Section 18. Serial Peripheral Interface (SPI)

HIGHLIGHTS

This section of the manual contains the following major topics:

Note: This family reference manual section is meant to serve as a complement to device data sheet. Depending on the device variant, this manual section may not apply to all dsPIC33E/PIC24E devices.

Please consult the note at the beginning of the **"Serial Peripheral Interface (SPI)"** chapter in the current device data sheet to check whether this document supports the device you are using.

Device data sheets and family reference manual sections are available for download from the Microchip Worldwide Web site at: http://www.microchip.com

18.1 INTRODUCTION

The Serial Peripheral Interface (SPI) module is a synchronous serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices can be serial EEPROMs, shift registers, display drivers, A/D converters, and so on. The SPI module is compatible with Motorola's SPI and SIOP interfaces.

Depending on the device variant, the dsPIC33E/PIC24E device family offers two or four SPI modules on a single device. These modules, which are designated as SPI1, SPI2, SPI3 and SPI4, are functionally identical. The SPI1 and SPI2 modules are available on all dsPIC33E/PIC24E devices, while the SPI3 and SPI4 module are available in many of the higher pin count packages. Each SPI module includes an eight-word FIFO buffer and allows DMA bus connections. When using the SPI module with DMA, FIFO operation can be disabled.

Note: In this section, the SPI modules are referred together as SPIx, or separately as SPI1, SPI2, SPI3 and SPI4. Special Function Registers (SFRs) follow a similar notation. For example, SPIxCON refers to the control register for the SPI1, SPI2, SPI3 or SPI4 module.

The SPIx serial interface consists of four pins, as follows:

- SDIx: Serial Data Input
- SDOx: Serial Data Output
- SCKx: Shift Clock Input or Output
- $\overline{SS}$x/FSYNCx: Active-Low Slave Select or Frame Synchronization I/O Pulse

The SPIx module can be configured to operate with two, three or four pins. In 2-pin mode, neither the SDOx nor the $\overline{SS}$x pin is used. In 3-pin mode, the $\overline{SS}$x pin is not used.

Figure 18-1 and Figure 18-2 illustrate the block diagrams of the SPI module in Standard and Enhanced mode.

Figure 18-1: SPIx Module Block Diagram (Standard Mode)

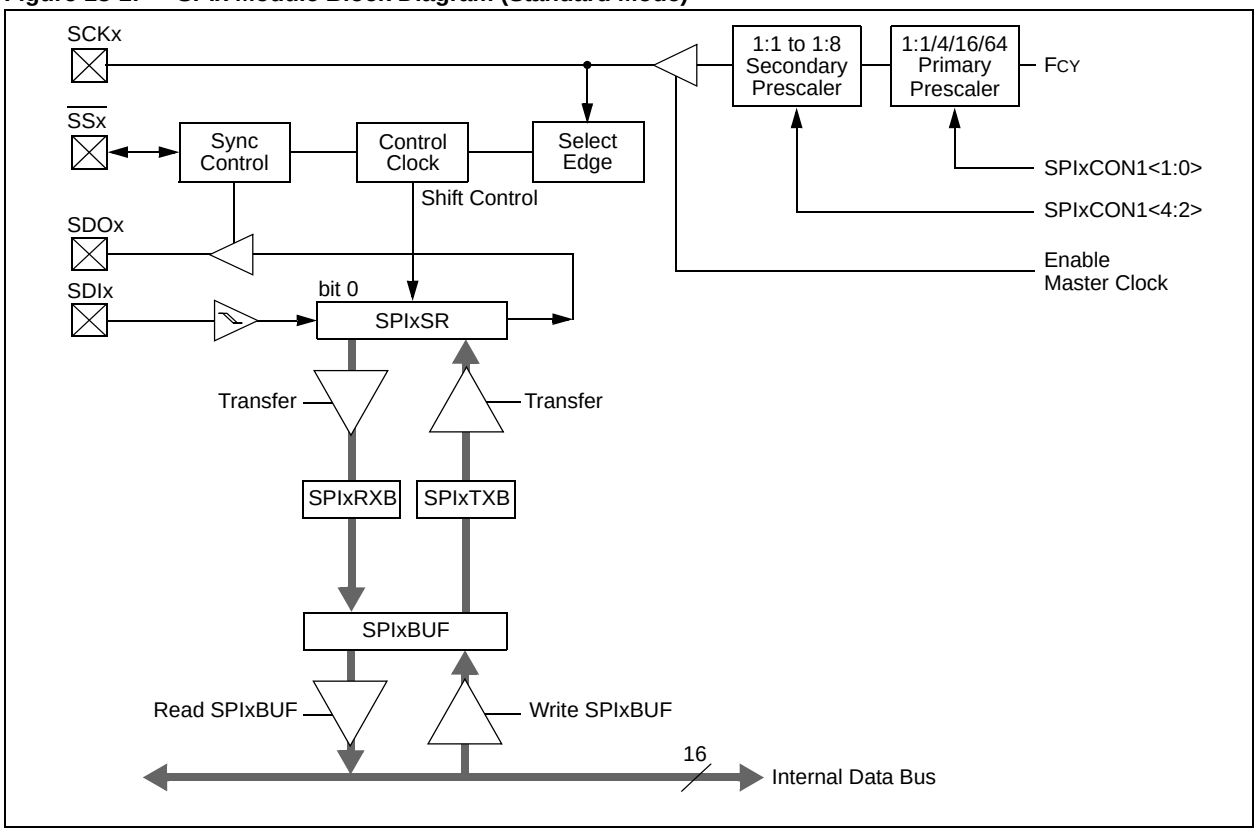
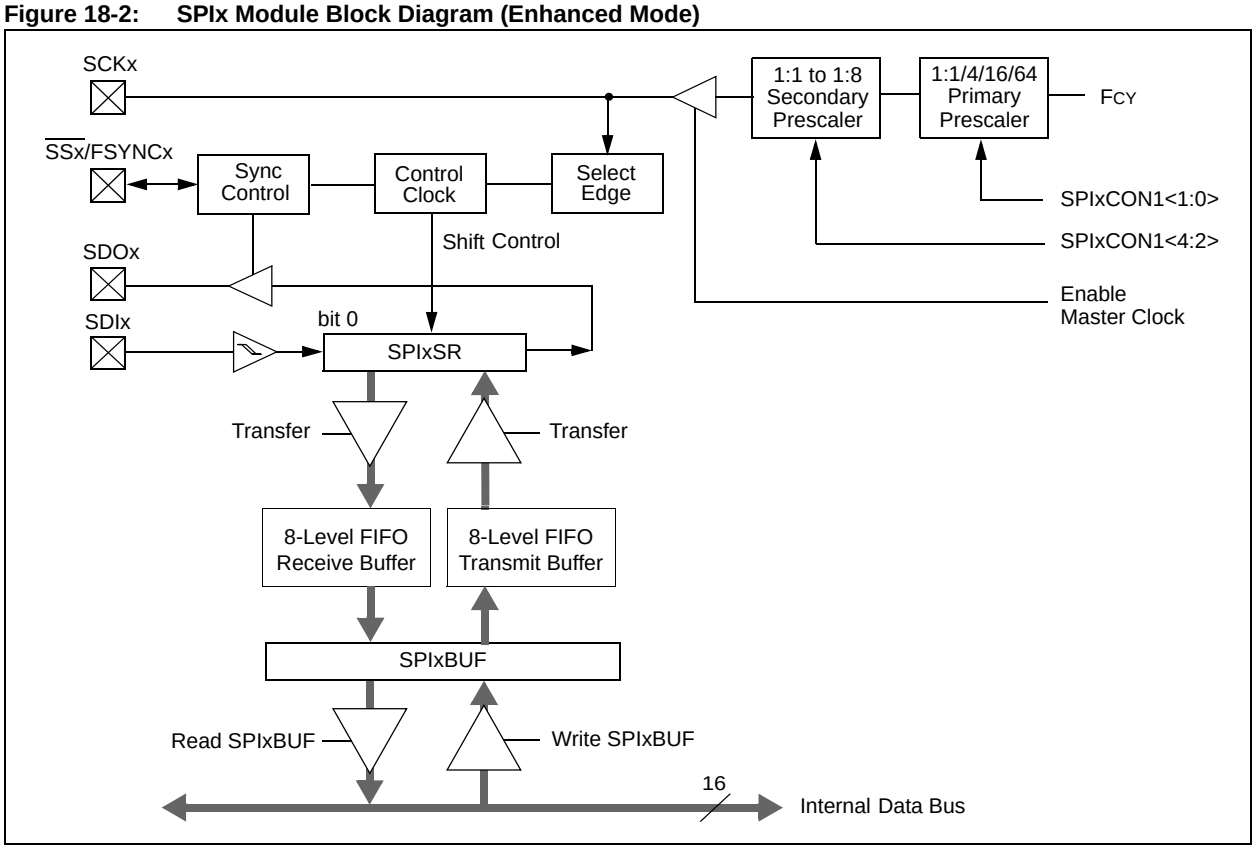


Figure 18-2: SPIx Module Block Diagram (Enhanced Mode)



18.2 SPI REGISTERS

The SPI module has the following Control and Status registers:

- **SPIxSTAT: SPIx Status and Control Register**

  The SPIx Status and Control register (SPIxSTAT) indicates the various status conditions, such as receive overflow, transmit buffer full and receive buffer full. This register specifies the operation of the module during Idle mode. It also contains a bit that can enable or disable the module.

- **SPIxCON1: SPIx Control Register 1**

  The SPIx Control Register 1 (SPIxCON1) specifies the clock prescaler, Master/Slave mode, Word/Byte communication, clock polarity and clock/data pin operation.

- **SPIxCON2: SPIx Control Register 2**

  The SPIx Control Register 2 (SPIxCON2) enables/disables the Framed SPI operation. This register also specifies the frame synchronization pulse direction, polarity and edge selection.

- **SPIxBUF: SPIx Data Receive/Transmit Buffer Register**

  In Standard mode, the SPIx Data Receive/Transmit Buffer register (SPIxBUF) consists of two separate internal registers: the Transmit Buffer register (SPIxTXB) and the Receive Buffer register (SPIxRXB). These two unidirectional, 16-bit registers share the SFR address of the SPIxBUF register. If the user application writes data to be transmitted to the SPIxBUF register, internally the data is written to the SPIxTXB register. Similarly, when the user application reads the received data from the SPIxBUF register, internally the data is read from the SPIxRXB register.

  When the enhanced buffer is enabled, the SPIxBUF register becomes the data interface to two 8-level FIFOs: one for reception and another for transmission. Each buffer can hold up to eight pending data transfers. When the CPU writes data to the SPIxBUF register, the data is moved into the next transmit buffer location. The SPIx peripheral begins to transfer data after the first CPU write to the SPIxBUF register, and continues until all pending transfers are completed. After each transfer, the SPIx updates the next receive buffer location with the received data and is available for the CPU to read. After the CPU read, data is read from the next receive buffer location.

The double-buffers transmit/receive operation allows continuous data transfer in the background, and the transmission and reception occur simultaneously. In addition, there is an internal 16-bit shift register (SPIxSR) that is not memory mapped. It shifts data in and out of the SPI port.

Register 18-1: SPIxSTAT: SPIx Status and Control Register

R/W-0	U-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
SPIEN	—	SPISIDL	—	—	SPIBEC<2:0>		
bit 15							bit 8

R/W-0	R/C-0, HS	R/W-0	R/W-0	R/W-0	R/W-0	R-0, HS, HC	R-0, HS, HC
SRMPT	SPIROV	SRXMPT	SISEL<2:0>			SPITBF	SPIRBF
bit 7							bit 0

Legend:	C = Clearable bit	x = Bit is unknown
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared
HS = Set in Hardware bit	HC = Cleared in Hardware bit	

bit 15 **SPIEN:** SPIx Enable bit
1 = Enables the module and configures SCKx, SDOx, SDIx and $\overline{SSx}$ as serial port pins
0 = Disables the module

bit 14 **Unimplemented:** Read as '0'

bit 13 **SPISIDL:** Stop in Idle Mode bit
1 = Discontinue the module operation when device enters Idle mode
0 = Continue the module operation in Idle mode

bit 12-11 **Unimplemented:** Read as '0'

bit 10-8 **SPIBEC<2:0>:** SPIx Buffer Element Count bits (valid in Enhanced Buffer mode)
Master mode:
Number of SPIx transfers are pending.
Slave mode:
Number of SPIx transfers are unread.

bit 7 **SRMPT:** Shift Register (SPIxSR) Empty bit (valid in Enhanced Buffer mode)
1 = SPIx Shift register is empty and ready to send or receive the data
0 = SPIx Shift register is not empty

bit 6 **SPIROV:** Receive Overflow Flag bit
1 = A new byte/word is completely received and discarded. The user application has not read the previous data in the SPIxBUF register
0 = No overflow has occurred

bit 5 **SRXMPT:** Receive FIFO Empty bit (valid in Enhanced Buffer mode)
1 = RX FIFO is empty
0 = RX FIFO is not empty

bit 4-2 **SISEL<2:0>:** SPIx Buffer Interrupt Mode bits (valid in Enhanced Buffer mode)
111 = Interrupt when the SPIx transmit buffer is full (SPIxTBF bit is set)
110 = Interrupt when last bit is shifted into SPIxSR, and as a result, the TX FIFO is empty
101 = Interrupt when the last bit is shifted out of SPIxSR, and the transmit is complete
100 = Interrupt when one data is shifted into the SPIxSR, and as a result, the TX FIFO has one open memory location
011 = Interrupt when the SPIx receive buffer is full (SPIxRBF bit set)
010 = Interrupt when the SPIx receive buffer is 3/4 or more full
001 = Interrupt when data is received in the receive buffer (SRMPT bit is set)
000 = Interrupt when the last data in the receive buffer is read, as a result, the buffer is empty (SRXMPT bit set)

Register 18-1: SPIxSTAT: SPIx Status and Control Register (Continued)

bit 1 **SPITBF:** SPIx Transmit Buffer Full Status bit

1 = Transmit not yet started, SPIxTXB is full
0 = Transmit started, SPIxTXB is empty

Standard Buffer mode:
Automatically set in hardware when core writes to the SPIxBUF location, loading SPIxTXB. Automatically cleared in hardware when SPIx module transfers data from SPIxTXB to SPIxSR.

Enhanced Buffer mode:
Automatically set in hardware when CPU writes to the SPIxBUF location, loading the last available buffer location. Automatically cleared in hardware when a buffer location is available for a CPU write operation.

bit 0 **SPIRBF:** SPIx Receive Buffer Full Status bit

1 = Receive complete, SPIxRXB is full
0 = Receive is incomplete, SPIxRXB is empty

Standard Buffer mode:
Automatically set in hardware when SPIx transfers data from SPIxSR to SPIxRXB. Automatically cleared in hardware when core reads the SPIxBUF location, reading SPIxRXB.

Enhanced Buffer mode:
Automatically set in hardware when SPIx transfers data from SPIxSR to the buffer, filling the last unread buffer location. Automatically cleared in hardware when a buffer location is available for a transfer from SPIxSR.

Register 18-2: SPIxCON1: SPIx Control Register 1

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	DISSCK	DISSDO	MODE16	SMP	CKE(1)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SSEN	CKP	MSTEN	SPRE<2:0>			PPRE<1:0>	
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 15-13 **Unimplemented:** Read as '0'

bit 12 **DISSCK:** Disable SCKx Pin bit (SPI Master modes only)
1 = Internal SPI clock is disabled; pin functions as I/O
0 = Internal SPI clock is enabled

bit 11 **DISSDO:** Disable SDOx Pin bit
1 = SDOx pin is not used by the module; pin functions as I/O
0 = SDOx pin is controlled by the module

bit 10 **MODE16:** Word/Byte Communication Select bit
1 = Communication is word-wide (16 bits)
0 = Communication is byte-wide (8 bits)

bit 9 **SMP:** SPIx Data Input Sample Phase bit
Master mode:
1 = Input data is sampled at end of data output time
0 = Input data is sampled at middle of data output time
Slave mode:
The SMPbit must be cleared when the SPIx module is used in Slave mode.

bit 8 **CKE:** SPIx Clock Edge Select bit(1)
1 = Serial output data changes on transition from active clock state to Idle clock state (refer to bit 6)
0 = Serial output data changes on transition from Idle clock state to active clock state (refer to bit 6)

bit 7 **SSEN:** Slave Select Enable bit (Slave mode)
1 = $\overline{SSx}$ pin is used for Slave mode
0 = SSx pin is not used by the module. Pin is controlled by port function.

bit 6 **CKP:** Clock Polarity Select bit
1 = Idle state for clock is a high level; active state is a low level
0 = Idle state for clock is a low level; active state is a high level

bit 5 **MSTEN:** Master Mode Enable bit
1 = Master mode
0 = Slave mode

bit 4-2 **SPRE<2:0>:** Secondary Prescale bits (Master mode)
111 = Reserved
110 = Secondary prescale 2:1
•
•
•
000 = Secondary prescale 8:1

bit 1-0 **PPRE<1:0>:** Primary Prescale bits (Master mode)
11 = Reserved
10 = Primary prescale 4:1
01 = Primary prescale 16:1
00 = Primary prescale 64:1

Note 1: The CKE bit is not used in Framed SPI modes. Program this bit to '0' for Framed SPI modes (FRMEN = 1).

Register 18-3: SPIxCON2: SPIx Control Register 2

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
FRMEN	SPIFSD	FRMPOL	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	FRMDLY	SPIBEN
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 15 **FRMEN:** Framed SPIx Support bit
1 = Framed SPIx support is enabled ($\overline{SS}$x pin is used as frame sync pulse input/output)
0 = Framed SPIx support is disabled

bit 14 **SPIFSD:** Frame Sync Pulse Direction Control bit
1 = Frame sync pulse input (slave)
0 = Frame sync pulse output (master)

bit 13 **FRMPOL:** Frame Sync Pulse Polarity bit
1 = Frame sync pulse is active-high
0 = Frame sync pulse is active-low

bit 12-2 **Unimplemented:** Read as '0'

bit 1 **FRMDLY:** Frame Sync Pulse Edge Select bit
1 = Frame sync pulse coincides with first bit clock
0 = Frame sync pulse precedes first bit clock

bit 0 **SPIBEN:** Enhanced Buffer Enable bit
1 = Enhanced Buffer is enabled
0 = Enhanced Buffer is disabled (Legacy mode)

Register 18-4: SPIxBUF: SPIx Data Receive/Transmit Buffer Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPIx Transmit and Receive Buffer Register							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPIx Transmit and Receive Buffer Register							
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 15-0 Transmit/Receive Buffer bits

18.3 MODES OF OPERATION

The SPI module uses these flexible operating modes:

- 8-bit and 16-bit Data Transmission/Reception
- Master and Slave Modes
- Enhanced Buffer Master and Slave Modes
- Framed SPIx Modes
- SPIx Receive-Only Operation
- SPIx Error Handling

Note 1: In Framed SPI mode, these four pins are used: SDIx, SDOx, SCKx and $\overline{SSx}$.

2: If Slave Select feature is used, then all four pins are used: SDIx, SDOx, SCKx and $\overline{SSx}$.

3: If standard SPI is used but CKE = 1, then enabling/using the Slave Select feature is mandatory, and therefore, all four pins are used: SDIx, SDOx, SCKx and $\overline{SSx}$.

4: If standard SPI is used but DISSDO = 1, then only two pins are used: SDIx and SCKx (unless slave select is also enabled).

5: In all other cases, three pins are used: SDIx, SDOx and SCKx.

18.3.1 8-bit and 16-bit Data Transmission/Reception

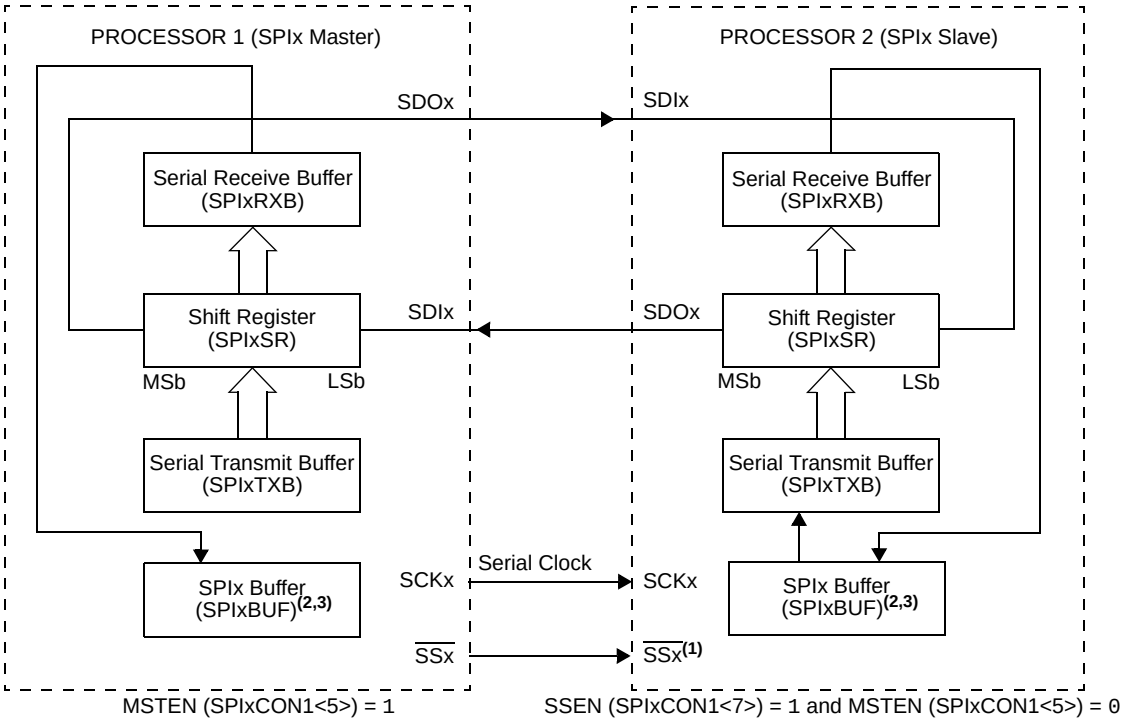
The Word/Byte Communication Select bit (MODE16) in the SPIx Control Register 1 (SPIxCON1<10>) allows the module to communicate in either 8-bit or 16-bit mode. The functionality is same for each mode except for the number of bits that are received and transmitted. In this context:

- The module is Reset when the value of the MODE16 bit is changed. Consequently, the bit should not be changed during normal operation.
- Data is transmitted out of bit 7 of the SPIx Shift Register (SPIxSR<7>) for 8-bit operation, while it is transmitted out of bit 15 (SPIxSR<15>) for 16-bit operation. In both modes, data is shifted into bit 0 (SPIxSR<0>).
- In 8-bit mode, eight clock pulses are required at the SCKx pin to shift data in/out when transmitting or receiving data. In 16-bit mode, 16 clock pulses are required at the SCKx pin.

18.3.2 Master and Slave Modes

Data can be thought of as taking a direct path between the Most Significant bit (MSb) of one module's shift register and the Least Significant bit (LSb) of the other, and then into the appropriate transmit or receive buffer. A module configured as the master module provides the serial clock and synchronization signals (as required) to the slave device. Figure 18-3 illustrates the connection of the master and slave modules.

Figure 18-3: SPI Master/Slave Connection



Note 1: The use of the $\overline{SSx}$ pin in Slave mode is optional.

2: The user application must write transmit data to read received data from the SPIxBUF register. The SPIxTXB and SPIxRXB registers are memory mapped to the SPIxBUF register.

3: If the user application does not change the data in the SPIxBUF register, every new transmission shifts the SPIxBUF value instead of shifting the value received in the shift register.

18.3.2.1 MASTER MODE

In Master mode, the system clock is prescaled and then used as the serial clock. The prescaling is based on the settings in the Primary Prescale bits (PPRE<1:0>) and the Secondary Prescale bits (SPRE<2:0>) in the SPIx Control Register 1 (SPIxCON1). The serial clock is output via the SCKx pin to slave devices. The clock pulses are generated only when there is data to be transmitted. For more information, see **18.4 "Master Mode Clock Frequency"**. The CKP (SPIxCON1<6>) and CKE (SPIxCON1<8>) bits determine the edge of the clock pulse on which data transmission occurs. Both, data to be transmitted and data received are respectively written into, or read from, the SPIxBUF register.

The SPIx module operation in Master mode is described as follows:

1. Once the module is set up in Master mode and enabled to operate, data to be transmitted is written to the SPIxBUF register. The SPIx Transmit Buffer Full Status bit (SPITBF) in the SPIx Status and Control register (SPIxSTAT<1>) is set.
2. The content of the SPIx Transmit Buffer register (SPIxTXB) is moved to the SPIx Shift register (SPIxSR), and the SPITBF bit (SPIxSTAT<1>) is cleared by the module.
3. A series of 8/16 clock pulses shift out 8/16 bits of transmit data from the SPIx Shift register (SPIxSR) to the SDOx pin and simultaneously shift the data at the SDIx pin into the SPIxSR register.
4. When the transfer is complete, the following events occur in the Interrupt controller:
   a) The appropriate interrupt flag bit is set in the Interrupt controller:
      - The SPI1IF bit is set in the Interrupt Flag Status Register 0 (IFS0<10>)
      - The SPI2IF bit is set in the Interrupt Flag Status Register 2 (IFS2<1>)
      - The SPI3IF bit is set in the Interrupt Flag Status Register 5 (IFS5<11>)
      - The SPI4IF bit is set in the Interrupt Flag Status Register 7 (IFS7<11>)

   b) These interrupts are enabled by setting the corresponding interrupt enable bits:
      - The SPI1IE bit is enabled in the Interrupt Enable Control Register 0 (IEC0<10>)
      - The SPI2IE bit is enabled in the Interrupt Enable Control Register 2 (IEC2<1>)
      - The SPI3IE bit is enabled in the Interrupt Enable Control Register 5 (IEC5<11>)
      - The SPI4IE bit is enabled in the Interrupt Enable Control Register 7 (IEC7<11>)

      The SPIxIF flags are not cleared automatically by the hardware.

   c) When the ongoing transmit and receive operations are completed, the content of the SPIx Shift register (SPIxSR) is moved to the SPIx Receive Buffer (SPIxRXB).
   d) The SPIx Receive Buffer Full Status bit (SPIRBF) in the SPIx Status and Control register (SPIxSTAT<0>) is set by the module, indicating that the receive buffer is full. Once the SPIxBUF register is read by the user application, the hardware clears the SPIRBF bit.
5. If the SPIRBF bit (SPIxSTAT<0>) is set (receive buffer is full) when the SPIx module needs to transfer data from the SPIxSR register to SPIxRXB register, the module sets the Receive Overflow Flag bit (SPIROV) in the SPIxSTAT register (SPIxSTAT<6>), indicating an overflow condition.
6. Data to be transmitted can be written to the SPIxBUF register by the user application at any time as long as the SPITBF bit (SPIxSTAT<1>) is clear. The write can occur while the SPIxSR register is transferring the previously written data, allowing continuous transmission.

Note:	The user application cannot write directly into the SPIxSR register. All writes to the SPIxSR register are performed through the SPIxBUF register.

18.3.2.1.1 Master Mode Setup Procedures

The following procedure is used to set up the SPIx module for Master mode of operation:

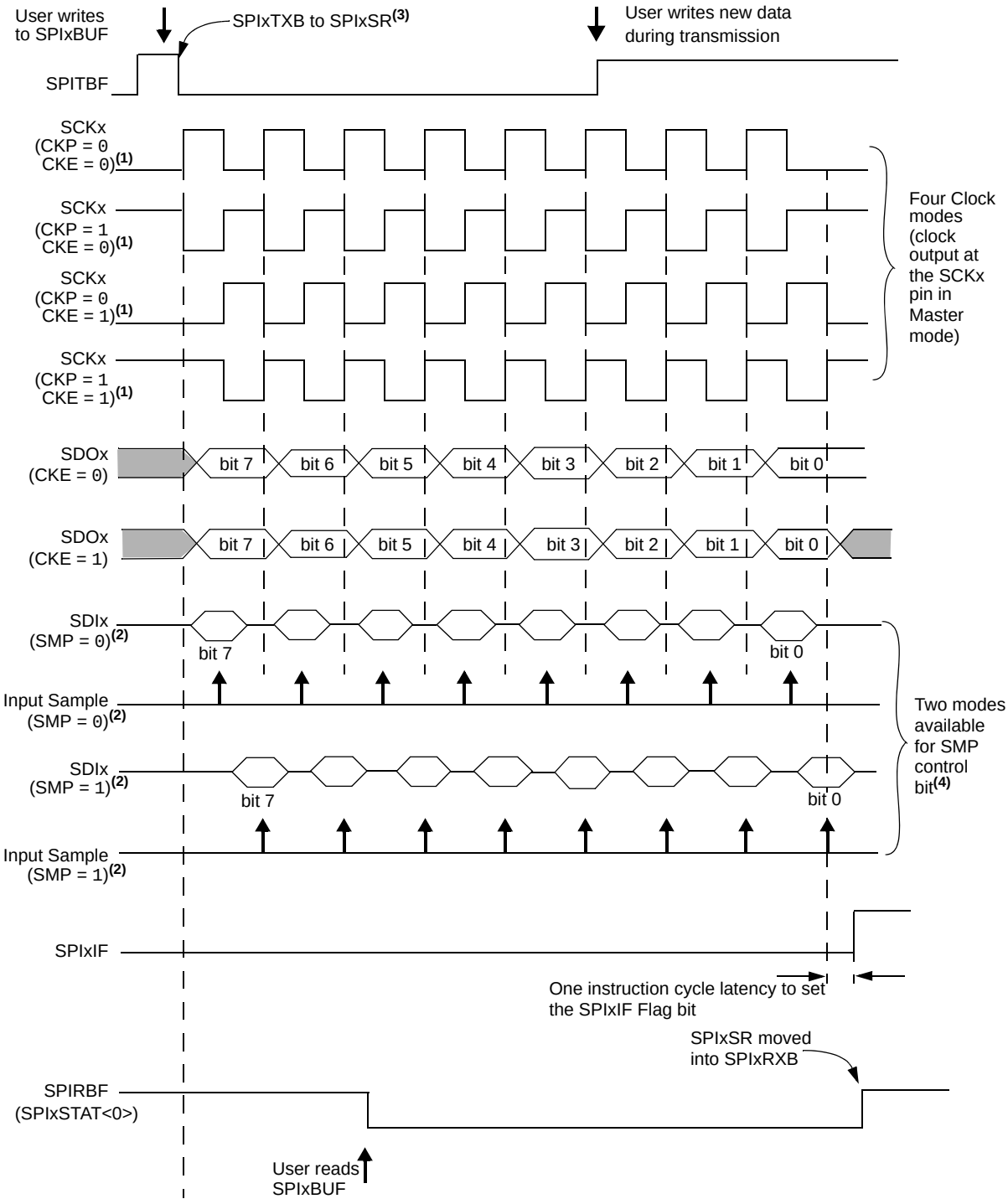
1. If using interrupts, configure the Interrupt controller:
   a) Clear the SPIx Interrupt Flag Status bit (SPIxIF) in the respective Interrupt Flag Status register (IFS0<10>, IFS2<1>, IFS5<11>, or IFS7<11>) in the Interrupt controller.
   b) Set the SPIx Event Interrupt Enable bit (SPIxIE) in the respective Interrupt Event Control register (IEC0<10>, IEC2<1>, IEC5<11> or IEC7<11>) in the Interrupt controller.
   c) Write the SPIx Event Interrupt Priority bits (SPIxIP) in the respective Interrupt Priority Control register (IPC2<10:8>, IPC8<6:4>, IPC22<14:12> or IPC30<6:4>) to set the interrupt priority.
2. Set the Master Mode Enable bit (MSTEN) in the SPIxCON1 register (SPIxCON1<5> = 1).
3. Clear the Receive Overflow Flag bit (SPIROV) in the SPIxSTAT register (SPIxSTAT<6> = 0).
4. Enable the SPIx operation by setting the SPIx Enable bit (SPIEN) in the SPIxSTAT register (SPIxSTAT<15> = 1).
5. Write the data to be transmitted to the SPIxBUF register. Transmission (and reception) starts as soon as data is written to the SPIxBUF register.

Example 18-1 illustrates the code sequence that shows the SPI register configuration for Master mode.

Example 18-1: SPI Configuration – Master Mode

```
/* The following code sequence shows SPI register configuration for Master mode */

IFS0bits.SPI1IF = 0;                // Clear the Interrupt flag
IEC0bits.SPI1IE = 0;                // Disable the interrupt

// SPI1CON1 Register Settings

SPI1CON1bits.DISSCK = 0;            // Internal serial clock is enabled
SPI1CON1bits.DISSDO = 0;            // SDOx pin is controlled by the module
SPI1CON1bits.MODE16 = 1;            // Communication is word-wide (16 bits)
SPI1CON1bits.SMP = 0;               // Input data is sampled at the middle of data output time
SPI1CON1bits.CKE = 0;               // Serial output data changes on transition from
                                    // Idle clock state to active clock state
SPI1CON1bits.CKP = 0;               // Idle state for clock is a low-level;
                                    // active state is a high-level
SPI1CON1bits.MSTEN = 1;             // Master mode enabled
SPI1STATbits.SPIEN = 1;             // Enable SPI module
SPI1BUF = 0x0000;                   // Write data to be transmitted

// Interrupt Controller Settings

IFS0bits.SPI1IF = 0;                // Clear the Interrupt flag
IEC0bits.SPI1IE = 1;                // Enable the interrupt
```

Figure 18-4: SPIx Master Mode Timing



Note 1: Four SPIx Clock modes are shown to demonstrate the CKP (SPIxCON1<6>) and CKE (SPIxCON1<8>) bits functionality only. Only one of the four modes can be chosen for the operation.

2: The SDIx and input sample are shown for two different values of the SMP bit (SPIxCON1<9>), for demonstration purposes only. Only one of the two configurations of the SMP bit can be chosen during the operation.

3: If there are no pending transmissions, the data in the SPIxTXB register is transferred to the SPIxSR register as soon as the user application writes to the SPIxBUF register.

4: Operation for 8-bit mode is shown. The operation for 16-bit mode is similar to 8-bit except for the number of clock

18.3.2.2 SLAVE MODE

In Slave mode, data is transmitted and received as the external clock pulses appear on the SCKx pin. The SPIx Clock Polarity Select (CKP) bit (SPIxCON<6>) and SPIx Clock Edge Select (CKE) bit (SPIxCON<8>) determine the edge of the clock pulse when the data transmission occurs. Data to be transmitted and data that is received are written into or read from the SPIxBUF register. The remaining module operation is identical to that of Master mode.

Note:	In Slave mode, if no data is written to the SPIxBUF register, when the SPI master initiates a read operation, SPI mode will transmit the last data that was written into the SPIxBUF register.

18.3.2.2.1 Slave Mode Setup Procedure

Use the following procedure to set up the SPIx module for the Slave mode of operation:

1. Clear the SPIxBUF register.
2. If using interrupts, configure the Interrupt controller:
   a) Clear the SPIx Interrupt Flag Status (SPIxIF) bit in the respective IFSx register.
   b) Set the SPIx Event Interrupt Enable (SPIxIE) bit in the respective IECx register.
   c) Write the SPIx Event Interrupt Priority (SPIxIP) bits in the respective IPCx register to set the interrupt priority.
3. Configure the SPIxCON1 register:
   a) Clear the Master Mode Enable (MSTEN) bit (SPIxCON1<5> = 0).
   b) Clear the Data Input Sample Phase (SMP) bit (SPIxCON1<9> = 0).
   c) If the Clock Edge Select (CKE) bit is set, set the Slave Select Enable (SSEN) bit to enable the $\overline{SS}$x pin (SPIxCON1<7> = 1).
4. Configure the SPIxSTAT register:
   a) Clear the Receive Overflow Flag (SPIROV) bit (SPIxSTAT<6> = 0).
   b) Set the SPIx Enable (SPIEN) bit (SPIxSTAT<15> = 1) to enable the SPIx operation.

Example 18-2 illustrates the code sequence that shows the SPI register configuration for Slave mode.

Example 18-2: SPI Register Configuration – Slave Mode

```
/* The following code sequence shows SPI register configuration for Slave mode */

SPI1BUF = 0;
IFS0bits.SPI1IF = 0;       // Clear the Interrupt flag
IEC0bits.SPI1IE = 0;       // Disable the interrupt

// SPI1CON1 Register Settings

SPI1CON1bits.DISSCK = 0;   // Internal Serial Clock is enabled
SPI1CON1bits.DISSDO = 0;   // SDOx pin is controlled by the module
SPI1CON1bits.MODE16 = 1;   // Communication is word-wide (16 bits)
SPI1CON1bits.SMP = 0;      // Input data is sampled at the middle of data
                           // output time.
SPI1CON1bits.CKE = 0;      // Serial output data changes on transition
                           // from Idle clock state to active clock state
SPI1CON1bits.CKP = 0;      // Idle state for clock is a low level; active
                           // state is a high level
SPI1CON1bits.MSTEN = 0;    // Master mode disabled
SPI1STATbits.SPIROV=0;     // No Receive Overflow has occurred
SPI1STATbits.SPIEN = 1;    // Enable SPI module

// Interrupt Controller Settings

IFS0bits.SPI1IF = 0;       // Clear the Interrupt flag
IEC0bits.SPI1IE = 1;       // Enable the interrupt
```

18.3.2.2.2 Slave Select Synchronization

The $\overline{SSx}$ pin allows Synchronous Slave mode. If the Slave Select Enable bit (SSEN) is set (SPIxCON1<7> = 1), transmission and reception are enabled in Slave mode, only if the $\overline{SSx}$ pin is driven to a low state (see Figure 18-6). The port output or other peripheral outputs must not be driven in order to allow the $\overline{SSx}$ pin to function as an input. If the SSEN bit is set and the $\overline{SSx}$ pin is driven high, the SDOx pin is no longer driven and will tri-state even if the module is in the middle of a transmission.

An aborted transmission is retried when the $\overline{SSx}$ pin is driven low again using the data held in the SPIxTXB register. If the SSEN bit is not set, the $\overline{SSx}$ pin does not affect the module operation in Slave mode.

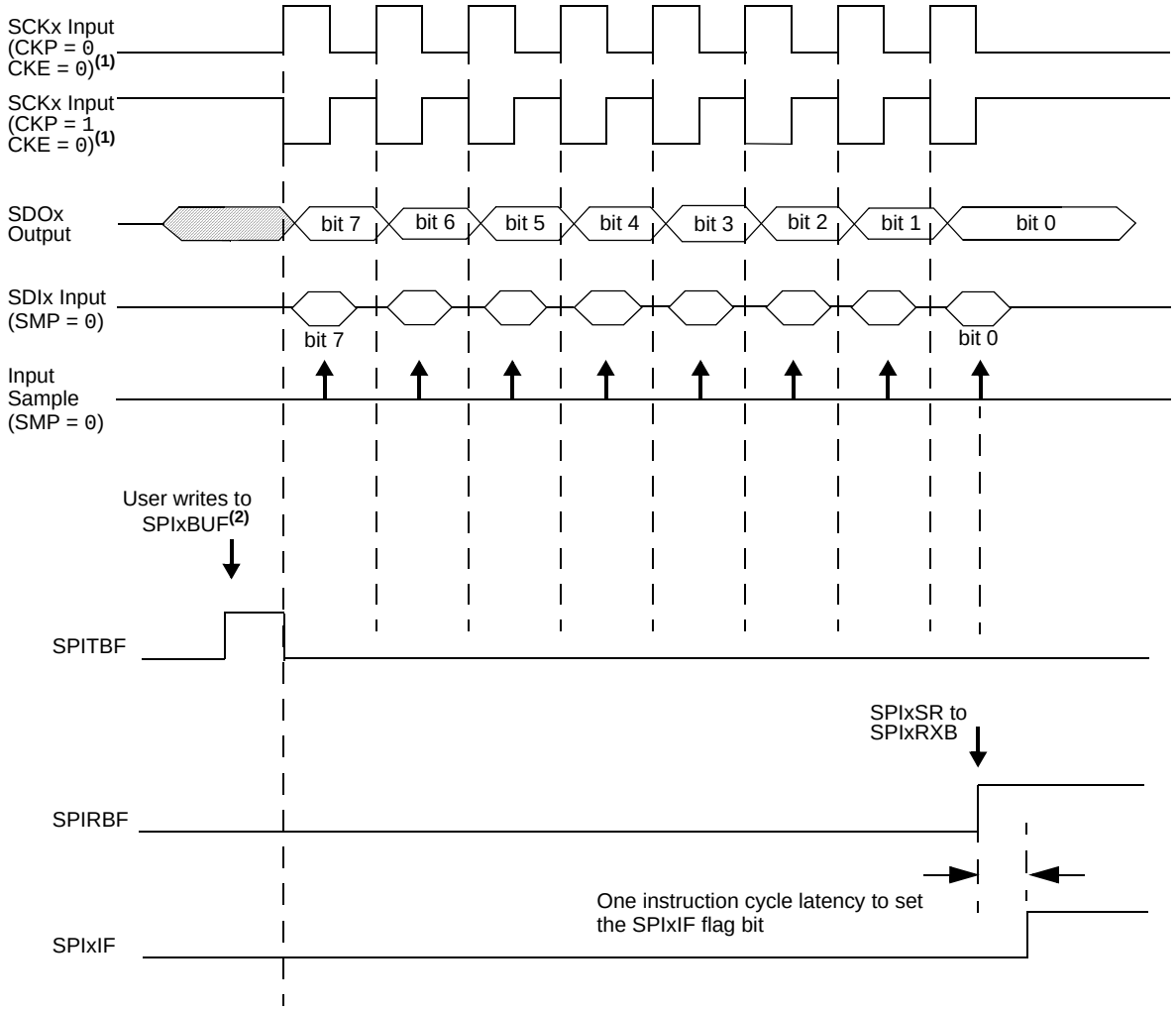
Note:	To meet the module timing requirements, the $\overline{SSx}$ pin must be enabled in Slave mode when CKE = 1 (see Figure 18-7 for details).

18.3.2.2.3 SPITBF Status Flag Operation

The Transmit Buffer Full Status bit (SPITBF) in the SPIxSTAT register (SPIxSTAT<1>) functions differently in Slave mode than it does in Master mode.

- If the SSEN bit is cleared (SPIxCON1<7> = 0), the SPITBF bit is set when the SPIxBUF register is loaded by the user application. It is cleared when the module transfers data from SPIxTXB to SPIxSR. This is similar to the SPITBF bit function in Master mode.
- If the SSEN bit is set (SPIxCON1<7> = 1), the SPITBF bit is set when the SPIxBUF register is loaded by the user application. However, it is cleared only when the SPIx module completes data transmission. The transmission is aborted when the $\overline{SSx}$ pin goes high, and may be retried later. Each data word is held in the SPIxTXB register until all bits are transmitted to the receiver.

Figure 18-5: SPIx Slave Mode Timing (Slave Select Pin Disabled)(3)

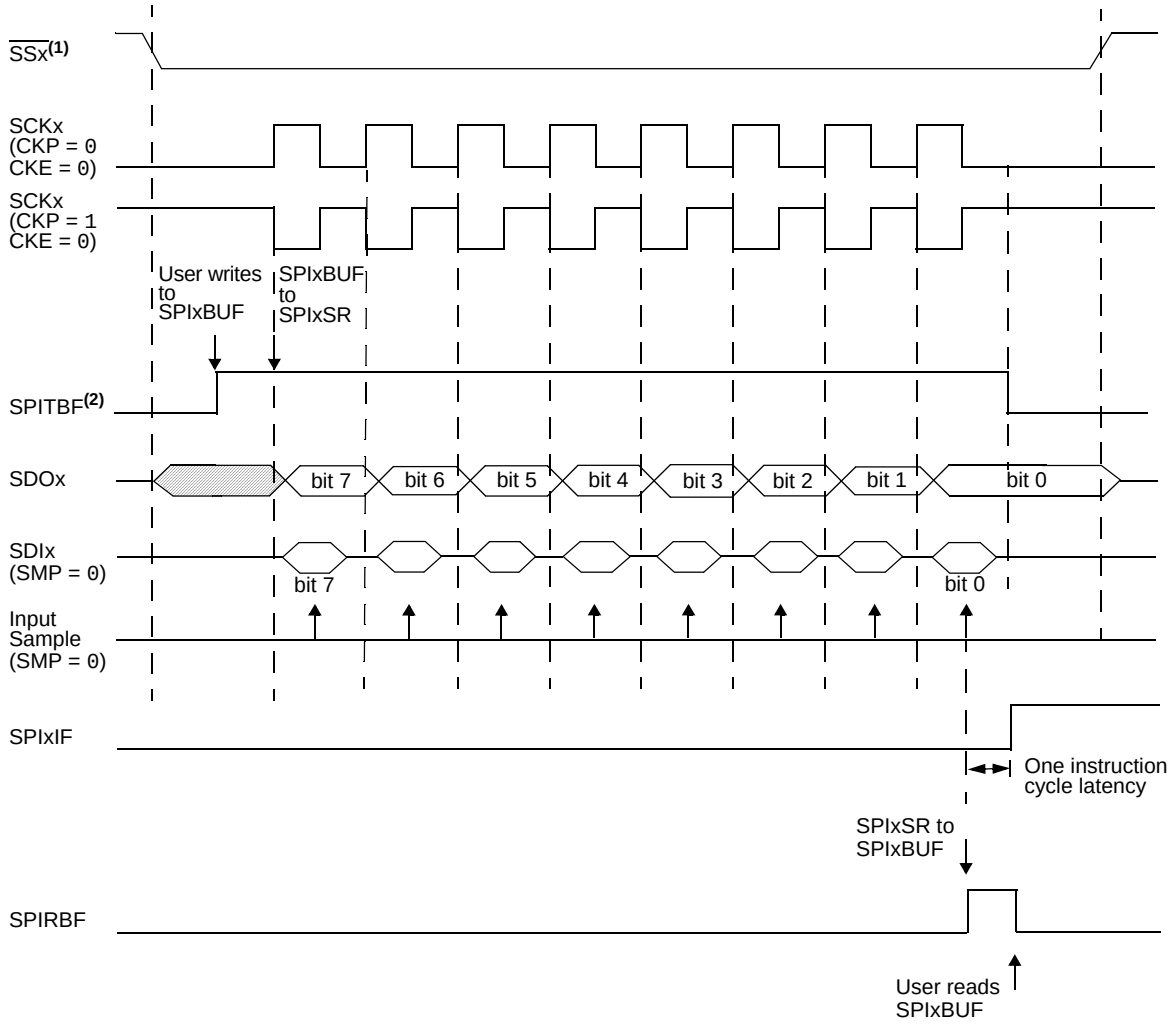


Note 1: Two SPIx Clock modes are shown only to demonstrate the CKP (SPIxCON<6>) and CKE (SPIxCON<8>) bits functionality. Any combination of the CKP and CKE bits can be chosen for the module operation.

2: If there are no pending transmissions or a transmission is in progress, the data in the SPIxBUF register is transferred to the SPIxSR register as soon as the user application writes to the SPIxBUF register.

3: Operation for 8-bit mode is shown; the operation for 16-bit mode is similar.

Figure 18-6: SPIx Slave Mode Timing (Slave Select Pin Enabled)(3)

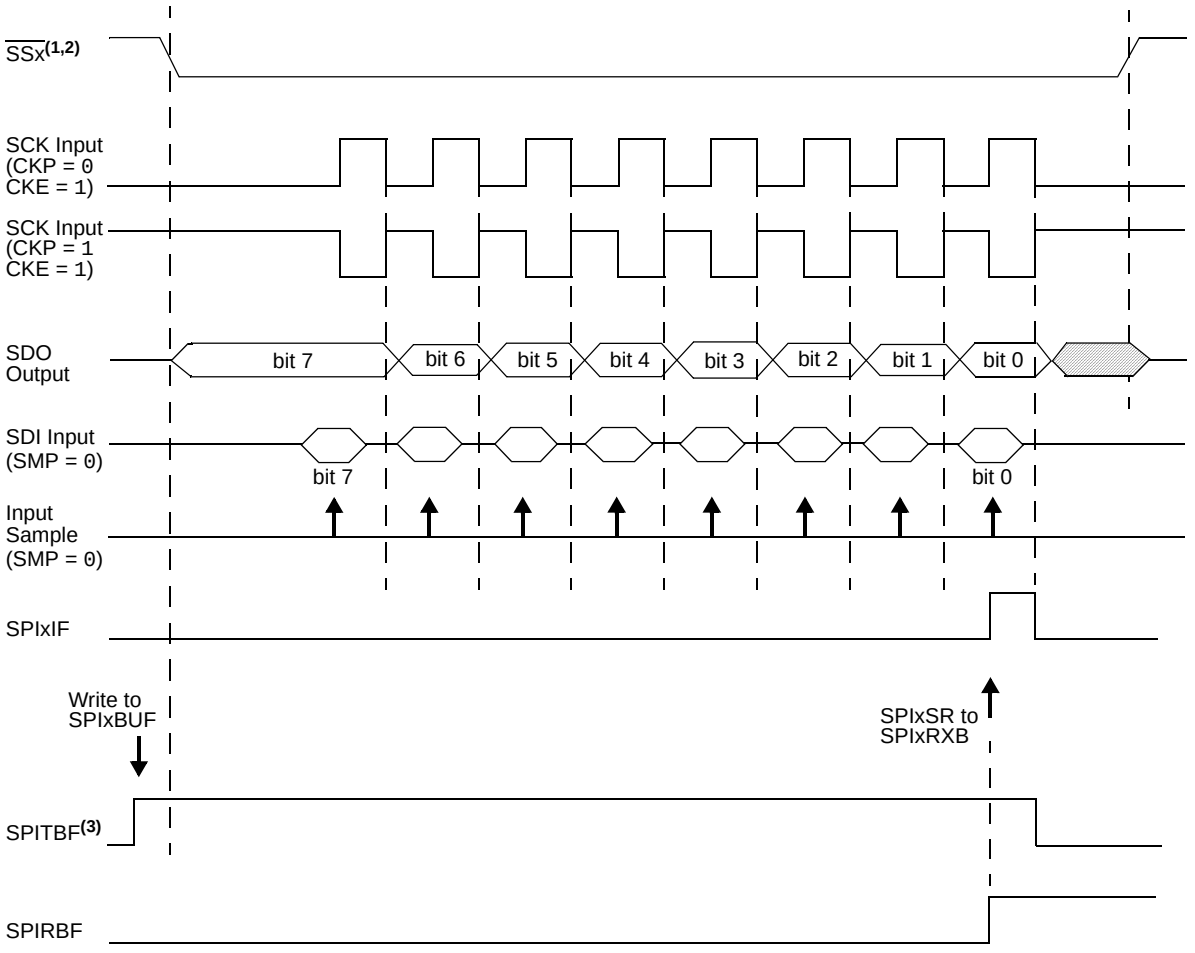


Note 1: When the SSEN bit is set (SPIxCON1<7> = 1), the $\overline{SSx}$ pin must be driven low to enable transmission and reception in Slave mode.

2: Transmit data is held in the SPIxTXB register, and SPITBF remains set until all bits are transmitted.

3: Operation for 8-bit mode is shown; the operation for 16-bit mode is similar.

Figure 18-7: SPIx Slave Mode Timing (CKE = 1)(4)



Note 1: The $\overline{SSx}$ pin must be used for Slave mode operation when CKE = 1.

2: When the SSEN bit is set (SPIxCON1<7> = 1), the $\overline{SSx}$ pin must be driven low to enable transmission and reception in Slave mode.

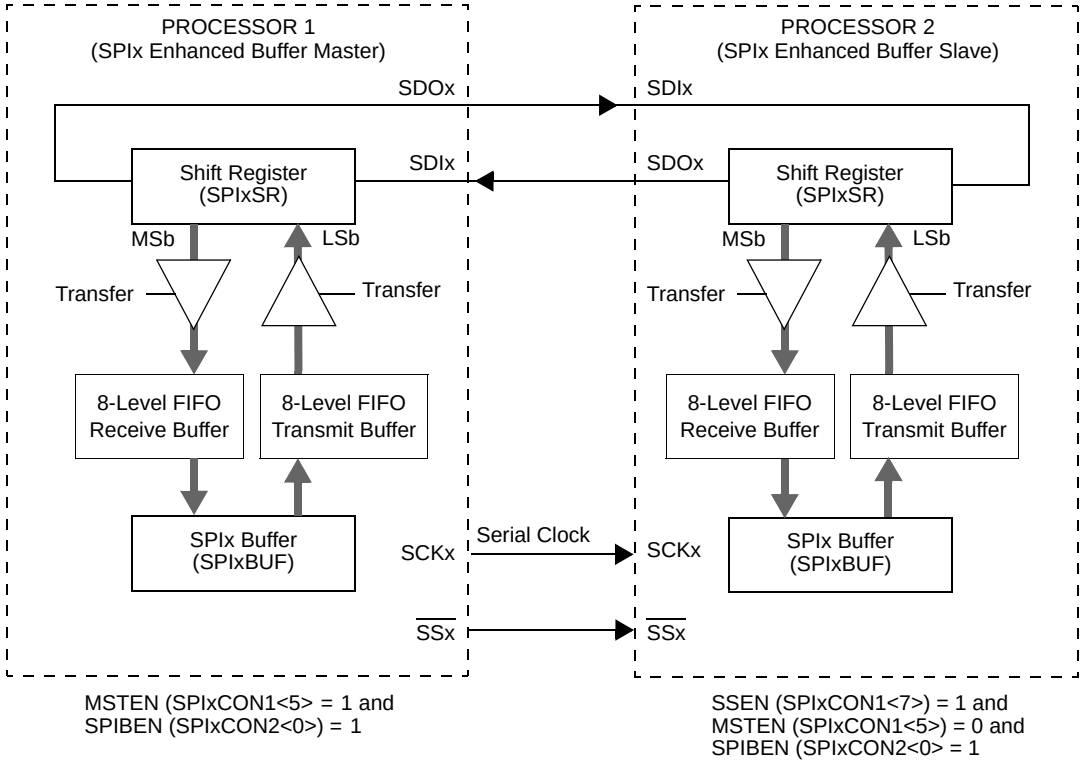
3: Transmit data is held in the SPIxTXB register, and the SPITBF remains set until all bits are transmitted.

4: Operation for 8-bit mode is shown; the operation for 16-bit mode is similar.

18.3.3 Enhanced Buffer Master and Slave Modes

The operation of Enhanced Buffer Master and Slave modes is very similar to standard Master and Slave modes. The difference is that data can be thought of as moving from the Shift register to a receive FIFO buffer and moving from the transmit FIFO buffer to the Shift register. The relationships in Enhanced Buffer mode are shown in Figure 18-8.

Figure 18-8: SPIx Master/Slave Connection (Enhanced Buffer Modes)



Note 1: The use of $\overline{SSx}$ pin in Slave mode is optional.

2: The user application must write transmit data to read received data from SPIxBUF. The SPIxTXB and SPIxRXB registers are memory mapped to SPIxBUF.

18.3.3.1 ENHANCED BUFFER MASTER AND SLAVE MODE

In Enhanced Buffer Master mode, the system clock is prescaled and then used as the serial clock. The prescaling is based on the settings in the Primary Prescale bits (PPRE<1:0>) and Secondary Prescale bits (SPRE<1:0>) in the SPIx Control Register 1 (SPIxCON1). The serial clock is output via the SCKx pin to slave devices. Clock pulses are generated only when there is data to be transmitted. For more information, see **18.4 "Master Mode Clock Frequency"**. The CKP and CKE bits determine on which edge of the clock data transmission occurs.

The CPU loads data to be transmitted into the transmit buffer by writing to the SPIxBUF register. An SPIx transmission begins after the first buffer write. Up to eight data elements can be loaded. The number of pending transfers is indicated by the Buffer Element Count bits (SPIBEC<2:0>) in the SPIx Status and Control Register (SPIxSTAT<10:8>).

In Master mode, the buffer element count reflects the number of transfers pending in the transmit buffer. In Slave mode, it reflects the number of unread receptions in the receive buffer. If the Shift register is empty, the first write will immediately load the Shift register, leaving eight transmit buffer locations available.

After completion of an SPIx transfer, the receive buffer location is updated with the received data. The CPU accesses the received data by reading the SPIxBUF register. After each CPU read, the SPIxBUF points to the next buffer location. The SPIx transfers continue until all pending data transfers have completed.

The SPIx module follows these sequences in Enhanced Buffer Master mode:

1. After the module is set up for Master mode of operation and is enabled, data to be transmitted is written to the SPIxBUF register and is loaded into the next available transmit buffer location. The SPIxTBF bit (SPIxSTAT<1>) and SPIxIF bit are set after eight pending transfers are loaded.
2. The contents of the current buffer location are moved to the Shift register, SPIxSR. The SPIxTBF bit is cleared by the module if a buffer location is available for a CPU write.
3. A series of 8/16 clock pulses, shift out 8/16 bits of transmit data from the SPIxSR register to the SDOx pin, and simultaneously shift in the data at the SDIx pin into the SPIxSR register.
4. When the transfer is complete, the following occurs:
   - When ongoing transmit and receive operation is complete, the contents of the SPIxSR register are moved into the next available location in the receive buffer.
   - If the last unread location is written by the SPIx module, the SPIxRBF bit (SPIxSTAT<0>) is set by the module, indicating that all buffer locations are full. The SPIx interrupts can be enabled by selecting an Interrupt mode with the SISEL<2:0> bits (SPIxSTAT<4:2>), and by setting the SPIx Interrupt Enable bit (SPIxIE). The SPIxIF flag is not cleared automatically by the hardware.
   - After the SPIxBUF register is read by the user application, the hardware clears the SPIxRBF bit, and the SPIxBUF register increments to the next unread receive buffer location. If the SPIxBUF register reads beyond the last unread location, it will not increment the buffer location. The SRXMPT bit (SPIxSTAT<5>) shows the status of the RX FIFO. This bit is set if the RX FIFO is empty.
5. If the SPIxRBF bit is set (receive buffer is full) when the SPIx module needs to transfer data from the SPIxSR register to the buffer, the module will set the SPIROV bit (SPIxSTAT<6>), indicating an overflow condition and set the SPIxIF bit.
6. Data to be transmitted can be written to the SPIxBUF register by the user application at any time as long as the SPIxTBF bit (SPIxSTAT<1>) is clear. Up to eight pending transfers can be loaded into the buffer allowing continuous transmission. The SRMPT bit (SPIxSTAT<7>) shows the status of the SPIxSR register. The SRMPT status bit is set when the SPIxSR is empty and ready to send or receive the data.

The timing of events in Enhanced Buffer Master mode operation is essentially the same as that for Standard Master mode, shown in Figure 18-4.

To set up the SPIx module for the Enhanced Buffer Master mode of operation, complete the following steps:

1. If using interrupts:
   a) Clear the SPIxIF bit in the respective IFSx register.
   b) Select an interrupt mode using the SISEL<2:0> bits (SPIxSTAT<4:2>).
   c) Set the SPIxIE bit in the respective IECx register.
   d) Write the SPIxIP bits in the respective IPCx register.
2. When MSTEN (SPIxCON1<5>) = 1, write the desired settings to the SPIxCON1 and SPIxCON2 registers.
3. Clear the SPIROV bit (SPIxSTAT<6>).
4. Select Enhanced Buffer mode by setting the SPIBEN bit (SPIxCON2<0>).
5. Enable the SPIx operation by setting the SPIEN bit (SPIxSTAT<15>).
6. Write the data to be transmitted to the SPIxBUF register. The transmission (and reception) starts as soon as data is written to the SPIxBUF register.

18.3.3.2 ENHANCED BUFFER SLAVE MODE

In Enhanced Buffer Slave mode, data is transmitted and received as the external clock pulses appear on the SCKx pin. The CKP (SPIxCON1<6>) and CKE (SPIxCON1<8>) bits determine on which edge of the clock data transmission occurs.

The rest of the operation of the module is identical to that in the Master mode. Specific timings for Enhanced Buffer Slave mode operations are identical to those for Standard Slave mode, as shown in Figure 18-5, Figure 18-6 and Figure 18-7.

To set up the SPIx module for the Enhanced Buffer Slave mode of operation, complete the following steps:

1. Clear the SPIxBUF register.
2. If using interrupts:
   a) Clear the SPIxIF bit in the respective IFSx register.
   b) Select an Interrupt mode using the SISEL<2:0> bits (SPIxSTAT<4:2>).
   c) Set the SPIxIE bit in the respective IECx register.
   d) Write the SPIxIP bits in the respective IPCx register to set the interrupt priority.
3. When MSTEN (SPIxCON1<5>) = 0, write the desired settings to the SPIxCON1 and SPIxCON2 registers.
4. Clear the SMP bit (SPIxCON1<9>).
5. If the CKE bit is set, the SSEN bit (SPIxCON1<7>) must be set, thus enabling the $\overline{SSx}$ pin.
6. Clear the SPIROV bit (SPIxSTAT<6>).
7. Select Enhanced Buffer mode by setting the SPIBEN bit (SPIxCON2<0>).
8. Enable the SPIx operation by setting the SPIEN bit (SPIxSTAT<15>).

18.3.3.2.1 Slave Select Synchronization

The $\overline{SSx}$ pin allows a Synchronous Slave mode. If the SSEN bit (SPIxCON1<7>) is set, transmission and reception is enabled in Slave mode, only if the $\overline{SSx}$ pin is driven to a low state (see Figure 18-6). The port output or other peripheral outputs must not be driven in order to allow the $\overline{SSx}$ pin to function as an input. If the SSEN bit is set and the $\overline{SSx}$ pin is driven high, the SDOx pin is no longer driven and will tri-state even if the module is in the middle of a transmission. An aborted transmission will be retried the next time the $\overline{SSx}$ pin is driven low using the data held in the SPIxTXB register. If the SSEN bit is not set, the $\overline{SSx}$ pin does not affect the module operation in Slave mode.

Note:	To meet the module timing requirements, the $\overline{SSx}$ pin must be enabled in Slave mode when CKE = 1 (see Figure 18-7 for details).

18.3.3.2.2 SPIxTBF Status Flag Operation

The function of the SPIxTBF bit (SPIxSTAT<1>) is different in the Slave mode of operation. If the SSEN bit is cleared, the SPIxTBF bit is set when the last available buffer location is loaded by the user application. It is cleared when the module transfers data from the buffer to the SPIxSR register and a buffer location is available for a CPU write. This is similar to the SPIxTBF bit function in Master mode.

If the SSEN bit is set, the SPIxTBF bit is set when the last available buffer location is loaded by the user application. However, it is cleared only when the SPIx module completes data transmission, leaving a buffer location available for a CPU write. A transmission will be aborted when the $\overline{SSx}$ pin goes high and may be retried at a later time. Each data word is held in the buffer until all bits are transmitted to the receiver.

18.3.4 Framed SPIx Modes

The SPI module supports a basic framed SPIx protocol while operating in either Master or Slave modes. Four control bits that configure the framed SPIx operation are:

- Framed SPIx Support (FRMEN)

  The FRMEN bit (SPIxCON2<15>) enables Framed SPIx modes and causes the $\overline{SSx}$ pin to be used as a frame synchronization pulse input or output pin. The state of the SSEN bit (SPIxCON1<7>) is ignored.

- Frame Sync Pulse Direction Control (SPIFSD)

  The SPIFSD bit (SPIxCON2<14>) determines whether the $\overline{SSx}$ pin is an input or an output (i.e., whether the module receives or generates the frame synchronization pulse).

- Frame Sync Pulse Polarity (FRMPOL)

  The FRMPOL bit (SPIxCON2<13>) selects the polarity of the frame synchronization pulse (active-high or active-low) for a single SPIx data frame.

- Frame Sync Pulse Edge Select (FRMDLY)

  The FRMDLY bit (SPIxCON2<1>) selects the synchronization pulse to either coincide with, or precede, the first serial clock pulse.

In Framed Master mode, the SPIx module generates the frame synchronization pulse and provides this pulse to other devices at the $\overline{SSx}$ pin.

In Framed Slave mode, the SPIx module uses a frame synchronization pulse received at the $\overline{SSx}$ pin.

Note:	The $\overline{SSx}$ and SCKx pins must be used in all Framed SPIx modes.

The Framed SPIx modes are supported in conjunction with the unframed Master and Slave modes. This makes four framed SPIx configurations available to the user application:

- SPIx Master mode and Framed Master mode
- SPIx Master mode and Framed Slave mode
- SPIx Slave mode and Framed Master mode
- SPIx Slave mode and Framed Slave mode

These modes determine whether the SPIx module generates the serial clock and the frame synchronization pulse.

- When FRMEN (SPIxCON<14>) = 1 and MSTEN (SPIxCON<5>) = 1, the SCKx pin becomes an output and the SPI clock at SCKx becomes a free running clock.
- When FRMEN = 1 and MSTEN = 0, the SCKx pin becomes an input. The source clock provided to the SCKx pin is assumed to be a free running clock.

The polarity of the clock is selected by the CKP bit (SPIxCON<6>). The CKE bit (SPIxCON<8>) is not used for Framed SPI modes and should be programmed to '0' by the user application.

- When CKP = 0, the frame synchronization pulse output and the SDOx data output change on the rising edge of the clock pulses at the SCKx pin. Input data is sampled at the SDIx input pin on the falling edge of the serial clock.
- When CKP = 1, the frame synchronization pulse output and the SDOx data output change on the falling edge of the clock pulses at the SCKx pin. Input data is sampled at the SDIx input pin on the rising edge of the serial clock.

18.3.4.1 FRAME MASTER AND FRAME SLAVE MODES

- When SPIFSD (SPIxCON2<14>) = 0, the SPIx module is in Framed Master mode. In this mode, the frame synchronization pulse is initiated by the module when the user application writes the transmit data to the SPIxBUF location (thus writing the SPIxTXB register with transmit data). At the end of the frame synchronization pulse, the data is transferred from the SPIxTXB register to the SPIxSR register and data transmission/reception begins.
- When SPIFSD = 1, the module is in Framed Slave mode. In this mode, the frame synchronization pulse is generated by an external source. When the module samples the frame synchronization pulse, it transfers the contents of the SPIxTXB register to the SPIxSR register and data transmission/reception begins. The user application must ensure that the correct data is loaded into the SPIxBUF register for transmission before the frame synchronization pulse is received.

Note:	Receiving a frame synchronization pulse starts a transmission, regardless of whether data is written to the SPIxBUF register. If no write operation is performed, the existing contents of the SPIxTXB register are transmitted.

18.3.4.2 SPIx MASTER/FRAMED MASTER MODE

In SPI Master/Framed Master mode, the SPIx module generates both the clock and the frame synchronization signals, as shown in Figure 18-9. This configuration is enabled by setting the MSTEN and FRMEN bits (SPIxCON1<5> and SPIxCON2<15>) to '1', and the SPIFSD bit (SPIxCON2<14>) to '0'.

In this mode, the serial clock is output continuously at the SCKx pin, regardless of whether the module is transmitting. When the SPIxBUF register is written, the $\overline{SSx}$ pin is driven to its active state (as determined by the FRMPOL bit) on the appropriate transmit edge of the SCKx clock, and remains active for one data frame. Figure 18-10 shows that if the FRMDLY control bit (SPIxCON2<1>) is cleared, the frame synchronization pulse precedes the data transmission. Figure 18-11 illustrates that if FRMDLY is set, the frame synchronization pulse coincides with the beginning of the data transmission. The module starts transmitting data on the next transmit edge of the SCKx.

Figure 18-9: SPIx Master/Framed Master Connection Diagram

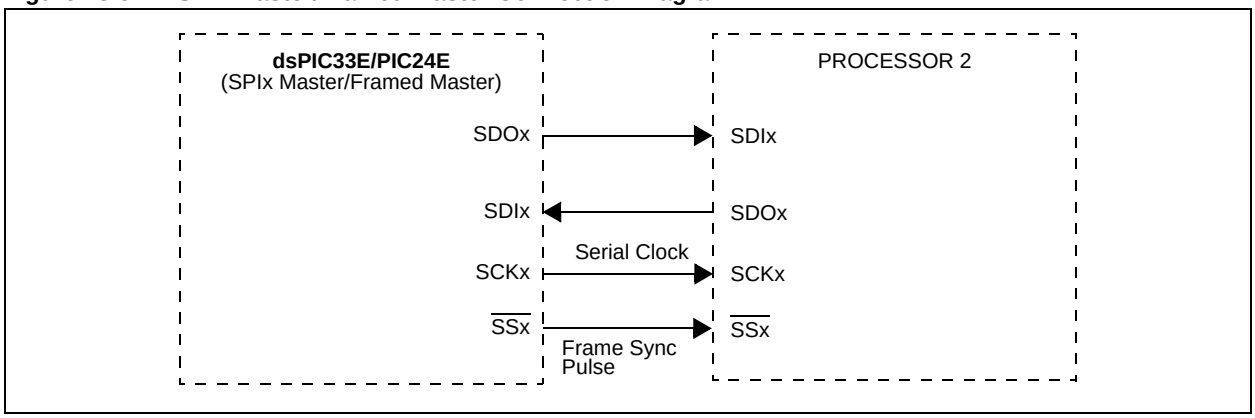


Figure 18-10: SPIx Master/Framed Master Timing (FRMDLY = 0)

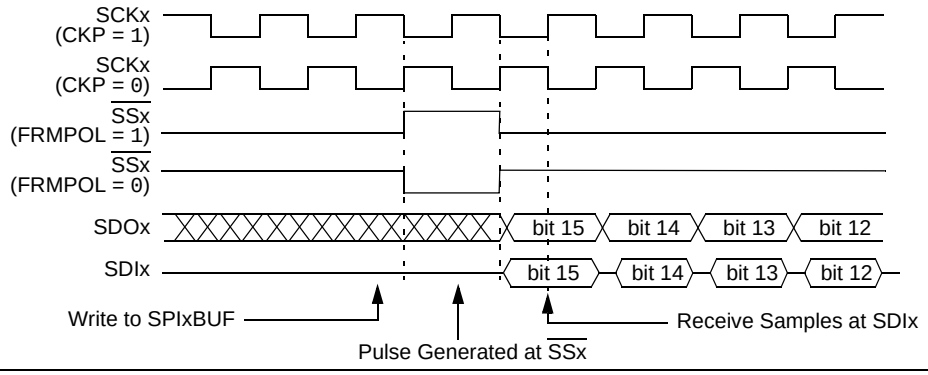
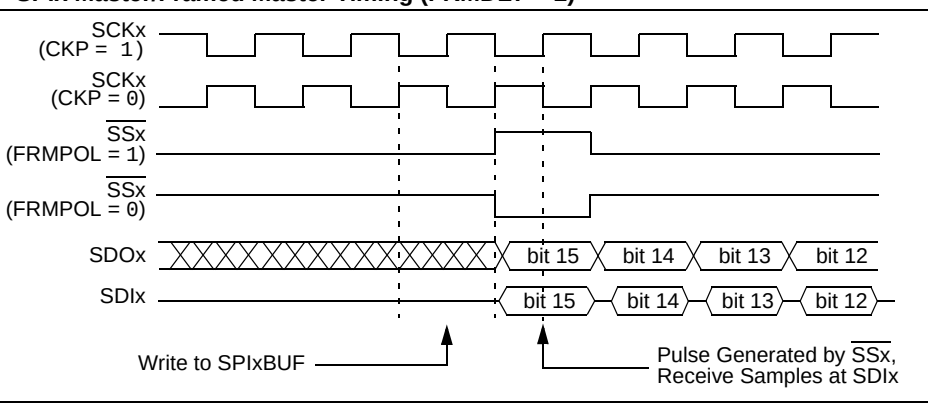


Figure 18-11: SPIx Master/Framed Master Timing (FRMDLY = 1)



18.3.4.3 SPIx MASTER/FRAMED SLAVE MODE

In SPI Master/Framed Slave mode, the module generates the clock signal but uses the Slave module's frame synchronization signal for data transmission (see Figure 18-12). It is enabled by setting the MSTEN, FRMEN and SPIFSD bits (SPIxCON1<5>, SPIxCON2<15> and SPIxCON2<14>) to '1'.

In this mode, the $\overline{SSx}$ pin is an input. It is sampled on the sample edge of the SPIx clock. When it is sampled in its active state, data is transmitted on the subsequent transmit edge of the SPIx clock. The interrupt flag, SPIxIF, is set when the transmission is complete. The user application must make sure that the correct data is loaded into the SPIxBUF register for transmission before the signal is received at the $\overline{SSx}$ pin.

Figure 18-12: SPIx Master/Framed Slave Connection Diagram

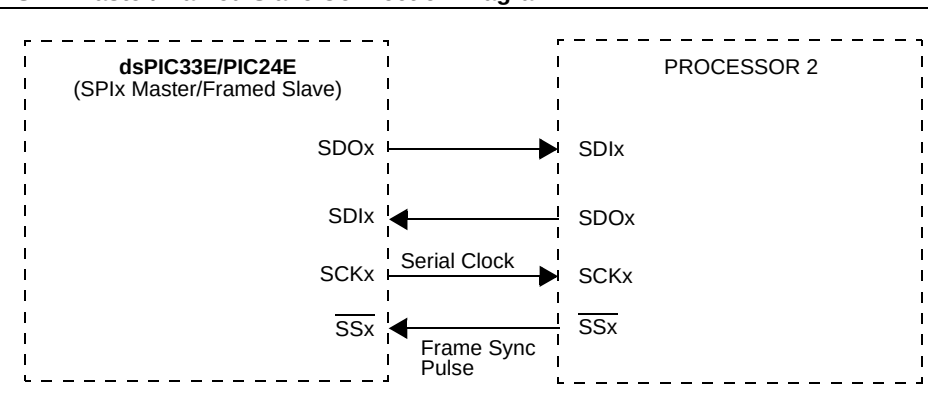


Figure 18-13: SPIx Master/Framed Slave Timing (FRMDLY = 0)

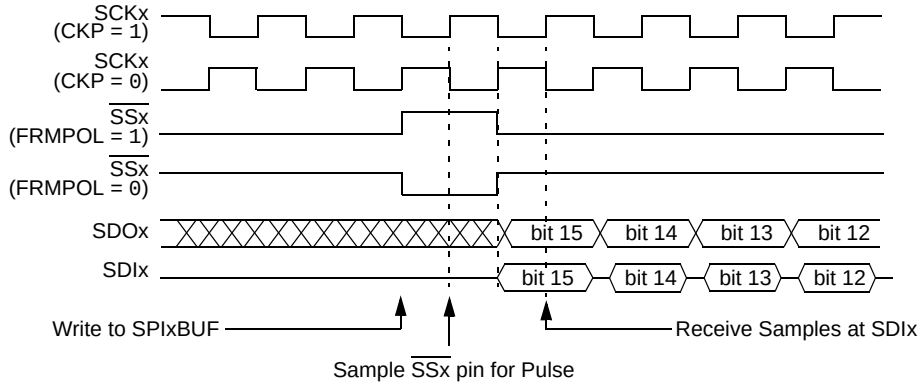
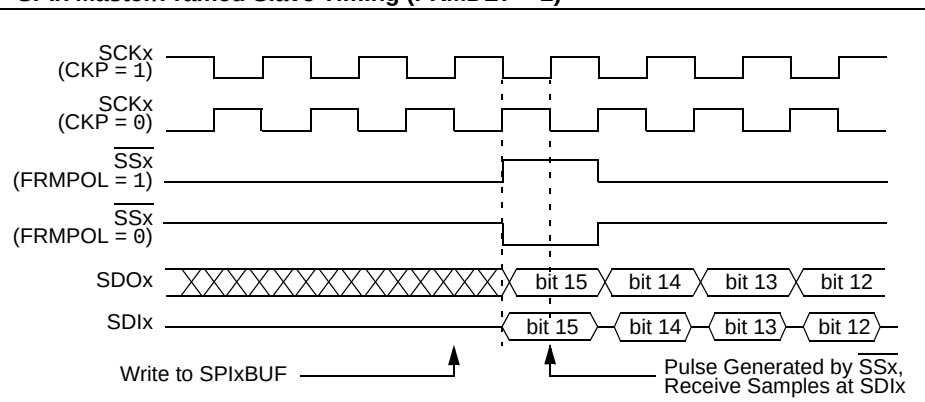


Figure 18-14: SPIx Master/Framed Slave Timing (FRMDLY = 1)

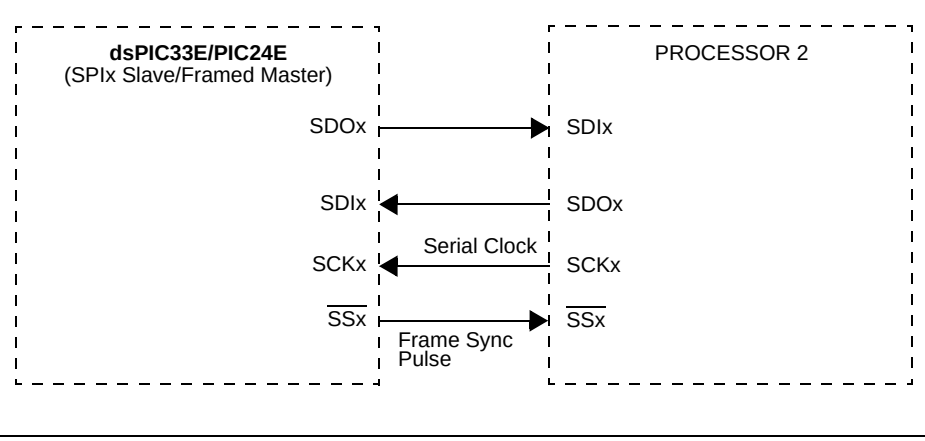


18.3.4.4 SPIx SLAVE/FRAMED MASTER MODE

In SPI Slave/Framed Master mode, the module acts as an SPIx slave and takes its clock from the other SPIx module; however, it produces frame synchronization signals to control data transmission (see Figure 18-15). It is enabled by setting the MSTEN bit (SPIxCON1<5>) to '0', the FRMEN bit (SPIxCON2<15>) to '1', and the SPIFSD bit (SPIxCON2<14>) to '0'.

The input SPIx clock is continuous in Slave mode. The SSx pin is an output pin when the SPIFSD bit is low. Therefore, when the SPIxBUF register is written, the module drives the SSx pin to the active state on the appropriate transmit edge of the SPIx clock for one SPIx clock cycle. Data starts transmitting on the appropriate SPIx clock transmit edge.

Figure 18-15: SPIx Slave/Framed Master Connection Diagram

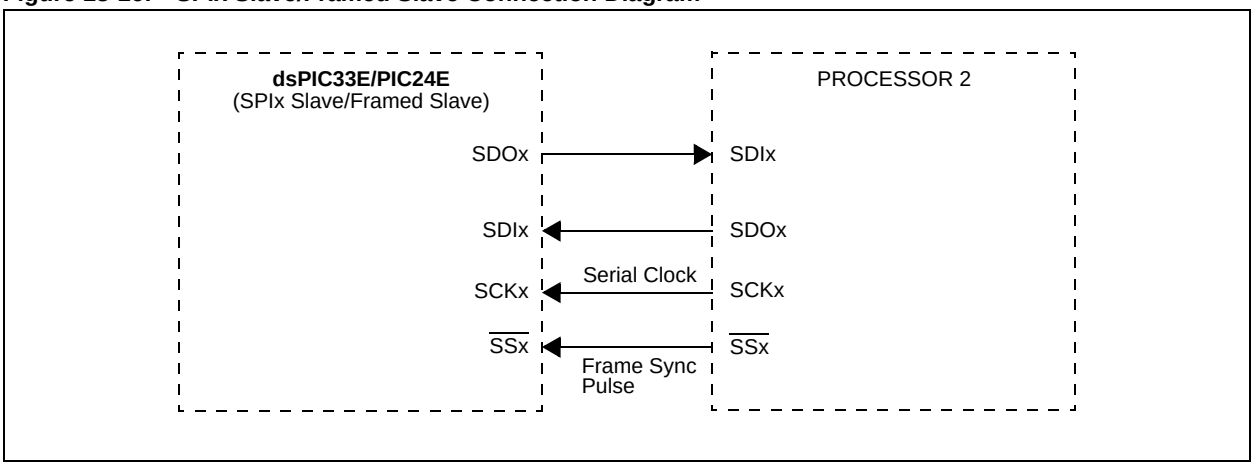


18.3.4.5 SPIx SLAVE/FRAMED SLAVE MODE

In SPI Slave/Framed Slave mode, the module gets both its clock and frame synchronization signal from the master module (see Figure 18-16). This mode is enabled by setting the MSTEN bit (SPIxCON1<5>) to '0', the FRMEN bit (SPIxCON2<15>) to '1', and the SPIFSD bit (SPIxCON2<14>) to '1'.

In this mode, both SCKx and $\overline{SSx}$ pins are the inputs. The $\overline{SSx}$ pin is sampled on the sample edge of the SPIx clock. When $\overline{SSx}$ is sampled at its active state, data is transmitted on the appropriate transmit edge of SCKx.

Figure 18-16: SPIx Slave/Framed Slave Connection Diagram



18.3.5 SPIx Receive-Only Operation

Setting the DISSDO control bit (SPIxCON1<11>) disables transmission at the SDOx pin. This allows the SPIx module to be configured for a Receive-only mode of operation. The SDOx pin is controlled by the respective port function if the DISSDO bit is set.

The DISSDO function is applicable to all SPIx operating modes.

18.3.6 SPIx Error Handling

If a new data word has been shifted into SPIxSR and the previous SPIxBUF contents have not been read, the SPIROV bit (SPIxSTAT<6>) is set. Any received data in SPIxSR is not transferred, and further data reception is disabled until the SPIROV bit is cleared. The SPIROV bit is not cleared automatically by the module; it must be cleared by the user application.

The SPIx Interrupt Flag (SPIxIF) is set when the SPIRBF (SPIxSTAT<0>) or SPITBF (SPIxSTAT<1>) bit is set. The interrupt flag cannot be cleared by hardware. It must be reset in software. The actual SPIx interrupt is generated only when the corresponding SPIxIE bit is set in the IECx control register.

In addition, the SPIx Error Interrupt Flag (SPIxEIF) is set when the SPIROV bit is set. This interrupt flag must be cleared in software. The actual SPIx Error Interrupt is generated only when the corresponding SPIxEIE bit is set in the IECx control register.

18.4 MASTER MODE CLOCK FREQUENCY

In Master mode, the clock provided to the SPIx module is the instruction cycle (TCY). This clock is then prescaled by the primary prescaler, specified by the Primary Prescale (PPRE<1:0>) bits (SPIxCON1<1:0>), and the secondary prescaler, specified by the Secondary Prescale (SPRE<2:0>) bits (SPIxCON1<4:2>). The prescaled instruction clock becomes the serial clock and is provided to external devices through the SCKx pin.

Note:	The SCKx signal clock is not free running for normal SPI modes. It only runs for 8 or 16 pulses when the SPIxBUF is loaded with data; however, it is continuous for Framed modes.

Equation 18-1 is used to calculate the SCKx clock frequency as a function of the primary and secondary prescaler settings.

Equation 18-1: SPI Clock Frequency

$$F_{SCK} = \frac{F_{CY}}{Primary\ Prescaler * Secondary\ Prescaler}$$

Some sample SPIx clock frequencies (in kHz) are shown in Table 18-1.

Note:	Not all clock rates are supported. For detail information, refer to the SPIx timing specifications in the specific device data sheet.

Table 18-1: Sample SCKx Frequencies(1)

FCY = 40 MHz		**Secondary Prescaler Settings**				
		1:1	**2:1**	**4:1**	**6:1**	**8:1**
Primary Prescaler Settings	1:1	Invalid	Invalid	10000	6666.67	5000
	4:1	10000	5000	2500	1666.67	1250
	16:1	2500	1250	625	416.67	312.50
	64:1	625	312.5	156.25	104.17	78.125
FCY = 60 MHz						
Primary Prescaler Settings	1:1	Invalid	Invalid	Invalid	10000	7500
	4:1	Invalid	7500	3750	2500	1875
	16:1	3750	1875	937.5	625	468.75
	64:1	937.5	468.75	234.37	156.25	117.18

Note 1: SCKx frequencies are in kHz.

18.5 SPI OPERATION WITH DMA

The DMA module transfers data between the CPU and SPI without CPU assistance. Refer to the specific device data sheet to see if DMA is present on your particular device. When using the SPI module with DMA, the ENHBUF bit can be programmed to '0', thereby disabling FIFO operation. For more information on the DMA module, refer to **Section 22. "Direct Memory Access (DMA)"** (DS70348).

If the DMA channel is associated with the SPI receiver, the SPI issues a DMA request every time data is ready to be moved from SPI to RAM. DMA transfers data from the SPIxBUF register into RAM and issues a CPU interrupt after a predefined number of transfers.

Similarly, if the DMA channel is associated with the SPI transmitter, the SPI issues a DMA request after each successful transmission. After each DMA request, the DMA transfers new data into the SPIxBUF register and issues a CPU interrupt after a predefined number of transfers. Since DMA channels are unidirectional, two DMA channels are required if SPI is used for both receive and transmit. Each DMA channel must be initialized, as shown in Table 18-2.

Table 18-2: DMA Channel Register Initialization for SPI to DMA Association

Peripheral to DMA Association	DMAxREQ Register IRQSEL<6:0> Bits	DMAxPAD Register Values to Read from Peripheral/Write to Peripheral
SPI1TX/RX – SPI1 Transmit/Receive	0001010	0x0248 (SPI1BUF)
SPI2TX/RX – SPI2 Transmit/Receive	0100001	0x0268 (SPI2BUF)
SPI3TX/RX – SPI3 Transmit/Receive	1011011	0x02A8 (SPI3BUF)
SPI4TX/RX – SPI4 Transmit/Receive	1111011	0x02C8 (SPI4BUF)

Starting DMA transfer to/from the SPI peripheral depends on SPI data direction and whether operation occurs in Slave or Master mode.

- **TX only in Master mode**

  In this configuration, no DMA request is issued until the first block of SPI data is sent. To initiate DMA transfers, the user application must first send data using DMA Manual Transfer mode, or it must first write data into the SPI buffer (SPIxBUF) independently of the DMA.

- **RX only in Master mode**

  In this configuration, no DMA request is issued until the first block of SPI data is received. However, in Master mode, no data is received until SPI transmits first. To initiate DMA transfers, the user application must use DMA Null Data Write mode and start DMA Manual Transfer mode.

- **RX and TX in Master mode**

  In this configuration, no DMA request is issued until the first block of SPI data is received. However, in Master mode, no data is received until the SPI transmits it. To initiate DMA transfers, the user application must first send data using DMA Manual Transfer mode, or it must first write data into the SPI buffer (SPIxBUF) independently of the DMA.

- **TX only in Slave mode**

  In this configuration, no DMA request is issued until the first block of SPI data is received. To initiate DMA transfers, the user application must first send data using DMA Manual Transfer mode, or it must first write data into the SPI buffer (SPIxBUF) independently of the DMA.

- **RX only in Slave mode**

  This configuration generates a DMA request as soon as the first SPI data has arrived. No special steps are required by the user application to initiate DMA transfer.

- **RX and TX in Slave mode**

  In this configuration, no DMA request is issued until the first SPI data block is received. To initiate DMA transfers, the user application must first send data using DMA Manual Transfer mode, or it must first write data into the SPI buffer (SPIxBUF) independently of the DMA.

18.5.1 SPI Transmission and Reception with DMA

Example 18-3 illustrates the SPI transmission and reception with DMA. The SPI module is configured in Master mode. Two DMA channels are used, Channel 0 for data transmission and Channel 1 for data reception.

DMA Channel 0 is configured for SPI transmission with these parameters:

- Transfer data from RAM to SPI continuously
- Register indirect with post-increment
- Using two ping-pong buffers
- 16 transfers per buffer

DMA Channel 1 is configured for SPI reception with these parameters:

- Transfer data from SPI to RAM continuously
- Register indirect with post-increment
- Using two ping-pong buffers
- 16 transfers per buffer

Example18-3: SPI Transmission and Reception with DMA

Setup for SPI1 Master Mode:

```
// Interrupt Controller Settings

IFS0bits.SPI1IF = 0;

// SPI1CON1 Register Settings

SPI1CON1bits.MODE16 = 1;  // Communication is word-wide (16 bits)
SPI1CON1bits.MSTEN = 1;   // Master mode enabled

// SPI1CON2 Register Settings

SPI1CON2bits.FRMEN = 0;   // Framed mode disabled

// SPI1STAT Register Settings

SPI1STATbits.SPISIDL = 0; // Continue module operation in Idle mode
SPI1STATbits.SPIBEC = 0;  // Buffer Length = 1 Word
SPI1STATbits.SPIROV = 0;  // No Receive Overflow has occurred
SPI1STATbits.SPIEN = 1;   // Enable SPI module

// Force First Word After Enabling SPI

DMA0REQbits.FORCE=1;
while (DMA0REQbits.FORCE == 1);

IEC0bits.SPI1IE = 1;
```

Set up DMA Channel 0 to Transmit in Continuous Ping-Pong Mode:

```
unsigned int TxBufferA[16] __attribute__((space(dma)));
unsigned int TxBufferB[16] __attribute__((space(dma)));

IFS0bits.DMA0IF = 0;
IEC0bits.DMA0IE = 1;
DMACS0 = 0;
DMA0CON = 0x2002;
DMA0STA = __builtin_dmaoffset(TxBufferA);
DMA0STB = __builtin_dmaoffset(TxBufferB);
DMA0PAD = (volatile unsigned int) &SPI1BUF;
DMA0CNT = 15;
DMA0REQ = 0x000A;
DMA0CONbits.CHEN = 1;
```

Set up DMA Channel 1 to Receive in Continuous Ping-Pong Mode:

```
unsigned int RxBufferA[16] __attribute__((space(dma)));
unsigned int RxBufferB[16] __attribute__((space(dma)));

IFS0bits.DMA1IF = 0;
IEC0bits.DMA1IE = 1;
DMA1CON = 0x0002;
DMA1STA = __builtin_dmaoffset(RxBufferA);
DMA1STB = __builtin_dmaoffset(RxBufferB);
DMA1PAD = (volatile unsigned int) &SPI1BUF;
DMA1CNT = 15;
DMA1REQ = 0x000A;
DMA1CONbits.CHEN = 1;
```

Example 18-3: SPI Transmission and Reception with DMA (Continued)

SPI and DMA Interrupt Handlers:

```
void __attribute__((__interrupt__)) _SPI1Interrupt(void)
{
    IFS0bits.SPI1IF = 0;
}

void __attribute__((__interrupt__)) _DMA0Interrupt(void)
{
static unsigned int BufferCount = 0;  // Keep record of the buffer that
                                      // contains TX data
if(BufferCount == 0)
{
TxData(BufferA);                      // Transmit SPI data in
                                      // DMA RAM Primary buffer
}
else
{
TxData(BufferB);                      // Transmit SPI data in DMA RAM
                                      // Secondary buffer
}
BufferCount ^= 1;
IFS0bits.DMA0IF = 0;                  // Clear the DMA0 Interrupt flag
}

void __attribute__((__interrupt__)) _DMA1Interrupt(void)
{
static unsigned int BufferCount = 0;  // Keep record of the buffer
                                      // that contains RX data
if(BufferCount == 0)
{
ProcessRxData(BufferA);               // Process received SPI data in
                                      // DMA RAM Primary buffer
}
else
{
ProcessRxData(BufferB);               // Process received SPI data in
                                      // DMA RAM Secondary buffer
}
BufferCount ^= 1;
IFS0bits.DMA1IF = 0;                  // Clear the DMA1 Interrupt flag
}
```

18.5.2 SPI and DMA with Null Data Write Mode

When the SPI is configured in Master mode, and only received data is of interest, some data must be written to the SPI Transmit buffer in order to start the SPI clock and receive the external data. For this situation, use Null Data Write mode of the DMA. For more information on DMA Null Data Write mode, refer to **Section 22. "Direct Memory Access (DMA)"** (DS70348).

18.6 SPI OPERATION IN POWER-SAVING MODES

The dsPIC33E/PIC24E family of devices has three power modes, which consist of normal (Full-Power) mode, and two power-saving modes invoked by the PWRSAV instruction. Depending on the SPIx mode selected, entry into a power-saving mode may also affect the operation of the module.

18.6.1 Sleep Mode

When the device enters Sleep mode, the system clock is disabled. The consequences of entering Sleep mode depend on which mode (Master or Slave) the module is configured for at the time Sleep mode is invoked.

18.6.1.1 MASTER MODE OPERATION

The effects of entering Sleep mode when the SPIx module is configured for Master operation are as follows:

- The Baud Rate Generator (BRG) in the SPIx module stops and is reset
- The transmitter and receiver stop in Sleep mode. The transmitter or receiver does not continue with a partially completed transmission at wake-up.
- If the SPIx module enters Sleep mode in the middle of a transmission or reception, the transmission or reception is aborted. Because there is no automatic way to prevent an entry into Sleep mode if a transmission or reception is pending, the user application must synchronize entry into Sleep mode with the SPIx module operation to avoid aborted transmissions.

18.6.1.2 SLAVE MODE OPERATION

As the clock pulses at the SCKx pin are externally provided for Slave mode, the module continues to function in Sleep mode. It completes any transactions during the transition into Sleep mode. On completion of a transaction, the SPIRBF flag is set. Consequently, the SPIxIF bit is set.

If SPIx interrupts are enabled (SPIxIE = 1), the device wakes up from Sleep. If the SPIx interrupt priority level is greater than the present CPU priority level, code execution resumes at the SPIx interrupt vector location. Otherwise, code execution continues with the instruction following the PWRSAV instruction that previously invoked Sleep mode. The module is not Reset on entering Sleep mode if it is operating as a slave device.

The register contents are not affected when the SPIx module is going into, or coming out of, Sleep mode.

18.6.2 Idle Mode

When the device enters Idle mode, the system clock sources remain functional. The SPISIDL bit (SPIxSTAT<13>) determines whether the module stops in Idle mode or continues to operate in Idle mode.

If SPISIDL = 1, the SPIx module stops communication on entering Idle mode. It operates in the same manner as it does in Sleep mode. If SPISID = 0 (default selection), the module continues operation in Idle mode.

18.7 REGISTER MAP

A summary of the registers associated with the dsPIC33E/PIC24E family's Serial Peripheral Interface (SPI) module is provided in Table 18-3.

Table 18-3: SPI1 Register Map

SFR Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
SPIxSTAT	SPIEN	—	SPISIDL	—	—	SPIBEC<2:0>			SRMPT	SPIROV	SRXMPT	SISEL<2:0>			SPITBF	SPIRBF	0000
SPIxCON1	—	—	—	DISSCK	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE<2:0>			PPRE<1:0>		0000
SPIxCON2	FRMEN	SPIFSD	FRMPOL	—	—	—	—	—	—	—	—	—	—	—	FRMDLY	SPIBEN	0000
SPIxBUF	SPIx Transmit and Receive Buffer Register																0000

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal

18.8 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the dsPIC33E/PIC24E device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Serial Peripheral Interface (SPI) module are:

Title	Application Note #
Interfacing Microchip's MCP41XXX and MCP42XXX Digital Potentiometers to a PIC® Microcontroller	AN746
Interfacing Microchip's MCP3201 Analog-to-Digital Converter to the PIC® Microcontroller	AN719

Note: Please visit the Microchip web site (www.microchip.com) for additional Application Notes and code examples for the dsPIC33E/PIC24E family of devices.

18.9 REVISION HISTORY

Revision A (December 2008)

This is the initial release of this document.

Revision B (June 2010)

This revision includes the following changes:

- Changed the document name from dsPIC33E Family Reference Manual to dsPIC33E/PIC24E Family Reference Manual
- All references to dsPIC33E in the document have been updated to dsPIC33E/PIC24E.
- Notes:
  - Added a shaded note at the beginning of the section, which provides information on complimentary documentation.
- Removed the watermark “Untested Code - For Information Purposes Only” from all code examples
- Changed “SPIx Slave, Framed Slave” to “SPIx Master/Framed Master Mode” in the dsPIC33E block in Figure 18-9.
- Changed “SPIx Slave, Framed Slave” to “SPIx Slave/Framed Master” in the dsPIC33E block in Figure 18-15.
- Changed (SPIx Master, Framed Slave) to (SPIx Slave/Framed Slave) in the dsPIC33E block in Figure 18-16.
- Minor changes to the text and formatting have been incorporated through the document.

NOTES:

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

ISBN: 978-1-60932-271-7

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-6578-300
Fax: 886-3-6578-370

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/05/10