

---

## Section 18. Serial Peripheral Interface (SPI)

---

### HIGHLIGHTS

This section of the manual contains the following major topics:

|      |  |       |
|------|--|-------|
| 18.1 | Introduction .....   | 18-2  |
| 18.2 | SPI Registers .....  | 18-3  |
| 18.3 | Modes of Operation .....                                     | 18-7  |
| 18.4 | Master Mode Clock Frequency .....                            | 18-20 |
| 18.5 | SPI Operation with DMA .....                                 | 18-21 |
| 18.6 | Operation in Power-Saving Modes .....                        | 18-24 |
| 18.7 | Special Function Registers Associated with SPI Modules ..... | 18-25 |
| 18.8 | Related Application Notes.....                               | 18-26 |
| 18.9 | Revision History .....                                       | 18-27 |

**Note:** This family reference manual section is meant to serve as a complement to device data sheets. Depending on the device variant, this manual section may not apply to all dsPIC33F/PIC24H 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. The peripheral devices can be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The SPI module is compatible with Motorola’s SPI and SIOP interfaces.

Depending on the variant, the dsPIC33F/PIC24H family offers one or two SPI modules on a single device. The modules, designated SPI1 and SPI2, are functionally identical. The SPI1 module is available on all devices, while the SPI2 module is available in many of the higher pin count packages.

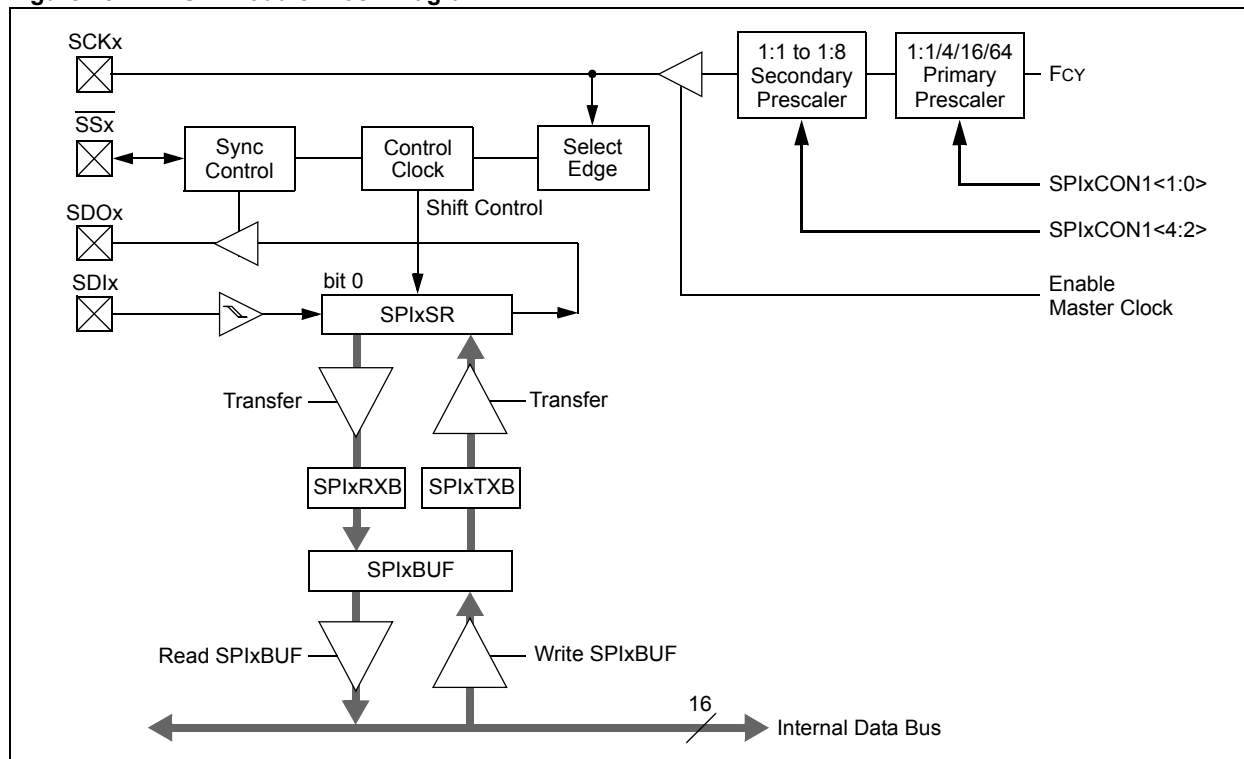
**Note:** In this section, the SPI modules are referred together as SPIx, or separately as SPI1 and SPI2. Special Function Registers (SFRs) follow a similar notation. For example, SPIxCON refers to the Control register for the SPI1 or SPI2 module.

The SPIx serial interface consists of four pins:

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

Figure 18-1 is a block diagram of the SPI module.

**Figure 18-1: SPI Module Block Diagram**



### 18.2 SPI REGISTERS

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

The SPIx Status and Control register (SPIxSTAT) indicates 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 enables and disables 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**

The SPIx Data Receive/Transmit Buffer register (SPIxBUF) is actually two separate internal registers: the Transmit Buffer (SPIxTXB) and the Receive Buffer (SPIxRXB). These two unidirectional, 16-bit registers share the SFR address of SPIxBUF. If a user application writes data to be transmitted to the SPIxBUF address, internally the data is written to the SPIxTXB register.

Similarly, when the user application reads the received data from SPIxBUF, internally the data is read from the SPIxRXB register.

The technique double-buffers transmit/receive operations and allows continuous data transfers in the background. 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.

# dsPIC33F/PIC24H Family Reference Manual

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

|        |     |         |     |     |     |     |       |
|--------|-----|---------|-----|-----|-----|-----|-------|
| R/W-0  | U-0 | R/W-0   | U-0 | U-0 | U-0 | U-0 | U-0   |
| SPIEN  | —   | SPISIDL | —   | —   | —   | —   | —     |
| bit 15 |     |         |     |     |     |     | bit 8 |

|       |        |     |     |     |     |        |        |
|-------|--------|-----|-----|-----|-----|--------|--------|
| U-0   | R/C-0  | U-0 | U-0 | U-0 | U-0 | R-0    | R-0    |
| —     | SPIROV | —   | —   | —   | —   | SPITBF | SPIRBF |
| bit 7 |        |     |     |     |     |        | bit 0  |

|                   |                      |
|-------------------|----------------------|
| <b>Legend:</b>    | C = Clearable bit    |
| R = Readable bit  | W = Writable bit     |
| -n = Value at POR | '1' = Bit is set     |
|                   | '0' = Bit is cleared |
|                   | x = Bit is unknown   |

- bit 15      **SPIEN:** SPIx Enable bit  
1 = Enables module and configures SCKx, SDOx, SDIx and  $\overline{SSx}$  as serial port pins  
0 = Disables module
- bit 14      **Unimplemented:** Read as '0'
- bit 13      **SPISIDL:** Stop in Idle Mode bit  
1 = Discontinue module operation when device enters Idle mode  
0 = Continue module operation in Idle mode
- bit 12-7    **Unimplemented:** Read as '0'
- bit 6      **SPIROV:** Receive Overflow Flag bit  
1 = A new byte/word is completely received and discarded. The user software has not read the previous data in the SPIxBUF register  
0 = No overflow has occurred
- bit 5-2    **Unimplemented:** Read as '0'
- bit 1      **SPITBF:** SPIx Transmit Buffer Full Status bit  
1 = Transmit not yet started, SPIxTXB is full  
0 = Transmit started, SPIxTXB is empty  
Automatically set in hardware when CPU writes SPIxBUF location, loading SPIxTXB.  
Automatically cleared in hardware when SPIx module transfers data from SPIxTXB to SPIxSR.
- bit 0      **SPIRBF:** SPIx Receive Buffer Full Status bit  
1 = Receive complete, SPIxRXB is full  
0 = Receive is not complete, SPIxRXB is empty  
Automatically set in hardware when SPIx transfers data from SPIxSR to SPIxRXB.  
Automatically cleared in hardware when core reads SPIxBUF location, reading SPIxRXB.

## Section 18. Serial Peripheral Interface (SPI)

**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 <sup>(1)</sup>       | CKE <sup>(2)</sup> |
| 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> <sup>(3)</sup> |        |        | PPRE<1:0> <sup>(3)</sup> |                    |
| 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 | <b>Unimplemented:</b> Read as '0'  |
| bit 12    | <b>DISSCK:</b> Disable SCKx pin bit (SPI Master mode only)<br>1 = Internal SPI clock is disabled, pin functions as I/O<br>0 = Internal SPI clock is enabled  |
| bit 11    | <b>DISSDO:</b> Disable SDOx pin bit<br>1 = SDOx pin is not used by the module; pin functions as I/O<br>0 = SDOx pin is controlled by the module  |
| bit 10    | <b>MODE16:</b> Word/Byte Communication Select bit<br>1 = Communication is word-wide (16 bits)<br>0 = Communication is byte-wide (8 bits)   |
| bit 9     | <b>SMP:</b> SPIx Data Input Sample Phase bit <sup>(1)</sup><br><u>Master mode:</u><br>1 = Input data sampled at end of data output time<br>0 = Input data sampled at middle of data output time<br><u>Slave mode:</u><br>SMP bit must be cleared when SPIx module is used in Slave mode. |
| bit 8     | <b>CKE:</b> SPIx Clock Edge Select bit <sup>(2)</sup><br>1 = Serial output data changes on transition from active clock state to Idle clock state (refer to bit 6)<br>0 = Serial output data changes on transition from Idle clock state to active clock state (refer to bit 6)          |
| bit 7     | <b>SSEN:</b> Slave Select Enable bit (Slave mode)<br>1 = $\overline{SSx}$ pin is used for Slave mode<br>0 = $\overline{SSx}$ pin is not used by the module. Pin controlled by port function  |
| bit 6     | <b>CKP:</b> Clock Polarity Select bit<br>1 = Idle state for clock is a high level; active state is a low level<br>0 = Idle state for clock is a low level; active state is a high level  |
| bit 5     | <b>MSTEN:</b> Master Mode Enable bit<br>1 = Master mode<br>0 = Slave mode  |
| bit 4-2   | <b>SPRE&lt;2:0&gt;:</b> Secondary Prescale bits (Master mode) <sup>(3)</sup><br>111 = Secondary prescale 1:1<br>110 = Secondary prescale 2:1<br>•<br>•<br>•<br>000 = Secondary prescale 8:1  |
| bit 1-0   | <b>PPRE&lt;1:0&gt;:</b> Primary Prescale bits (Master mode) <sup>(3)</sup><br>11 = Primary prescale 1:1<br>10 = Primary prescale 4:1<br>01 = Primary prescale 16:1<br>00 = Primary prescale 64:1   |

**Note 1:** The SMP bit must be set only after setting the MSTEN bit. The SMP bit remains cleared if MSTEN = 0.

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

**3:** Do not set Primary and Secondary Prescale bit values to 1:1.

# dsPIC33F/PIC24H Family Reference Manual

**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  | U-0   |
| —     | —   | —   | —   | —   | —   | FRMDLY | —     |
| 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 enabled (SSx pin is used as frame sync pulse input/output)  
 0 = Framed SPIx support 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 **Unimplemented**: This bit must not be set to '1' by the user application

**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
- Framed SPIx Modes
- SPIx Receive-only Operation
- SPIx Error Handling

### 18.3.1 8-Bit vs. 16-Bit Operation

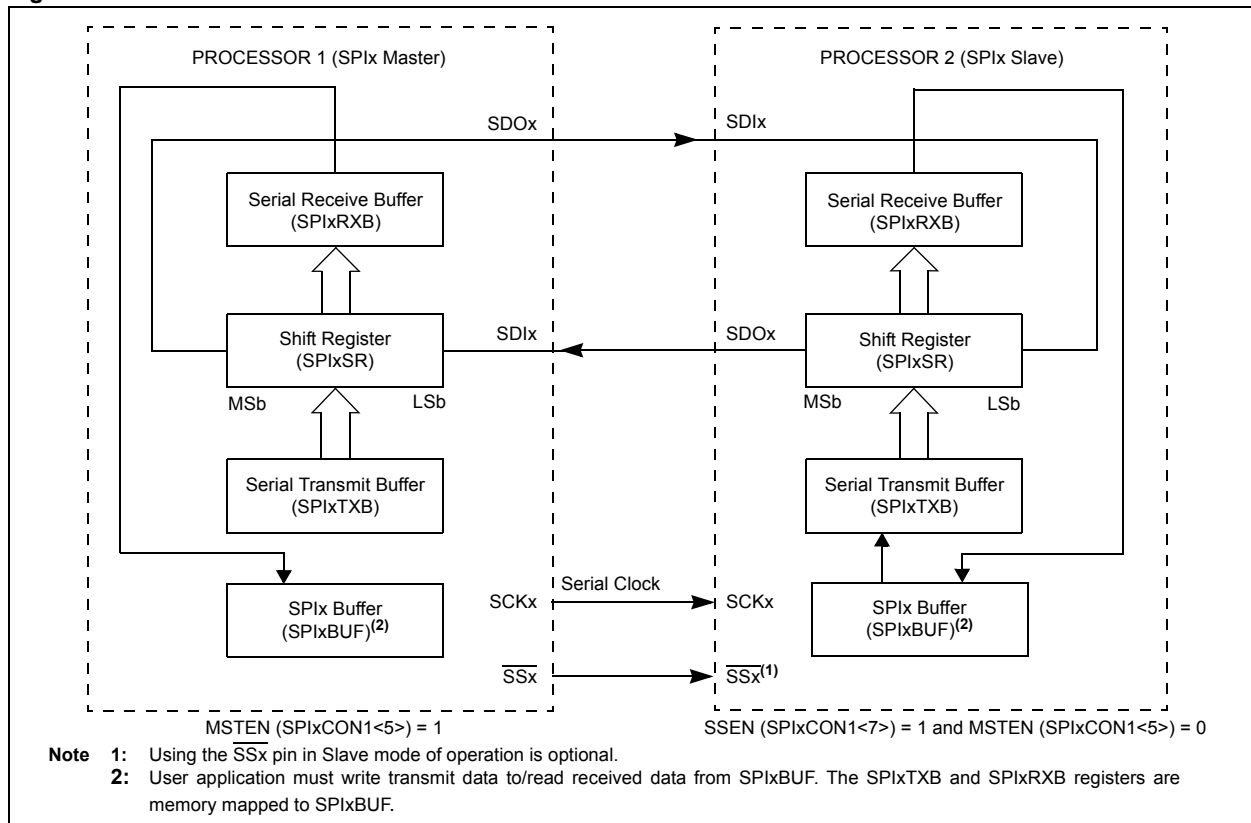
The Word/Byte Communication Select control bit (MODE16) in SPIx Control Register 1 (SPIxCON1<10>) allows the module to communicate in either 8-bit or 16-bit mode. The functionality is the same for each mode except for the number of bits that are received and transmitted. In this context:

- 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) 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>)
- When transmitting or receiving data, 8 clock pulses are required at the SCKx pin to shift data in/out in 8-bit mode. 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. The module configured as the Master module provides the serial clock and synchronization signals (as required) to the slave device. Figure 18-2 illustrates the connection of Master and Slave modules.

**Figure 18-2: SPI Master/Slave Connection**



## 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>) in SPIx Control Register 1 (SPIxCON1<1:0>) and the Secondary Prescale bits (SPRE<2:0>) in SPIxCON1<4:2>. The serial clock is output via the SCKx pin to slave devices. The clock pulses are only generated when there is data to be transmitted. For further information, see **18.4 “Master Mode Clock Frequency”**. The CKP and CKE 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.

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 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 Shift Register, SPIxSR, to the SDOx pin and simultaneously shift the data at the SDIx pin into SPIxSR.
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:
    - SPI1IF is set in Interrupt Flag Status Register 0 (IFS0<10>)
    - SPI2IF is set in Interrupt Flag Status Register 2 (IFS2)These interrupts are enabled by setting the corresponding interrupt enable bits:
    - SPI1IE is enabled in Interrupt Enable Control Register 0 (IEC0<10>)
    - SPI2IE is enabled in Interrupt Enable Control Register 2 (IEC2<1>)The SPIxIF flags are not cleared automatically by the hardware.
  - b) When the ongoing transmit and receive operations are completed, the content of the SPIx Shift register (SPIxSR) is moved to the SPIx Receive Buffer register (SPIxRXB).
  - c) 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 code, the hardware clears the SPIRBF bit.
5. If the SPIRBF bit is set (receive buffer is full) when the SPIx module needs to transfer data from SPIxSR to SPIxRXB, the module sets the Receive Overflow Flag bit (SPIROV) in the SPIx Status and Control register (SPIxSTAT<6>), indicating an overflow condition.
6. Data to be transmitted can be written to SPIxBUF by the user software at any time as long as the SPITBF bit (SPIxSTAT<1>) is clear. The write can occur while SPIxSR is shifting out the previously written data, allowing continuous transmission.

|   |
|---|
| <b>Note:</b> 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

Follow these procedures to set up the SPIx module for the Master mode of operation:

1. If using interrupts, configure the Interrupt controller:
  - a) Clear the SPIx Event Interrupt Flag Status bit (SPIxIF) in the respective Interrupt Flag Status register (IFS0<10> or IFS2<1>) in the Interrupt controller.
  - b) Set the SPIx Event Interrupt Enable bit (SPIxIE) in the respective Interrupt Event Control register (IEC0<10> or IEC2<1>) in the Interrupt controller.
  - c) Write the SPIx Event Interrupt Priority bits (SPIxIP) in the respective Interrupt Priority Control register (IPC2<10-8> or IPC8<6-4>) in the Interrupt controller register 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 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 soon after the data is written to the SPIxBUF register.

Example 18-1 is a code snippet that shows SPI register configuration for Master mode.

### Example 18-1: SPI Configuration – Master Mode

```
/* The following code shows the 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
```

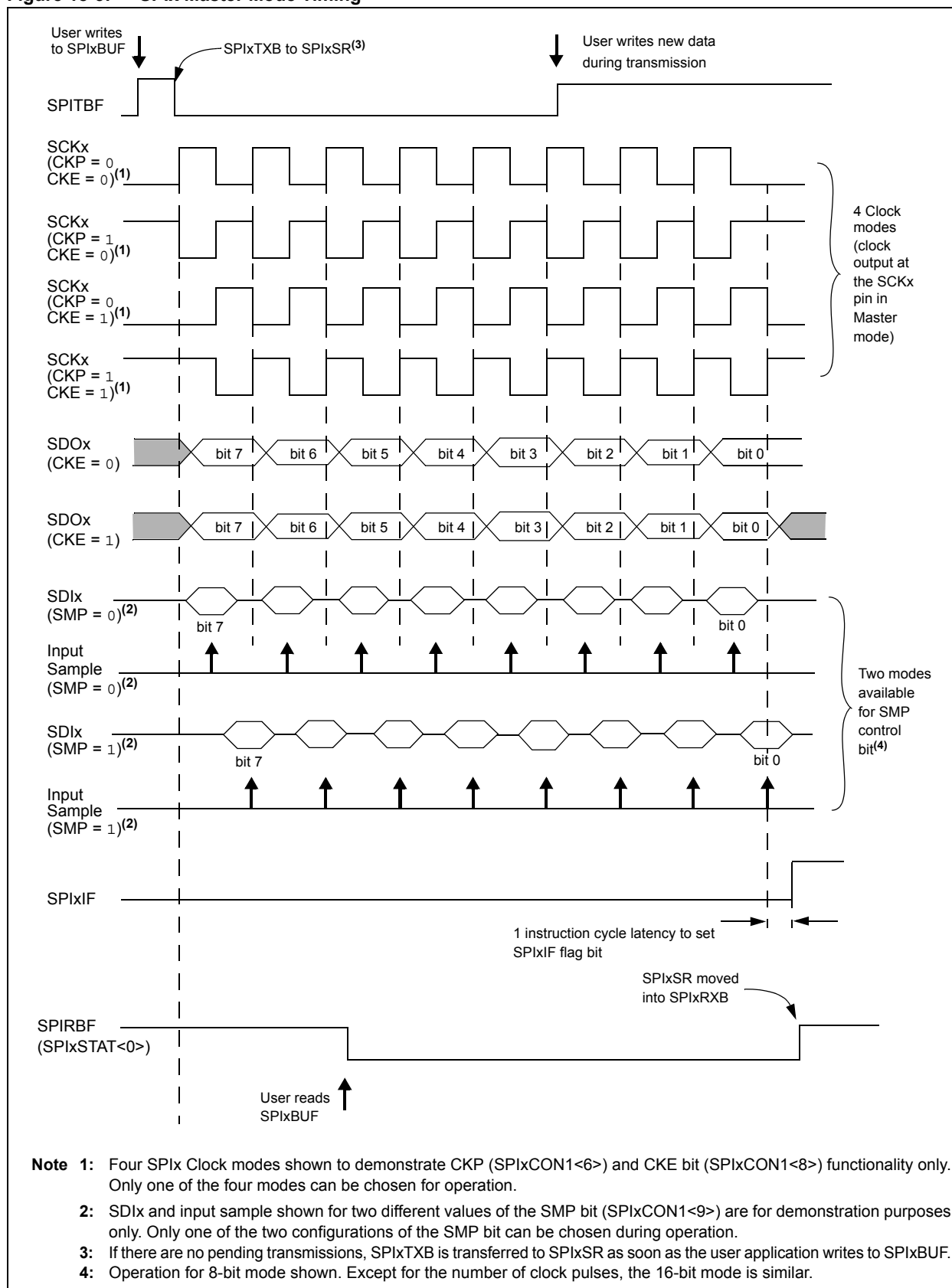
## 18.3.2.1.2 External Clocking in Master Mode

In Master mode, the module can also be configured to operate with an external data clock. SPIx clock operation is controlled by the Disable SCKx Pin bit (DISSCK) in SPIx Control Register 1 (SPIxCON1<12>).

When this bit is set, the internal data clock is disabled. Data is transferred when external clock pulses are presented on the SCKx pin. All other aspects of Master mode operation are the same.

**Note:** The DISSCK bit is available only in SPI Master mode.

**Figure 18-3: SPIx Master Mode Timing**



## 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 bit (CKP) and SPIx Clock Edge Select bit (CKE) in the SPIx Control Register 1 (SPIxCON1) determine the edge of the clock pulse on which the data transmission occurs. Both data to be transmitted and data that is received are written into or read from the SPIxBUF register. The remaining operation of the module is identical to that in the Master mode.

### 18.3.2.2.1 Slave Mode Setup 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 Event Interrupt Flag Status bit (SPIxIF) in the respective Interrupt Flag Status register (IFS0<10> or IFS2<1> in the Interrupt controller).
  - b) Set the SPIx Event Interrupt Enable bit (SPIxIE) in the respective IECn register.
  - c) Write the SPIx Event Interrupt Priority bits (SPIxIP) in the respective IPCn register to set the interrupt priority.
3. Configure the SPIxCON1 register:
  - a) Clear the Master Mode Enable bit (MSTEN) in the SPIxCON1 register (SPIxCON1<5> = 0).
  - b) Clear the SPIx Data Input Sample Phase bit (SMP) in the SPIxCON1 register (SPIxCON1<9> = 0).
  - c) If the SPIx Clock Edge Select bit (CKE) is set, set the Slave Select Enable bit (SSEN) to enable the SSx pin (SPIxCON1<7> = 1).
4. Configure the SPIxSTAT register:
  - a) Clear the Receive Overflow Flag bit (SPIROV) in the SPIxSTAT register (SPIxSTAT<6> = 0).
  - b) Set the SPIx Enable bit (SPIEN) in the SPIxSTAT register (SPIxSTAT<15> = 1) to enable SPIx operation.

Example 18-2 is a code snippet that shows the SPI register configuration for Slave mode.

#### Example 18-2: SPI Configuration – Slave Mode

```
/* The following code shows the 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  $\overline{\text{SSx}}$  pin enables the Synchronous Slave mode. If the Slave Select Enable bit (SSEN) is set to '1', the transmission and reception are enabled in Slave mode, only if the  $\overline{\text{SSx}}$  pin is driven to a low state (see Figure 18-5). The port output or other peripheral outputs must not be driven in order to allow the  $\overline{\text{SSx}}$  pin to function as an input. If the SSEN bit is set and the  $\overline{\text{SSx}}$  pin is driven high, the SDOx pin is no longer driven and tri-states even if the module is in the middle of a transmission.

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

|  |
|--|
| <b>Note:</b> To meet module timing requirements, the $\overline{\text{SSx}}$ pin must be enabled in Slave mode when CKE = 1 (see Figure 18-6 for details). |
|--|

## 18.3.2.2.3 SPITBF Status Flag Operation

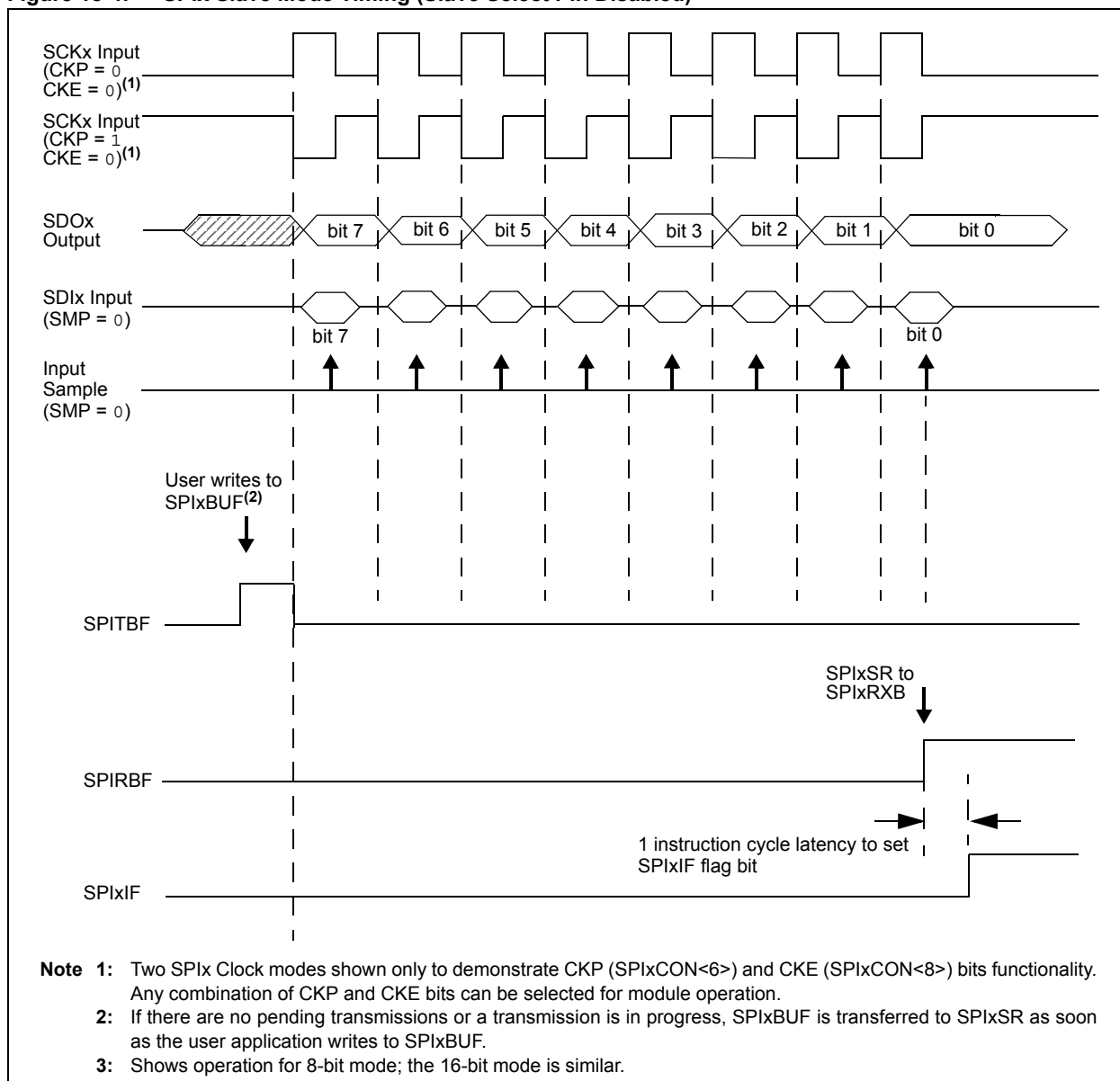
The Transmit Buffer Full Status bit (SPITBF) in the SPIxSTAT register functions differently in Slave mode of operation than it does in Master mode.

If SSEN is cleared (= 0), the SPITBF bit is set when the SPIxBUF is loaded by the user application. It is cleared when the module transfers SPIxTXB to SPIxSR. This is similar to the SPITBF bit function in Master mode.

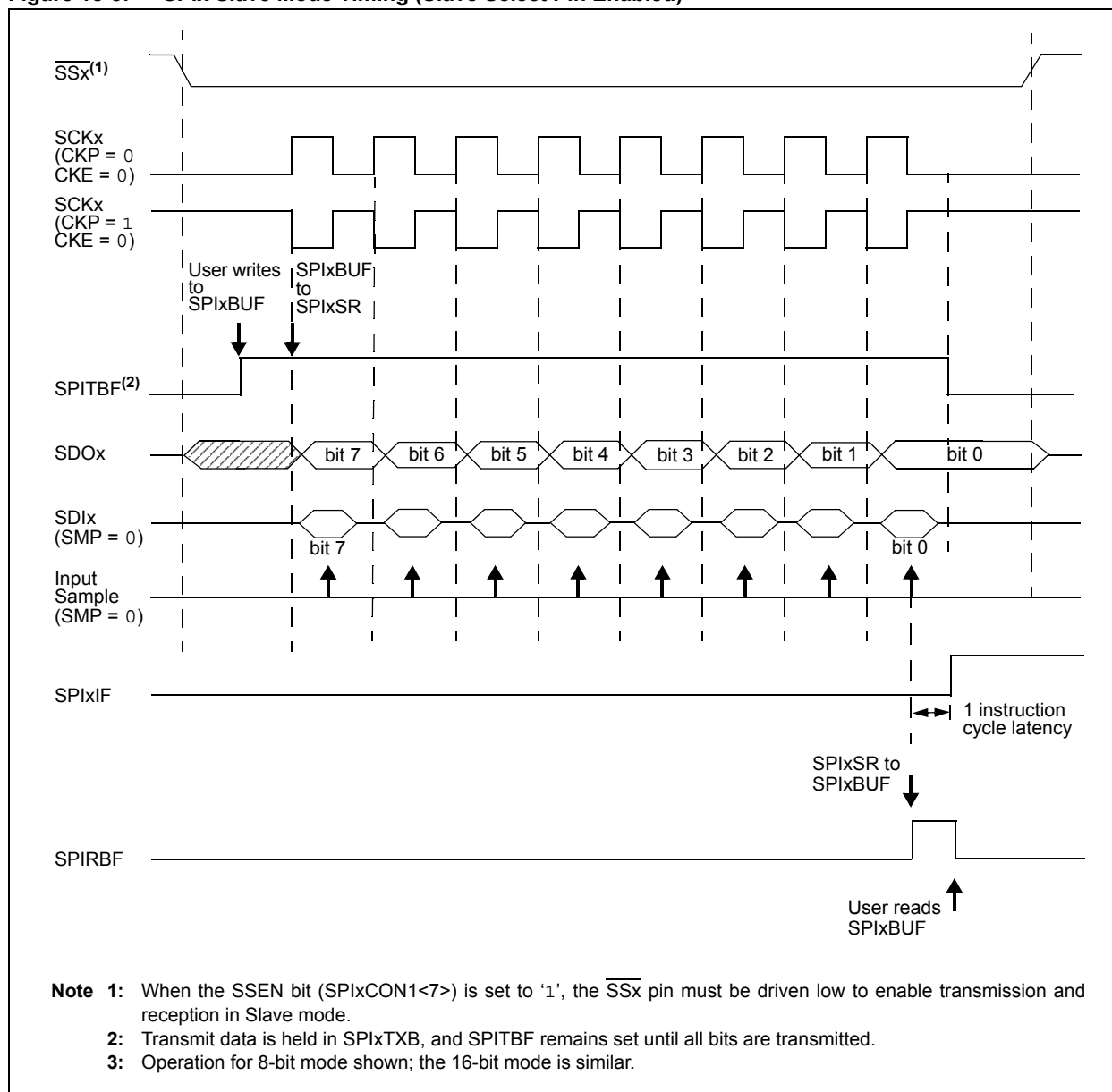
If SSEN is set (= 1), the SPITBF bit is set when the SPIxBUF is loaded by the user application. However, it is cleared only when the SPIx module completes data transmission. A transmission is aborted when the  $\overline{\text{SSx}}$  pin goes high, and may be retried later. Each data word is held in SPIxTXB until all bits are transmitted to the receiver.

## Section 18. Serial Peripheral Interface (SPI)

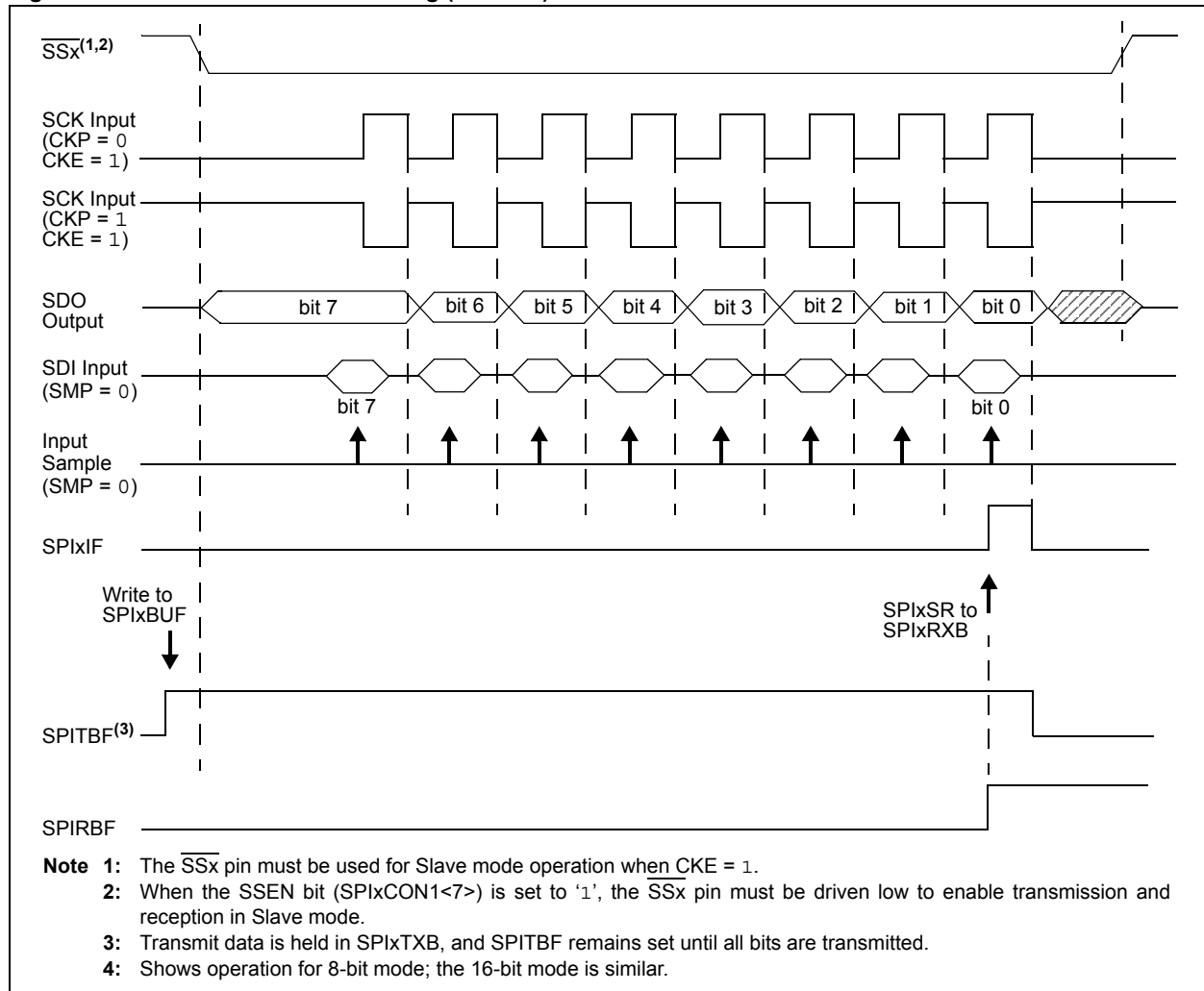
**Figure 18-4: SPIx Slave Mode Timing (Slave Select Pin Disabled)<sup>(3)</sup>**



**Figure 18-5: SPIx Slave Mode Timing (Slave Select Pin Enabled)<sup>(3)</sup>**



**Figure 18-6: SPIx Slave Mode Timing (CKE = 1)<sup>(4)</sup>**



## 18.3.3 Framed SPIx Modes

The SPI module supports a basic framed SPIx protocol while operating in Master or Slave mode. Four control bits configure framed SPIx operation:

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

**Note:** The  $\overline{\text{SSx}}$  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 the following four framed SPIx configurations available to the user:

- 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 = 1 and MSTEN = 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. The CKE bit is not used for the Framed SPI modes and should be programmed to '0' by the user software.

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.

### 18.3.3.1 FRAME MASTER AND FRAME SLAVE MODES

When SPIFSD = 0, in the SPIx Control Register 2 (SPIxCON2<14>), the SPIx module is in the Framed Master mode of operation. In this mode, the frame synchronization pulse is initiated by the module when the user software writes the transmit data to the SPIxBUF location (thus writing the SPIxTXB register with transmit data). At the end of frame synchronization pulse, the SPIxTXB is transferred to the SPIxSR 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 and data transmission/reception begins. The user application must ensure that the correct data is loaded into the SPIxBUF 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 SPIxBUF. If no write is performed, the current contents of the SPIxTXB are transmitted.

### 18.3.3.2 SPIx MASTER/FRAMED MASTER MODE

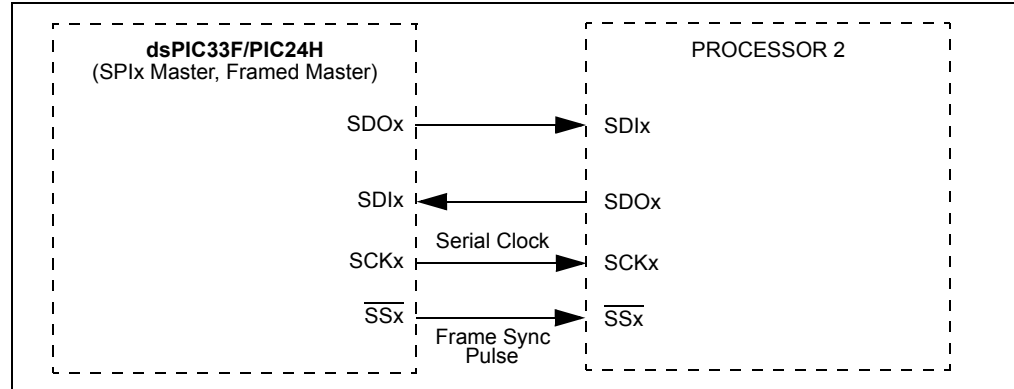
In the SPIx Master/Framed Master mode, the SPIx module generates both the clock and frame synchronization signals, as shown in Figure 18-7. This configuration is enabled by setting the MSTEN and FRMEN bits to '1' and the SPIFSD bit to '0'.

In this mode, the serial clock is outputted continuously at the SCKx pin, regardless of whether the module is transmitting. When SPIxBUF is written, the  $\overline{\text{SSx}}$  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 18-8 illustrates, if the FRMDLY control bit (SPIxCON2<1>) is cleared, the frame synchronization pulse precedes the data transmission. Figure 18-9 illustrates, if FRMDLY bit 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 pin.

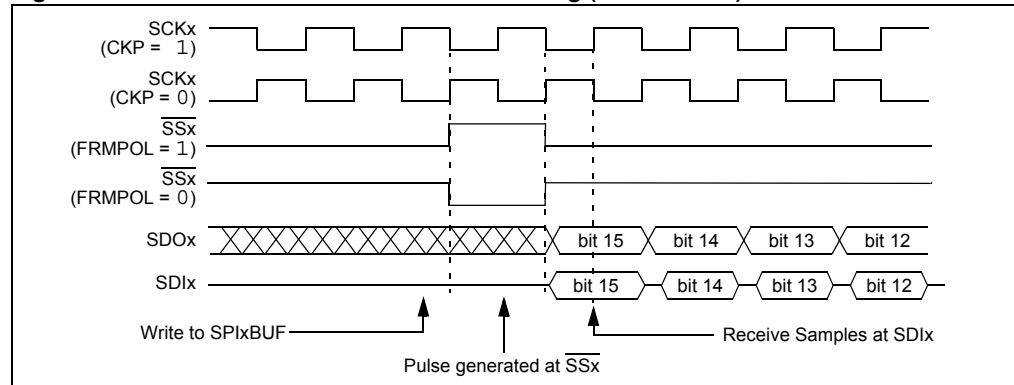


## Section 18. Serial Peripheral Interface (SPI)

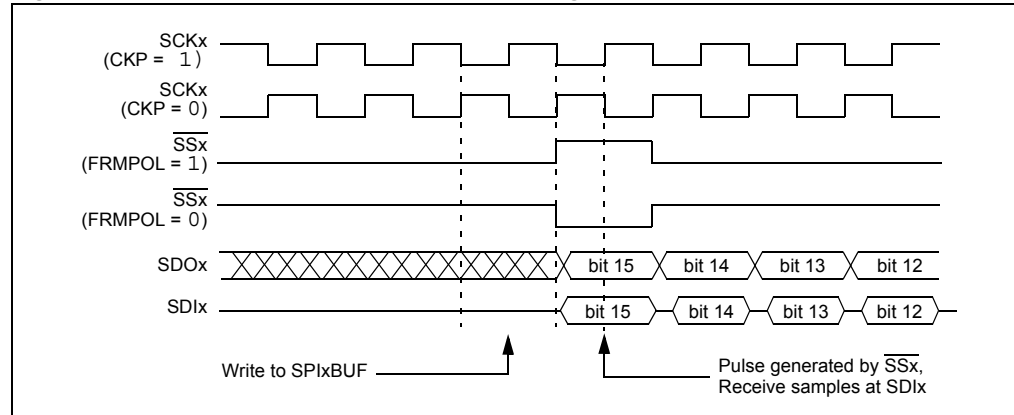
**Figure 18-7: SPIx Master/Framed Master Connection Diagram**



**Figure 18-8: SPIx Master/Framed Master Timing (FRMDLY = 0)**



**Figure 18-9: SPIx Master/Framed Master Timing (FRMDLY = 1)**

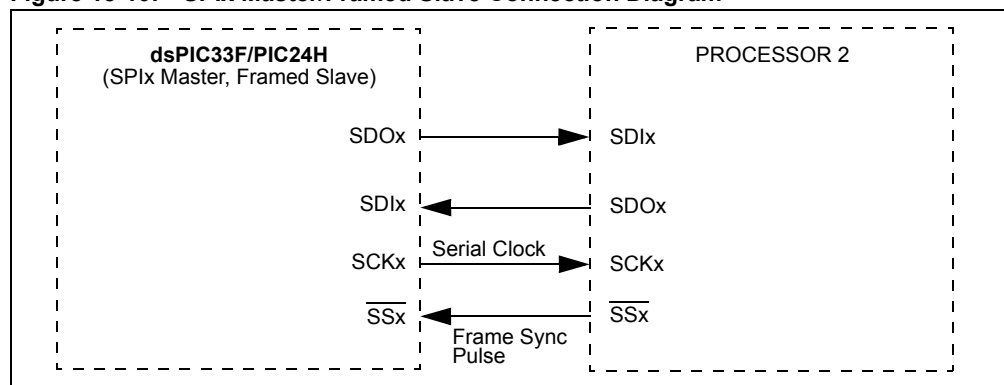


### 18.3.3.3 SPIx MASTER/FRAMED SLAVE MODE

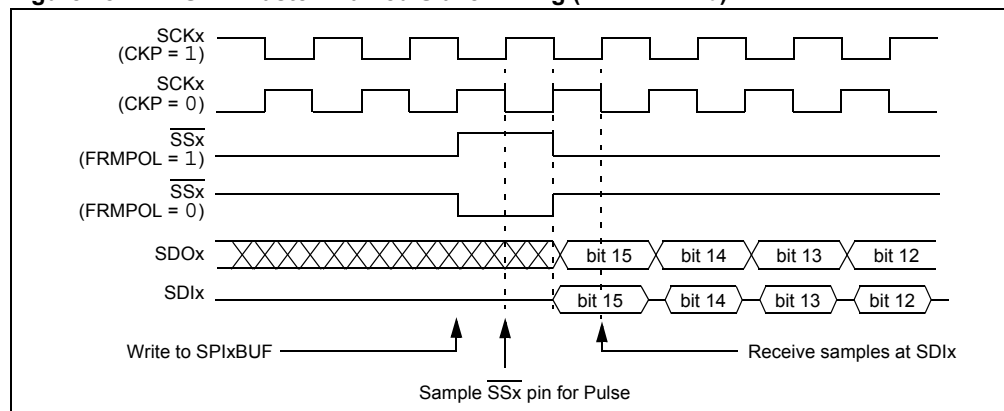
In the SPIx Master/Framed Slave mode, the module generates the clock signal but uses the Slave module frame synchronization signal for data transmission (see Figure 18-10). It is enabled by setting the MSTEN, FRMEN and SPIFSD bits to '1'.

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 for transmission before the signal is received at the  $\overline{SSx}$  pin.

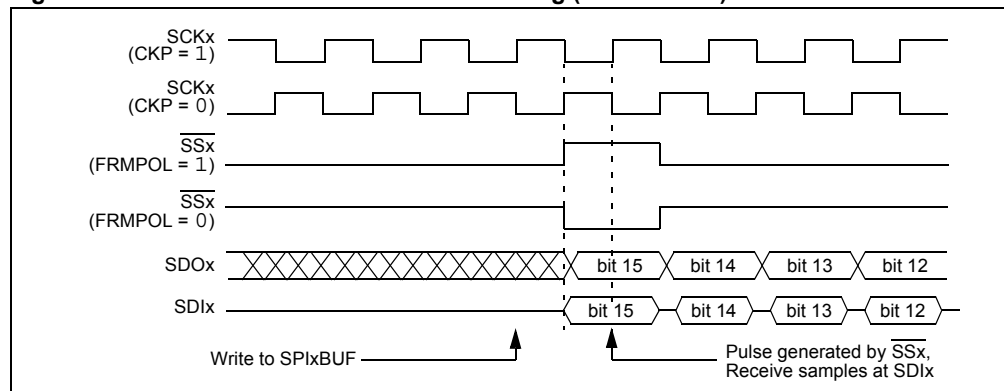
**Figure 18-10: SPIx Master/Framed Slave Connection Diagram**



**Figure 18-11: SPIx Master/Framed Slave Timing (FRMDLY = 0)**



**Figure 18-12: SPIx Master/Framed Slave Timing (FRMDLY = 1)**

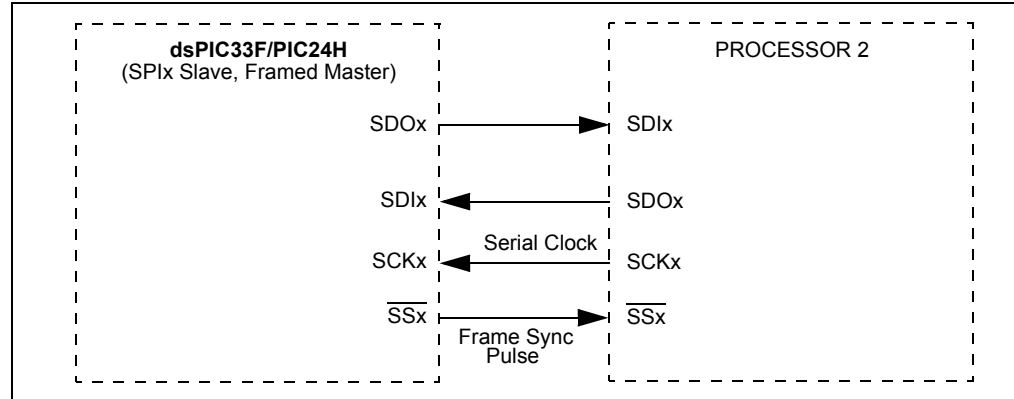


## 18.3.3.4 SPIx SLAVE/FRAMED MASTER MODE

In the SPIx Slave/Framed Master mode, the module acts as the SPIx slave and takes its clock from the other SPIx module; however, it produces frame synchronization signals to control data transmission (see Figure 18-13). It is enabled by setting the MSTEN bit to '0', the FRMEN bit to '1' and the SPIFSD bit to '0'.

The input SPIx clock is continuous in Slave mode. The  $\overline{SSx}$  pin is an output when the SPIFSD bit is low. Therefore, when the SPIxBUF is written, the module drives the  $\overline{SSx}$  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 18-13: SPIx Slave/Framed Master Connection Diagram**

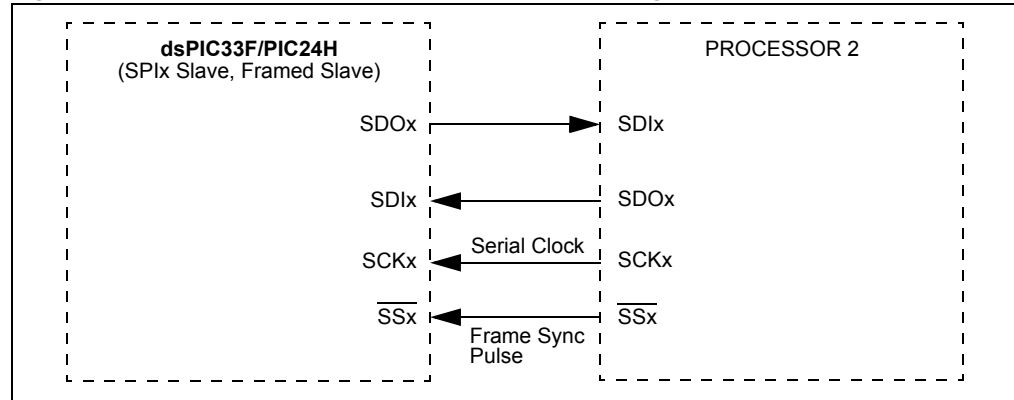


## 18.3.3.5 SPIx SLAVE/FRAMED SLAVE MODE

In the SPIx Slave/Framed Slave mode, the module obtains both its clock and frame synchronization signal from the Master module (see Figure 18-14). It is enabled by setting MSTEN to '0', FRMEN to '1' and SPIFSD 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 18-14: SPIx Slave/Framed Slave Connection Diagram**



## 18.3.4 SPIx Receive-Only Operation

Setting the DISSDO control bit (SPIxCON1<11>) disables the 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.

## 18.3.5 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 Event Interrupt Flag Status bit (SPIxIF) is set when the SPIx Receive Buffer Full Status bit (SPIRBF) or SPIx Transmit Buffer Full Status bit (SPITBF) in SPIxSTAT register 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 IECn Control register.

In addition, SPIx Error Interrupt Flag bit (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 IECn 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 bits (PPRE<1:0>) and the secondary prescaler, specified by the Secondary Prescale bits (SPRE<2:0>) in the SPIxCON1 register. 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 mode.

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**

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

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

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

**Table 18-1: Sample SCKx Frequencies<sup>(1)</sup>**

| 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 = 5 MHz                |      |                              |         |        |         |        |
| Primary Prescaler Settings | 1:1  | 5000                         | 2500    | 1250   | 833     | 625    |
|                            | 4:1  | 1250                         | 625     | 313    | 208     | 156    |
|                            | 16:1 | 313                          | 156     | 78     | 52      | 39     |
|                            | 64:1 | 78                           | 39      | 20     | 13      | 10     |

**Note 1:** SCKx frequencies are shown in kHz.

## 18.5 SPI OPERATION WITH DMA

On some of the dsPIC33F/PIC24H devices, the Direct Memory Access (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. For more information on the DMA module, refer to **Section 22. “Direct Memory Access (DMA)”** (DS70182).

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)  |

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 the 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 the 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 the 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 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

### Example 18-3: SPI Transmission and Reception with DMA

#### Setup for SPI1 Master mode:

```
// Interrupt Controller Settings
IFS0bits.SPI1IF = 0;

// SPI1CON1 Register Settings
// Communication is word-wide (16 bits)
SPI1CON1bits.MODE16 = 1;
// Master mode Enabled
SPI1CON1bits.MSTEN = 1;

// SPI1CON2 Register Settings
// Framed mode Disabled
SPI1CON2bits.FRMEN = 0;

// SPI1STAT Register Settings
// Continue module operation in Idle mode
SPI1STATbits.SPISIDL = 0;
// Buffer Length = 1 Word
SPI1STATbits.BUFELM = 0;
// No Receive Overflow has occurred
SPI1STATbits.SPIROV = 0;
// Enable SPI module
SPI1STATbits.SPIEN = 1;

// 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;
DMAC0 = 0;
DMA0CON = 0x2002;
DMA0STA = __builtin_dmaoffset(TxBufferA);
DMA0STB = __builtin_dmaoffset(TxBufferB);
DMA0PAD = (volatile unsigned int) &SPI1BUF;
DMA0CNT = 15;
DMA0REQ = 0x000A;
DMA0CONbits.CHEN = 1;
```

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

### 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;
```

### SPI and DMA Interrupt Handler:

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

void __attribute__((__interrupt__)) _DMA0Interrupt(void)
{
    static unsigned int BufferCount = 0; // Keep record of which buffer
                                         // 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 which buffer
                                         // 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 to start the SPI clock and receive the external data. For this situation, use the Null Data Write mode of the DMA. For more information on the DMA Null Data Write mode, refer to **Section 22. “Direct Memory Access (DMA)”** (DS70182).

## 18.6 OPERATION IN POWER-SAVING MODES

The dsPIC33F/PIC24H family of devices have three Power modes, normal (Full-Power) mode and two Power-Saving modes invoked by the `PWRSV` 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 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. 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. Since there is no automatic way to prevent an entry into Sleep mode if a transmission or reception is pending, the user software must synchronize entry into Sleep with SPIx module operation to avoid aborted transmissions

#### 18.6.1.2 SLAVE MODE OPERATION

Since the clock pulses at SCKx are externally provided for Slave mode, the module continues to function in Sleep mode. It completes any transactions during the transition into Sleep. 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 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 `PWRSV` 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 SPIIDL bit (`SPIxSTAT<13>`) selects whether the module stops or continues functioning in Idle mode.

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



## 18.7 SPECIAL FUNCTION REGISTERS ASSOCIATED WITH SPI MODULES

**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 |
|----------|---|--------|---------|--------|--------|--------|-------|-------|-------|--------|-------|-----------|-------|-------|-----------|--------|------------|
| SPI1STAT | SPIEN                                     | —      | SPISIDL | —      | —      | —      | —     | —     | —     | SPIROV | —     | —         | —     | —     | SPITBF    | SPIRBF | 0000       |
| SPI1CON1 | —   | —      | —       | DISSCK | DISSDO | MODE16 | SMP   | CKE   | SSEN  | CKP    | MSTEN | SPRE<2:0> |       |       | PPRE<1:0> |        | 0000       |
| SPI1CON2 | FRMEN                                     | SPIFSD | FRMPOL  | —      | —      | —      | —     | —     | —     | —      | —     | —         | —     | —     | FRMDLY    | —      | 0000       |
| SPI1BUF  | SPI1 Transmit and Receive Buffer Register |        |         |        |        |        |       |       |       |        |       |           |       |       |           |        | 0000       |

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

**Table 18-4: SPI2 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 |
|----------|---|--------|---------|--------|--------|--------|-------|-------|-------|--------|-------|-----------|-------|-------|-----------|--------|------------|
| SPI2STAT | SPIEN                                     | —      | SPISIDL | —      | —      | —      | —     | —     | —     | SPIROV | —     | —         | —     | —     | SPITBF    | SPIRBF | 0000       |
| SPI2CON1 | —   | —      | —       | DISSCK | DISSDO | MODE16 | SMP   | CKE   | SSEN  | CKP    | MSTEN | SPRE<2:0> |       |       | PPRE<1:0> |        | 0000       |
| SPI2CON2 | FRMEN                                     | SPIFSD | FRMPOL  | —      | —      | —      | —     | —     | —     | —      | —     | —         | —     | —     | FRMDLY    | —      | 0000       |
| SPI2BUF  | SPI2 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 dsPIC33F/PIC24H device families, 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              |

|   |
|---|
| <b>Note:</b> For additional application notes and code examples for the dsPIC33F/PIC24H family of devices, visit the Microchip web site ( <a href="http://www.microchip.com">www.microchip.com</a> ). |
|---|

### 18.9 REVISION HISTORY

#### Revision A (April 2007)

This is the initial released version of this document.

#### Revision B (January 2008)

This revision incorporates the following content updates:

- Registers:
  - SPIxCON1: SPIx Control Register 1 register (see Register 18-2) – Note 1 has been added.
- Additional minor corrections such as language and formatting updates are incorporated throughout the document.

#### Revision C (January 2010)

This revision incorporates the following updates:

- Added a shaded note at the beginning of the section, which provides information on complementary documentation
- Renamed the Family Reference Manual from dsPIC33F to dsPIC33F/PIC24H
- All references to dsPIC33F in the document are updated to dsPIC33F/PIC24H
- Figures:
  - In Figure 18-3, the CKE bit value has been changed from '0' to '1' for the fourth Clock mode
  - The text in Figure 18-7, (SPIx Slave, Framed Slave) has been replaced by (SPIx Master, Framed Master)
  - The text in Figure 18-13, (SPIx Slave/Framed Slave) has been replaced by (SPIx Slave, Framed Master)
  - The text in Figure 18-14, (SPIx Master/Framed Slave) has been replaced by (SPIx Slave, Framed Slave)
- Added Note 3 to Register 18-2
- Additional minor corrections such as language and formatting updates are incorporated throughout the document

NOTES: