- Selectable external hardware timer resets
- Programmable prescaler (1:1 to 1:128)
- Programmable postscaler (1:1 to 1:16)
- Selectable synchronous/asynchronous operation
- Alternate clock sources
- Interrupt on period
- Three modes of operation:
  - Free Running Period
  - One-Shot
  - Monostable

See Figure 22-1 for a block diagram of Timer2. See Figure 22-2 for the clock source block diagram.

Note:	Three identical Timer2 modules are implemented on this device. The timers are named Timer2, Timer4, and Timer6. All references to Timer2 apply as well to Timer4 and Timer6. All references to T2PR apply as well to T4PR and T6PR.

FIGURE 22-1: TIMER2 BLOCK DIAGRAM



23.2 Capture Mode

Capture mode makes use of the 16-bit Timer1 resource. When an event occurs on the capture source, the 16-bit CCPRxH:CCPRxL register pair captures and stores the 16-bit value of the TMRxH:TMRxL register pair, respectively. An event is defined as one of the following and is configured by the MODE<3:0> bits of the CCPxCON register:

- Every falling edge of CCPx input
- Every rising edge of CCPx input
- Every 4th rising edge of CCPx input
- Every 16th rising edge of CCPx input
- Every edge of CCPx input (rising or falling)

When a capture is made, the Interrupt Request Flag bit CCPxIF of the respective PIR register is set. The interrupt flag must be cleared in software. If another capture occurs before the value in the CCPRxH:CCPRxL register pair is read, the old captured value is overwritten by the new captured value.

Note:	If an event occurs during a 2-byte read, the high and low-byte data will be from different events. It is recommended while reading the CCPRxH:CCPRxL register pair to either disable the module or read the register pair twice for data integrity.

Figure 23-1 shows a simplified diagram of the capture operation.

23.2.1 CAPTURE SOURCES

In Capture mode, the CCPx pin should be configured as an input by setting the associated TRIS control bit.

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

The capture source is selected by configuring the CTS<2:0> bits of the CCPxCAP register. Refer to CCPxCAP register (Register 23-3) for a list of sources that can be selected.

23.2.2 TIMER1 MODE RESOURCE

Timer1 must be running in Timer mode or Synchronized Counter mode for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

- See **Section 21.0 "Timer1/3/5 Module with Gate Control"** for more information on configuring Timer1.

Note:	Clocking Timer1 from the system clock (FOSC) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (FOSC/4) or from an external clock source.

31.12 Flow Control

This section does not apply to the LIN, DALI, or DMX modes.

Flow control is the means by which a sending UART data stream can be suspended by a receiving UART. Flow control prevents input buffers from overflowing without software intervention. The UART supports both hardware and XON/XOFF methods of flow control.

The flow control method is selected with the FLO<1:0> bits in the UxCON2 register. Flow control is disabled when are both bits are cleared.

31.12.1 HARDWARE FLOW CONTROL

Hardware flow control is selected by setting the FLO<1:0> bits to '10'.

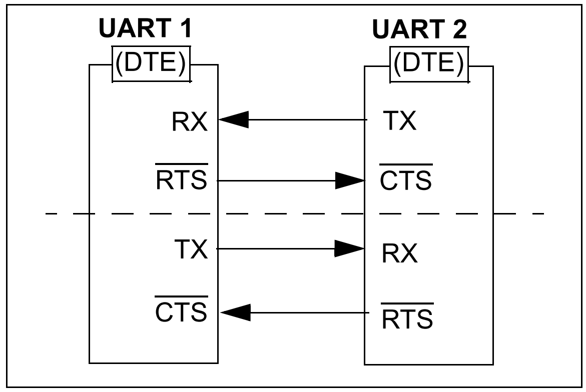
Hardware flow control consists of three lines. The RS-232 signal names for two of these are $\overline{\text{RTS}}$, and $\overline{\text{CTS}}$. Both are low true. The third line may be used to control an RS-485 transceiver. The signal name for this is TXDE for transmit drive enable. This output is high when the TX output is actively sending a character and low at all other times. The UART is configured as DTE (computer) equipment which means $\overline{\text{RTS}}$ is an output and $\overline{\text{CTS}}$ is an input.

The $\overline{\text{RTS}}$ and $\overline{\text{CTS}}$ signals work as a pair to control the transmission flow. A DTE-to-DTE configuration connects the $\overline{\text{RTS}}$ output of the receiving UART to the $\overline{\text{CTS}}$ input of the sending UART. Refer to Figure 31-10.

The UART receiving data asserts the $\overline{\text{RTS}}$ output low when the input FIFO is empty. When a character is received, the $\overline{\text{RTS}}$ output goes high until the UxRXB is read to free up both FIFO locations.

When the $\overline{\text{CTS}}$ input goes high after a byte has started to transmit, the transmission will complete normally. The receiver accommodates this by accepting the character in the second FIFO location even when the $\overline{\text{CTS}}$ input is high.

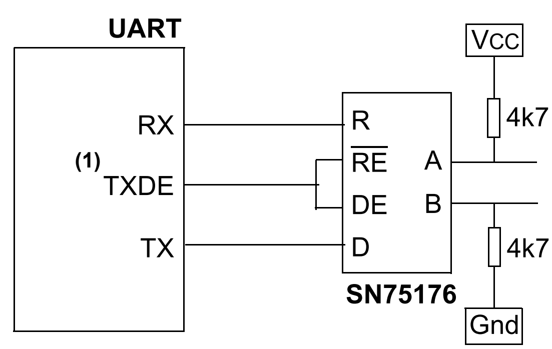
FIGURE 31-10: FLOW CONTROL



31.12.2 RS-485 TRANSCEIVER CONTROL

Hardware flow control can be used to control the direction of an RS-485 transceiver as shown in Figure 31-11. Configure the CTS input to be always enabled by setting the UxCTSPPS selection to an unimplemented port pin such as RD0. When the signal and control lines are configured as shown in Figure 31-11, then the UART will not receive its own transmissions. To verify that there are no collisions on the RS-485 lines then the transceiver RE control can be disconnected from TXDE and tied low thereby enabling loop-back reception of all transmissions. See **Section 31.14 "Collision Detection"** for more information.

FIGURE 31-11: RS-485 CONFIGURATION



Note 1: Configure UxCTSPPS to an unimplemented input such as RD0 (UxCTSPPS = 0x18).

32.4 Transfer Counter

In all master modes, the transfer counter can be used to determine how many data transfers the SPI will send/receive. The transfer counter is comprised of the SPIxTCTH/L set of registers, and is also partially controlled by the SPIxTWIDTH register. The Transfer Counter has two primary modes, determined by the BMODE bit of the SPIxCON0 register. Each mode uses the SPIxTCTH/L and SPIxTWIDTH registers to determine the number and size of the transfers. In both modes, when the transfer counter reaches zero, the TCZIF interrupt flag is set.

> **Note:** When BMODE=`1` in all master modes (and at all times in slave modes), the Transfer Counter will still decrement as transfers occur and can be used to count the number of messages sent/received, as well as to control SS(out) and to trigger TCZIF. Also when BMODE = `1`, the SPIxTWIDTH register can be used in Master and Slave modes to determine the size of messages sent and received by the SPI, even if the Transfer Counter is not being actively used to control the number of messages being sent/received by the SPI module.

32.4.1 TOTAL BIT COUNT MODE (BMODE = `0`)

In this mode, SPIxTCTH/L and SPIxTWIDTH are concatenated to determine the total number of bits to be transferred. These bits will be loaded from/into the transmit/receive FIFOs in 8-bit increments and the transfer counter will be decremented by eight until the total number of remaining bits is less than eight. If there are any remaining bits (SPIxTWIDTH ≠ `0`), the transmit FIFO will send out one final message with any extra bits greater than the remainder ignored. The SPIxTWIDTH is the remaining bit count but the value does not change as it does for the SPIxTCT value. Similarly, the receiver will load a final byte into the receiver FIFO, and pad the extra bits with zeros. The LSBF bit of SPIxCON0 determines whether the Most Significant or Least Significant bits of this final byte are ignored/padded. For example, when LSBF = `0` and the final transfer contains only two bits then if the last byte sent was 5Fh then the RXB of the receiver will contain 40h which are the two MSbits of the final byte padded with zeros in the LSbits.

In this mode, the SPI master will only transmit messages when the SPIxTCT value is greater than zero, regardless of TXR and RXR settings. In Master Transmit mode, the transfer starts with the data write to the SPIxTXB register or the count value written to the SPIxTCTL register, which ever occurs last. In Master Receive-only mode, the transfer clocks start when the SPIxTCTL value is written. Transfer clocks are suspended when the receive FIFO is full and resume as the FIFO is read.

32.4.2 VARIABLE TRANSFER SIZE MODE (BMODE = `1`)

In this mode, SPIxTWIDTH specifies the width of every individual piece of the data transfer in bits. SPIxTCTH/SPIxTCTL specifies the number of transfers of this bit length. If SPIxTWIDTH = `0`, each piece is a full byte of data. If SPIxTWIDTH ≠ `0`, then only the specified number of bits from the transmit FIFO are shifted out, with the unused bits ignored. Received data is padded with zeros in the unused bit areas when transfered into the receive FIFO. The LSBF bit of SPIxCON0 determines whether the Most Significant or Least Significant bits of the transfers are ignored/padded. In this mode, the transfer counter being zero only stops messages from being sent/received when in “Receive only” mode.

> **Note:** With BMODE = `1`, it is possible for the transfer counter (SPIxTCTH/L) to decrement below zero, although when in “Receive only” Master mode, transfer clocks will cease when the transfer counter reaches zero.

32.5.6 MASTER MODE SPI CLOCK CONFIGURATION

32.5.6.1 SPI Clock Selection

The clock source for SPI master modes is selected by the SPIxCLK register. Selections include the following:

- FOSC
- HFINTOSC
- CLKREF
- Timer0_overflow
- Timer2_Postscaled
- Timer4_Postscaled
- Timer6_Postscaled
- SMT_match

The SPIxBAUD register allows for dividing this clock. The frequency of the SCK output is defined by Equation 32-1:

EQUATION 32-1: FREQUENCY OF SCK OUTPUT SIGNAL

$$F_{BAUD} = \frac{F_{CSEL}}{(2 \cdot (BAUD + 1))}$$

where FBAUD is the baud rate frequency output on the SCK pin, FCSEL is the frequency of the input clock selected by the SPIxCLK register, and BAUD is the value contained in the SPIxBAUD register.

32.5.6.2 CKE, CKP and SMP

The CKP, CKE, and SMP bits control the relationship between the SCK clock output, SDO output data changes, and SDI input data sampling. The bit functions are as follows:

- CKP - SCK output polarity
- CKE - SDO output change relative to the SCK clock
- SMP - SDI input sampling relative to the clock edges

The CKE bit, when set, inverts the low Idle state of the SCK output to a high Idle state.

Figure 32-7 through Figure 32-10 illustrate the eight possible combinations of the CKP, CKE, and SMP bit selections.

When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. When the CKE bit is cleared, the SDO data is undefined prior to the first SCK edge.

Note:	All timing diagrams assume the LSBF bit of SPIxCON0 is cleared.

FIGURE 32-7: CLOCKING DETAIL-MASTER MODE, CKE/SMP = 0/0

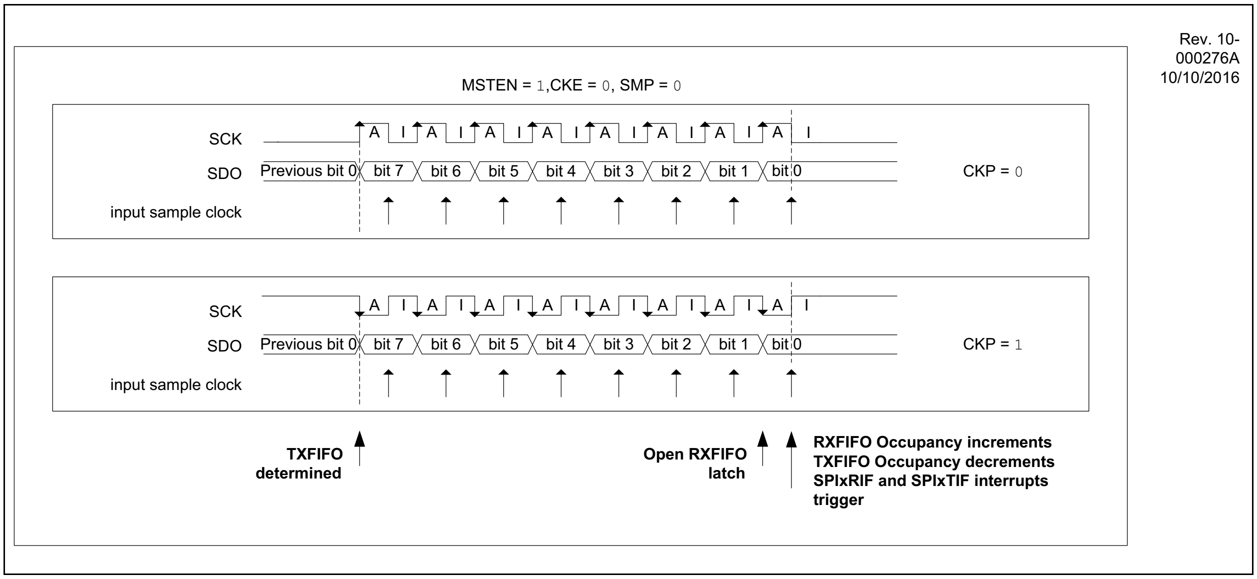
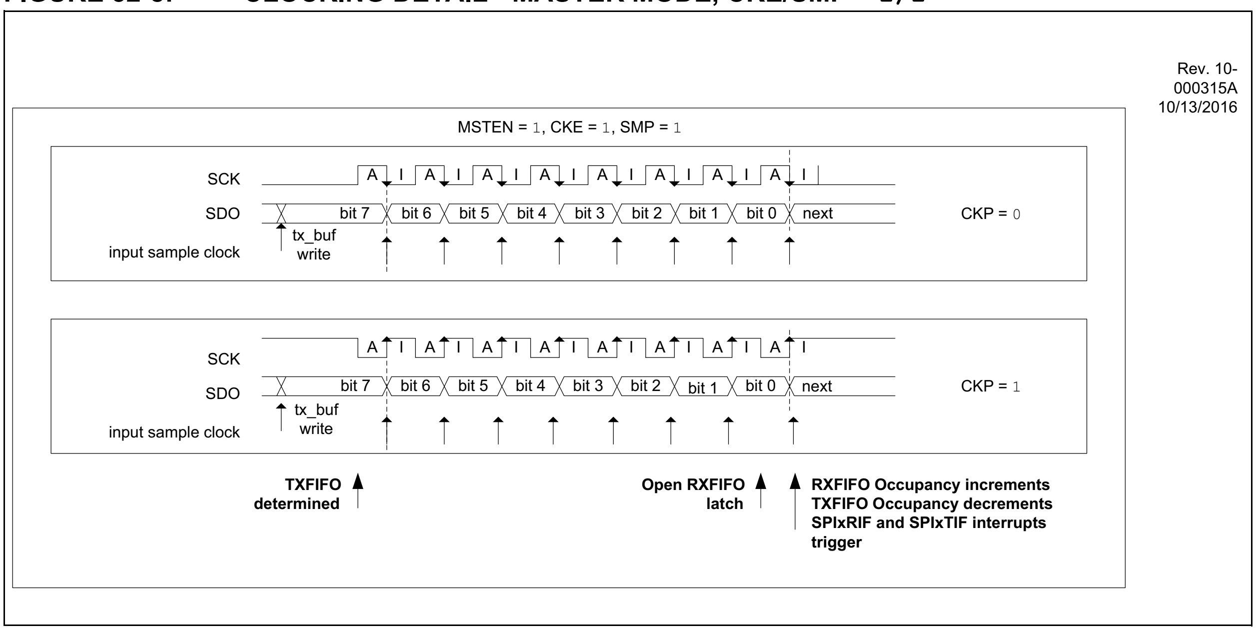


FIGURE 32-8: CLOCKING DETAIL - MASTER MODE, CKE/SMP = 1/1



32.8.3.1 Shift Register Empty Interrupt

The Shift Register Empty interrupt flag and enable are the SRMTIF and SRMTIE bits respectively. This interrupt is only available in master mode and triggers when a data transfer completes and conditions are not present to start a new transfer, as dictated by the TXR and RXR bits (see Table 32-1 for conditions for starting a new Master mode data transfer with different TXR/RXR settings). This interrupt will be triggered at the end of the last full bit period, after SCK has been low for one ½-baud period. See Figure 32-14 for more details of the timing of this interrupt as well as other interrupts. This bit will not clear itself when the conditions for starting a new transfer occur, and must be cleared in software.

32.8.3.2 Transfer Counter is Zero Interrupt

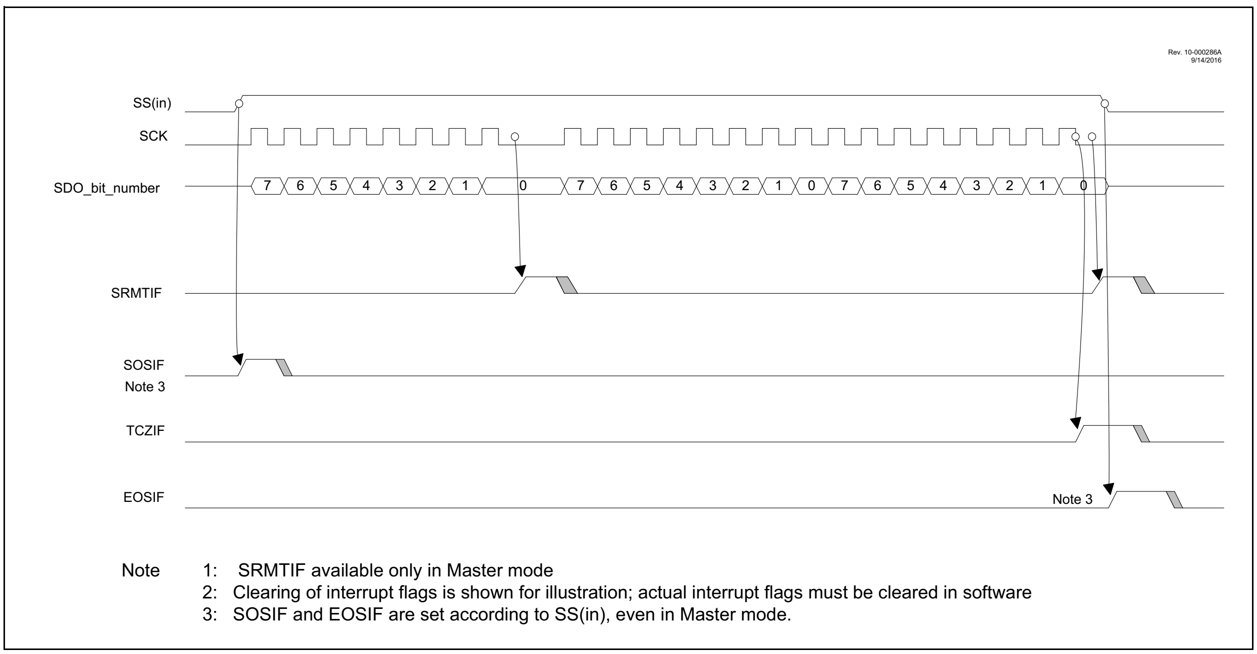
The Transfer Counter is zero interrupt flag and enable are the TCZIF and TCZIE bits, respectively. This interrupt will trigger when the transfer counter (defined by BMODE, SPIxTCTH/L and SPIxTWIDTH) decrements from one to zero. See Figure 32-14 for more details on the timing of this interrupt as well as other interrupts. This bit must be cleared in software.

Note:	The TCZIF flag only indicates that the transfer counter has decremented from one to zero, and may not indicate that the entire data transfer process is complete. Either poll the BUSY bit of SPIxCON2 and wait for it to be cleared or use the Shift Register Empty Interrupt (SRMTIF) to determine if a data transfer is fully complete.

32.8.3.3 Start of Slave Select and End of Slave Select Interrupts

The start of slave select interrupt flag and enable are the SOSIF and SOSIE bits, respectively, and the end of slave select interrupt flag and enable are similarly designated by the EOSIF and EOSIE bits. These interrupts trigger at the leading and trailing edges of the slave select input. Note that the interrupts are active in both master and slave mode, and will trigger on transitions of the slave select input regardless of which mode the SPI is in. In Master mode, PPS should be used to route the slave select input to the same pin as the slave select output, allowing these interrupts to trigger on changes to the slave select output. Also note that in slave mode, changing the SSET bit can trigger these interrupts, as it changes the effective input value of slave select. Both SOSIF and EOSIF must be cleared in software

FIGURE 32-14: TRANSFER AND SLAVE SELECT INTERRUPT TIMINGS



34.0 FIXED VOLTAGE REFERENCE (FVR)

The Fixed Voltage Reference, or FVR, is a stable voltage reference, independent of VDD, with 1.024V, 2.048V or 4.096V selectable output levels. The output of the FVR can be configured to supply a reference voltage to the following:

- ADC input channel
- ADC positive reference
- Comparator input
- Digital-to-Analog Converter (DAC)

The FVR can be enabled by setting the EN bit of the FVRCON register.

Note:	Fixed Voltage Reference output cannot exceed VDD.

34.1 Independent Gain Amplifiers

The output of the FVR, which is connected to the ADC, Comparators, and DAC, is routed through two independent programmable gain amplifiers. Each amplifier can be programmed for a gain of 1x, 2x or 4x, to produce the three possible voltage levels.

The ADFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the ADC module. Reference **Section 36.0 "Analog-to-Digital Converter with Computation (ADC2) Module"** for additional information.

The CDAFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the DAC and comparator module. Reference **Section 37.0 "5-Bit Digital-to-Analog Converter (DAC) Module"** and **Section 38.0 "Comparator Module"** for additional information.

34.2 FVR Stabilization Period

When the Fixed Voltage Reference module is enabled, it requires time for the reference and amplifier circuits to stabilize. Once the circuits stabilize and are ready for use, the RDY bit of the FVRCON register will be set.

FIGURE 34-1: VOLTAGE REFERENCE BLOCK DIAGRAM

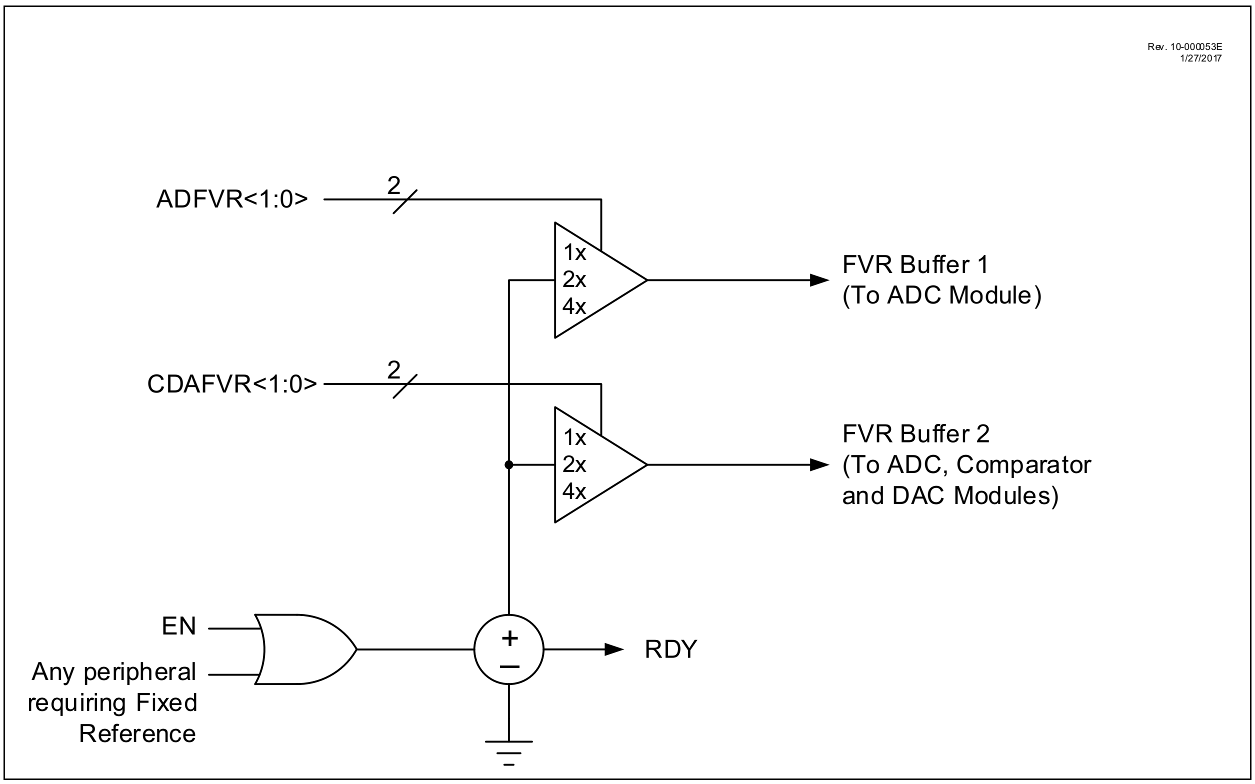
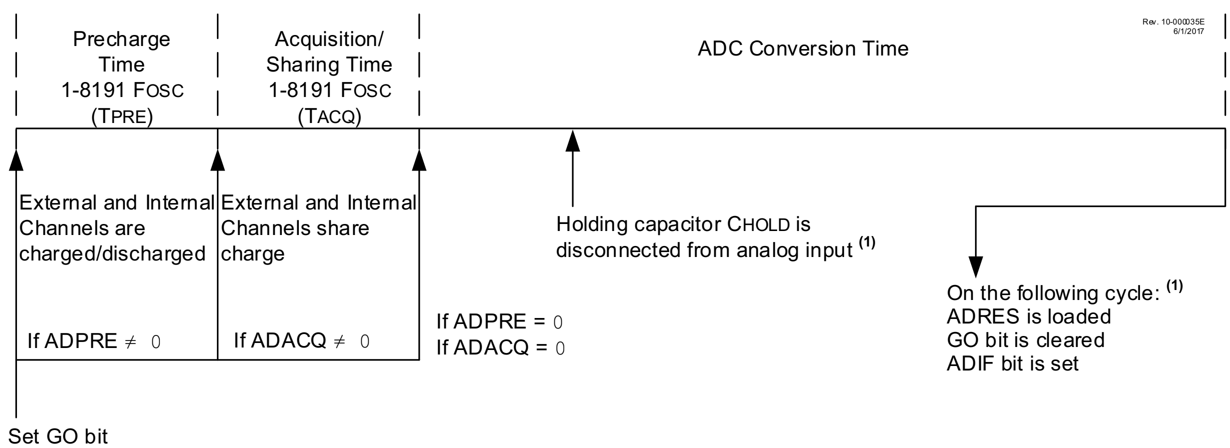


TABLE 36-1: ADC CLOCK PERIOD (TAD) Vs. DEVICE OPERATING FREQUENCIES(1,4)

ADC Clock Period (TAD)		Device Frequency (FOSC)						
ADC Clock Source	**CS<5:0>**	**64 MHz**	**32 MHz**	**20 MHz**	**16 MHz**	**8 MHz**	**4 MHz**	**1 MHz**
FOSC/2	000000	31.25 ns(2)	62.5 ns(2)	100 ns(2)	125 ns(2)	250 ns(2)	500 ns(2)	2.0 μs
FOSC/4	000001	62.5 ns(2)	125 ns(2)	200 ns(2)	250 ns(2)	500 ns(2)	1.0 μs	4.0 μs
FOSC/6	000010	125 ns(2)	187.5 ns(2)	300 ns(2)	375 ns(2)	750 ns(2)	1.5 μs	6.0 μs
FOSC/8	000011	187.5 ns(2)	250 ns(2)	400 ns(2)	500 ns(2)	1.0 μs	2.0 μs	8.0 μs
...	...	...	...	...	...	...	...	...
FOSC/16	000111	250 ns(2)	500 ns(2)	800 ns(2)	1.0 μs	2.0 μs	4.0 μs	16.0 μs(3)
...	...	...	...	...	...	...	...	...
FOSC/128	111111	2.0 μs	4.0 μs	6.4 μs	8.0 μs	16.0 μs(3)	32.0 μs(2)	128.0 μs(2)
FRC	CS(ADCON0<4>) = 1	1.0-6.0 μs	1.0-6.0 μs	1.0-6.0 μs	1.0-6.0 μs	1.0-6.0 μs	1.0-6.0 μs	1.0-6.0 μs

Legend: Shaded cells are outside of recommended range.
Note 1: See TAD parameter for FRC source typical TAD value.
2: These values violate the required TAD time.
3: Outside the recommended TAD time.
4: The ADC clock period (TAD) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock FOSC. However, the FRC oscillator source must be used when conversions are to be performed with the device in Sleep mode.

FIGURE 36-2: ANALOG-TO-DIGITAL CONVERSION CYCLES



Note 1: Refer to Table 44-16 for ADC Conversion Timing Specifications.

BTFSC — Bit Test File, Skip if Clear

Syntax: BTFSC f, b {,a}

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

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 2-cycle instruction.
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 41.2.3 "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: $0 \leq f \leq 255$
$0 \leq b < 7$
$a \in [0,1]$

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 2-cycle instruction.
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 41.2.3 "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)`