



Serial Peripheral Interface (SPI) with Audio Codec Support

HIGHLIGHTS

This section of the manual contains the following topics:

1.0	Introduction	2
2.0	Status and Control Registers	6
3.0	Modes of Operation	20
4.0	Audio Protocol Interface Mode.....	35
5.0	Interrupts.....	55
6.0	Operation in Power-Saving and Debug Modes.....	57
7.0	Effects of Various Resets	58
8.0	Peripherals Using SPI Modules	58
9.0	Related Application Notes.....	59
10.0	Revision History	60

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 dsPIC33/PIC24 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>

1.0 INTRODUCTION

The Serial Peripheral Interface (SPI) module is a synchronous serial interface useful for communicating with external peripherals and other microcontroller devices. These peripheral devices may be a serial EEPROM, shift register, display driver, Analog-to-Digital Converter (ADC) or an audio codec. The dsPIC33/PIC24 family SPI module is compatible with Motorola® SPI and SIOP interfaces. [Figure 1-1](#) shows a block diagram of the SPI module.

Some of the key features of this module are:

- Master and Slave modes support
- Four different clock formats
- Framed SPI protocol support
- Standard and Enhanced Buffering modes (Enhanced Buffering mode is not available on all devices)
- User-configurable 8-bit, 16-bit and 32-bit data width
- Two separate shift registers for transmission and reception
- SPIx receive and transmit buffers are FIFO buffers, which are 4/8/16-deep in Enhanced Buffering mode
- User-configurable variable data width, from 2 to 32-bit
- Programmable interrupt event on every 8-bit, 16-bit and 32-bit data transfer
- Audio Protocol Interface mode

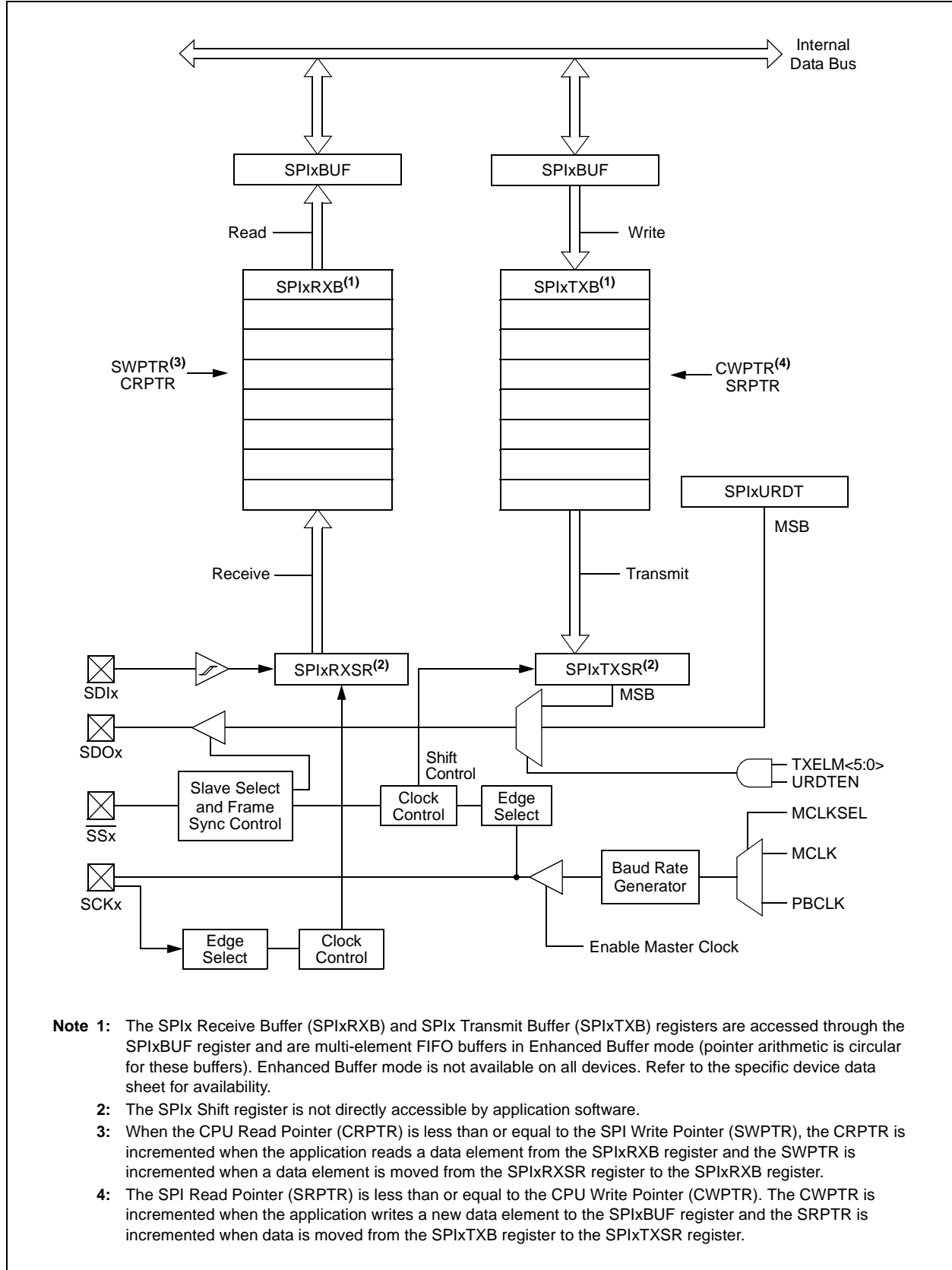
Some dsPIC33/PIC24 devices support audio codec serial protocols, such as Inter-IC Sound (I²S), Left Justified, Right Justified and PCM/DSP modes for 16, 24 and 32-bit audio data. Refer to the specific device data sheet for availability of these features.

The SPI serial interface consists of four pins:

- SDIx: Serial Data Input
- SDOx: Serial Data Output
- SCKx: Shift Clock Input or Output
- SSx: Active-Low Slave Select or Frame Synchronization I/O Pulse

Serial Peripheral Interface (SPI) with Audio Codec Support

Figure 1-1: SPLx Module Block Diagram



1.1 Normal Mode SPI Operation

In Normal mode operation, the SPI master controls the generation of the serial clock. The number of output clock pulses corresponds to the transfer data width: 8, 16 or 32 bits, or depending upon variable data width configuration, from 2 to 32-bit. Figure 1-2 and Figure 1-3 illustrate SPI master-to-slave and slave-to-master device connections.

Figure 1-2: Typical SPIx Master-to-Slave Device Connection Diagram

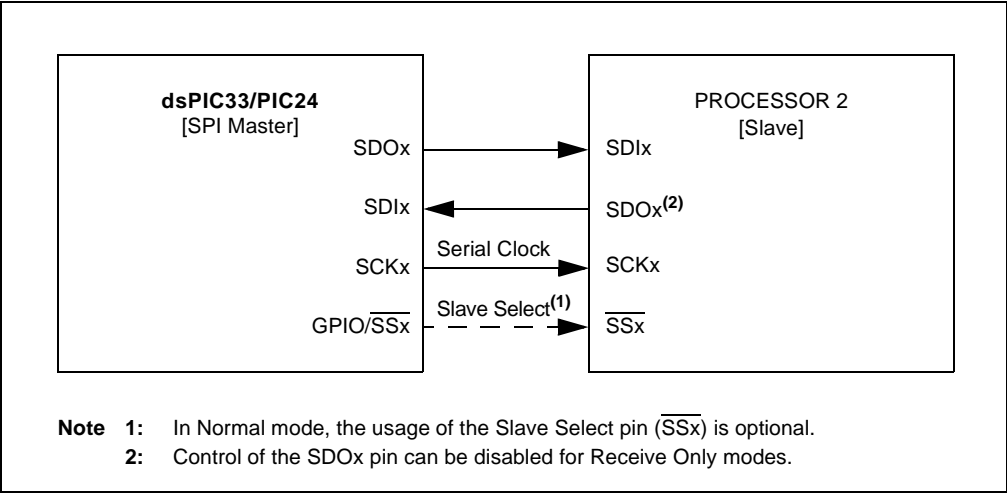
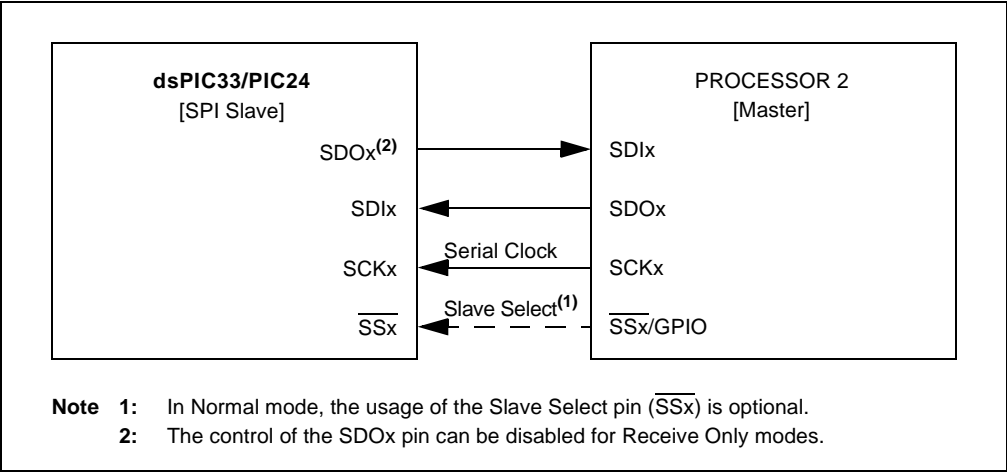


Figure 1-3: Typical SPIx Slave-to-Master Device Connection Diagram



Serial Peripheral Interface (SPI) with Audio Codec Support

1.2 Framed Mode SPI Operation

In Framed mode operation, the frame master controls the generation of the Frame Synchronization pulse. The SPI clock is still generated by the SPI master and is continuously running. Figure 1-4 and Figure 1-5 illustrate SPI frame master and frame slave device connections.

Figure 1-4: Typical SPlx Master, Frame Master Connection Diagram

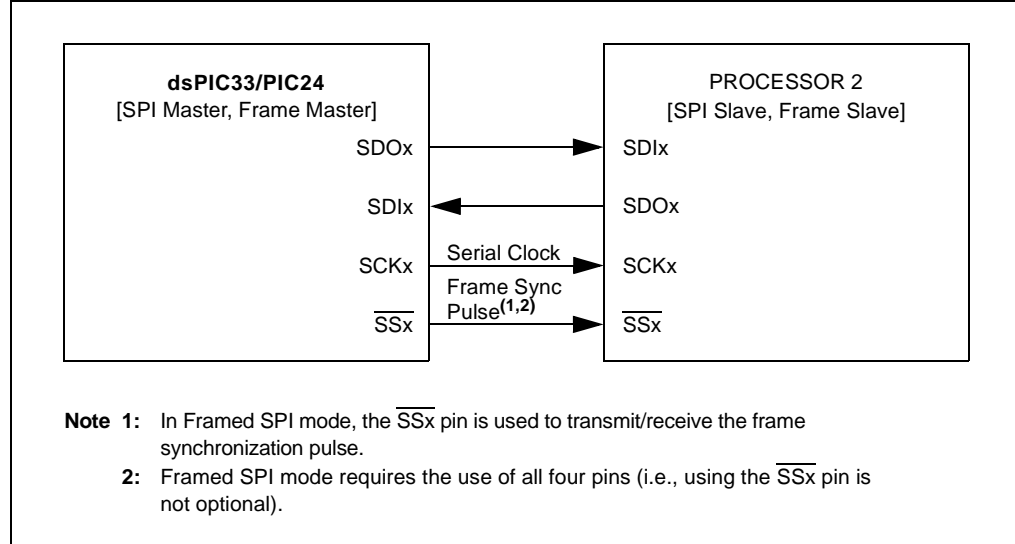
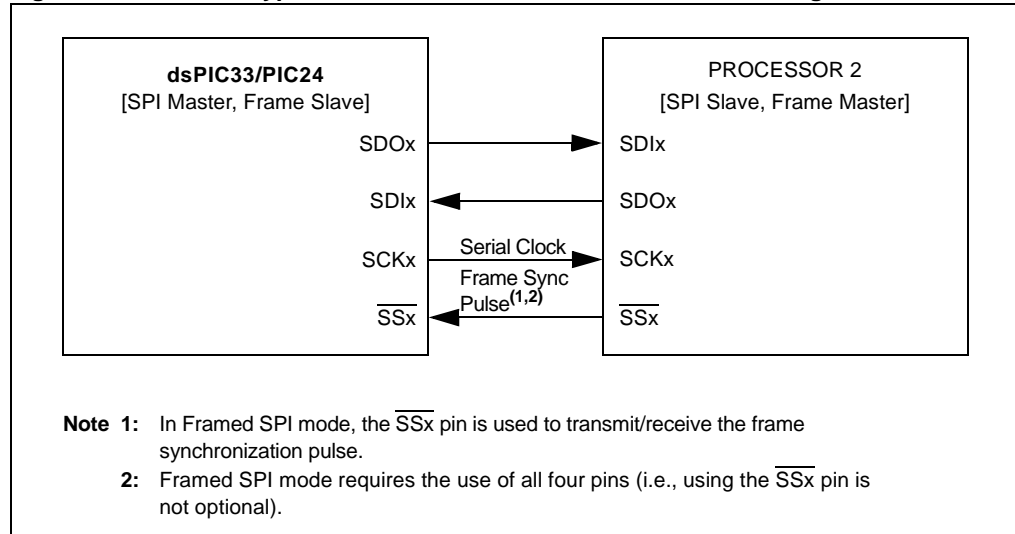


Figure 1-5: Typical SPlx Master, Frame Slave Connection Diagram

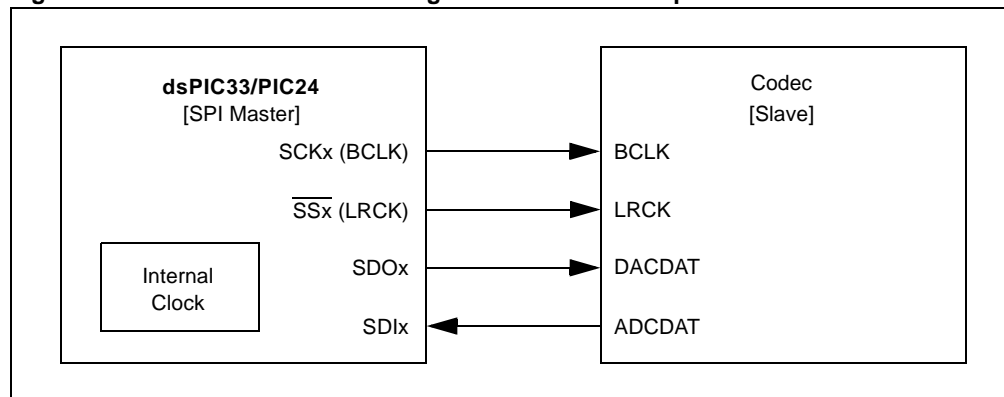


1.3 Audio Protocol Interface Mode

1.3.1 SPI IN AUDIO MASTER MODE CONNECTED TO A CODEC SLAVE

Figure 1-6 shows the Bit Clock (BCLK) and Left/Right Channel Clock (LRCK) as generated by the dsPIC33/PIC24 SPI module.

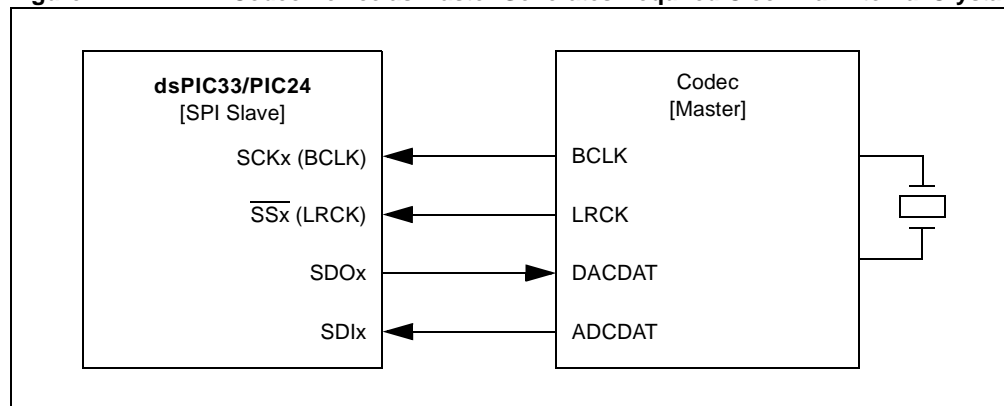
Figure 1-6: Master Generating its Own Clock – Output BCLK and LRCK



1.3.2 SPI IN AUDIO SLAVE MODE CONNECTED TO A CODEC MASTER

Figure 1-7 shows the BCLK and LRCK as generated by the codec master.

Figure 1-7: Codec Device as Master Generates Required Clock via External Crystal



2.0 STATUS AND CONTROL REGISTERS

Note: Each dsPIC33/PIC24 family device variant may have one or more SPI modules. An 'x' used in the names of pins, control/status bits and registers denotes the particular module. Refer to the specific device data sheets for more details.

The SPI module consists of the following Special Function Registers (SFRs):

- SPIxCON1L, SPIxCON1H and SPIxCON2L: SPIx Control Registers
- SPIxSTAT1L and SPIxSTAT1H: SPIx Status Registers
- SPIxBUFL and SPIxBUFH: SPIx Buffer Registers
- SPIxBRGL and SPIxBRGH: SPIx Baud Rate Registers
- SPIxIMSKL and SPIxIMSKH: SPIx Interrupt Mask Registers
- SPIxURDTL and SPIxURDTH: SPIx Underrun Data Registers

Serial Peripheral Interface (SPI) with Audio Codec Support

Register 2-1: SPIxCON1L: SPIx Control Register 1 Low

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPIEN ⁽³⁾	—	SPISIDL	DISSDO	MODE32 ^(1,4)	MODE16 ^(1,4)	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 ⁽²⁾	CKP	MSTEN	DISSDI	DISSCK	MCLKEN ⁽³⁾	SPIFE	ENHBUF
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 **SPIEN:** SPIx On bit⁽³⁾

1 = Enables module

0 = Turns off and resets module, disables clocks, disables interrupt event generation and allows SFR modifications

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

bit 13 **SPISIDL:** SPIx Stop in Idle Mode bit

1 = Halts in CPU Idle mode

0 = Continues to operate in CPU Idle mode

bit 12 **DISSDO:** Disable SDOx Output Port bit

1 = SDOx pin is not used by the module; pin is controlled by the port function

0 = SDOx pin is controlled by the module

bit 11-10 **MODE<32,16>:** Serial Word Length bits^(1,4)

AUDEN = 0:

MODE32	MODE16	COMMUNICATION
1	x	32-bit
0	1	16-bit
0	0	8-bit

AUDEN = 1:

MODE32	MODE16	COMMUNICATION
1	1	24-bit data, 32-bit FIFO, 32-bit channel/64-bit frame
1	0	32-bit data, 32-bit FIFO, 32-bit channel/64-bit frame
0	1	16-bit data, 16-bit FIFO, 32-bit channel/64-bit frame
0	0	16-bit data, 16-bit FIFO, 16-bit channel/32-bit frame

bit 9 **SMP:** SPIx Data Input Sample Phase bit

Master Mode:

1 = Input data is sampled at the end of data output time

0 = Input data is sampled at the middle of data output time

Slave Mode:

Input data is always sampled at the middle of data output time, regardless of the SMP setting.

Note 1: When AUDEN = 1, this module functions as if CKE = 0, regardless of its actual value.

2: When FRMEN = 1, SSEN is not used.

3: MCLKEN can only be written when the SPIEN bit = 0.

4: This channel is not meaningful for DSP/PCM mode as LRC follows FRMSYPW.

dsPIC33/PIC24 Family Reference Manual

Register 2-1: SPIxCON1L: SPIx Control Register 1 Low (Continued)

bit 8	CKE: SPIx Clock Edge Select bit ⁽¹⁾ 1 = Transmit happens on transition from active clock state to Idle clock state 0 = Transmit happens on transition from Idle clock state to active clock state
bit 7	SSEN: Slave Select Enable bit (Slave mode) ⁽²⁾ 1 = \overline{SSx} pin is used by the macro in Slave mode; \overline{SSx} pin is used as the Slave Select input 0 = \overline{SSx} pin is not used by the macro (\overline{SSx} pin will be controlled by the port I/O)
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	DISSDI: Disable SDIx input port 1 = SDIx pin is not used by the module; pin is controlled by the port function 0 = SDIx pin is controlled by the module
bit 3	DISSCK: Disable SCKx output port 1 = SCKx pin is not used by the module; pin is controlled by the port function 0 = SCKx pin is controlled by the module
bit 2	MCLKEN: Master Clock Enable bit ⁽³⁾ 1 = MCLK is used by the BRG 0 = PBCLK is used by the BRG
bit 1	SPIFE: Frame Sync Pulse Edge Select bit 1 = Frame Sync pulse (Idle-to-active edge) coincides with the first bit clock 0 = Frame Sync pulse (Idle-to-active edge) precedes the first bit clock
bit 0	ENHBUF: Enhanced Buffer Enable bit 1 = Enhanced Buffer mode is enabled 0 = Enhanced Buffer mode is disabled

Note 1: When AUDEN = 1, this module functions as if CKE = 0, regardless of its actual value.

2: When FRMEN = 1, SSEN is not used.

3: MCLKEN can only be written when the SPIEN bit = 0.

4: This channel is not meaningful for DSP/PCM mode as LRC follows FRMSYPW.

Serial Peripheral Interface (SPI) with Audio Codec Support

Register 2-2: SPIxCON1H: SPIx Control Register 1 High

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
AUDEN ⁽¹⁾	SPISGNEXT	IGNROV	IGNTUR	AUDMONO	URDTEN	AUDMOD1 ⁽³⁾	AUDMOD0 ⁽³⁾
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
FRMEN ⁽²⁾	FRMSYNC	FRMPOL	MSEN	FRMSYPW	FRMCNT2	FRMCNT1	FRMCNT0
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 **AUDEN:** Audio Codec Support Enable bit⁽¹⁾
1 = Audio protocol is enabled; MSTEN controls the direction of both SCKx and frame (a.k.a. LRC), and this module functions as if FRMEN = 1, FRMSYNC = MSTEN, FRMCNT<2:0> = 001 and SMP = 0, regardless of their actual values
0 = Audio protocol is disabled
- bit 14 **SPISGNEXT:** SPIx Sign-Extend RX FIFO Read Data Enable bit
1 = Data from RX FIFO is sign-extended
0 = Data from RX FIFO is not sign-extended
- bit 13 **IGNROV:** Ignore Receive Overflow bit
1 = A Receive Overflow (ROV) is NOT a critical error; during ROV, data in the FIFO is not overwritten by the receive data
0 = A ROV is a critical error that stops SPI operation
- bit 12 **IGNTUR:** Ignore Transmit Underrun bit
1 = A Transmit Underrun (TUR) is NOT a critical error and data indicated by URDTEN is transmitted until the SPIxTXB is not empty
0 = A TUR is a critical error that stops SPI operation
- bit 11 **AUDMONO:** Audio Data Format Transmit bit
1 = Audio data is mono (i.e., each data word is transmitted on both left and right channels)
0 = Audio data is stereo
- bit 10 **URDTEN:** Transmit Underrun Data Enable bit
1 = Transmits data out of the SPIxURDT register during Transmit Underrun conditions
0 = Transmits the last received data during Transmit Underrun conditions
- bit 9-8 **AUDMOD<1:0>:** Audio Protocol Mode Selection bits⁽³⁾
11 = PCM/DSP mode
10 = Right Justified mode: This module functions as if SPIFE = 1, regardless of its actual value
01 = Left Justified mode: This module functions as if SPIFE = 1, regardless of its actual value
00 = I²S mode: This module functions as if SPIFE = 0, regardless of its actual value
- bit 7 **FRMEN:** Framed SPIx Support bit⁽²⁾
1 = Framed SPIx support is enabled (\overline{SSx} pin is used as FSYNC input/output)
0 = Framed SPIx support is disabled
- bit 6 **FRMSYNC:** Frame Sync Pulse Direction Control bit
1 = Frame Sync pulse input (slave)
0 = Frame Sync pulse output (master)

- Note 1:** When AUDEN = 1, this module functions as if CKE = 0, regardless of its actual value.
2: When FRMEN = 1, SSEN is not used.
3: This channel is not meaningful for DSP/PCM mode as LRC follows FRMSYPW.

dsPIC33/PIC24 Family Reference Manual

Register 2-2: SPIxCON1H: SPIx Control Register 1 High (Continued)

- bit 5 **FRMPOL:** Frame Sync/Slave Select Polarity bit
1 = Frame Sync pulse/Slave Select is active-high
0 = Frame Sync pulse/Slave Select is active-low
- bit 4 **MSEN:** Master Mode Slave Select Enable bit
1 = SPIx Slave Select support is enabled with polarity determined by FRMPOL (\overline{SSx} pin is automatically driven during transmission in Master mode)
0 = SPIx Slave Select support is disabled (\overline{SSx} pin will be controlled by port IO)
- bit 3 **FRMSYPW:** Frame Sync Pulse-Width bit
1 = Frame Sync pulse is one serial word length wide (as defined by MODE<32,16>/WLENGTH<4:0>).
0 = Frame Sync pulse is one clock (SCKx) wide.
- bit 2-0 **FRMCNT<2:0>:** Frame Sync Pulse Counter bits
Controls the number of serial words transmitted per Sync pulse.
111 = Reserved
110 = Reserved
101 = Generates a Frame Sync pulse on every 32 serial words
100 = Generates a Frame Sync pulse on every 16 serial words
011 = Generates a Frame Sync pulse on every 8 serial words
010 = Generates a Frame Sync pulse on every 4 serial words
001 = Generates a Frame Sync pulse on every 2 serial words (value used by audio protocols)
000 = Generates a Frame Sync pulse on each serial word

- Note 1:** When AUDEN = 1, this module functions as if CKE = 0, regardless of its actual value.
2: When FRMEN = 1, SSEN is not used.
3: This channel is not meaningful for DSP/PCM mode as LRC follows FRMSYPW.

Serial Peripheral Interface (SPI) with Audio Codec Support

Register 2-3: SPIxCON2L: SPIx Control Register 2 Low

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	WLENGTH4 ^(1,2)	WLENGTH3 ^(1,2)	WLENGTH2 ^(1,2)	WLENGTH1 ^(1,2)	WLENGTH0 ^(1,2)
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 15-5 **Unimplemented:** Read as '0'

bit 4-0 **WLENGTH<4:0>:** Variable Word Length bits^(1,2)

11111 = 32-bit data
 11110 = 31-bit data
 11101 = 30-bit data
 11100 = 29-bit data
 11011 = 28-bit data
 11010 = 27-bit data
 11001 = 26-bit data
 11000 = 25-bit data
 10111 = 24-bit data
 10110 = 23-bit data
 10101 = 22-bit data
 10100 = 21-bit data
 10011 = 20-bit data
 10010 = 19-bit data
 10001 = 18-bit data
 10000 = 17-bit data
 01111 = 16-bit data
 01110 = 15-bit data
 01101 = 14-bit data
 01100 = 13-bit data
 01011 = 12-bit data
 01010 = 11-bit data
 01001 = 10-bit data
 01000 = 9-bit data
 00111 = 8-bit data
 00110 = 7-bit data
 00101 = 6-bit data
 00100 = 5-bit data
 00011 = 4-bit data
 00010 = 3-bit data
 00001 = 2-bit data
 00000 = See MODE<32,16> bits (SPIxCON1L<11:10>)

Note 1: These bits are effective when AUDEN = 0 only.

2: Varying the length by changing these bits does not affect the depth of the TX/RX FIFO.

dsPIC33/PIC24 Family Reference Manual

Register 2-4: SPIxCON2H: SPIx Control Register 2 High

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-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 15-0 **Unimplemented:** Read as '0'

Register 2-5: SPIxSTATL: SPIx Status Register Low

U-0	U-0	U-0	R/C-0, HS	R-0, HSC	U-0	U-0	R-0, HSC
—	—	—	FRMERR	SPIBUSY	—	—	SPITUR ⁽¹⁾
bit 15							bit 8

R-0, HSC	R/C-0, HS	R-1, HSC	U-0	R-1, HSC	U-0	R-0, HSC	R-0, HSC
SRMT	SPIROV	SPIRBE	—	SPITBE	—	SPITBF	SPIRBF
bit 7							bit 0

Legend:

C = Clearable bit HS = Hardware Settable bit
 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
 HSC = Hardware Settable/Clearable bit

bit 15-13 **Unimplemented:** Read as '0'

bit 12 **FRMERR:** SPIx Frame Error Status bit

1 = Frame error is detected
 0 = No frame error is detected

bit 11 **SPIBUSY:** SPIx Activity Status bit

1 = Module is currently busy with some transactions
 0 = No ongoing transactions (at time of read)

bit 10-9 **Unimplemented:** Read as '0'

bit 8 **SPITUR:** SPIx Transmit Underrun Status bit⁽¹⁾

1 = Transmit buffer has encountered a Transmit Underrun condition
 0 = Transmit buffer does not have a Transmit Underrun condition

bit 7 **SRMT:** Shift Register Empty Status bit

1 = No current or pending transactions (i.e., neither SPIxTXB or SPIxTXSR contains data to transmit)
 0 = Current or pending transitions

Note 1: SPITUR is cleared when SPIEN = 0. When IGNTUR = 1, SPITUR provides dynamic status of the Transmit Underrun condition, but does not stop RX/TX operation and does not need to be cleared by software.

Serial Peripheral Interface (SPI) with Audio Codec Support

Register 2-5: SPIxSTATL: SPIx Status Register Low (Continued)

- bit 6 **SPIROV:** SPIx Receive Overflow Status bit
1 = A new byte/half-word/word has been completely received when the SPIxRXB was full
0 = No overflow
- bit 5 **SPIRBE:** SPIx RX Buffer Empty Status bit
1 = RX buffer is empty
0 = RX buffer is not empty
Standard Buffer Mode:
Automatically set in hardware when SPIxBUF is read from, reading SPIxRXB. Automatically cleared in hardware when SPIx transfers data from SPIxRXSR to SPIxRXB.
Enhanced Buffer Mode:
Indicates RX FIFO is empty.
- bit 4 **Unimplemented:** Read as '0'
- bit 3 **SPITBE:** SPIx Transmit Buffer Empty Status bit
1 = SPIxTXB is empty
0 = SPIxTXB is not empty
Standard Buffer Mode:
Automatically set in hardware when SPIx transfers data from SPIxTXB to SPIxTXSR. Automatically cleared in hardware when SPIxBUF is written, loading SPIxTXB.
Enhanced Buffer Mode:
Indicates TX FIFO is empty.
- bit 2 **Unimplemented:** Read as '0'
- bit 1 **SPITBF:** SPIx Transmit Buffer Full Status bit
1 = SPIxTXB is full
0 = SPIxTXB not full
Standard Buffer Mode:
Automatically set in hardware when SPIxBUF is written, loading SPIxTXB. Automatically cleared in hardware when SPIx transfers data from SPIxTXB to SPIxTXSR.
Enhanced Buffer Mode:
Indicates TX FIFO is full.
- bit 0 **SPIRBF:** SPIx Receive Buffer Full Status bit
1 = SPIxRXB is full
0 = SPIxRXB is not full
Standard Buffer Mode:
Automatically set in hardware when SPIx transfers data from SPIxRXSR to SPIxRXB. Automatically cleared in hardware when SPIxBUF is read from, reading SPIxRXB.
Enhanced Buffer Mode:
Indicates RX FIFO is full.

Note 1: SPITUR is cleared when SPIEN = 0. When IGTUR = 1, SPITUR provides dynamic status of the Transmit Underrun condition, but does not stop RX/TX operation and does not need to be cleared by software.

dsPIC33/PIC24 Family Reference Manual

Register 2-6: SPIxSTATH: SPIx Status Register High

U-0	U-0	R-0, HS	R-0, HS	R-0, HSC	R-0, HS	R-0, HS	R-0, HSC
—	—	RXELM5 ⁽³⁾	RXELM4 ⁽²⁾	RXELM3 ⁽¹⁾	RXELM2	RXELM1	RXELM0 ⁽¹⁾
bit 15							bit 8

U-0	U-0	R-0, HSC	R-0, HSC	R-0, HSC	R-0, HSC	R-0, HSC	R-0, HSC
—	—	TXELM5 ⁽³⁾	TXELM4 ⁽²⁾	TXELM3 ⁽¹⁾	TXELM2	TXELM1	TXELM0
bit 7							bit 0

Legend:	HS = Hardware Settable bit	HSC = Hardware Settable/Clearable bit
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 15-14 **Unimplemented:** Read as '0'

bit 13-8 **RXELM<5:0>:** Receive Buffer Element Count bits (valid in Enhanced Buffer mode)^(1,2,3)

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **TXELM<5:0>:** Transmit Buffer Element Count bits (valid in Enhanced Buffer mode)^(1,2,3)

Note 1: RXELM3 and TXELM3 bits are only present when FIFODEPTH = 8 or higher.

Note 2: RXELM4 and TXELM4 bits are only present when FIFODEPTH = 16 or higher.

Note 3: RXELM5 and TXELM5 bits are only present when FIFODEPTH = 32.

Serial Peripheral Interface (SPI) with Audio Codec Support

Register 2-7: SPIxBUFL: SPIx Buffer Register Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DATA<15:8>							
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
DATA<7:0>							
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 15-0 **DATA<15:0>**: SPIx FIFO Data bits

When the MODE<32,16> or WLENGTH<4:0> bits select 16 to 9-bit data, the SPIx only uses DATA<15:0>. When the MODE<32,16> or WLENGTH<4:0> bits select 8 to 2-bit data, the SPIx only uses DATA<7:0>.

Register 2-8: SPIxBUFH: SPIx Buffer Register High

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DATA<31:24>							
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
DATA<23:16>							
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 **DATA<31:16>**: SPIx FIFO Data bits

When the MODE<32,16> or WLENGTH<4:0> bits select 32 to 25-bit data, the SPIx uses DATA<31:16>.

When the MODE<32,16> or WLENGTH<4:0> bits select 24 to 17-bit data, the SPIx only uses DATA<23:16>.

dsPIC33/PIC24 Family Reference Manual

Register 2-9: SPIxBRGL: SPIx Baud Rate Generator Register Low

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	BRG<12:8> ⁽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
BRG<7:0> ⁽¹⁾							
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 12-0 **BRG<12:0>:** SPIx Baud Rate Generator Divisor bits⁽¹⁾

Note 1: Changing the BRG value when SPIEN = 1 causes undefined behavior.

Register 2-10: SPIxBRGH: SPIx Baud Rate Generator Register High

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-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 15-0 **Unimplemented:** Read as '0'

Serial Peripheral Interface (SPI) with Audio Codec Support

Register 2-11: SPIxIMSKL: SPIx Interrupt Mask Register Low

U-0	U-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	—	—	FRMERREN	BUSYEN	—	—	SPITUREN
bit 15			bit 8				

R/W-0		R/W-0		R/W-0		U-0		R/W-0		U-0		R/W-0		R/W-0																	
SRMTEN		SPIROVEN		SPIRBEN		—			SPITBEN		—			SPITBFEN		SPIRBFEN															
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 **FRMERREN:** Enable Interrupt Events via FRMERR bit

1 = Frame error generates an interrupt event

0 = Frame error does not generate an interrupt event

bit 11 **BUSYEN:** Enable Interrupt Events via SPIBUSY bit

1 = SPIBUSY generates an interrupt event

0 = SPIBUSY does not generate an interrupt event

bit 10-9 **Unimplemented:** Read as '0'

bit 8 **SPITUREN:** Enable Interrupt Events via SPITUR bit

1 = Transmit Underrun (TUR) generates an interrupt event

0 = Transmit Underrun does not generate an interrupt event

bit 7 **SRMTEN:** Enable Interrupt Events via SRMT bit

1 = Shift register empty generates an interrupt event

0 = Shift register empty does not generate an interrupt event

bit 6 **SPIROVEN:** Enable Interrupt Events via SPIROV bit

1 = SPIx Receive Overflow (ROV) generates an interrupt event

0 = SPIx Receive Overflow does not generate an interrupt event

bit 5 **SPIRBEN:** Enable Interrupt Events via SPIRBE bit

1 = SPIx receive buffer empty generates an interrupt event

0 = SPIx receive buffer empty does not generate an interrupt event

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

bit 3 **SPITBEN:** Enable Interrupt Events via SPITBE bit

1 = SPIx transmit buffer empty generates an interrupt event

0 = SPIx transmit buffer empty does not generate an interrupt event

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

bit 1 **SPITBFEN:** Enable Interrupt Events via SPITBF bit

1 = SPIx transmit buffer full generates an interrupt event

0 = SPIx transmit buffer full does not generate an interrupt event

bit 0 **SPIRBFEN:** Enable Interrupt Events via SPIRBF bit

1 = SPIx receive buffer full generates an interrupt event

0 = SPIx receive buffer full does not generate an interrupt event

dsPIC33/PIC24 Family Reference Manual

Register 2-12: SPIxIMSKH: SPIx Interrupt Mask Register High

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RXWIEN	—	RXMSK5 ⁽¹⁾	RXMSK4 ^(1,4)	RXMSK3 ^(1,3)	RXMSK2 ^(1,2)	RXFMSK1 ⁽¹⁾	RXMSK0 ⁽¹⁾
bit 15							bit 8

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TXWIEN	—	TXMSK5 ⁽¹⁾	TXMSK4 ^(1,4)	TXMSK3 ^(1,3)	TXMSK2 ^(1,2)	TXMSK1 ⁽¹⁾	TXMSK0 ⁽¹⁾
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 **RXWIEN:** Receive Watermark Interrupt Enable bit
1 = Triggers receive buffer element watermark interrupt when RXMSK<5:0> = RXELM<5:0>
0 = Disables receive buffer element watermark interrupt
- bit 14 **Unimplemented:** Read as '0'
- bit 13-8 **RXMSK<5:0>:** RX Buffer Mask bits^(1,2,3)
RX mask bits; used in conjunction with the RXWIEN bit.
- bit 7 **TXWIEN:** Transmit Watermark Interrupt Enable bit
1 = Triggers transmit buffer element watermark interrupt when TXMSK<5:0> = TXELM<5:0>
0 = Disables transmit buffer element watermark interrupt
- bit 6 **Unimplemented:** Read as '0'
- bit 5-0 **TXMSK<5:0>:** TX Buffer Mask bits^(1,2,3)
TX mask bits; used in conjunction with the TXWIEN bit.

- Note 1:** Mask values higher than FIFODEPTH are not valid. The module will not trigger a match for any value in this case.
- 2:** RXMSK2 and TXMSK2 bits are only present when FIFODEPTH = 8 or higher.
- 3:** RXMSK3 and TXMSK3 bits are only present when FIFODEPTH = 16 or higher.
- 4:** RXMSK4 and TXMSK4 bits are only present when FIFODEPTH = 32.

Serial Peripheral Interface (SPI) with Audio Codec Support

Register 2-13: SPIxURDTL: SPIx Underrun Data Register Low

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
URDATA<15:8>							
bit 15				bit 8			

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
URDATA<7:0>							
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 15-0 **URDATA<15:0>**: SPIx Underrun Data bits

These bits are only used when URDTEN = 1. This register holds the data to transmit when a Transmit Underrun condition occurs.

When the MODE<32,16> or WLENGTH<4:0> bits select 16 to 9-bit data, the SPIx only uses URDATA<15:0>. When the MODE<32,16> or WLENGTH<4:0> bits select 8 to 2-bit data, the SPIx only uses URDATA<7:0>.

Register 2-14: SPIxURDTH: SPIx Underrun Data Register High

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 URDATA<31:24>							
bit 15				bit 8			

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPIx URDATA<23:16>							
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 **URDATA<31:16>**: SPIx Underrun Data bits

These bits are only used when URDTEN = 1. This register holds the data to transmit when a Transmit Underrun condition occurs.

When the MODE<32,16> or WLENGTH<4:0> bits select 32 to 25-bit data, the SPIx only uses URDATA<31:16>. When the MODE<32,16> or WLENGTH<4:0> bits select 24 to 17-bit data, the SPIx only uses URDATA<23:16>.

3.0 MODES OF OPERATION

The SPI module offers the following operating modes:

- 8-Bit, 16-Bit and 32-Bit Data Transmission modes
- 8-Bit, 16-Bit and 32-Bit Data Reception modes
- Master and Slave modes
- Framed SPI modes
- Audio Protocol Interface mode

- | |
|---|
| <p>Note 1: In Framed SPI mode, these four pins are used: SDIx, SDOx, SCKx and $\overline{\text{SSx}}$.</p> <p>2: If the Slave Select feature is used, all four pins listed in Note 1 are used.</p> <p>3: If standard SPI is used, but CKE = 1, enabling/using the Slave Select feature is mandatory, and therefore, all four pins listed in Note 1 are used.</p> <p>4: If standard SPI is used, but DISSDO = 1, only two pins are used: SDIx and SCKx; unless Slave Select is also enabled.</p> <p>5: In all other cases, three pins are used: SDIx, SDOx and SCKx.</p> |
|---|

3.1 8-Bit, 16-Bit and 32-Bit Operation

The SPI module allows three types of data widths when transmitting and receiving data over an SPI bus. The selection of data width determines the minimum length of SPI data. For example, when the selected data width is 32, all transmission and receptions are performed in 32-bit values. All reads and writes from the CPU are also performed in 32-bit values. Accordingly, the application software should select the appropriate data width to maximize its data throughput.

Two control bits, MODE32 and MODE16 (SPIxCON1L<11:10>), which are referred to as MODE<32,16>, define the mode of operation. To change the mode of operation on-the-fly, the SPI module must be Idle (i.e., not performing any transactions). If the SPI module is switched off (SPIxCON1L<15> = 0), the new mode will be available when the module is again switched on.

Additionally, the following items should be noted in this context:

- The MODE<32,16> bits should not be changed when a transaction is in progress
- The first bit to be shifted out from SPIxTXSR varies with the selected mode of operation:
 - 8-bit mode, bit 7
 - 16-bit mode, bit 15
 - 32-bit mode, bit 31
- In each mode, data is shifted into bit 0 of the SPIxRXSR
- The number of clock pulses at the SCKx pin are also dependent on the selected mode of operation:
 - 8-bit mode, 8 clocks
 - 16-bit mode, 16 clocks
 - 32-bit mode, 32 clocks

3.2 Buffer Modes

There are two SPI Buffering modes: Standard and Enhanced.

3.2.1 STANDARD BUFFER MODE

The SPIx Data Receive/Transmit Buffer (SPIxBUF) register is actually two separate internal registers: the Transmit Buffer (SPIxTXB) and the Receive Buffer (SPIxRXB). These two unidirectional registers share the SFR address of SPIxBUF.

When a complete byte/word is received, it is transferred from SPIxRXSR to SPIxRXB and the SPIRBF bit is set. If the software reads the SPIxBUF buffer, the SPIRBF bit is cleared.

As the software writes to SPIxBUF, the data is loaded into the SPIxTXB and the SPITBF bit is set by hardware. As the data is transmitted out of SPIxTXSR, the SPITBF bit is cleared.

Serial Peripheral Interface (SPI) with Audio Codec Support

The SPI module double-buffers transmit/receive operations and allows continuous data transfers in the background. Transmission and reception occur simultaneously in SPIxTXSR and SPIxRXSR, respectively.

Note: SPIxBUF refers to SPIxBUFL/SPIxBUFH. If data length is greater than 16, then both SPIxBUFL and SPIxBUFH have to be used, else only SPIxBUFL has to be used.

3.2.2 ENHANCED BUFFER MODE

The Enhanced Buffer Enable (ENHBUF) bit in the SPIx Control Register 1 Low (SPIxCON1L<0>) can be set to enable the Enhanced Buffer mode.

In Enhanced Buffer mode, two 128-bit FIFO buffers are used for the SPIx Transmit Buffer (SPIxTXB) and the SPIx Receive Buffer (SPIxRXB). SPIxBUF provides access to both the receive and transmit FIFOs. The data transmission and reception in the SPIxSR buffer is identical to that in the Standard Buffer mode. The FIFO depth depends on the data width chosen by the Word/Half-Word Byte Communication Select (MODE<32,16>) bits in the SPIx Control Register 1 Low (SPIxCON1L<11:10>). If the MODE field selects 32-bit data lengths, the FIFO is 4 deep; if the MODE selects 16-bit data lengths, the FIFO is 8 deep, or if MODE selects 8-bit data lengths, the FIFO is 16 deep.

Note: FIFO depth does not change when variable word length is configured.

The SPITBF status bit is set when all of the elements in the transmit FIFO buffer are full and is cleared if one or more of those elements are empty. The SPIRBF status bit is set when all of the elements in the receive FIFO buffer are full and is cleared if the SPIxBUF buffer is read by the software.

Note: When data is more than 16 bits, always write SPIxBUFL first and then write SPIxBUFH. Similarly, read SPIxBUFL first and then read SPIxBUFH.

The SPITBE status bit is set if all the elements in the transmit FIFO buffer are empty and is cleared otherwise. The SPIRBE bit is set if all of the elements in the receive FIFO buffer are empty and is cleared otherwise.

There is underrun or overflow protection against reading an empty receive FIFO element or writing a full transmit FIFO element. The SPIxSTATL register provides the SPIx Transmit Underrun bit (SPITUR) and the Receive Overflow Status bit (SPIROV). Depending on the requirements, IGNTUR and IGNROV can be configured for SPI operation to be continued or not, at the time of error. When a Transmit Underrun occurs, the last received data or the data in the SPIxURDT register can be transmitted by configuring the URDTEN bit (SPIxCON1H<10>).

The Receive Buffer Element Count bits (RXELM<5:0>) in the SPIx Status Register High (SPIxSTATH<13:8>) indicate the number of unread elements in the receive FIFO. The Transmit Buffer Element Count bits (TXELM<5:0>) in the SPIx Status Register High (SPIxSTATH<5:0>) indicate the number of elements not transmitted in the transmit FIFO.

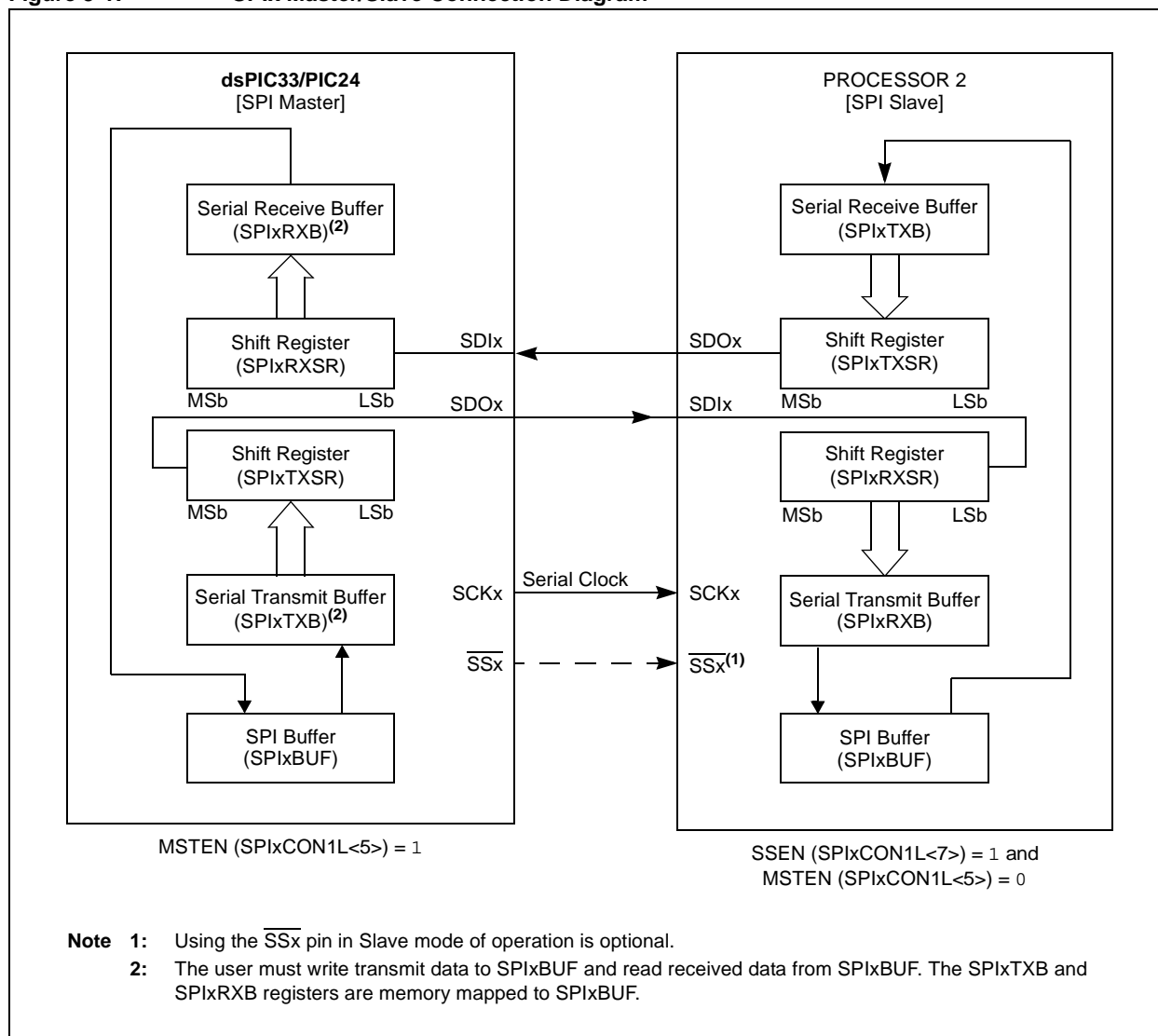
3.3 Variable Word Length Operation

The SPI module allows variable word length when transmitting and receiving data over an SPI bus. Word length can vary from 2 to 32 bits. Different word lengths can be configured by changing the WLENGTH<4:0> bits (SPIxCON2L<4:0>). The number of clock pulses at the SCKx pin will correspond to the word length that is selected.

3.4 Master and Slave Modes

In 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 moving them 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. The relationship between the master and slave modules is shown in [Figure 3-1](#).

Figure 3-1: SPIx Master/Slave Connection Diagram



Serial Peripheral Interface (SPI) with Audio Codec Support

3.4.1 MASTER MODE OPERATION

Perform the following steps to set up the SPI module for Master mode operation:

1. Disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit.
3. Clear the receive buffer.
4. Clear the ENHBUF bit (SPIxCON1L<0>) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
5. If SPIx interrupts are not going to be used, skip this step. Otherwise, the following additional steps are performed:
 - a) Clear the SPIx interrupt flags/events in the respective IFSx register.
 - b) Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
 - c) Set the SPIx interrupt enable bits in the respective IECx register.
6. Write the Baud Rate register, SPIxBRGL.
7. Clear the SPIROV bit (SPIxSTATL<6>).
8. Write the desired settings to the SPIxCON1L register with MSTEN (SPIxCON1L<5>) = 1.
9. Enable SPI operation by setting the SPIEN bit (SPIxCON1L<15>).
10. Write the data to be transmitted to the SPIxBUFL and SPIxBUFH registers. Transmission (and reception) will start as soon as data is written to the SPIxBUFL/H register.

Note: The SPI device must be turned OFF prior to changing the mode from Slave to Master. (When using the Slave Select mode, the \overline{SSx} pin or another GPIO pin is used to control the slave's \overline{SSx} input. The pin must be controlled in software.)

In Master mode, the PBCLK is divided and then used as the serial clock. The division is based on the settings in the SPIxBRGL register. The serial clock is output through the SCKx pin to the slave devices. Clock pulses are only generated when there is data to be transmitted; except when in Framed mode, when the clock is generated continuously. For further information, refer to [Section 3.8 “SPI Master Mode Clock Frequency”](#).

The Master Mode Slave Select Enable (MSEN) bit in the SPIx Control Register 1 High (SPIxCON1H<4>) can be set to automatically drive the Slave Select signal (\overline{SSx}) in Master mode. Clearing this bit disables the Slave Select signal support in Master mode. The FRMPOL bit (SPIxCON1H<5>) determines the polarity for the Slave Select signal in Master mode.

Note: The MSEN bit is not available on all devices. This bit should not be set when the SPI Framed mode is enabled (i.e., FRMEN = 1). Refer to the specific device data sheet for details.

In devices that do not feature the MSEN bit, the Slave Select signal (in Non-Framed SPI mode) must be generated by using the \overline{SSx} pin or another general purpose I/O pin under software control.

The CKP (SPIxCON1L<6>) and CKE (SPIxCON1L<8>) bits determine on which edge of the clock data transmission occurs.

Note: The user must turn OFF the SPI device prior to changing the CKE or CKP bits. Otherwise, the behavior of the device is not ensured.

Both data to be transmitted and data that is received are written to, or read from, the SPIxBUF register, respectively.

The following progression describes the SPI module operation in Master mode:

1. Once the module is set up for Master mode operation and enabled, data to be transmitted is written to the SPIxBUF register. The SPITBE bit (SPIxSTATL<3>) is cleared.
2. The contents of SPIxTXB are moved to the SPIx Shift register, SPIxTXSR (see [Figure 3-1](#)), and the SPITBE bit is set by the module.
3. A series of 8/16/32 clock pulses shifts 8/16/32 bits of transmit data from SPIxTXSR to the SDOx pin and simultaneously shifts the data at the SDIx pin into SPIxRXSR.
4. When the transfer is complete, the following events will occur:
 - a) The SPIxRXIF interrupt flag bit is set. SPIx interrupts can be enabled by setting the SPIxRXIE interrupt enable bit. The SPIxRXIF flag is not cleared automatically by the hardware.
 - b) Also, when the ongoing transmit and receive operation is completed, the contents of SPIxRXSR are moved to SPIxRXB.
 - c) The SPIRBF bit (SPIxSTATL<0>) is set by the module, indicating that the receive buffer is full. Once SPIxBUF is read by the user code, the hardware clears the SPIRBF bit. In Enhanced Buffer mode, the SPIRBE bit (SPIxSTATL<5>) is set when the SPIxRXB FIFO buffer is completely empty and cleared when not empty.
5. If the SPIRBF bit is set (the receive buffer is full) when the SPI module needs to transfer data from SPIxRXSR to SPIxRXB, the module will set the SPIROV bit (SPIxSTATL<6>) indicating an overflow condition.
6. Data to be transmitted can be written to SPIxBUF by the user software at any time, if the SPITBE bit (SPIxSTATL<3>) is set. The write can occur while SPIxTXSR is shifting out the previously written data, allowing continuous transmission. In Enhanced Buffer mode, the SPITBF bit (SPIxSTATL<1>) is set when the SPIxTXB FIFO buffer is completely full and clear when it is not full.

Note: The SPIxTXSR register cannot be written directly by the user. All writes to the SPIxTXSR register are performed through the SPIxBUF register.

Example 3-1: Initialization Code for 16-Bit SPI Master Mode

```
/* The following code example will initialize the SPI1 in Master mode. */
int rData;

IPC2bits.SPI1TXIP = 4;          //Set SPI Interrupt Priorities

SPI1BRGL = 0x1;                // use FPB/4 clock frequency
SPI1STATLbits.SPIROV = 0;      // clear the Overflow
SPI1CON1L = 0x0420;            // 16 bits transfer, Master mode, ckp=0, cke=0, smp=0
SPI1IMSKLbits.SPITBFEN = 1;    // SPI1 transmit buffer full generates interrupt event

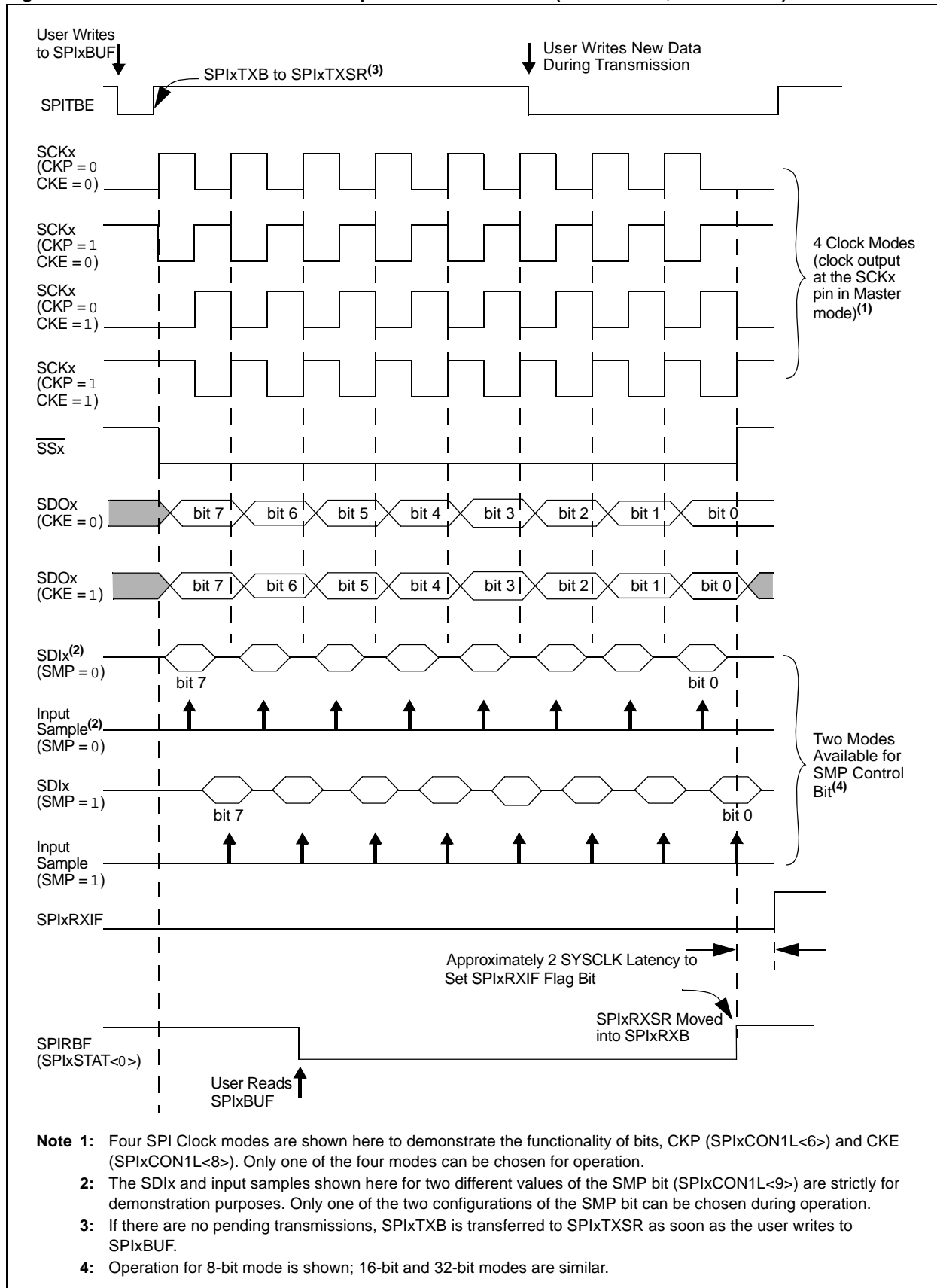
IEC0bits.SPI1TXIE = 1;         // Enable interrupts

SPI1CON1Lbits.SPIEN = 1;

// from here, the device is ready to transmit and receive data. Buffer can be loaded to
transmit data.
```


Serial Peripheral Interface (SPI) with Audio Codec Support

Figure 3-2: SPIx Master Mode Operation in 8-Bit Mode (MODE32 = 0, MODE16 = 0)



3.4.2 SLAVE MODE OPERATION

The following steps are used to set up the SPI module for the Slave mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit.
3. Clear the receive buffer.
4. Clear the ENHBUF bit (SPIxCON1L<0>) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
5. If using interrupts, the following additional steps are performed:
6. Clear the SPIx interrupt flags/events in the respective IFSx register.
7. Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
8. Set the SPIx interrupt enable bits in the respective IECx register.
9. Clear the SPIROV bit (SPIxSTATL<6>).
10. Write the desired settings to the SPIxCON1L register with MSTEN (SPIxCON1L<5>) = 0.
11. Enable SPI operation by setting the SPIEN bit (SPIxCON1L<15>).
12. Transmission (and reception) will start as soon as the master provides the serial clock.

Note: The SPI module must be turned OFF prior to changing the mode from master to slave.

In Slave mode, data is transmitted and received as the external clock pulses appear on the SCKx pin. The CKP bit (SPIxCON1L<6>) and the CKE bit (SPIxCON1L<8>) determine on which edge of the clock data transmission occurs.

Both data to be transmitted and data that is received are respectively written into or read from the SPIxBUFL and SPIxBUFH registers.

The rest of the operation of the module is identical to that in the Master mode, including Enhanced Buffer mode.

3.4.2.1 Slave Mode Additional Features

The following additional features are provided in the Slave mode:

- Slave Select Synchronization

The \overline{SSx} pin allows a Synchronous Slave mode. If the SSEN bit (SPIxCON1L<7>) is set, transmission and reception are enabled in Slave mode only if the \overline{SSx} pin is driven to a low state. 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.

- SPITBE Status Flag Operation

The SPITBE bit (SPIxSTATL<3>) has a different function in the Slave mode of operation. The following describes the function of SPITBE for various settings of the Slave mode of operation:

- If SSEN (SPIxCON1L<7>) is cleared, the SPITBE bit is cleared when SPIxBUF is loaded by the user code. It is set when the module transfers SPIxTXB to SPIxTXSR. This is similar to the SPITBE bit function in Master mode.
- If SSEN is set, SPITBE is cleared when SPIxBUF is loaded by the user code. However, it is set only when the SPI module completes data transmission. A transmission will be aborted when the \overline{SSx} pin goes high and may be retried at a later time. So, each data word is held in SPIxTXB until all bits are transmitted to the receiver.

Note: Slave Select cannot be used when operating in Frame mode.

Serial Peripheral Interface (SPI) with Audio Codec Support

Example 3-2: Initialization Code for 16-Bit SPI Slave Mode

```
/* The following code example will initialize the SPI1 in Slave mode. */
int rData;

IPC14bits.SPI1RXIP = 4;          //Set SPI Interrupt Priorities

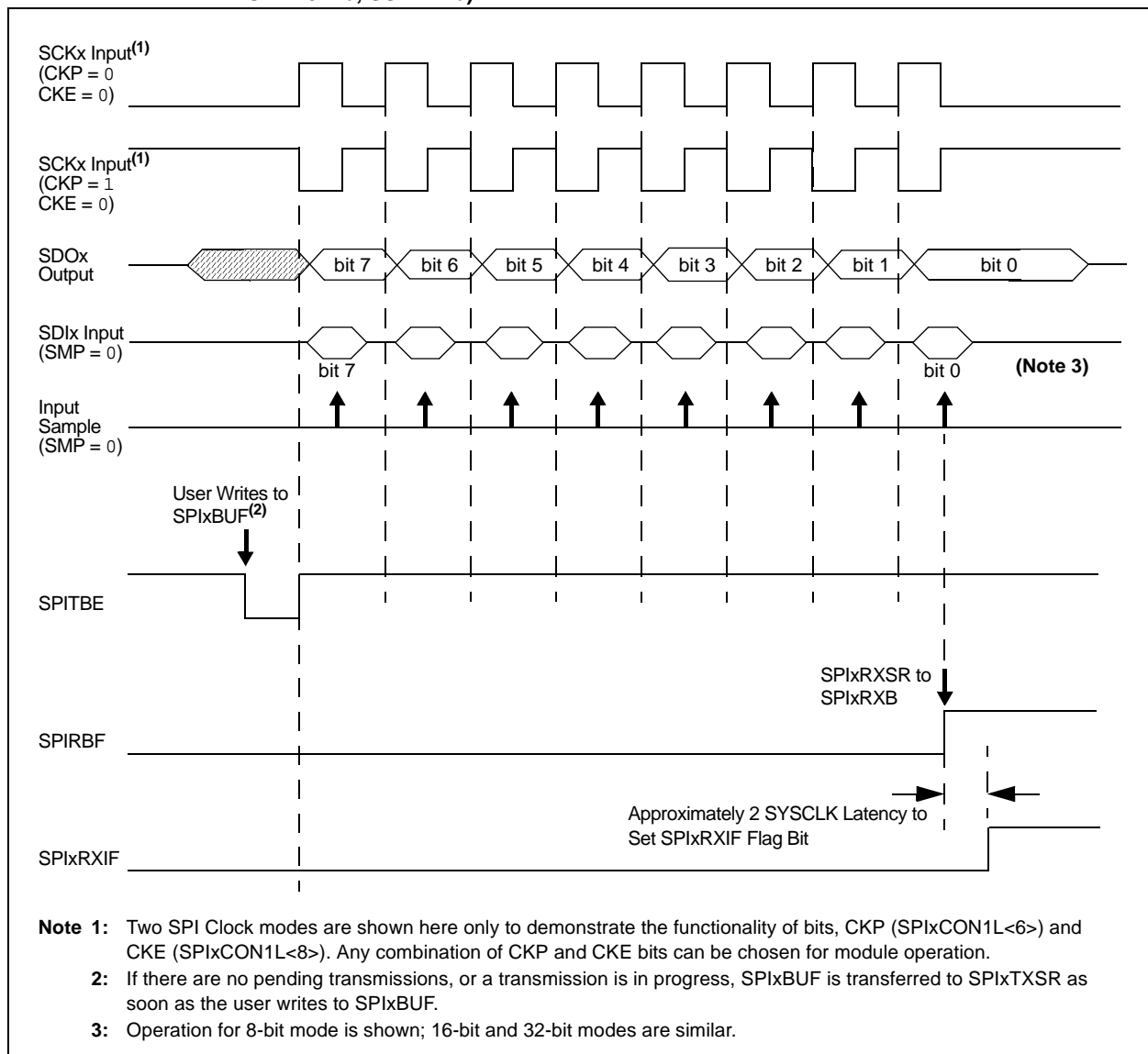
SPI1STATLbits.SPIROV = 0;        // clear the Overflow
SPI1CON1L = 0x0400;              // 16 bits transfer, Slave mode, ckp=0, cke=0, smp=0
SPI1IMSKLbits.SPIRBFE = 1;      // SPI1 receive buffer full generates interrupt event

IEC3bits.SPI1RXIE = 1;          // Enable interrupts

SPI1CON1Lbits.SPIEN = 1;

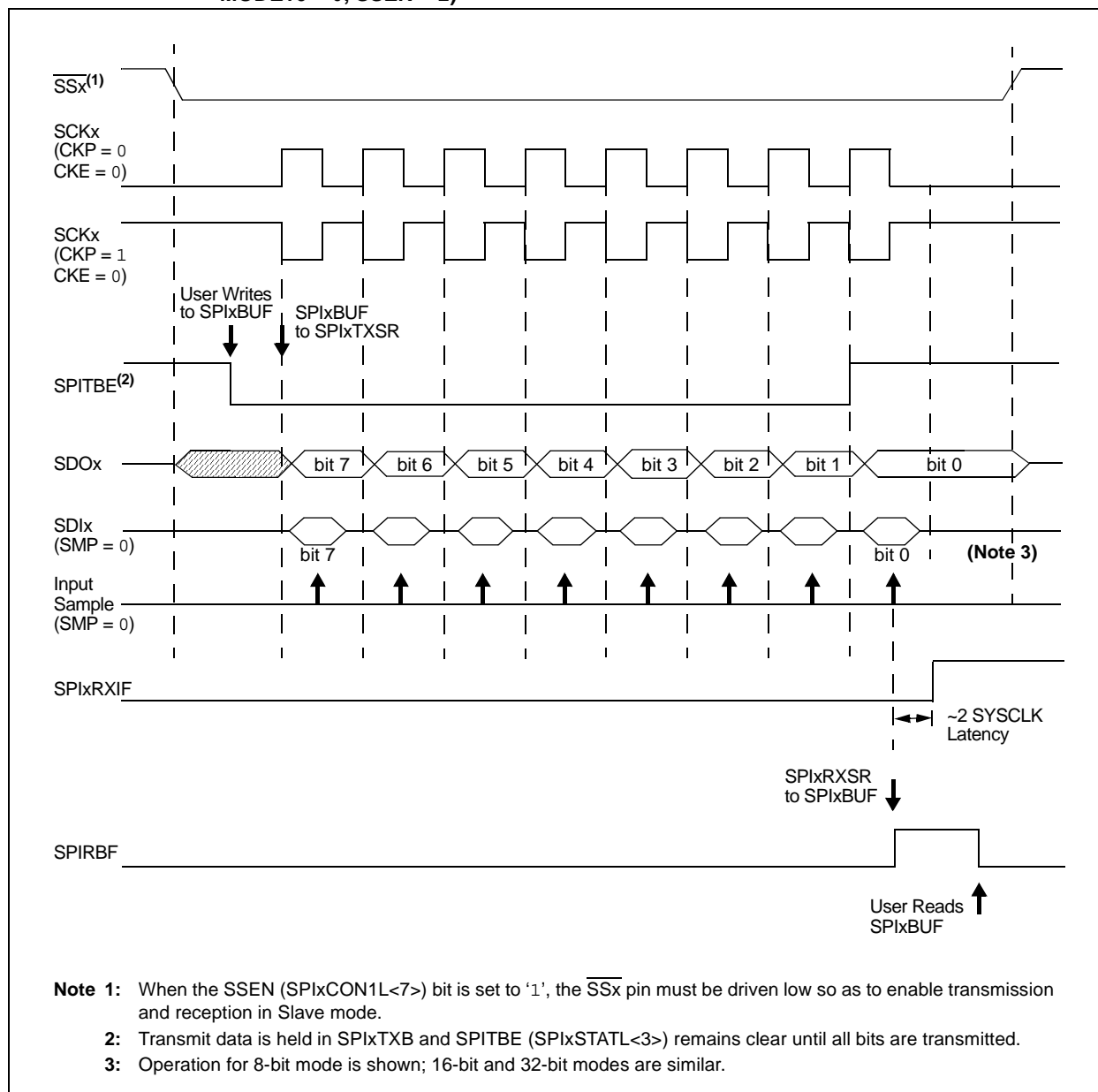
// from here, the device is ready to transmit and receive data. Buffer can be loaded to
// transmit data.
```

Figure 3-3: SPIx Slave Mode Operation in 8-Bit Mode with Slave Select Pin Disabled (MODE32 = 0, MODE16 = 0, SSEN = 0)



dsPIC33/PIC24 Family Reference Manual

Figure 3-4: SPIx Slave Mode Operation in 8-Bit Mode with Slave Select Pin Enabled (MODE32 = 0, MODE16 = 0, SSEN = 1)



3.5 SPI Error Handling

When a new data word has been shifted into the SPIx Shift register, SPIxRXSR, and the previous contents of the SPIx Receive register, SPIxRXB, have not been read by the user software, the SPIROV bit (SPIxSTATL<6>) will be set. The module will not transfer the received data from SPIxRXSR to the SPIxRXB. Further data reception is disabled until the SPIROV bit is cleared. The SPIROV bit is not cleared automatically by the module and must be cleared by the user software.

3.6 SPI Receive Only Operation

Setting the DISSDO control bit (SPIxCON1L<12>) disables transmission at the SDOx pin. This allows the SPI module to be configured for a Receive Only mode of operation. The SDOx pin will be controlled by the respective port function if the DISSDO bit is set.

The DISSDO function is applicable to all SPI operating modes.

3.7 Framed SPI Modes

The module supports a very basic framed SPI protocol while operating in either Master or Slave modes. The following features are provided in the SPI module to support Framed SPI modes:

- The FRMEN control bit (SPIxCON1H<7>) enables Framed SPI mode and causes the \overline{SSx} pin to be used as a Frame Synchronization pulse input or output pin. The state of SSxEN (SPIxCON1L<7>) is ignored.
- The FRMSYNC control bit (SPIxCON1H<6>) determines whether the \overline{SSx} pin is an input or an output (i.e., whether the module receives or generates the Frame Synchronization pulse).
- The FRMPOL control bit (SPIxCON1H<5>) determines the Frame Synchronization pulse polarity for a single SPI clock cycle.
- The FRMSYPW control bit (SPIxCON1H<3>) can be set to configure the width of the Frame Synchronization pulse to one character wide.

Note: The FRMSYPW bit is not available on all devices. Refer to the specific device data sheet for details.
--

- The FRMCNT<2:0> control bits (SPIxCON1H<2:0>) can be set to configure the number of data characters transmitted per Frame Synchronization pulse.

The following Framed SPI modes are supported by the SPI module:

- Frame Master mode

The SPI module generates the Frame Synchronization pulse and provides this pulse to other devices at the \overline{SSx} pin.

- Frame Slave mode

The SPI module uses a Frame Synchronization pulse received at the \overline{SSx} pin.

The Framed SPI modes are supported in conjunction with the Master and Slave modes. Therefore, the following Framed SPI mode configurations are available:

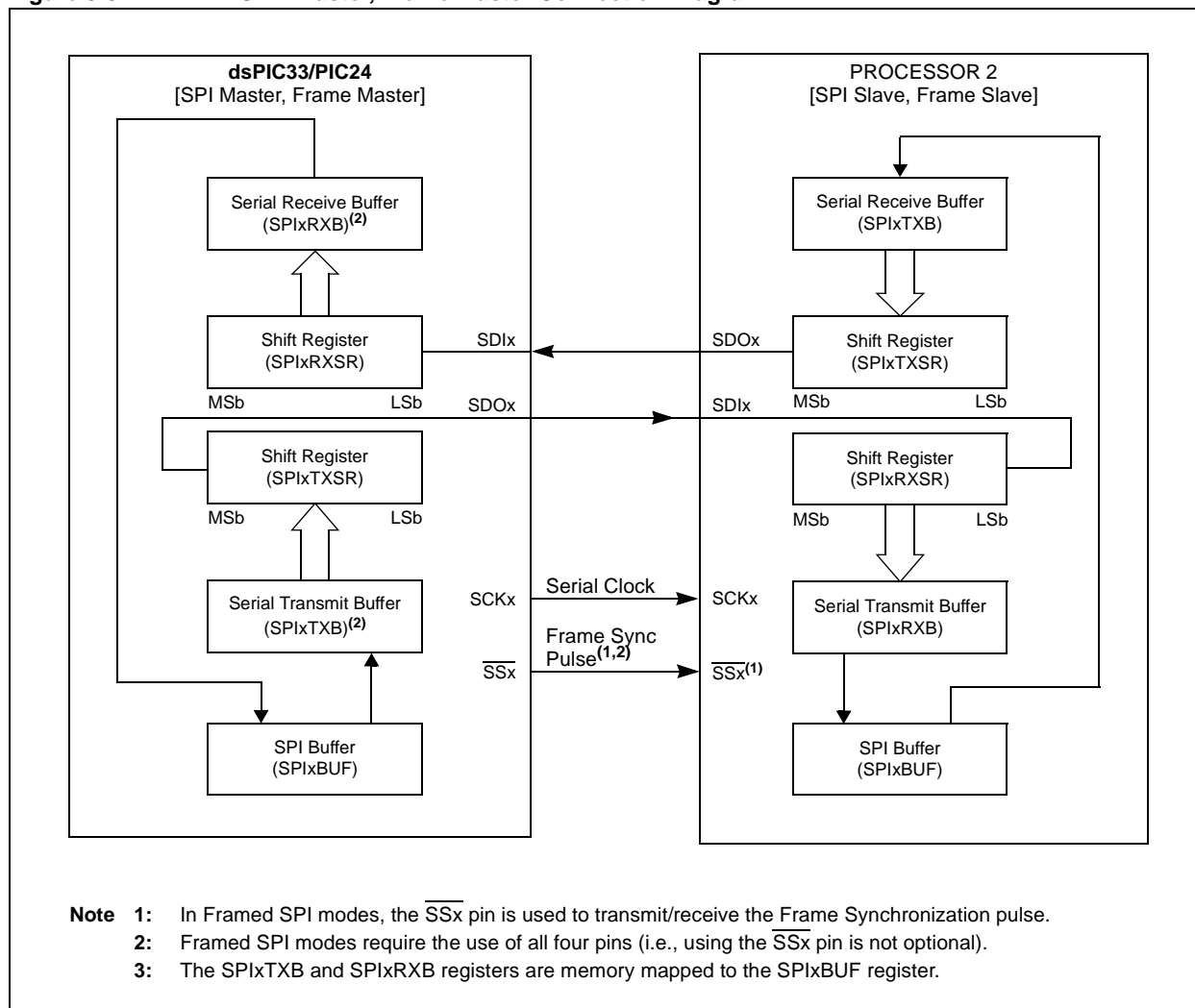
- SPI Master mode and Frame Master mode
- SPI Master mode and Frame Slave mode
- SPI Slave mode and Frame Master mode
- SPI Slave mode and Frame Slave mode

These four modes determine whether or not the SPI module generates the serial clock and the Frame Synchronization pulse.

The ENHBUF bit (SPIxCON1L<0>) can be configured to use the Standard Buffering mode or Enhanced Buffering mode in Framed SPI mode.

In addition, the SPI module can be used to interface to external audio DAC/ADC and codec devices in Framed SPI mode.

Figure 3-5: SPIx Master, Frame Master Connection Diagram



3.7.1 SCKx IN FRAMED SPI MODES

When $FRMEN$ ($SPIxCON1H<7>$) = 1 and $MSTEN$ ($SPIxCON1L<5>$) = 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 ($SPIxCON1L<6>$). The CKE bit ($SPIxCON1L<8>$) is not used for the Framed SPI modes.

When CKP or CKE = 0, the Frame Sync 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 xor CKE = 1, the Frame Sync 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.

Serial Peripheral Interface (SPI) with Audio Codec Support

3.7.2 SPI BUFFERS IN FRAMED SPI MODES

When FRMSYNC (SPIxCON1H<6>) = 0, the SPI module is in the Frame Master mode of operation. In this mode, the Frame Sync pulse is initiated by the module when the user software writes the transmit data to a SPIxBUF location (thus, writing the SPIxTXB register with transmit data). At the end of the Frame Sync pulse, SPIxTXB is transferred to SPIxTXSR and data transmission/reception begins.

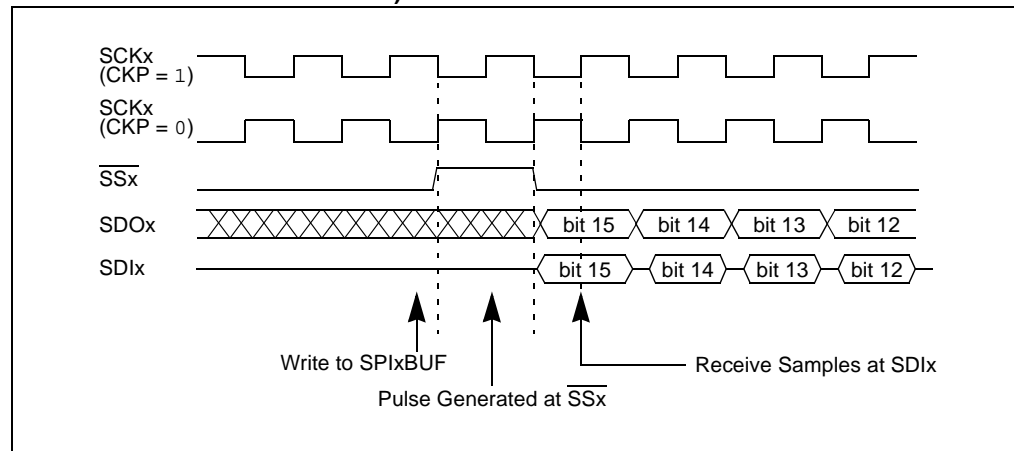
When FRMSYNC = 1, the module is in Frame Slave mode. In this mode, the Frame Sync pulse is generated by an external source. When the module samples the Frame Sync pulse, it will transfer the contents of the SPIxTXB register to SPIxTXSR, and data transmission/reception begins. The user must make sure that the correct data is loaded into the SPIxBUF for transmission before the Frame Sync pulse is received.

Note: Receiving a Frame Sync pulse will start a transmission, regardless of whether or not data was written to SPIxBUF. If a write was not performed, zeros will be transmitted.

3.7.3 SPI MASTER MODE AND FRAME MASTER MODE

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1L<5>) and the FRMEN bit (SPIxCON1H<7>) to '1', and the FRMSYNC bit (SPIxCON1H<6>) to '0'. In this mode, the serial clock will be output continuously at the SCKx pin, regardless of whether the module is transmitting. When SPIxBUF is written, the SSx pin will be driven active-high or active-low, depending on the FRMPOL bit (SPIxCON1H<5>), on the next transmit edge of the SCKx clock. The SSx pin will be high for one SCKx clock cycle. The module will start transmitting data on the next transmit edge of SCKx, as shown in Figure 3-6. A connection diagram indicating signal directions for this operating mode is shown in Figure 3-6.

Figure 3-6: SPIx Master, Frame Master (MODE32 = 0, MODE16 = 1, SPIFE = 0, FRMPOL = 1)



3.7.4 SPI MASTER MODE AND FRAME SLAVE MODE

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1L<5>), the FRMEN bit (SPIxCON1H<7>) and the FRMSYNC bit (SPIxCON1H<6>) to '1'. The \overline{SSx} pin is an input and it is sampled on the sample edge of the SPI clock. When it is sampled active-high or active-low, depending on the FRMPOL bit (SPIxCON1H<5>), data will be transmitted on the subsequent transmit edge of the SPI clock, as shown in Figure 3-7. The SPIx Interrupt Flag, SPIxIF, is set when the transmission is complete. The user must make sure that the correct data is loaded into SPIxBUF for transmission before the signal is received at the \overline{SSx} pin. A connection diagram indicating signal directions for this operating mode is shown in Figure 3-8.

Figure 3-7: SPIx Master, Frame Slave (MODE32 = 0, MODE16 = 1, SPIFE = 0, FRMPOL = 1)

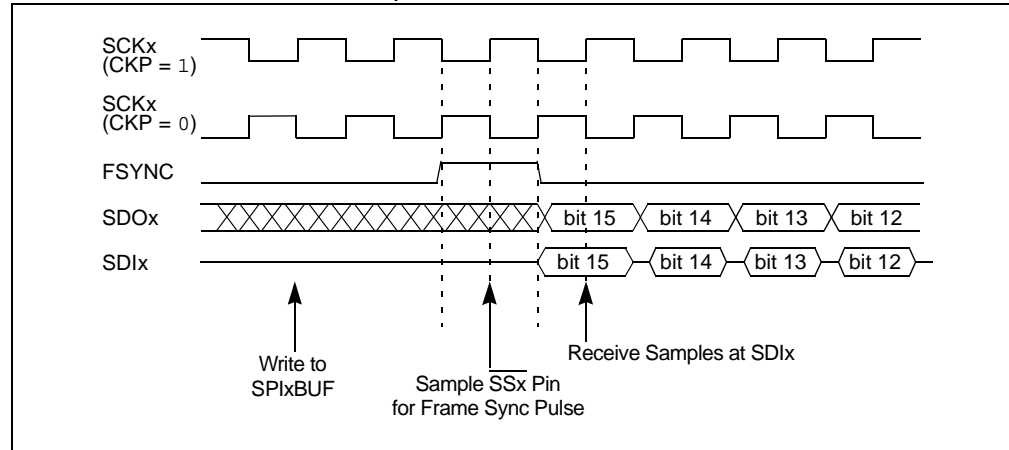
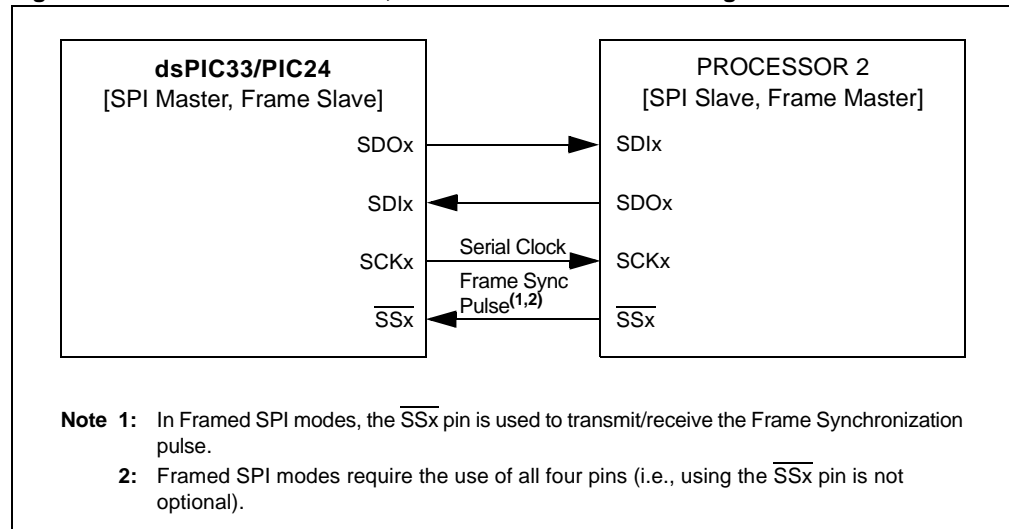


Figure 3-8: SPIx Master, Frame Slave Connection Diagram

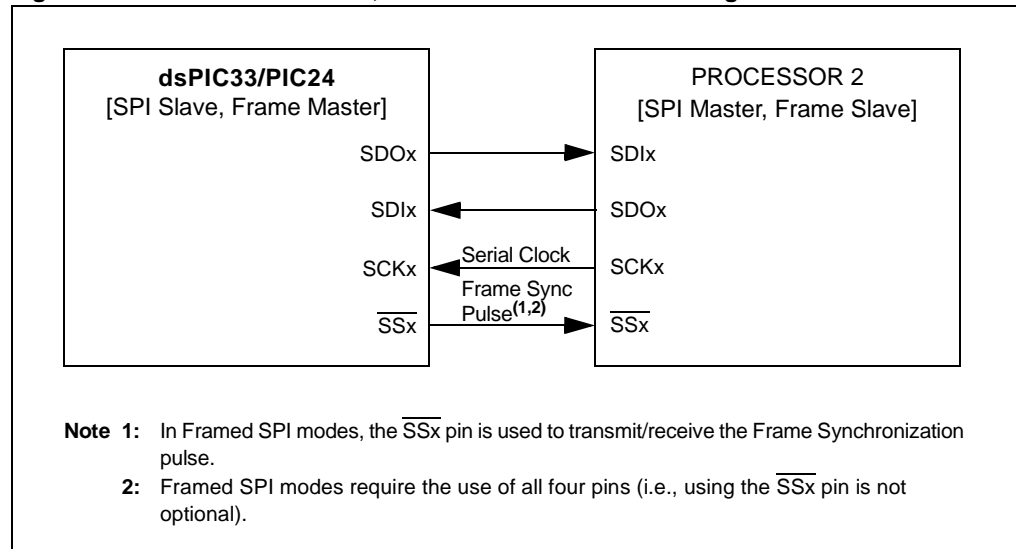


Serial Peripheral Interface (SPI) with Audio Codec Support

3.7.5 SPI SLAVE MODE AND FRAME MASTER MODE

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1L<5>) to '0', the FRMEN bit (SPIxCON1H<7>) to '1' and the FRMSYNC bit (SPIxCON1H<6>) to '0'. The input SPI clock will be continuous in Slave mode. The SSx pin will be an output when bit, FRMSYNC, is low. Therefore, when SPIxBUF is written, the module will drive the SSx pin active-high or active-low, depending on the FRMPOL bit (SPIxCON1H<5>), on the next transmit edge of the SPI clock. The SSx pin will be driven high for one SPI clock cycle. Data transmission will start on the next SPI clock transmit edge. A connection diagram indicating signal directions for this operating mode is shown in [Figure 3-9](#).

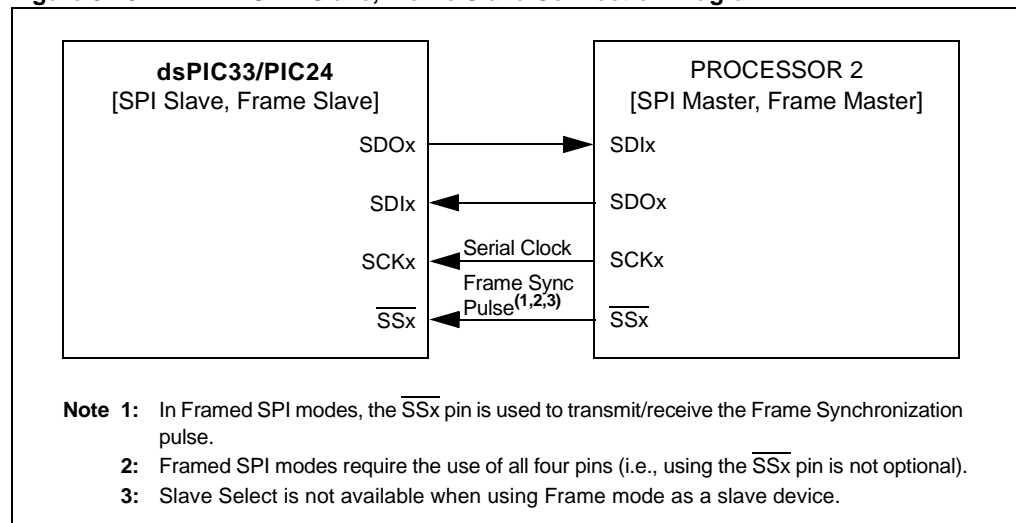
Figure 3-9: SPIx Slave, Frame Master Connection Diagram



3.7.6 SPI SLAVE MODE AND FRAME SLAVE MODE

This Framed SPI mode is enabled by setting the MSTEN bit (SPIxCON1L<5>) to '0', the FRMEN bit (SPIxCON1H<7>) to '1' and the FRMSYNC bit (SPIxCON1H<6>) to '1'. Therefore, both the SCKx and SSx pins will be inputs. The SSx pin will be sampled on the sample edge of the SPI clock. When SSx is sampled active-high or active-low, depending on the FRMPOL bit (SPIxCON1H<5>), data will be transmitted on the next transmit edge of SCKx. A connection diagram indicating signal directions for this operating mode is shown in [Figure 3-10](#).

Figure 3-10: SPIx Slave, Frame Slave Connection Diagram



3.8 SPI Master Mode Clock Frequency

The SPI module allows flexibility in baud rate generation through the 13-bit SPIxBRGL register. SPIxBRGL is readable and writable, and determines the baud rate. The peripheral clock, PBCLK, provided to the SPI module is a divider function of the CPU core clock. This clock is divided based on the value loaded into SPIxBRGL. The SCKx clock, obtained by dividing PBCLK, is 50% duty cycle and it is provided to the external devices through the SCKx pin.

Note: The SCKx clock is not free running for Non-Framed SPI modes. It will only run for 8, 16 or 32 pulses when SPIxBUF is loaded with data. It will, however, be continuous for Framed modes.

Equation 3-1 defines the SCKx clock frequency as a function of SPIxBRGL settings.

Equation 3-1: SCKx Frequency

$$F_{SCK} = \frac{F_{PB}}{2 \cdot (SPIxBRG + 1)}$$

Therefore, the maximum baud rate possible is $F_{PB}/2$ (SPIxBRGL = 0) and the minimum baud rate possible is $F_{PB}/16384$.

Some sample SPI clock frequencies (in kHz) are shown in Table 3-1.

Table 3-1: Sample SCKx Frequencies⁽¹⁾

SPIxBRGL Setting	0	15	31	63	85	127	255	511
FPB = 32 MHz	16.00 MHz	10.0 MHz	500 kHz	257 kHz	190.48 kHz	125 kHz	62.5 kHz	31.25 kHz
FPB = 25 MHz	12.50 MHz	781.25 kHz	390.63 kHz	145.35 kHz	97.66 kHz	281.25 kHz	48.83 kHz	24.41 kHz
FPB = 20 MHz	10.00 MHz	625 kHz	312.50 kHz	156.25 kHz	116.28 kHz	78.13 kHz	39.06 kHz	19.53 kHz
FPB = 12 MHz	6.00 MHz	375 MHz	187.50 kHz	93.75 kHz	69.77 kHz	46.88 kHz	23.44 kHz	11.72 kHz
FPB = 10 MHz	5.00 MHz	312.50 kHz	156.25 kHz	78.13 kHz	58.14 kHz	39.06 kHz	19.53 kHz	9.77kHz
FPB = 8 MHz	4.00 MHz	250 kHz	125 kHz	62.50 kHz	46.51 kHz	31.25 kHz	15.63 kHz	7.81 kHz

Note 1: Not all clock rates are supported. For further information, refer to the SPI timing specifications in the “Electrical Characteristics” chapter of the specific device data sheet.

4.0 AUDIO PROTOCOL INTERFACE MODE

The SPI module can be interfaced to most codec devices available today to provide dsPIC33/PIC24 microcontroller-based audio solutions. The SPI module provides support to the audio protocol functionality through four standard I/O pins. The four pins that make up the audio protocol interface modes are:

- SDIx: Serial Data Input for receiving sample Digital Audio Data (ADCDAT)
- SDOx: Serial Data Output for transmitting Digital Audio Data (DACDAT)
- SCKx: Serial Clock, also known as the Bit Clock (BCLK)
- $\overline{\text{SSx}}$: Left/Right Channel Clock (LRCK)

BCLK provides the clock required to drive the data out or into the module, while LRCK provides the synchronization of the frame based on the Protocol mode selected.

In some codecs, Serial Clock (SCK) refers to the Baud/Bit Clock (BCLK). Throughout this section, the signal, $\overline{\text{SSx}}$, is to be referred to as LRCK to be consistent with codec naming conventions. The SPI module has the ability to function in Audio Protocol Master and Audio Protocol Slave modes. In Master mode, the module generates both the BCLK on the SCKx pin and the LRCK on the $\overline{\text{SSx}}$ pin. In certain devices, while in Slave mode, the module receives these two clocks from its I²S partner, which is operating in Master mode.

While in Master mode, the SPI module has the ability to generate its own clock internally through the Master Clock (MCLK) from various internal sources, such as the primary clock, PBCLK, USB clock, FRC and other internal sources. In addition, the SPI module has the ability to provide the MCLK to the codec device, which is a common requirement.

To start the Audio Protocol mode, first disable the peripheral by setting the SPIEN bit (SPIxCON1L<15>) = 0. Next, set the AUDEN bit (SPIxCON1H<15>) = 1 and then re-enable the peripheral by setting the SPIEN bit = 1.

When configured in Master mode, the leading edge of SCKx and the LRCK is driven out within one SCKx period of starting the audio protocol. Serial data is shifted in or out with timing determined by the Protocol mode set by the AUDMOD<1:0> bits (SPIxCON1H<9:8>). If the transmit FIFO is empty, zeros are transmitted.

In Slave mode, the peripheral drives zeros out of SDOx, but does not transmit the contents of the transmit FIFO until it sees the leading edge of the LRCK, after which time, it starts receiving data (provided SDIx has not been disabled). It will continue to transmit zeros as long as the transmit FIFO is empty.

While in Slave or Master mode, the SPI module does not generate an underrun on the TX FIFO after start-up. This allows software to set up the SPI, set up the DMA, turn on the SPI module's audio protocol and then turn on the DMA without getting an error.

After the first write to the TX FIFO (SPIxBUF), the SPI enables underrun detection and generation. To keep the RX FIFO empty until the DMA is enabled, set DISSDI (SPIxCON1L<4>) = 1. After enabling the DMA, set DISSDI = 0 to start receiving.

4.1 Master Mode

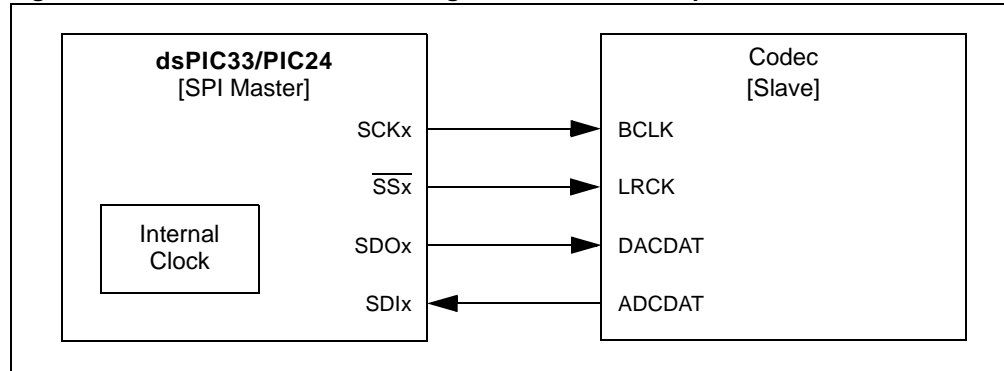
To configure the dsPIC33/PIC24 device in Audio Protocol Master mode, set both the MSTEN bit (SPIxCON1L<5>) and the AUDEN bit (SPIxCON1H<15>) to '1'.

A few characteristics of Master mode are:

- This mode enables the device to generate SCKx and LRCK pulses as long as the SPIEN bit (SPIxCON1L<15>) = 1.
- The SPI module generates LRCK and SCKx continuously in all cases, regardless of the transmit data while in Master mode.
- The SPI module drives the leading edge of LRCK and SCKx within 1 SCKx period, and the serial data shifts in and out continuously, even when the TX FIFO is empty.

Figure 4-1 shows a typical interface between master and slave while in Master mode.

Figure 4-1: Master Generating its Own Clock – Output BCLK and LRCK



4.2 Slave Mode

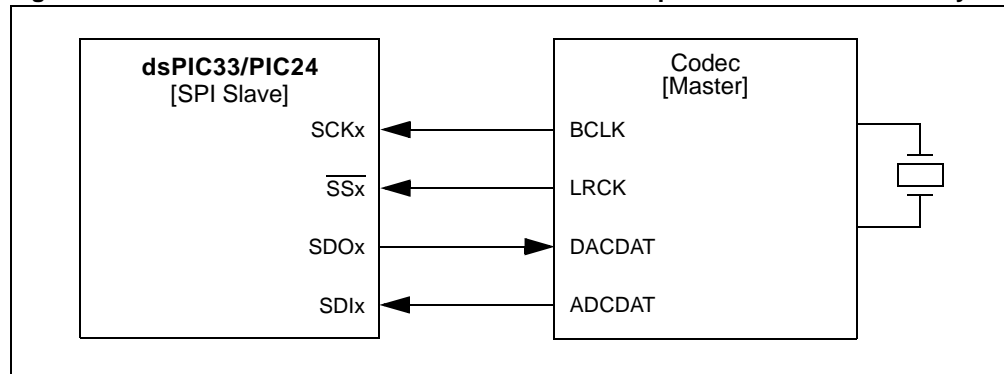
The SPI module can be configured in Audio Protocol Slave mode by setting the MSTEN bit = 0 (SPIxCON1L<5>) and the AUDEN bit = 1 (SPIxCON1H<15>).

A few characteristics of Slave mode are:

- This mode enables the device to receive SCKx and LRCK pulses as long as the SPIEN bit (SPIxCON1L<15>) = 1.
- The SPI module drives zeros out of SDOx, but does not shift data out or in (SDIx) until the module receives the LRCK (i.e., the edge that precedes the left channel).
- Once the module receives the leading edge of LRCK, it starts receiving data if DISSDI (SPIxCON1L<4>) = 0 and the serial data shifts out continuously, even when the TX FIFO is empty.

Figure 4-2 shows the interface between a SPI module in Audio Slave Interface mode to a codec master device.

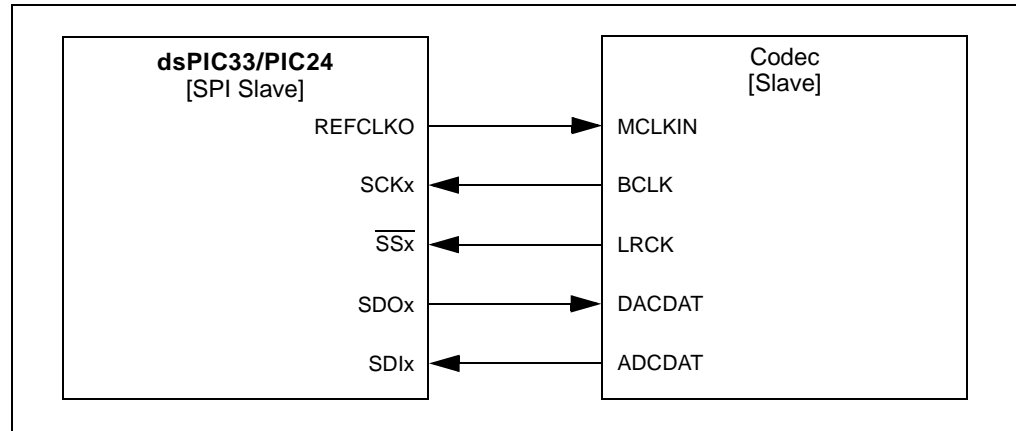
Figure 4-2: Codec Device as Master Generates Required Clock via External Crystal



Serial Peripheral Interface (SPI) with Audio Codec Support

Figure 4-3 shows the interface between an SPI module in Audio Slave Interface mode to a codec master device, in which the Master Clock is being derived from the SPI reference clock out function.

Figure 4-3: Codec Device as Master Derives MCLK from dsPIC33/PIC24 Reference Clock Out



4.3 Audio Data Length and Frame Length

While codec devices may generate audio data samples of various word lengths of 8, 16, 20, 24, 32, the dsPIC33/PIC24 SPI module supports transmit/receive audio data lengths of 16, 24 and 32.

Note: Actual sample data can be any length, with a maximum of 32 bits, and the data must be packed in one of three (16/24/32) formats.

Table 4-1 illustrates how the MODE<32,16> bits (SPIxCON1L<11:10>) control the maximum allowable sample length and frame length (LRCK period on SSx).

Table 4-1: Audio Data Length vs. LRCK Period

SPIxCON1L<11:10>		Data Length (bits)	FIFO Width (bits)	Left/Right Channel Sample Length (bits)	Enhanced Buffer FIFO Depth (samples)	LRCK Period Frame Length (bits)
MODE32	MODE16					
0	0	16	16	≤ 16	8	32
0	1	16	16	≤ 32	8	64
1	1	24	32	≤ 32	4	64
1	0	32	32	≤ 32	4	64

The parameters of the MODE<32,16> bits (SPIxCON1L<11:10>) have the following behavior:

- Controls left/right channel data length, frame length
- In 16-Bit Sample mode, 32/64-bit frame length is supported
- In 24/32-Bit Sample mode, 64-bit frame length is supported
- Defines FIFO width and depth (for example, 24-bit data has a 32-bit wide and 4-location deep FIFO)
- If the written data is greater than the data selected, the upper bytes are ignored
- If the written data is less than the data selected, the FIFO Pointers change on the write to the Most Significant Byte (MSB) of the selected length

If this data is written to the transmit FIFO in more than one write, the write order must be from Least Significant to Most Significant.

For example, assuming that audio data is 24 bits per sample, with 8 bits available at a time. According to Table 4-1, the FIFO width is 32 bits per sample. Therefore, the 8 Most Significant bits (MSBs), bits<31:24>, in each FIFO sample are ignored.

Bits<15:8> and <7:0> can be written to the SPIxBUFL register in any order; however, bits<23:16> must be written last to the SPIxBUFH, as writing the Most Significant bit (bit 24) triggers a change in the pointers of the transmit buffer.

Data written to unused bytes is ignored. Also, transactions that are only to unused bytes are also ignored. Therefore, a byte write to address offset, 0x0023, is completely ignored and does not cause a FIFO push if the data is less than 32 bits wide.

4.4 Frame Error/LRCK Errors

The SPI module provides detection of frame/LRCK errors for debugging. The frame/LRCK error occurs when the LRCK edge, that is defining a channel start, happens before the correct number of bits (as defined by MODE<32,16>).

The SPI module immediately sets the FRMERR bit (SPIxSTATL<12>), pushes data in from the SPIxRXSR register into the SPIxRXB register, and pops data from the SPIxTXB register into the SPIxTXSR register. The module can be configured to detect frame/LRCK related errors by setting the FRMERREN bit (SPIxIMSKL<12>).

Note:	In Audio Protocol mode, both the BCLK (on the SCKx pin) and the LRCK (on the SSx pin) are free running, meaning they are continuous. Normally, the LRCK is a fixed number of BCLKs long. In all cases, the SPI module will realign to the new frame edge and will set the FRMERR bit. If operating in a Non-PCM mode, the SPI module will also push the abbreviated data onto the FIFO when the frame is too short.
--------------	---

4.5 Audio Protocol Modes

The SPI module supports four Audio Protocol modes and can be operated in any one of these modes:

- I²S mode (not available on all devices; refer to the specific device data sheet for availability)
- Left Justified mode
- Right Justified mode
- PCM/DSP mode

These Audio Protocol modes can be enabled by configuring the AUDMOD<1:0> bits (SPIxCON1H<9:8>). These modes enable communication to different types of codecs, and control the edge relationships of LRCK and SDIx/SDOx with respect to SCKx.

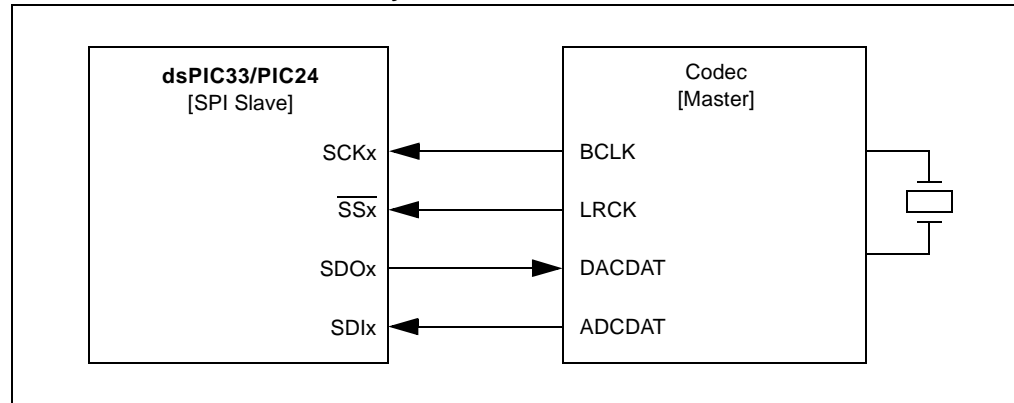
With respect to data transmit, in all of the Protocol modes, the MSB is first transmitted followed by MSB-1, and so on, until the Least Significant Byte (LSB) transmits. The length of the data is discussed in [Section 4.3 “Audio Data Length and Frame Length”](#). If there are SCKx periods left over after the LSb is transmitted, zeros are sent to fill up the frame.

When in Slave mode, the relationship between the BCLK (on the SCKx pin) and the period (or frame length) of the LRCK (on the SSx pin) is far less constrained than that of Master mode. In Master mode, the frame length equals 32 or 64 BCLKs, depending on the MODE<32,16> (SPIxCON1L<11:10>) bit settings. However, in Slave mode, the frame length can be greater than or equal to 32 or 64 BCLKs, but the FRMERR bit (SPIxSTATL<12>) will be set if the frame LRCK edge arrives early.

Serial Peripheral Interface (SPI) with Audio Codec Support

Figure 4-4 illustrates the general interface between the codec device and the SPI module in Audio mode.

Figure 4-4: SPIx Module in Audio Slave Mode – BCLK and WS or LRCK Generated by Master



4.5.1 I²S MODE

Note: This feature is not available on all devices. Refer to the specific device data sheet for availability.

The Inter-IC Sound (I²S) protocol enables transmission of two channels of digital audio data over a single serial interface. The I²S protocol defines a 3-wire interface that handles the stereo data using the WS/LRCK line. The I²S specification defines a half-duplex interface that supports transmit or receive, but not both at the same time. With both SDOx and SDIx available, full-duplex operation is supported by this peripheral, as shown in Figure 4-5.

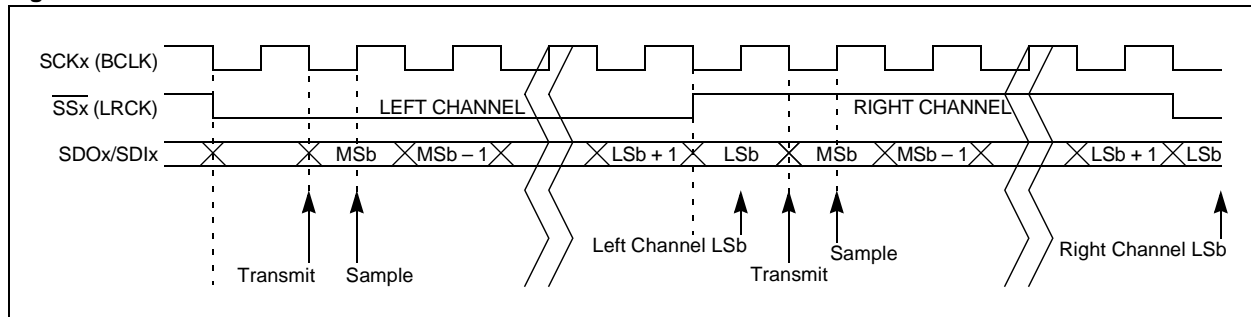
- Data Transmit and Clocking:
 - The transmitter shifts the audio sample data's MSb, on the first falling edge of SCKx, after an LRCK transition
 - The receiver samples the MSb on the second rising edge of SCKx
 - The left channel data shifts out while LRCK is low and the right channel data is shifted out while LRCK is high
 - The data in the left and right channels consists of a single frame
- Required Configuration Settings

To set the module to I²S mode, the following bits must be set:

 - AUDMOD<1:0> = 00 (SPIxCON1H<9:8>)
 - FRMPOL = 0 (SPIxCON1H<5>)
 - CKP = 1 (SPIxCON1L<6>)

Setting these bits enables the SDOx and LRCK (\overline{SSx}) transitions to occur on the falling edge of SCKx (BCLK) and sampling of SDIx to occur on the rising edge of SCKx. Refer to the diagrams shown in Figure 4-5.

Figure 4-5: I²S with 16-Bit Data/Channel or 32-Bit Data/Channel



4.5.1.1 I²S Audio Slave Mode of Operation

Use the following steps to set up the SPI module for the I²S Audio Slave mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit (SPIxCON1L<15>).
3. Reset the SPIx Control Register 1 High, SPIxCON1H.
4. Clear the receive buffer.
5. Clear the ENHBUF bit (SPIxCON1L<0>) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
6. If using interrupts, the following additional steps need to be performed:
 - a) Clear the SPIx interrupt flags/events in the respective IFSx register.
 - b) Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
 - c) Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTATL<6>).
8. Write the desired settings to the SPIxCON1H register:
 - a) AUDMOD<1:0> bits (SPIxCON1H<9:8>) = 00
 - b) AUDEN bit (SPIxCON1H<15>) = 1
9. Write the desired settings to the SPIxCON1L register:
 - a) MSTEN (SPIxCON1L<5>) = 0
 - b) CKP (SPIxCON1L<6>) = 1
 - c) MODE<32,16> (SPIxCON1L<11:10>) = 0 for 16-bit audio channel data.
 - d) Enable SPI operation by setting the SPIEN bit (SPIxCON1L<15>).
10. Transmission (and reception) will start as soon as the master provides the BCLK and LRCK.

Example 4-1: I²S Slave Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in I2S Slave mode. */
int rData;
IPC14bits.SPI1RXIP = 4;

SPI1STATLbits.SPIROV = 0;      // clear the Overflow
SPI1CON1H=0x8000;             // AUDEN=1, I2S mode, stereo mode
SPI1CON1L=0x0440;             // 16 bits/32 channel transfer, Slave mode,ckp=1
SPI1IMSKLbits.SPIRBFEN = 1;   // SPI1 receive buffer full generates interrupt event

IEC3bits.SPI1RXIE = 1;        // Enable interrupts

SPI1CON1Lbits.SPIEN = 1;

// from here, the device is ready to receive and transmit data
```


Serial Peripheral Interface (SPI) with Audio Codec Support

4.5.1.2 I²S Audio Master Mode of Operation

A typical application could be to play PCM data (8 kHz sample frequency, 16-bit data, 32-bit frame size) when interfaced to a codec slave device. In this case, the SPI module is initialized to generate BCLK @ 625 kbps. Assuming a 20 MHz peripheral bus clock, $F_{PB} = 20e6$, the baud rate would be determined using [Equation 4-1](#).

Equation 4-1: Baud Rate Calculation

$$Baud\ Rate = \frac{F_{PB}}{2 \cdot (SPIxBRG + 1)}$$

Solving for the value of SPIxBRGL is shown in [Equation 4-2](#).

Equation 4-2: Baud Rate Calculation

$$SPIxBRG = \frac{F_{PB}}{2(Baud\ Rate)} - 1$$

The Baud Rate is now equal to 256e3. [Equation 4-3](#) shows the resulting calculation.

Equation 4-3: Baud Rate Calculation

$$SPIxBRG = \frac{40e6}{2(256e3)} - 1 = 15$$

If the result of [Equation 4-3](#) is rounded to the nearest integer, SPIxBRGL is now equal to 15; therefore, the effective Baud Rate is that of [Equation 4-4](#).

Equation 4-4: Baud Rate Calculation

$$\frac{20e6}{2 \cdot (15 + 1)} = \frac{20e6}{32} = 625000\ bits\ per\ second$$

The following steps can be used to set up the SPI module for operation in I²S Audio Master mode:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit (SPIxCON1L<15>).
3. Reset the SPIx Control Register 1 High, SPIxCON1H.
4. Reset the SPIx Baud Rate Register Low, SPIxBRGL.
5. Clear the receive buffer.
6. Clear the ENHBUF bit (SPIxCON1L<0>) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
7. If using interrupts, perform these additional steps:
 - a) Clear the SPIx interrupt flags/events in the respective IFSx register.
 - b) Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
 - c) Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTATL<6>).
9. Write the desired settings to the SPIxCON1H register. The AUDMOD<1:0> bits (SPIxCON1H<9:8>) must be set to '00' for I²S mode and the AUDEN bit (SPIxCON1H<15>) must be set to '1' to enable the audio protocol.
10. Set the SPIxBRGL register to 0x4F (to generate approximately 625 kbps sample rate with PBCLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1L register:
 - a) MSTEN (SPIxCON1L<5>) = 1.
 - b) CKP (SPIxCON1L<6>) = 1.
 - c) MODE<32,16> (SPIxCON1L<11:10>) = 0 for 16-bit audio channel data.
 - d) Enable SPI operation by setting the SPIEN bit (SPIxCON1L<15>).
12. Transmission (and reception) will start immediately after the SPIEN bit is set.

Example 4-2: I²S Master Mode, 625 kbps BCLK, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in I2S Master mode. */
int rData;

IPC2bits.SPI1TXIP = 4;

SPI1STATLbits.SPIROV = 0;      // clear the Overflow
SPI1CON1H = 0x8000;           // AUDEN=1, I2S mode, stereo mode
SPI1BRGL = 0x000F;            // to generate 625 kbps sample rate, PBCLK @ 20 MHz
SPI1CON1L = 0x0460;           // 16 bits/32 channel transfer, Master mode, ckp=1
SPI1IMSKLbits.SPITBFEN = 1;   // SPI1 transmit buffer full generates interrupt event

IEC0bits.SPI1TXIE = 1;        // Enable interrupts

SPI1CON1Lbits.ENHBUF = 1;
SPI1CON1Lbits.SPIEN = 1;

// from here, the device is ready to receive and transmit data
```

Serial Peripheral Interface (SPI) with Audio Codec Support

4.5.2 LEFT JUSTIFIED MODE

The Left Justified mode is similar to I²S mode, however, in this mode, the SPI shifts the audio data's MSb on the first SCKx edge that is coincident with an LRCK transition. On the receiver side, the SPI module samples the MSb on the next SCKx edge.

In general, a codec using justified protocols defaults to transmitting data on the rising edge of SCKx and receiving data on the falling edge of SCKx.

- Required Configuration Settings

To set the module to Left Justified mode, the following bits must to be set:

- AUDMOD<1:0> = 01 (SPIxCON1H<9:8>)
- FRMPOL = 1 (SPIxCON1H<5>)
- CKP = 0 (SPIxCON1L<6>)

This enables the SDOx and LRCK transitions to occur on the rising edge of SCKx. Refer to the sample waveform diagrams shown in [Figure 4-6](#) and [Figure 4-7](#) for 16, 24 and 32-bit audio data transfers.

Figure 4-6: Left Justified with 16-Bit Data/Channel or 32-Bit Data/Channel

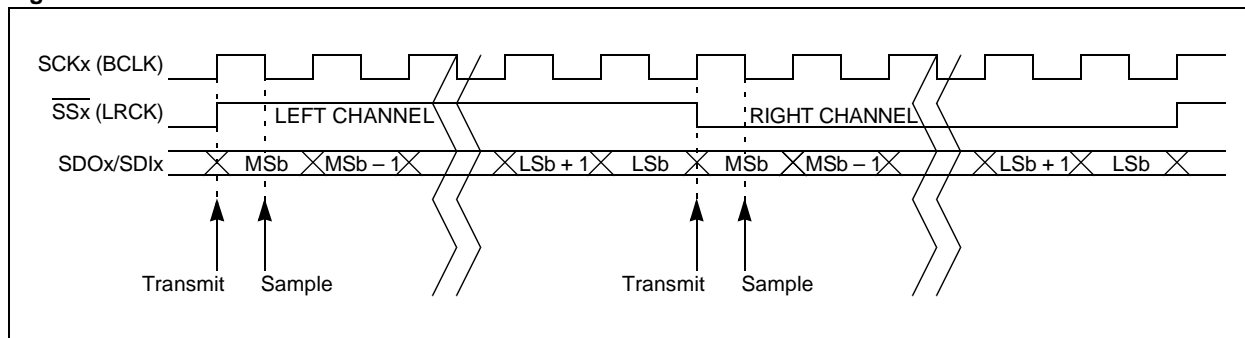
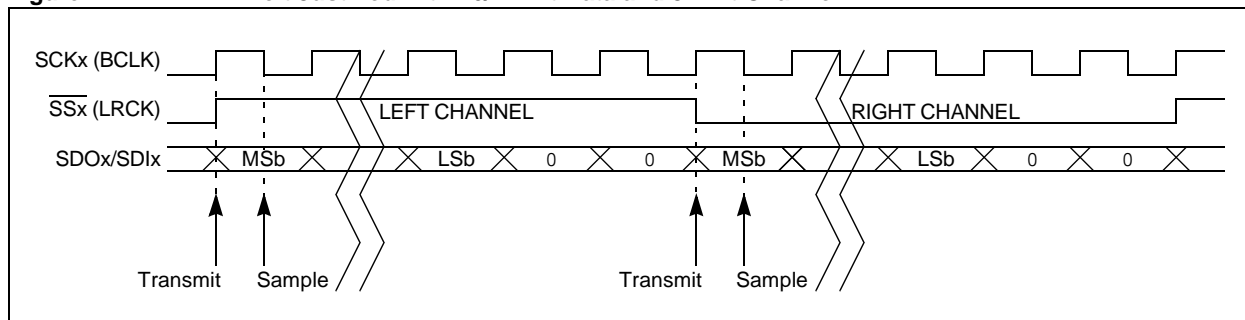


Figure 4-7: Left Justified with 16/24-Bit Data and 32-Bit Channel



4.5.2.1 Left Justified Audio Slave Mode Operation

Use the following steps to set up the SPI module for the Left Justified Audio Slave mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit (SPIxCON1L<15>).
3. Reset the SPIx Control Register 1 High, SPIxCON1H.
4. Clear the receive buffer.
5. Clear the ENHBUF bit (SPIxCON1L<0>) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
6. If using interrupts, the following additional steps are performed:
 - a) Clear the SPIx interrupt flags/events in the respective IFSx register.
 - b) Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
 - c) Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTATL<6>).
8. Write the desired settings in the SPIxCON1H register. The AUDMOD<1:0> bits (SPIxCON1H<9:8>) must be set to '01' for Left Justified mode and the AUDEN bit (SPIxCON1H<15>) must be set to '1' to enable the audio protocol.
9. Write the desired settings to the SPIxCON1L register:
 - a) Set to Slave mode, MSTEN (SPIxCON1L<5>) = 0.
 - b) Set Clock Polarity, CKP (SPIxCON1L<6>) = 0.
 - c) Set Frame Polarity, FRMPOL (SPIxCON1H<5>) = 1.
 - d) Set MODE<32,16> (SPIxCON1L<11:10>) = 0 for 16-bit audio channel data
 - e) Enable SPI operation by setting the SPIEN bit (SPIxCON1L<15>).
10. Transmission (and reception) will start as soon as the master provides the BCLK and LRCK.

Example 4-3: Left Justified Slave Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Left-Justified Slave mode. */
int rData;
IPC14bits.SPI1RXIP = 4;

SPI1STATLbits.SPIROV = 0;      // clear the Overflow
SPI1CON1H = 0x8120;           // AUDEN=1, Left-Justified mode, stereo mode,FRMPOL=1
SPI1CON1L = 0x0400;           // 16 bits/32 channel transfer, Slave mode,ckp=0
SPI1IMSKLbits.SPIRBFEN = 1;   // SPI1 receive buffer full generates interrupt event

IEC3bits.SPI1RXIE = 1;        // Enable interrupts

SPI1CON1Lbits.ENHBUF = 1;
SPI1CON1Lbits.SPIEN = 1;
// from here, the device is ready to receive and transmit data
```

Serial Peripheral Interface (SPI) with Audio Codec Support

4.5.2.2 Left Justified Audio Master Mode Operation

Use the following steps to set up the SPI module for the Left Justified Audio Master mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit (SPIxCON1L<15>).
3. Reset the SPIx Control Register 1 High, SPIxCON1H.
4. Reset the SPIx Baud Rate Register Low, SPIxBRGL.
5. Clear the receive buffer.
6. Clear the ENHBUF bit (SPIxCON1L<0>) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
7. If using interrupts, the following additional steps are performed:
 - a) Clear the SPIx interrupt flags/events in the respective IFSx register.
 - b) Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
 - c) Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTATL<6>).
9. Write the desired settings in the SPIxCON1H register. The AUDMOD<1:0> bits (SPIxCON1H<9:8>) must be set to '01' for left justified and the AUDEN bit (SPIxCON1H<15>) must be set to '1' to enable the audio protocol.
10. Set the SPIx Baud Rate Register Low, SPIxBRGL, to 0x4F (to generate approximately 256 kbps sample rate with PBCLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1L register:
 - a) Set to Master mode, MSTEN (SPIxCON1L<5>) = 1.
 - b) Set Clock Polarity, CKP (SPIxCON1L<6>) = 0.
 - c) Set Frame Polarity, FRMPOL (SPIxCON1H<5>) = 1.
 - d) Set MODE<32,16> (SPIxCON1L<11:10>) = 0 for 16-bit audio channel data.
 - e) Enable SPI operation by setting the SPIEN bit (SPIxCON1L<15>).
12. Transmission (and reception) will start immediately after the SPIEN bit is set.

Example 4-4: Left Justified Master Mode, 625 kbps BLCK, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Left-Justified Master mode. */
int rData;
IPC2bits.SPI1TXIP = 4;

SPI1STATLbits.SPIROV = 0;      // clear the Overflow
SPI1CON1H = 0x8120;           // AUDEN=1, Left-Justified mode, stereo mode, FRMPOL = 1
SPI1BRGL = 0x000F;            // to generate 625 kbps sample rate, PBCLK @ 20 MHz
SPI1CON1L = 0x0420;           // 16 bits/32 channel transfer, Master mode, ckp=0
SPI1IMSKLbits.SPITBFEN = 1;   // SPI1 transmit buffer full generates interrupt event

IEC0bits.SPI1TXIE = 1;        // Enable interrupts

SPI1CON1Lbits.ENHBUF = 1;
SPI1CON1Lbits.SPIEN = 1;
// from here, the device is ready to receive and transmit data
```

4.5.3 RIGHT JUSTIFIED MODE

In Right Justified mode, the SPI module shifts the audio sample data's MSb after aligning the data to the last clock cycle. The bits preceding the audio sample data can be driven to logic level '0' by setting the DISSDO bit (SPIxCON1L<12>) to '0'. When DISSDO = 0, the module ignores the unused bit slot.

- Required Configuration Settings

To set the module to Right Justified mode, the following bits must to be set:

- AUDMOD<1:0> (SPIxCON1H<9:8>) = 10
- FRMPOL (SPIxCON1H<5>) = 1
- CKP (SPIxCON1L<6>) = 0

This enables the SDOx and LRCK transitions to occur on the rising edge of SCKx, after the LSb is aligned to the last clock cycle. Refer to the sample waveform diagrams shown in [Figure 4-8](#) and [Figure 4-9](#) for 16, 24 and 32-bit audio data transfers.

Figure 4-8: Right Justified with 16-Bit Data/Channel or 32-Bit Data/Channel

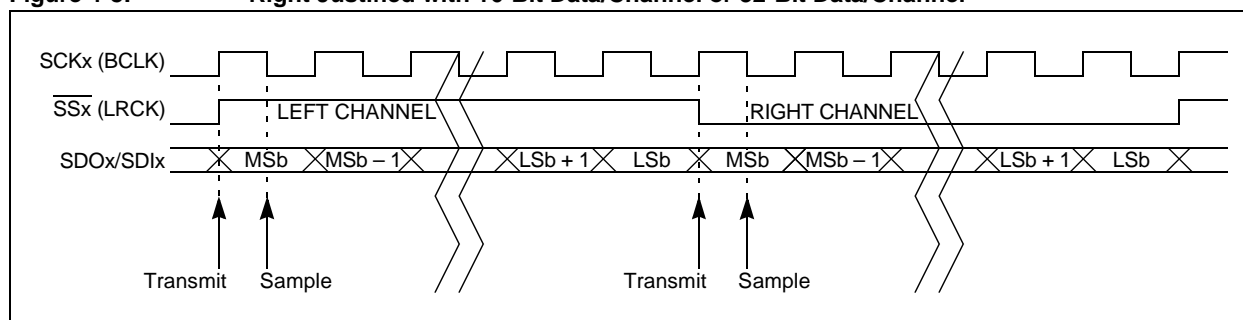
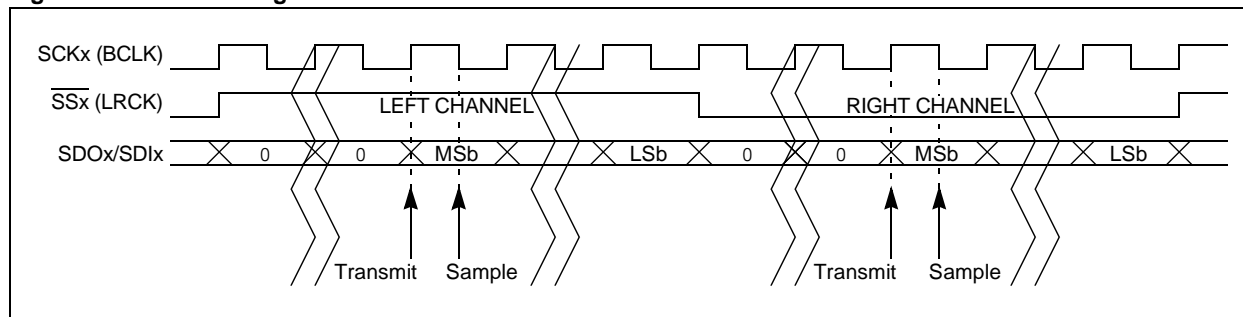


Figure 4-9: Right Justified with 16/24-Bit Data and 32-Bit Channel



Serial Peripheral Interface (SPI) with Audio Codec Support

4.5.3.1 Right Justified Audio Slave Mode Operation

Use the following steps to set up the SPI module for the Right Justified Audio Slave mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit (SPIxCON1L<15>).
3. Reset the SPIx Control Register 1 High, SPIxCON1H.
4. Clear the receive buffer.
5. Clear the ENHBUF bit (SPIxCON1L<0>) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
6. If using interrupts, perform the following steps:
 - a) Clear the SPIx interrupt flags/events in the respective IFSx register.
 - b) Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
 - c) Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTATL<6>).
8. Write the desired settings in the SPIxCON1H register. The AUDMOD<1:0> bits (SPIxCON1H<9:8>) must be set to '10' for Right Justified mode and the AUDEN bit (SPIxCON1H<15>) must be set to '1' to enable the audio protocol.
9. Write the desired settings to the SPIxCON1L register:
 - a) Set to Slave mode, MSTEN (SPIxCON1L<5>) = 0.
 - b) Set Clock Polarity, CKP (SPIxCON1L<6>) = 0.
 - c) Set Frame Polarity, FRMPOL (SPIxCON1H<5>) = 1.
 - d) Set MODE<32,16> (SPIxCON1L<11:10>) = 0 for 16-bit audio channel data.
 - e) Enable SPI operation by setting the SPIEN bit (SPIxCON1L<15>).
10. Transmission (and reception) will start as soon as the master provides the BCLK and LRCK.

Example 4-5: Right Justified Slave Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Right-Justified Slave mode. */
int rData;
IPC14bits.SPI1RXIP = 4;

SPI1STATLbits.SPIROV = 0;      // clear the Overflow
SPI1CON1H = 0x8220;           // AUDEN=1, Right-Justified mode, stereo mode, FRMPOL=1
SPI1CON1L = 0x0400;           // 16 bits/32 channel transfer, Slave mode, ckp=0
SPI1IMSKLbits.SPIRBFE = 1;    // SPI1 receive buffer full generates interrupt event

IEC3bits.SPI1RXIE = 1;        // Enable interrupts

SPI1CON1Lbits.ENHBUF = 1;
SPI1CON1Lbits.SPIEN = 1;
// from here, the device is ready to receive and transmit data
```

4.5.3.2 Right Justified Audio Master Mode Operation

Use the following steps to set up the SPI module for the Right Justified Audio Master mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit (SPIxCON1L<15>).
3. Reset the SPIx Control Register 1 High, SPIxCON1H.
4. Reset the SPIx Baud Rate Register Low, SPIxBRGL.
5. Clear the receive buffer.
6. Clear the ENHBUF bit (SPIxCON1L<0>) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
7. If using interrupts, the following additional steps are performed:
 - a) Clear the SPIx interrupt flags/events in the respective IFSx register.
 - b) Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
 - c) Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTATL<6>).
9. Write the desired settings in the SPIxCON1H register. The AUDMOD<1:0> bits (SPIxCON1H<9:8>) must be set to '10' for Right Justified mode and the AUDEN bit (SPIxCON1H<15>) must be set to '1' to enable the audio protocol.
10. Set the SPIx Baud Rate Register Low, SPIxBRGL, to 0x0F (to generate approximately 256 kbps sample rate with PBCLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1L register:
 - a) Set to Master mode, MSTEN (SPIxCON1L<5>) = 1.
 - b) Set Clock Polarity, CKP (SPIxCON1L<6>) = 0.
 - c) Set Frame Polarity, FRMPOL (SPIxCON1H<5>) = 1.
 - d) Set MODE<32,16> (SPIxCON1L<11:10>) = 0 for 16-bit audio channel data.
 - e) Enable SPI operation by setting the SPIEN bit (SPIxCON1L<15>).
12. Transmission (and reception) will start immediately after the SPIEN bit is set.

Example 4-6: Right Justified Master Mode, 625 kbps BLCK, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in Right-Justified Master mode. */
int rData;
IPC2bits.SPI1TXIP = 4;

SPI1STATbits.SPIROV = 0;      // clear the Overflow
SPI1CON1H = 0x8220;           // AUDEN=1, Right-Justified mode, stereo mode
SPI1BRGL = 0x000F;           // to generate 625 kbps sample rate, PBCLK @ 20 MHz
SPI1CON1L = 0x0420;           // 16 bits/32 channel transfer, Master mode, ckp=0
SPI1IMSKLbits.SPITBFEN = 1;  // SPI1 transmit buffer full generates interrupt event

IEC0bits.SPI1TXIE = 1;       // Enable interrupts

SPI1CON1Lbits.ENHBUF = 1;
SPI1CON1Lbits.SPIEN = 1;
// from here, the device is ready to receive and transmit data
```


Serial Peripheral Interface (SPI) with Audio Codec Support

4.5.4 PCM/DSP MODE

The PCM/DSP Protocol mode is available for communication with some codecs and certain DSP (Digital Signal Processor) devices. This mode modifies the behavior of LRCK and audio data spacing. In PCM/DSP mode, the LRCK can be a single bit wide (i.e., 1 SCKx) or as wide as the audio data (16, 24, 32 bits). The audio data is packed in the frame with the left channel data immediately followed by the right channel data. The frame length is still either 32 or 64 clocks when this device is the master.

In PCM/DSP mode, the transmitter drives the audio data's (left channel) MSb on the first or second transmit edge (see the SPIFE bit (SPIxCON1L<1>)) of SCKx (after an LRCK transition). Immediately after the (left channel) LSb, the transmitter drives the (right channel) MSb.

- Required Configuration Settings

To set the module to Left Justified mode, the following bits must to be set:

- AUDMOD<1:0> bits (SPIxCON1H<9:8>) = 11

Refer to the sample waveform diagrams shown in [Figure 4-10](#) and [Figure 4-11](#) for 16, 24 and 32-bit audio data transfers.

Figure 4-10: PCM/DSP with 16-Bit Data/Channel or 32-Bit Data/Channel

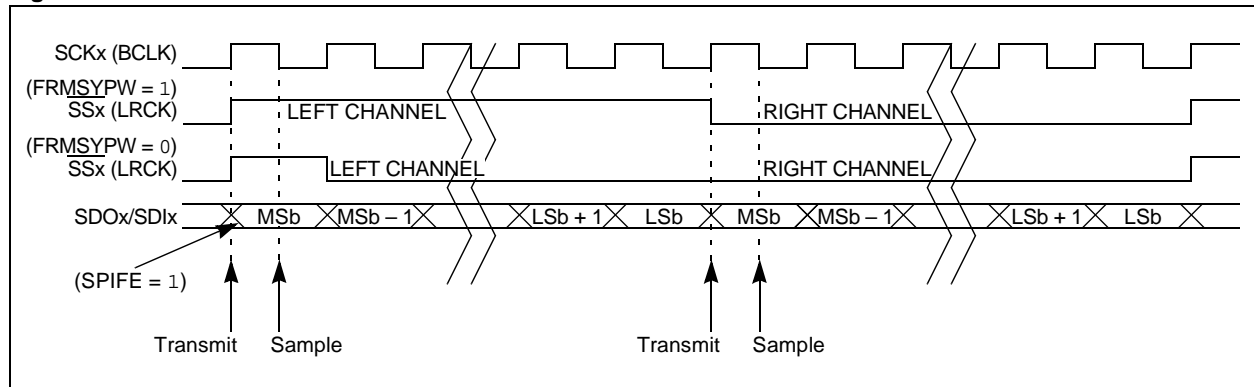
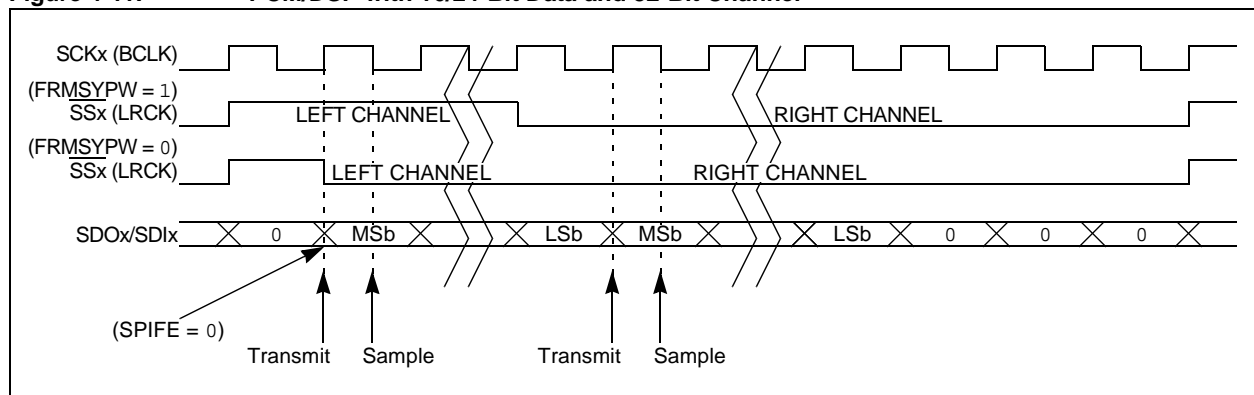


Figure 4-11: PCM/DSP with 16/24-Bit Data and 32-Bit Channel



4.5.4.1 PCM/DSP Audio Slave Mode of Operation

Use the following steps to set up the SPI module for the PCM/DSP Audio Slave mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit (SPIxCON1L<15>).
3. Reset the SPIx Control Register 1 High, SPIxCON1H.
4. Clear the receive buffer.
5. Clear the ENHBUF bit (SPIxCON1L<0>) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
6. If using interrupts, the following additional steps are performed:
 - a) Clear the SPIx interrupt flags/events in the respective IFSx register.
 - b) Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
 - c) Set the SPIx interrupt enable bits in the respective IECx register.
7. Clear the SPIROV bit (SPIxSTATL<6>).
8. Write the desired setting in the SPIxCON1H register. The AUDMOD<1:0> bits (SPIxCON1H<9:8>) must be set to '11' for DSP/PCM mode and the AUDEN bit (SPIxCON1H<15>) must be set to '1' to enable audio protocol.
9. Write the desired settings to the SPIxCON1L register:
 - a) Set to Slave mode, MSTEN (SPIxCON1L<5>) = 0.
 - b) Set MODE<32,16> (SPIxCON1L<11:10>) = 0 for 16-bit audio channel data.
 - c) Enable SPI operation by setting the SPIEN bit (SPIxCON1L<15>).
10. Transmission (and reception) will start as soon as the master provides the BCLK and LRCK.

Example 4-7: PCM/DSP Slave Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in PCM/DSP Slave Mode. */
int rData;
IPC14bits.SPI1RXIP = 4;

SPI1STATbits.SPIROV = 0;      // clear the Overflow
SPI1CON1H = 0x8320;           // AUDEN=1, PCM/DSP mode, stereo mode, FRMPOL=1
SPI1CON1L = 0x0400;           // 16 bits/32 channel transfer, Slave mode, ckp=0
SPI1IMSKLbits.SPIRBFFEN = 1;  // SPI1 receive buffer full generates interrupt event

IEC3bits.SPI1RXIE = 1;        // Enable interrupts

SPI1CON1Lbits.ENHBUF = 1;
SPI1CON1Lbits.SPIEN = 1;
// from here, the device is ready to receive and transmit data
```

Serial Peripheral Interface (SPI) with Audio Codec Support

4.5.4.2 PCM/DSP Audio Master Mode of Operation

Use the following steps to set up the SPI module for the PCM/DSP Audio Master mode of operation:

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit (SPIxCON1L<15>).
3. Reset the SPIx Control Register 1 High, SPIxCON1H.
4. Reset the SPIx Baud Rate Register Low, SPIxBRGL.
5. Clear the receive buffer.
6. Clear the ENHBUF bit (SPIxCON1L<0>) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
7. If using interrupts, perform the following steps:
 - a) Clear the SPIx interrupt flags/events in the respective IFSx register.
 - b) Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
 - c) Set the SPIx interrupt enable bits in the respective IECx register.
8. Clear the SPIROV bit (SPIxSTATL<6>).
9. Write the desired settings in the SPIxCON1H register. The AUDMOD<1:0> bits (SPIxCON1H<9:8>) must be set to '11' for DSP/PCM mode and the AUDEN bit (SPIxCON1H<15>) must be set to '1' to enable the audio protocol.
10. Set the SPIx Baud Rate Register Low, SPIxBRGL, to 0x0F (to generate approximately 256 kbps sample rate with PBCLK @ 20 MHz).
11. Write the desired settings to the SPIxCON1L register:
 - a) Set to Master mode, MSTEN (SPIxCON1L<5>) = 1.
 - b) Set MODE<32,16> (SPIxCON1L<11:10>) = 0 for 16-bit audio channel data.
 - c) Enable SPI operation by setting the SPIEN bit (SPIxCON1L<15>).
12. Transmission (and reception) will start immediately after the SPIEN bit is set.

Example 4-8: PCM/DSP Master Mode, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in PCM/DSP Master Mode. */
int rData;
IPC2bits.SPI1TXIP = 4;

SPI1STATLbits.SPIROV = 0;      // clear the Overflow
SPI1CON1H = 0x8320;           // AUDEN=1, PCM/DSP mode, stereo mode
SPI1BRGL = 0x000F;            // to generate 625 kbps sample rate, PBCLK @ 20 MHz
SPI1CON1L = 0x0420;           // 16 bits/32 channel transfer, Master mode, ckp=0
SPI1IMSKLbits.SPITBFEN = 1;   // SPI1 transmit buffer full generates interrupt event

IEC0bits.SPI1TXIE=1;          // Enable interrupts

SPI1CON1Lbits.ENHBUF = 1;
SPI1CON1Lbits.SPIEN = 1;
// from here, the device is ready to receive and transmit data
```

4.6 Audio Protocol Mode Features

4.6.1 BCLK/SCKx AND LRCK GENERATION

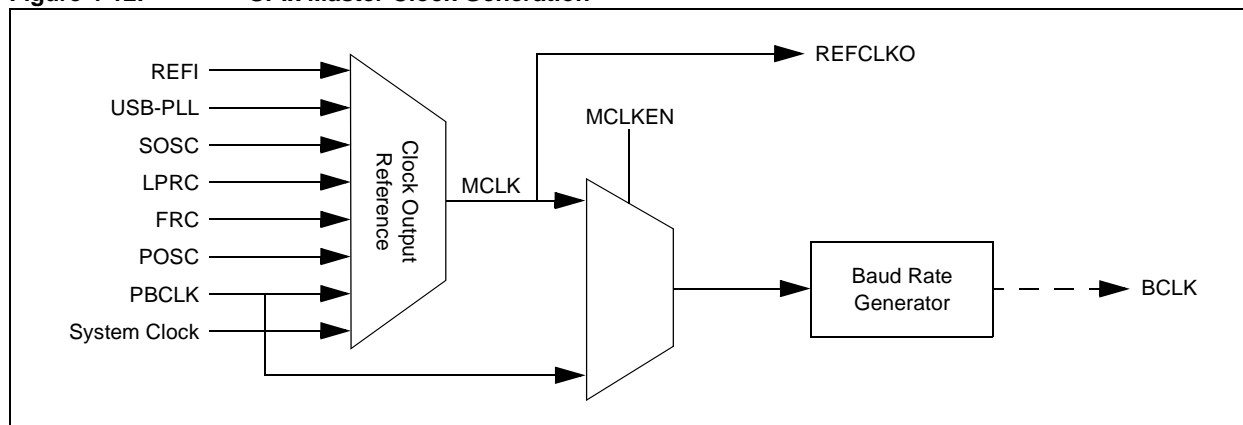
BCLK and LRCK generation is a key requirement in Master mode. The frame frequency of SCKx and LRCK is defined by the MODE<32,16> bits (SPIxCON1L<11:10>). When the frame is 64 bits, SCKx is 64 times the frequency of LRCK. Similarly, when the frame is 32 bits, SCKx is 32 times the frequency of LRCK. The frequency of SCKx must be derived from the toggling rate of LRCK and the frame size.

For example, to sample 16-bit channel data at 8 kHz with PBCLK = 36.864 MHz, set the SPIxBRGL register to 0x47 to generate an 8 kHz LRCK.

4.6.2 MASTER MODE CLOCKING AND MCLK

The SPI module as a master has the ability to generate BCLK and LRCK by internally generating using PBCLK (MCLKEN = 0). The SPI module can generate the clock for external codec devices using the reference output, REFCLKO, function (see [Figure 4-12](#)), although some codecs may have the ability to generate their own MCLK from a crystal to provide accurate audio sample rates. [Figure 4-13](#) shows that the REFCLKO clock can be used as MCLKIN by the codec.

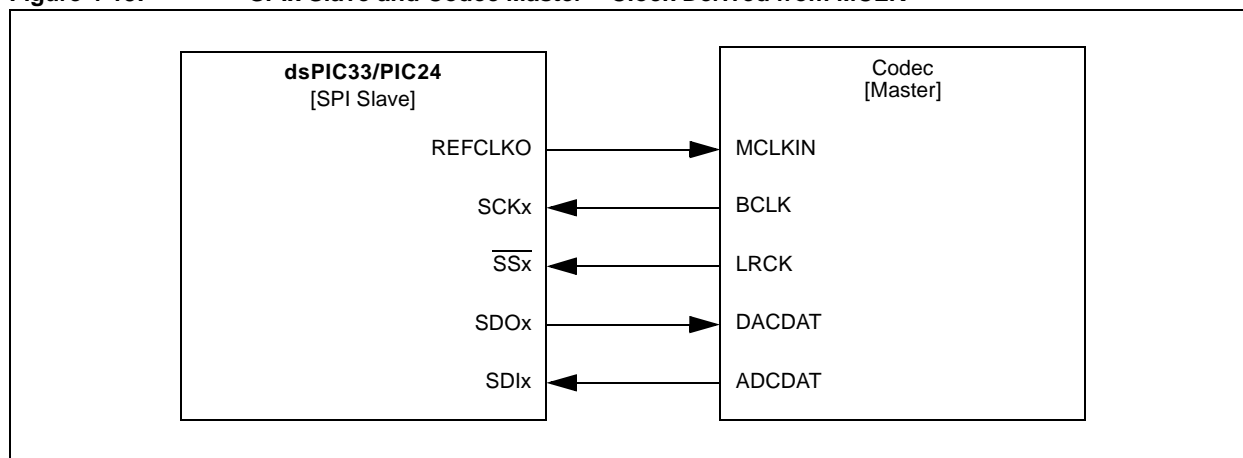
Figure 4-12: SPIx Master Clock Generation



For more information on the reference clock output interface, refer to the specific device data sheet.

[Figure 4-13](#) shows the interface between an SPI slave and a codec master, deriving the clock from the MCLK input interface.

Figure 4-13: SPIx Slave and Codec Master – Clock Derived from MCLK



Serial Peripheral Interface (SPI) with Audio Codec Support

4.6.2.1 I²S Audio Master Mode of Operation Using REFCLKO

The following steps can be used to set up the SPI module for the I²S Audio Master mode of operation with MCLK enabled. The SPI module is initialized to generate BCLK @ 625 kbps and MCLK is derived from PBCLK using the Reference Oscillator Controller register. A typical application could be to play PCM data (8 kHz sample frequency, 16-bit data, 32-bit frame) when interfaced to a codec slave device.

1. If using interrupts, disable the SPIx interrupts in the respective IECx register.
2. Stop and reset the SPI module by clearing the SPIEN bit (SPIxCON1L<15>).
3. Reset the SPIx Control Register 1 High, SPIxCON1H.
4. Reset the Reference Oscillator Controller register, REFOCON.
5. Reset the SPIx Baud Rate Register Low, SPIxBRGL.
6. Clear the receive buffer.
7. Clear the ENHBUF bit (SPIxCON1L<0>) if using Standard Buffer mode or set the bit if using Enhanced Buffer mode.
8. If using interrupts, the following additional steps are performed:
 - a) Clear the SPIx interrupt flags/events in the respective IFSx register.
 - b) Write the SPIx interrupt priority and sub-priority bits in the respective IPCx register.
 - c) Set the SPIx interrupt enable bits in the respective IECx register.
 - d) Clear the SPIROV bit (SPIxSTATL<6>).
9. Write the desired settings in the SPIxCON1H register. The AUDMOD<1:0> bits (SPIxCON1H<9:8>) must be set to '00' for I²S mode and the AUDEN bit (SPIxCON1H<15>) must be set to '1' to enable the audio protocol.
10. Set the Reference Oscillator Controller register, REFOCON:
 - a) RODIV<14:0> (REFOCONH<14:0>) = 0.
 - b) ROEN (REFOCONL<15>) = 1, reference oscillator is enabled.
 - c) ROOUT (REFOCONL<12>) = 1, output is enabled.
11. Set the SPIx Baud Rate Register Low, SPIxBRGL, to 0x1F (to generate approximately 625 kbps sample rate with PBCLK @ 20 MHz).
12. Write the desired settings to the SPIxCON1L register with:
 - a) MSTEN (SPIxCON1L<5>) = 1.
 - b) CKP (SPIxCON1L<6>) = 1.
 - c) MODE<32,16> (SPIxCON1L<11:10>) = 0 for 16-bit audio channel data.
 - d) MCLKEN (SPIxCON1L<2>) = 1, Master mode.
 - e) Enable SPI operation by setting the SPIEN bit (SPIxCON1L<15>).
13. Transmission (and reception) will start as soon as the master provides the BCLK and LRCK.

Example 4-9: I²S Master Mode, 625 kbps BCLK, 16-Bit Channel Data, 32-Bit Frame

```
/* The following code example will initialize the SPI1 Module in I2S Maaster mode.
int rData;
IPC2bits.SPI1TXIP = 4;

SPI1STATLbits.SPIROV = 0;          // clear the Overflow
SPI1CON1H = 0x8000;                // AUDEN=1, I2S mode, stereo mode
SPI1BRGL = 0x000F;                 // to generate 625 kbps sample rate, PBCLK @ 20 MHz
SPI1CON1L = 0x0464;                // 16 bits/32 channel transfer, Master mode,ckp=1, MCLKEN=1
REFOCONL = 0x8001;                // REFO ROEN = 1, ROSEL = 1 for PBCLK
SPI1IMSKLbits.SPITBFEN = 1;       // SPI1 transmit buffer full generates interrupt event

IEC0bits.SPI1TXIE = 1;             // Enable interrupts
IEC0bits.SPI1IE = 1;
IEC3bits.SPI1RXIE = 1;

SPI1CON1Lbits.ENHBUF = 1;
SPI1CON1Lbits.SPIEN = 1;
// from here, the device is ready to receive and transmit data
```

Note: The use of a reference clock output to generate MCLK for the codec may not be a perfect choice. Driving a clock out to an I/O pad induces jitter that may degrade audio fidelity of the codec. The best solution is for the codec to use a crystal and be the master I²S/audio device.

4.7 Mono Mode vs. Stereo Mode

The SPI module enables the audio data transmission in Mono or Stereo mode by setting the AUDMONO bit (SPIxCON1H<11>). When the AUDMONO bit is set to '0' (Stereo mode), the SPIx Shift register uses each FIFO location once, which gives each channel a unique stream of data for stereo data. When the AUDMONO bit is set to '1' (Mono mode), the SPIx Shift register uses each FIFO location twice, to give each channel the same mono stream of audio data.

Note: Receive data is not affected by AUDMONO bit settings.
--

4.8 Streaming Data Support and Error Handling

Most audio streaming applications transmit or receive data continuously. This is required to keep the channel active during the period of operation and ensures the best possible accuracy. Due to streaming audio, the data feeds could be bursty or packet loss can occur causing the module to encounter situations such as underrun. The software needs to be involved to recover from an underrun.

The Ignore Transmit Underrun (IGNTUR) bit (SPIxCON1H<12>), when set to a '1', ignores an underrun condition. This is helpful for cases when software does not care or does not need to know about underrun conditions. When an underrun is encountered, the SPI module sets the SPITUR bit (SPIxSTATL<8>) and when URDTEN = 1 (SPIxCON1H<10>), the module remains in an error state until the software clears the state or the SPIEN bit = 0 (SPIxCON1L<15>).

During the underrun condition, the SPI module loads the SPIxTXSR with the data in the SPIxURDT register when the URDTEN bit is set to '1'. If URDTEN is not set to '1', then the last received data during underrun is loaded to SPIxTXSR. The module samples the underrun condition on channel boundaries, so transmission of SPIxURDT data can start with either the left or right audio channel.

When the condition clears (i.e., SPIxTXB is not empty), the logic loads audio data from the transmit buffer into the SPIxTXSR on the next LRC frame boundary. Because recovery from the underrun condition occurs on the LRC frame boundary (i.e., at the end of a full left and right channel pair), software must ensure the left and right audio data is always transferred to the FIFO in pairs.

The Ignore Receive Overflow (IGNROV) bit (SPIxCON1H<13>), when set to a '1', ignores a Receive Overflow condition. This is useful when there is a general performance problem in the system that software must handle properly. An alternate method to handle the Receive Overflow is by setting the DISSDI bit = 1 (SPIxCON1L<4>) when the system does not need to receive audio data. After changing the DISSDI bit on-the-fly, the SPIx Receive Shift register starts a receive on the leading LRCK edge.

If an RX overflow occurs when IGNROV = 0, the I²S will behave just like it would in SPI mode, that is, it will stop writing to the RX FIFO. However, recovery from overflow is different from SPI mode. When the CPU gets around to reading the RX FIFO, the I²S will restart receiving into the RX FIFO only when two additional conditions are met:

- The I²S is on an LRC boundary
- There is a multiple of two locations free in the RX FIFO

These conditions will ensure the received data will start at the beginning of the LEFT channel and there is room to receive the RIGHT channel information immediately following.

5.0 INTERRUPTS

The SPI module has the ability to generate interrupts reflecting the events that occur during the data communication. The following types of interrupts can be generated:

- Receive data available interrupts are signaled by SPIxRXIF. This event occurs when:
 - RX watermark interrupt
 - SPIROV = 1
 - SPIRBF = 1
 - SPIRBE = 1, provided respective mask bits are enabled in SPIxIMSK
- Transmit buffer empty interrupts are signaled by SPIxTXIF. This event occurs when:
 - TX watermark interrupt
 - SPITUR = 1
 - SPITBF = 1
 - SPITBE = 1, provided respective mask bits are enabled in SPIxIMSK
- General interrupts are signaled by SPIxIF. This event occurs when:
 - FRMERR = 1
 - BUSY = 1
 - SRMT = 1, provided respective mask bits are enabled in SPIxIMSK

All of these interrupt flags, which must be cleared in software, are located in the IFSx registers. Refer to the specific device data sheet for more information.

To enable the SPIx interrupts, use the respective SPIx Interrupt Enable bits, SPIxRXIE, SPIxTXIE and SPIxIF, in the corresponding IECx registers.

The Interrupt Priority Level (IPL) bits must be also be configured using the SPIxIP bits in the corresponding IPCx registers.

When using Enhanced Buffer mode, the SPIx transmit buffer can be configured to interrupt at different FIFO levels using mask bits, TXMSK<5:0> in SPIxIMSKH<5:0>. Also, the Transmit Watermark Interrupt bit, TXWIEN (SPIxIMSKH<7>), should be enabled.

Similarly, the SPIx receive buffer can be configured to interrupt at different FIFO levels using mask bits, RXMSK<5:0> (SPIxIMSKH<13:8>). Also, the Receive Watermark Interrupt, RXWIEN (SPIxIMSKH<15>), should be enabled.

Refer to “**Interrupts**” (DS70000600) in the “*dsPIC33/PIC24 Family Reference Manual*” for further details.

5.1 Interrupt Configuration

Each SPI module has three dedicated interrupt flag bits: SPIxIF, SPIxRXIF and SPIxTXIF, and corresponding interrupt enable bits, SPIxIE, SPIxRXIE and SPIxTXIE. These bits are used to determine the source of an interrupt and to enable or disable an individual interrupt source. Note that all the interrupt sources for a specific SPI module share one interrupt vector. Each SPI module can have its own priority level independent of other SPI modules.

Note: SPIxTXIF, SPIxRXIF and SPIxIF bits will be set without regard to the state of the corresponding enable bit. The interrupt flag bits can be polled by software if desired.
--

The SPIxIE, SPIxTXIE and SPIxRXIE bits are used to define the behavior of the interrupt controller when a corresponding SPIxIF, SPIxTXIF or SPIxRXIF bit is set. When the corresponding interrupt enable bit is clear, the interrupt controller does not generate a CPU interrupt for the event. If the interrupt enable bit is set, the interrupt controller will generate an interrupt to the CPU when the corresponding interrupt flag bit is set (subject to the priority as outlined below).

It is the responsibility of the user's software routine, that services a particular interrupt, to clear the appropriate interrupt flag bit before the service routine is complete.

The priority of each SPI module can be set independently with the SPIxIP<2:0> bits. This priority defines the priority group to which the interrupt source will be assigned. The priority groups range from a value of 7 (the highest priority), to a value of 0 (which does not generate an interrupt). An interrupt being serviced will be preempted by an interrupt in a higher priority group.

The priority group bits allow more than one interrupt source to share the same priority. If simultaneous interrupts occur in this configuration, the natural order of the interrupt sources within a priority group pair determines the interrupt generated. The natural priority is based on the vector numbers of the interrupt sources. The lower the vector number, the higher the natural priority of the interrupt. Any interrupts that were overridden by natural order will then generate their respective interrupts based on priority and natural order, after the interrupt flag for the current interrupt is cleared.

After an enabled interrupt is generated, the CPU will jump to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. The CPU will then begin executing code at the vector address. The user's code at this vector address should perform any application-specific operations required, and clear interrupt flags, SPIxIF, SPIxTXIF or SPIxRXIF, and then exit. For more information on interrupts, refer to the vector address table details in **"Interrupts"** (DS70000600) in the *"dsPIC33/PIC24 Family Reference Manual"*.

For devices with Enhanced Buffering mode, the user application should clear the interrupt request flag after servicing the interrupt condition.

If an SPIx interrupt has occurred, the ISR should read the SPIx Data Buffer (SPIxBUF) register and then clear the SPIx interrupt flag.

6.0 OPERATION IN POWER-SAVING AND DEBUG MODES

6.1 Sleep Mode

When the device enters Sleep mode, the system clock is disabled. The exact SPI module operation during Sleep mode depends on the current mode of operation. The following sub-sections describe mode-specific behavior.

6.1.1 MASTER MODE IN SLEEP MODE

The following items should be noted in Sleep mode:

- The Baud Rate Generator (BRG) is stopped and may be reset (check the device data sheet).
- On-going transmission and reception sequences are aborted. The module may not resume aborted sequences when Sleep mode is exited. (Again, check the device data sheet.)
- Once in Sleep mode, the module will not transmit or receive any new data.

Note: To prevent unintentional abort of transmit and receive sequences, you may need to wait for the current transmission to be completed before activating Sleep mode.
--

6.1.2 SLAVE MODE IN SLEEP MODE

In Slave mode, the SPI module operates from the SCKx provided by an external SPI master. Since the clock pulses at SCKx are externally provided for Slave mode, the module will continue to function in Sleep mode. It will complete any transactions during the transition into Sleep. On completion of a transaction, the SPIRBF flag is set. Consequently, the SPIxRXIF bit will be set. If SPIx interrupts are enabled (SPIxRXIE = 1) and the SPI Interrupt Priority Level is greater than the present CPU priority level, the device will wake from Sleep mode and the code execution will resume at the SPIx interrupt vector location. If the SPI Interrupt Priority Level is lower than or equal to the present CPU priority level, the CPU will remain in Idle mode.

The module is not reset on entering Sleep mode if it is operating as a slave device. Register contents are not affected when the SPI module is going into or coming out of Sleep mode.

6.2 Idle Mode

When the device enters Idle mode, the system clock sources remain functional.

6.2.1 MASTER MODE IN IDLE MODE

Bit, SPISIDL (SPIxCON1L<13>), selects whether the module will stop or continue functioning in Idle mode.

- If SPISIDL = 1, the module will discontinue operation in Idle mode. The module will perform the same procedures when stopped in Idle mode that it does for Sleep mode.
- If SPISIDL = 0, the module will continue operation in Idle mode

6.2.2 SLAVE MODE IN IDLE MODE

The module will continue operation in Idle mode irrespective of the SPISIDL bit setting. The behavior is identical to the one in Sleep mode.

6.3 Debug Mode

6.3.1 OPERATION OF SPIxBUF

6.3.1.1 Reads During Debug Mode

During Debug mode, SPIxBUF can be read, but the read operation does not affect any status bits. For example, if the SPIRBF bit (SPIxSTATL<0>) is set when Debug mode is entered, it will remain set on an exit from Debug mode, even though the SPIxBUF register was read in Debug mode.

7.0 EFFECTS OF VARIOUS RESETS

7.1 Device Reset

All SPIx registers are forced to their Reset states upon a device Reset. When the asynchronous Reset input goes active, the SPI logic:

- Resets all bits in SPIxCON1L and SPIxSTAT
- Resets the SPIx Transmit and Receive Buffers (SPIxBUF) to the empty state
- Resets the Baud Rate Generator (BRG)

7.2 Power-on Reset

All SPIx registers are forced to their Reset states when a Power-on Reset occurs.

7.3 Watchdog Timer Reset

All SPIx registers are forced to their Reset states when a Watchdog Timer Reset occurs.

8.0 PERIPHERALS USING SPI MODULES

There are no other peripherals using the SPI module.

9.0 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 dsPIC33/PIC24 device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the 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 dsPIC33/PIC24 family of devices.

10.0 REVISION HISTORY

Revision A (September 2013)

This is the initial released version of this document.

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.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, FlashFlex, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC³² logo, rPIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MTP, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.


Analog-for-the-Digital Age, Application Maestro, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniclient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICTail, REAL ICE, rLAB, Select Mode, SQI, Serial Quad I/O, Total Endurance, TSHARC, UniWinDriver, WiperLock, ZENA and Z-Scale are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

GestIC and ULPP are registered trademarks of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2013, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 978-1-62077-493-9

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
= ISO/TS 16949 =

Microchip received ISO/TS-16949:2009 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://www.microchip.com/support>
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

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

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: 86-10-8569-7000
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 - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2943-5100
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: 86-755-8864-2200
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: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

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: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2508-8600
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

08/20/13