Section 23. Serial Peripheral Interface

HIGHLIGHTS

This section of the manual contains the following topics:

23.1 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 Serial EEPROMs, shift registers, display drivers, A/D converters, etc. The PIC32MX SPI module is compatible with Motorola® SPI and SIOP interfaces.

Following are some of the key features of this module:

- Master and Slave modes support
- Four different clock formats
- Framed SPI protocol support
- User configurable 8-bit, 16-bit, and 32-bit data width
- Separate SPI shift registers for receive and transmit
- Programmable interrupt event on every 8-bit, 16-bit, and 32-bit data transfer

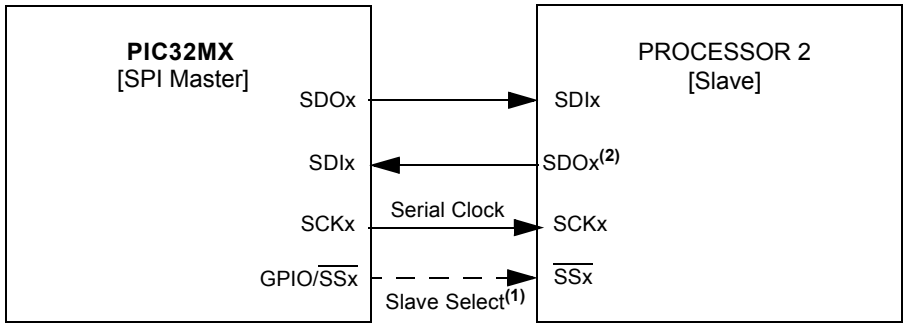
Table 23-1: SPI Features

Available SPI Modes	SPI Master	SPI Slave	Frame Master	Frame Slave	8-Bit, 16-Bit and 32-Bit Modes	Selectable Clock Pulses and Edges	Selectable Frame Sync Pulses and Edges	Slave Select Pulse
Normal Mode	Yes	Yes	—	—	Yes	Yes	—	Yes
Framed Mode	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No

23.1.1 Normal Mode SPI Operation

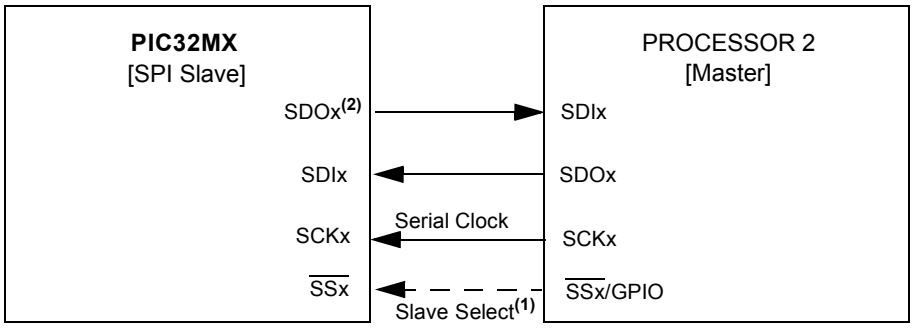
In Normal mode operation, the SPI Master controls the generation of the serial clock. The number of output clock pulses corresponds to the transfer data width: 8, 16, or 32 bits. Figures 23-1 and 23-2 illustrate SPI Master-to-Slave and Slave-to-Master device connections.

Figure 23-1: Typical SPI Master-to-Slave Device Connection Diagram



Note 1: In Normal mode, the usage of the Slave Select pin ($\overline{SSx}$) is optional.
2: Control of the SDO pin can be disabled for Receive-Only modes.

Figure 23-2: Typical SPI Slave-to-Master Device Connection Diagram

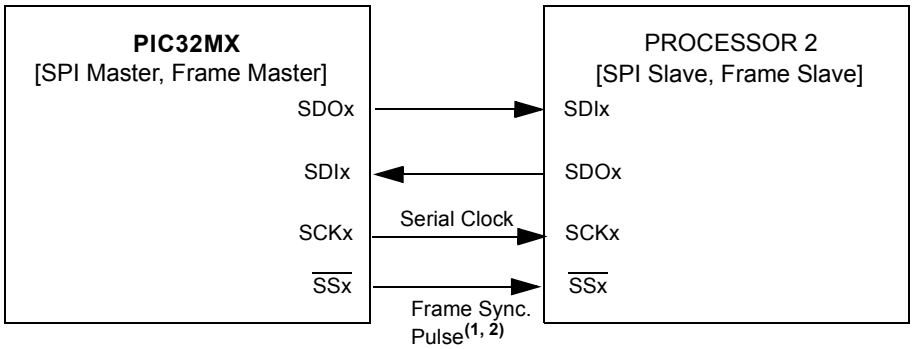


Note 1: In Normal mode, the usage of the Slave Select pin ($\overline{SSx}$) is optional.
2: The control of the SDO pin can be disabled for Receive-Only modes.

23.1.2 Framed Mode SPI Operation

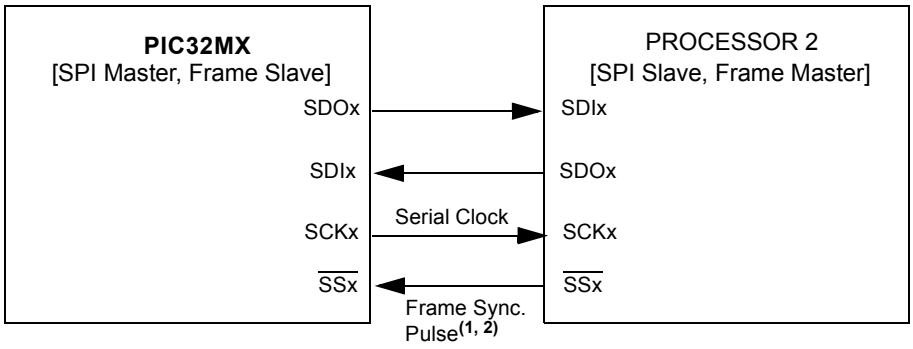
In Framed mode operation, the Frame Master controls the generation of the frame synchronization pulse. The SPI clock is still generated by the SPI Master and is continuously running. Figures 23-3 and 23-4 illustrate SPI Frame Master and Frame Slave device connections.

Figure 23-3: Typical SPI Master, Frame Master Connection Diagram



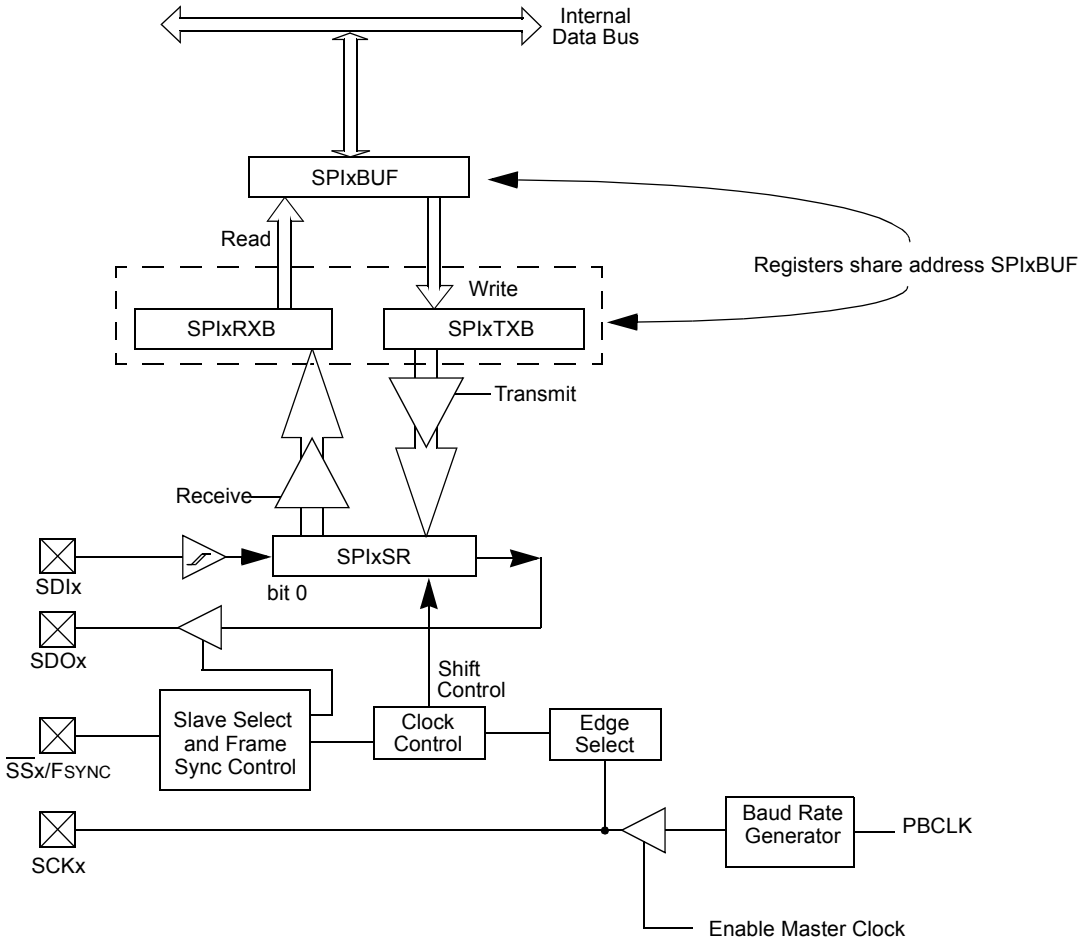
Note 1: In Framed SPI mode, the $\overline{SSx}$ pin is used to transmit/receive the frame synchronization pulse.
2: Framed SPI mode requires the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).

Figure 23-4: Typical SPI Master, Frame Slave Connection Diagram



Note 1: In Framed SPI mode, the $\overline{SSx}$ pin is used to transmit/receive the frame synchronization pulse.
2: Framed SPI mode requires the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).

Figure 23-5: SPI Module Block Diagram



Note: The SPIxTXB and SPIxRXB registers are accessed via the SPIxBUF register.

23.2 STATUS AND CONTROL REGISTERS

Note: Each PIC32MX 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):

- SPIxCON: SPI Control Register for the Module 'x'

  SPIxCONCLR, SPIxCONSET, SPIxCONINV: Atomic Bit Manipulation Write-only Registers for SPIxCON

- SPIxSTAT: SPI Status Register for the Module 'x'

  SPIxSTATCLR, SPIxSTATSET, SPIXSTATINV: Atomic Bit Manipulation Write-only Registers for SPIxSTAT

- SPIxBUF: SPI Transmit and Receive Buffer Register for the Module 'x'
- SPIxBRG: SPI Baud Rate Generator Register for the Module 'x'

  SPIxBRGCLR, SPIxBRGSET, SPIxBRGINV: Atomic Bit Manipulation Write-only Registers for SPIxBRG

Each SPI module also has the following associated bits for interrupt control:

- SPIxRXIF, SPIxTXIF, SPIxEIF: Interrupt Flag Status Bits for Receive, Transmit, and Error Events – in IFS0, IFS1 INT Registers
- SPIxRXIE, SPIxTXIE, SPIxEIE: Interrupt Enable Control Bits for Receive, Transmit, and Error Events – in IEC0, IEC1 INT Registers
- SPIxIP<2:0>: Interrupt Priority Control Bits – in IPC6, IPC7 INT Registers
- SPIxIS<1:0>: Interrupt Subpriority Control Bits – in IPC6, IPC7 INT Registers

The following table summarizes all SPI-related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

Table 23-2: SPI SFR Summary

Name		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
SPIxCON	31:24	FRMEN	FRMSYNC	FRMPOL	—	—	—	—	—
	23:16	—	—	—	—	—	—	SPIFE	—
	15:8	ON	FRZ	SIDL	DISSDO	MODE32	MODE16	SMP	CKE
	7:0	SSEN	CKP	MSTEN	—	—	—	—	—
SPIxCONCLR	31:0	Write clears selected bits in SPIxCON, read yields undefined value							
SPIxCONSET	31:0	Write sets selected bits in SPIxCON, read yields undefined value							
SPIxCONINV	31:0	Write inverts selected bits in SPIxCON, read yields undefined value							
SPIxSTAT	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	SPIBUSY	—	—	—
	7:0	—	SPIROV	—	—	SPITBE	—	—	SPIRBF
SPIxSTATCLR	31:0	Write clears selected bits in SPIxSTAT, read yields undefined value							
SPIxBUF	31:24	DATA<31:24>							
	23:16	DATA<23:16>							
	15:8	DATA<15:8>							
	7:0	DATA<7:0>							
SPIxBRG	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	—	BRG<8>
	7:0	BRG<7>	BRG<6>	BRG<5>	BRG<4>	BRG<3>	BRG<2>	BRG<1>	BRG<0>
SPIxBRGCLR	31:0	Write clears selected bits in SPIxBRG, read yields undefined value							
SPIxBRGSET	31:0	Write sets selected bits in SPIxBRG, read yields undefined value							
SPIxBRGINV	31:0	Write inverts selected bits in SPIxBRG, read yields undefined value							

Table 23-2: SPI SFR Summary (Continued)

Name		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
IFS0	31:24	I2C1MIF	I2C1SIF	I2C1BIF	U1TXIF	U1RXIF	U1EIF	SPI1RXIF	SPI1TXIF
	23:16	SPI1EIF	OC5IF	IC5IF	T5IF	INT4IF	OC4IF	IC4IF	T4IF
	15:8	INT3IF	OC3IF	IC3IF	T3IF	INT2IF	OC2IF	IC2IF	T2IF
	7:0	INT1IF	OC1IF	IC1IF	T1IF	INT0IF	CS1IF	CS0IF	CTIF
IFS0CLR	31:0	Write clears selected bits in IFS0, read yields undefined value							
IFS0SET	31:0	Write sets selected bits in IFS0, read yields undefined value							
IFS0INV	31:0	Write inverts selected bits in IFS0, read yields undefined value							
IFS1	31:24	—	—	—	—	—	—	USBIF	FCEIF
	23:16	—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
	15:8	RTCCIF	FSCMIF	I2C2MIF	I2C2SIF	I2C2BIF	U2TXIF	U2RXIF	U2EIF
	7:0	SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
IFS1CLR	31:0	Write clears selected bits in IFS1, read yields undefined value							
IFS1SET	31:0	Write sets selected bits in IFS1, read yields undefined value							
IFS1INV	31:0	Write inverts selected bits in IFS1, read yields undefined value							
IEC0	31:24	I2C1MIE	I2C1SIE	I2C1BIE	U1TXIE	U1RXIE	U1EIE	SPI1RXIE	SPI1TXIE
	23:16	SPI1EIE	OC5IE	IC5IE	T5IE	INT4IE	OC4IE	IC4IE	T4IE
	15:8	INT3IE	OC3IE	IC3IE	T3IE	INT2IE	OC2IE	IC2IE	T2IE
	7:0	INT1IE	OC1IE	IC1IE	T1IE	INT0IE	CS1IE	CS0IE	CTIE
IEC0CLR	31:0	Write clears selected bits in IEC0, read yields undefined value							
IEC0SET	31:0	Write sets selected bits in IEC0, read yields undefined value							
IEC0INV	31:0	Write inverts selected bits in IEC0, read yields undefined value							
IEC1	31:24	—	—	—	—	—	—	USBIE	FCEIE
	23:16	—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
	15:8	RTCCIE	FSCMIE	I2C2MIE	I2C2SIE	I2C2BIE	U2TXIE	U2RXIE	U2EIE
	7:0	SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
IEC1CLR	31:0	Write clears selected bits in IEC1, read yields undefined value							
IEC1SET	31:0	Write sets selected bits in IEC1, read yields undefined value							
IEC1INV	31:0	Write inverts selected bits in IEC1, read yields undefined value							
IPC5	31:24	—	—	—	SPI1IP<2:0>			SPI1IS<1:0>	
	23:16	—	—	—	OC5IP<2:0>			OC5IS<1:0>	
	15:8	—	—	—	IC5IP<2:0>			IC5IS<1:0>	
	7:0	—	—	—	T5IP<2:0>			T5IS<1:0>	
IPC5CLR	31:0	Write clears selected bits in IPC5, read yields undefined value							
IPC5SET	31:0	Write sets selected bits in IPC5, read yields undefined value							
IPC5INV	31:0	Write inverts selected bits in IPC5, read yields undefined value							
IPC7	31:24	—	—	—	SPI2IP<2:0>			SPI2IS<1:0>	
	23:16	—	—	—	CMP2IP<2:0>			CMP2IS<1:0>	
	15:8	—	—	—	CMP1IP<2:0>			CMP1IS<1:0>	
	7:0	—	—	—	PMPIP<2:0>			PMPIS<1:0>	
IPC7CLR	31:0	Write clears selected bits in IPC7, read yields undefined value							
IPC7SET	31:0	Write sets selected bits in IPC7, read yields undefined value							
IPC7INV	31:0	Write inverts selected bits in IPC7, read yields undefined value							

Register 23-1: SPIxCON: SPI Control Register

R/W-0	R/W-0	R/W-0	r-x	r-x	r-x	r-x	r-x
FRMEN	FRMSYNC	FRMPOL	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	R/W-0	r-x
—	—	—	—	—	—	SPIFE	—
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ON	FRZ	SIDL	DISSDO	MODE32	MODE16	SMP	CKE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	r-x	r-x	r-x	r-x	r-x
SSEN	CKP	MSTEN	—	—	—	—	—
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31 **FRMEN:** Framed SPI Support bit
1 = Framed SPI support is enabled ($\overline{SSx}$ pin used as FSYNC input/output)
0 = Framed SPI support is disabled

bit 30 **FRMSYNC:** Frame Sync Pulse Direction Control on $\overline{SSx}$ pin bit (Framed SPI mode only)
1 = Frame sync pulse input (Slave mode)
0 = Frame sync pulse output (Master mode)

bit 29 **FRMPOL:** Frame Sync Polarity bit (Framed SPI mode only)
1 = Frame pulse is active-high
0 = Frame pulse is active-low

bit 28-18 **Reserved:** Maintain as ‘0’; Ignore Read

bit 17 **SPIFE:** Frame Sync Pulse Edge Select bit (Framed SPI mode only)
1 = Frame synchronization pulse coincides with the first bit clock
0 = Frame synchronization pulse precedes the first bit clock

bit 16 **Reserved:** Maintain as ‘0’; Ignore Read

bit 15 **ON:** SPI Peripheral On bit
1 = SPI Peripheral is enabled
0 = SPI Peripheral is disabled

bit 14 **FRZ:** Freeze in DEBUG Exception Mode bit
1 = Freeze operation when CPU enters DEBUG Exception mode
0 = Continue operation when CPU enters DEBUG Exception mode
Note: FRZ is writable in DEBUG Exception mode only, it is forced to ‘0’ in Normal mode.

bit 13 **SIDL:** Stop in IDLE Mode bit
1 = Discontinue operation when CPU enters in IDLE mode
0 = Continue operation in IDLE mode

bit 12 **DISSDO:** Disable SDOx pin bit
1 = SDOx pin is not used by the module. Pin is controlled by associated PORT register
0 = SDOx pin is controlled by the module

Register 23-1: SPIxCON: SPI Control Register (Continued)

bit 11-10	**MODE<32,16>:** 32/16-Bit Communication Select bits 1x = 32-bit data width 01 = 16-bit data width 00 = 8-bit data width
bit 9	**SMP:** SPI Data Input Sample Phase bit Master mode (MSTEN = 1): 1 = Input data sampled at end of data output time 0 = Input data sampled at middle of data output time Slave mode (MSTEN = 0): SMP value is ignored when SPI is used in Slave mode. The module always uses SMP = 0.
bit 8	**CKE:** SPI Clock Edge Select bit 1 = Serial output data changes on transition from active clock state to Idle clock state (see CKP bit) 0 = Serial output data changes on transition from Idle clock state to active clock state (see CKP bit) **Note:** The CKE bit is not used in the Framed SPI mode. The user should program this bit to '0' for the Framed SPI mode (FRMEN = 1).
bit 7	**SSEN:** Slave Select Enable (Slave mode) bit 1 = $\overline{SSx}$ pin used for Slave mode 0 = $\overline{SSx}$ pin not used for Slave mode, pin controlled by port function.
bit 6	**CKP:** Clock Polarity Select bit 1 = Idle state for clock is a high level; active state is a low level 0 = Idle state for clock is a low level; active state is a high level
bit 5	**MSTEN:** Master Mode Enable bit 1 = Master mode 0 = Slave mode
bit 4-0	**Reserved:** Maintain as '0'; Ignore Read

Register 23-2: SPIxCONCLR: SPIxCON Clear Register

Write clears selected bits in SPIxCON, read yields undefined value
bit 31 bit 0

bit 31-0 **Clears selected bits in SPIxCON**

A write of '1' in one or more bit positions clears the corresponding bit(s) in SPIxCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `SPIxCONCLR = 0x00008020` will clear bits 15 and 5 in SPIxCON register.

Register 23-3: SPIxCONSET: SPIxCON Set Register

Write sets selected bits in SPIxCON, read yields undefined value
bit 31 bit 0

bit 31-0 **Sets selected bits in SPIxCON**

A write of '1' in one or more bit positions sets the corresponding bit(s) in SPIxCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `SPIxCONSET = 0x00008020` will set bits 15 and 5 in SPIxCON register.

Register 23-4: SPIxCONINV: SPIxCON Invert Register

Write inverts selected bits in SPIxCON, read yields undefined value
bit 31 bit 0

bit 31-0 **Inverts selected bits in SPIxCON**

A write of '1' in one or more bit positions inverts the corresponding bit(s) in SPIxCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `SPIxCONINV = 0x00008020` will invert bits 15 and 5 in SPIxCON register.

Register 23-5: SPIxSTAT: SPI Status Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	R-0	r-x	r-x	r-x
—	—	—	—	SPIBUSY	—	—	—
bit 15							bit 8

r-x	R/W-0	r-x	r-x	R-1	r-x	r-x	R-0
—	SPIROV	—	—	SPITBE	—	—	SPIRBF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-12 **Reserved:** Maintain as ‘0’; Ignore Read

bit 11 **SPIBUSY:** SPI Activity Status bit
1 = SPI peripheral is currently busy with some transactions
0 = SPI peripheral is currently idle

bit 10-7 **Reserved:** Maintain as ‘0’; Ignore Read

bit 6 **SPIROV:** Receive Overflow Flag bit
1 = A new data is completely received and discarded. The user software has not read the previous data in the SPIxBUF register.
0 = No overflow has occurred
This bit is set in hardware; can only be cleared (= 0) in software.

bit 5-4 **Reserved:** Maintain as ‘0’; Ignore Read

bit 3 **SPITBE:** SPI Transmit Buffer Empty Status bit
1 = Transmit buffer, SPIxTXB is empty
0 = Transmit buffer, SPIxTXB is not empty
Automatically set in hardware when SPI transfers data from SPIxTXB to SPIxSR.
Automatically cleared in hardware when SPIxBUF is written to, loading SPIxTXB.

bit 2 **Reserved:** Maintain as ‘0’; Ignore Read

bit 1 **Reserved:** Maintain as ‘0’; Ignore Read

bit 0 **SPIRBF:** SPI Receive Buffer Full Status bit
1 = Receive buffer, SPIxRXB is full
0 = Receive buffer, SPIXRXB is not full
Automatically set in hardware when SPI transfers data from SPIxSR to SPIxRXB.
Automatically cleared in hardware when SPIxBUF is read from, reading SPIxRXB.

Register 23-6: SPIxSTATCLR: SPIxSTAT Clear Register

Write clears selected bits in SPIxSTAT, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in SPIxSTAT**

A write of '1' in one or more bit positions clears the corresponding bit(s) in SPIxSTAT register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `SPIxSTATCLR = 0x00000040` will clear bit 6 in SPIxSTAT register.

Register 23-7: SPIxBUF: SPI Buffer Register

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DATA<31:24>							
bit 31							bit 24

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 23							bit 16

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	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-0 **DATA**<31:0>**:** SPI Transmit/Receive Buffer register
Serves as a memory-mapped value of Transmit (SPIxTXB) and Receive (SPIxSR) registers.

When 32-Bit Data mode is enabled (MODE[32,16] (SPIxCON<11:10>) = 1x):
All 32-bits (SPIxBUF<31:0>) of this register are used to form a 32-bit character.

When 16-Bit Data mode is enabled (MODE[32,16] (SPIxCON<11:10>) = 01):
Only lower 16-bits (SPIxBUF<15:0>) of this register are used to form the 16-bit character.

When 8-Bit Data mode is enabled (MODE[32,16] (SPIxCON<11:10>) = 00):
Only lower 8-bits (SPIxBUF<7:0>) of this register are used to form the 8-bit character.

Register 23-8: SPIxBRG: SPI Baud Rate Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	r-x	r-x	r-x	r-x	r-x	r-x	R/W-0
—	—	—	—	—	—	—	BRG<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
BRG<7>	BRG<6>	BRG<5>	BRG<4>	BRG<3>	BRG<2>	BRG<1>	BRG<0>
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-9 **Reserved:** Maintain as ‘0’; Ignore Read

bit 8-0 **BRG<8:0>:** Baud Rate Divisor bits

Register 23-9: SPIxBRGCLR: SPIxBRG Clear Register

Write clears selected bits in SPIxBRG, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in SPIxBRG**

A write of '1' in one or more bit positions clears the corresponding bit(s) in SPIxBRG register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `SPIxBRGCLR = 0x000001FF` will clear bits 8 through 0 in SPIxBRG register.

Register 23-10: SPIxBRGSET: SPIxBRG Set Register

Write sets selected bits in SPIxBRG, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in SPIxBRG**

A write of '1' in one or more bit positions sets the corresponding bit(s) in SPIxBRG register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `SPIxBRGSET = 0x000001FF` will set bits 8 through 0 in SPIxBRG register.

Register 23-11: SPIxBRGINV: SPIxBRG Invert Register

Write inverts selected bits in SPIxBRG, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in SPIxBRG**

A write of '1' in one or more bit positions inverts the corresponding bit(s) in SPIxBRG register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `SPIxBRGINV = 0x000001FF` will toggle bits 8 through 0 in SPIxBRG register.

Register 23-12: IFS0: Interrupt Flag Status Register 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
I2C1MIF	I2C1SIF	I2C1BIF	U1TXIF	U1RXIF	U1EIF	SPI1RXIF	SPI1TXIF
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPI1EIF	OC5IF	IC5IF	T5IF	INT4IF	OC4IF	IC4IF	T4IF
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT3IF	OC3IF	IC3IF	T3IF	INT2IF	OC2IF	IC2IF	T2IF
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0
INT1IF	OC1IF	IC1IF	T1IF	INT0IF	CS1IF	CS0IF	CTIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-26 Interrupt Flag bits for other peripheral devices

bit 25 **SPI1RXIF**: SPI1 Receive Buffer Full Interrupt Flag bit
1 = Receive buffer full interrupt pending
0 = No receive interrupt pending
Set by the hardware when a character is assembled in the SPI1 receive buffer.
Cleared by the software, usually in the ISR.

bit 24 **SPI1TXIF**: SPI1 Transmit Buffer Empty Interrupt Flag bit
1 = Transmit buffer empty interrupt pending
0 = No transmit interrupt pending
Set by the hardware when a character can be written into the SPI1 transmit buffer.
Cleared by the software, usually in the ISR.

bit 23 **SPI1EIF**: SPI1 Error Interrupt Flag bit
1 = SPI1 receive overflow interrupt pending
0 = No receive overflow interrupt pending
Set by the hardware when a character is assembled in the SPI1 receive buffer and the previous character hasn’t been read from the SPI buffer.
Cleared by the software as part of the error processing.

bit 22-0 Interrupt Flag bits for other peripheral devices

Note: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the SPI.

Register 23-13: IFS1: Interrupt Flag Status Register 1

r-x	r-x	r-x	r-x	r-x	r-x	R/W-0	R/W-0
—	—	—	—	—	—	USBIF	FCEIF
bit 31							bit 24

r-0	r-0	r-0	r-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RTCCIF	FSCMIF	I2C2MIF	I2C2SIF	I2C2BIF	U2TXIF	U2RXIF	U2EIF
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0
SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-26 **Unimplemented:** Read as ‘0’

bit 25-24 Interrupt flags for other peripheral devices

bit 23-20 **Reserved:** Maintain as ‘0’

bit 19-8 Interrupt flags for other peripheral devices

bit 7 **SPI2RXIF**: SPI2 Receive buffer full interrupt flag
1 = Receive buffer full interrupt pending
0 = No receive interrupt pending
Set by the hardware when a character is assembled in the SPI2 receive buffer.
Cleared by the software, usually in the ISR.

bit 6 **SPI2TXIF**: SPI2 Transmit buffer empty interrupt flag
1 = Transmit buffer empty interrupt pending
0 = No transmit interrupt pending
Set by the hardware when a character can be written into the SPI2 transmit buffer.
Cleared by the software, usually in the ISR.

bit 5 **SPI2EIF**: SPI2 Error interrupt flag
1 = SPI2 receive overflow interrupt pending
0 = No receive overflow interrupt pending
Set by the hardware when a character is assembled in the SPI2 receive buffer and the previous character hasn’t been read from the SPI buffer.
Cleared by the software as part of the error processing.

bit 4-0 Interrupt flags for other peripheral devices

Register 23-14: IEC0: Interrupt Enable Control Register 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
I2C1MIE	I2C1SIE	I2C1BIE	U1TXIE	U1RXIE	U1EIE	SPI1RXIE	SPI1TXIE
bit 31							bit 24

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPI1EIE	OC5IE	IC5IE	T5IE	INT4IE	OC4IE	IC4IE	T4IE
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT3IE	OC3IE	IC3IE	T3IE	INT2IE	OC2IE	IC2IE	T2IE
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
INT1IE	OC1IE	IC1IE	T1IE	INT0IE	CS1IE	CS0IE	CTIE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-26 Interrupt Enable Flag bits for other peripheral devices

bit 25 **SPI1RXIE:** SPI1 Receive Buffer Full Interrupt Enable bit

1 = Receive buffer full interrupt enabled
0 = Receive buffer full interrupt disabled

Set/cleared by the software to enable/disable SPI interrupts when a new character is assembled in the SPI1 receive buffer.

bit 24 **SPI1TXIE:** SPI1 Transmit Buffer Empty Interrupt Enable bit

1 = Transmit buffer empty interrupt enabled
0 = Transmit buffer empty interrupt disabled

Set/cleared by the software to enable/disable SPI interrupts when a new character can be written into the SPI1 transmit buffer.

bit 23 **SPI1EIE:** SPI1 Error Interrupt Enable bit

1 = SPI1 receive overflow interrupt enabled
0 = SPI1 receive overflow interrupt disabled

Set by the software to enable/disable SPI interrupts when a character is assembled in the SPI1 receive buffer and the previous character hasn’t been read from the SPI buffer.

bit 22-0 Interrupt Enable Flag bits for other peripheral devices

Note: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the SPI.

Register 23-15: IEC1: Interrupt Enable Control Register 1

r-x	r-x	r-x	r-x	r-x	r-x	R/W-0	R/W-0
—	—	—	—	—	—	USBIE	FCEIE
bit 31							bit 24

r-0	r-0	r-0	r-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
bit 23							bit 16

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RTCCIE	FSCMIE	I2C2MIE	I2C2SIE	I2C2BIE	U2TXIE	U2RXIE	U2EIE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-0
SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-26 **Unimplemented:** Read as ‘0’

bit 25-24 Interrupt flags for other peripheral devices

bit 23-20 **Reserved:** Maintain as ‘0’

bit 19-8 Interrupt flags for other peripheral devices

bit 7 **SPI2RXIE:** SPI2 Receive Buffer Full Interrupt Enable bit

1 = Receive buffer full interrupt enabled
0 = Receive buffer full interrupt disabled

Set/cleared by the software to enable/disable the interrupt when a character is assembled in the SPI2 receive buffer.

bit 6 **SPI2TXIE:** SPI2 Transmit Buffer Empty Interrupt Enable bit

1 = Transmit buffer empty interrupt enabled
0 = Transmit buffer empty interrupt disabled

Set/cleared by the software to enable/disable interrupts when a character can be written into the SPI2 transmit buffer.

bit 5 **SPI2EIE:** SPI2 Error Interrupt Enable bit

1 = SPI2 receive overflow interrupt enabled
0 = SPI2 receive overflow interrupt disabled

Set/cleared by the software to enable/disable overflow interrupts.

bit 4-0 Interrupt enable bits for other peripheral devices

Register 23-16: IPC5: Interrupt Priority Control Register 5

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	SPI1IP<2:0>			SPI1IS<1:0>	
bit 31							bit 24

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OC5IP<2:0>			OC5IS<1:0>	
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	IC5IP<2:0>			IC5IS<1:0>	
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	T5IP<2:0>			T5IS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Reserved:** Maintain as ‘0’; Ignore Read

bit 28-26 **SPI1IP<2:0>:** SPI1 Interrupt Vector Priority bits
111 = SPI1 interrupts have priority 7 (highest priority)
•
•
•
001 = SPI1 interrupts have priority 1
000 = SPI1 interrupts are disabled

bit 25-24 **SPI1IS<1:0>:** SPI1 Interrupt Vector Subpriority bits
11 = SPI1 interrupts have Subpriority 3 (highest subpriority)
•
•
00 = SPI1 interrupts have Subpriority 0 (lowest subpriority)

bit 23-21 **Reserved:** Maintain as ‘0’; Ignore Read

bit 20-16 Interrupt Priority Control bits for other peripheral devices

bit 15-13 **Reserved:** Maintain as ‘0’; Ignore Read

bit 12-8 Interrupt Priority Control bits for other peripheral devices

bit 7-5 **Reserved:** Maintain as ‘0’; Ignore Read

bit 4-0 Interrupt Priority Control bits for other peripheral devices

Note: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the SPI.

Register 23-17: IPC7: Interrupt Priority Control Register 7

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	SPI2IP<2:0>			SPI2IS<1:0>	
bit 31							bit 24

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CMP2IP<2:0>			CMP2IS<1:0>	
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CMP1IP<2:0>			CMP1IS<1:0>	
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	PMPIP<2:0>			PMPIS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-29 **Reserved:** Maintain as ‘0’; Ignore Read

bit 28-26 **SPI2IP<2:0>:** SPI2 Interrupt Vector Priority bits
111 = SPI2 interrupts have priority 7 (highest priority)
•
•
•
001 = SPI2 interrupts have priority 1
000 = SPI2 interrupts are disabled

bit 25-24 **SPI2IS<1:0>:** SPI2 Interrupt Vector Subpriority bits
11 = SPI2 interrupts have Subpriority 3 (highest subpriority)
•
•
00 = SPI2 interrupts have Subpriority 0 (lowest subpriority)

bit 23-21 **Reserved:** Maintain as ‘0’; Ignore Read

bit 20-16 Interrupt Priority Control bits for other peripheral devices

bit 15-13 **Reserved:** Maintain as ‘0’; Ignore Read

bit 12-8 Interrupt Priority Control bits for other peripheral devices

bit 7-5 **Reserved:** Maintain as ‘0’; Ignore Read

bit 4-0 Interrupt Priority Control bits for other peripheral devices

Note: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the SPI.

23.3 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

23.3.1 8-Bit, 16-Bit, and 32-Bit Operation

The PIC32MX 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 (SPIxCON<11:10>), 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 (SPIxCON<15> = 0), the new mode will be available when the module is again switched on.

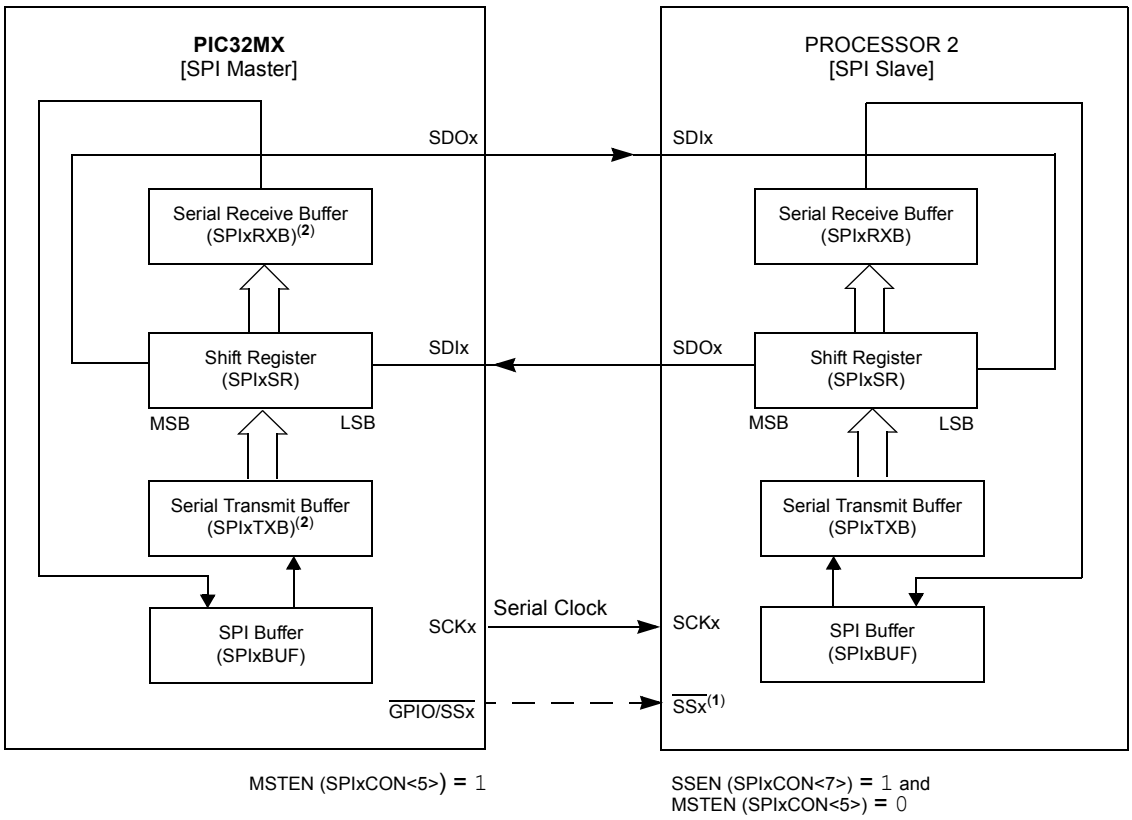
Additionally, the following items should be noted in this context:

- The MODE32 and MODE16 bits should not be changed when a transaction is in progress.
- The first bit to be shifted out from SPIxSR 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 SPIxSR.
- 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

23.3.2 Master and Slave Modes

Figure 23-6: SPI Master/Slave Connection Diagram



Note 1: Using the $\overline{SSx}$ pin in Slave mode of operation is optional.
2: User must write transmit data to SPIxBUF and read received data from SPIxBUF. The SPIxTXB and SPIxRXB registers are memory mapped to SPIxBUF.

23.3.2.1 Master Mode Operation

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

1. Disable the SPI interrupts in the respective IEC0/1 register.
2. Stop and reset the SPI module by clearing the ON bit.
3. Clear the receive buffer.
4. If SPI interrupts are not going to be used, skip this step and continue to step 5. Otherwise the following additional steps are performed:
   a) Clear the SPIx interrupt flags/events in the respective IFS0/1 register.
   b) Set the SPIx interrupt enable bits in the respective IEC0/1 register.
   c) Write the SPIx interrupt priority and subpriority bits in the respective IPC5/7 register.
5. Write the Baud Rate register, SPIxBRG.
6. Clear the SPIROV bit (SPIxSTAT<6>).
7. Write the desired settings to the SPIxCON register with MSTEN (SPIxCON<5>) = 1.
8. Enable SPI operation by setting the ON bit (SPIxCON<15>).
9. Write the data to be transmitted to the SPIxBUF register. Transmission (and reception) will start as soon as data is written to the SPIxBUF register.

Note:	The SPI device must be turned off prior to changing the mode from Slave to Master.

Note:	When using the Slave Select mode, the $\overline{SSx}$ 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 SPIxBRG register. The serial clock is output via the SCKx pin to slave devices. Clock pulses are only generated when there is data to be transmitted; except when in Framed mode, when clock is generated continuously. For further information, refer to **23.3.6 "SPI Master Mode Clock Frequency"**.

Bits CKP (SPIxCON<6>) and CKE (SPIxCON<8>) 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 guaranteed.

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 SPIxBUF register. The SPITBE (SPIxSTAT<3>) bit is cleared.
2. The contents of SPIxTXB are moved to the shift register SPIxSR (see Figure 23-6), 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 SPIxSR to the SDOx pin and simultaneously shifts the data at the SDIx pin into SPIxSR.
4. When the transfer is complete, the following events will occur:
   a) The interrupt flag bit SPIxRXIF is set. SPI interrupts can be enabled by setting the interrupt enable bit SPIxRXIE. The SPIxRXIF flag is not cleared automatically by the hardware.
   b) Also, when the ongoing transmit and receive operation is completed, the contents of SPIxSR are moved to SPIxRXB.
   c) The SPIRBF bit (SPIxSTAT<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.

5. If the SPIRBF bit is set (the receive buffer is full) when the SPI module needs to transfer data from SPIxSR to SPIxRXB, the module will set the SPIROV bit (SPIxSTAT<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 (SPIxSTAT<3>) bit is set. The write can occur while SPIxSR is shifting out the previously written data, allowing continuous transmission.

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

Example 23-1: Initialization Code for 16-Bit SPI Master Mode

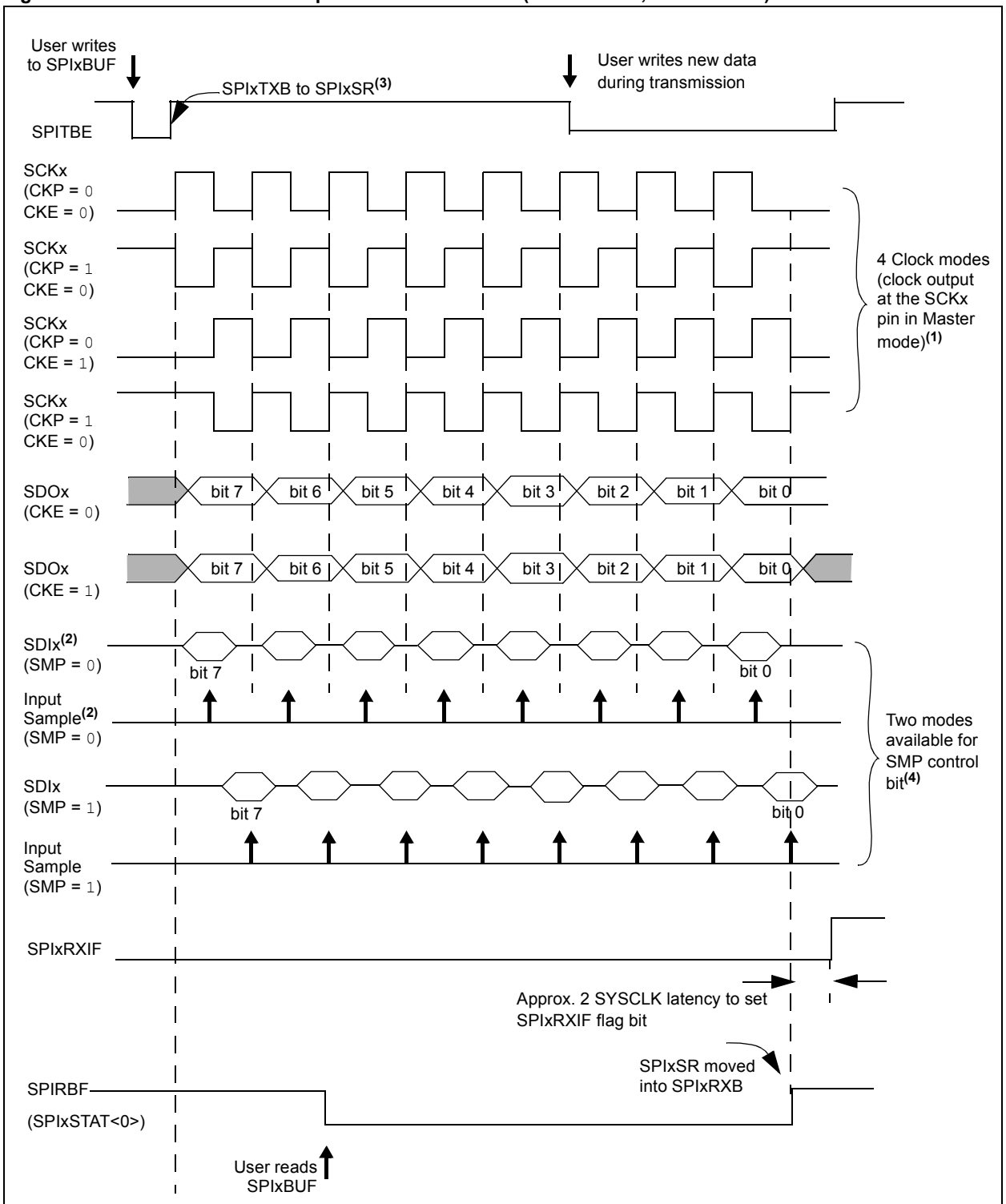
```
/*
    The following code example will initialize the SPI1 in Master mode.
    It assumes that none of the SPI1 input pins are shared with an analog input. If so, the
    AD1PCFG and corresponding TRIS registers have to be properly configured.
*/
    int rData;

    IEC0CLR=0x03800000;           // disable all interrupts
    SPI1CON = 0;                  // Stops and resets the SPI1.
    rData=SPI1BUF;                // clears the receive buffer
    IFS0CLR=0x03800000;           // clear any existing event
    IPC5CLR=0x1f000000;           // clear the priority
    IPC5SET=0x0d000000;           // Set IPL=3, Subpriority 1
    IEC0SET=0x03800000;           // Enable Rx, Tx and Error interrupts

    SPI1BRG=0x1;                  // use FPB/4 clock frequency
    SPI1STATCLR=0x40;             // clear the Overflow
    SPI1CON=0x8220;               // SPI ON, 8 bits transfer, SMP=1, Master mode

                                  // from now on, the device is ready to transmit and receive
                                     data
    SPI1BUF='A';                  // transmit an A character
```

Figure 23-7: SPI Master Mode Operation in 8-Bit Mode (MODE32 = 0, MODE16 = 0)



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

2: The SDI and input samples shown here for two different values of the SMP bit (SPIxCON<9>) are strictly for demonstration purposes. Only one of the two configurations of the SMP bit can be chosen during operation.

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

4: Operation for 8-bit mode shown. 16-bit and 32-bit modes are similar.

23.3.2.2 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 SPI interrupts in the respective IEC0/1 register.
2. Stop and reset the SPI module by clearing the ON bit.
3. Clear the receive buffer.
4. If using interrupts, the following additional steps are performed:
   a) Clear the SPIx interrupt flags/events in the respective IFS0/1 register.
   b) Set the SPIx interrupt enable bits in the respective IEC0/1 register.
   c) Write the SPIx interrupt priority and subpriority bits in the respective IPC5/7 register.
5. Clear the SPIROV bit (SPIxSTAT<6>).
6. Write the desired settings to the SPIxCON register with MSTEN (SPIxCON<5>) = 0.
7. Enable SPI operation by setting the ON bit (SPIxCON<15>).
8. Transmission (and reception) will start as soon as the master provides the serial clock.

Note:	The SPI device 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. Bits CKP (SPIxCON<6>) and CKE (SPIxCON<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 SPIxBUF register.

The rest of the operation of the module is identical to that in the Master mode.

23.3.2.2.1 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 (SPIxCON<7>) is set, transmission and reception is 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 (SPIxSTAT<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 (SPIxCON<7>) is cleared, the SPITBE is cleared when SPIxBUF is loaded by the user code. It is set when the module transfers SPIxTXB to SPIxSR. 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 SPIx 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.

Example 23-2: Initialization Code for 16-Bit SPI Slave Mode

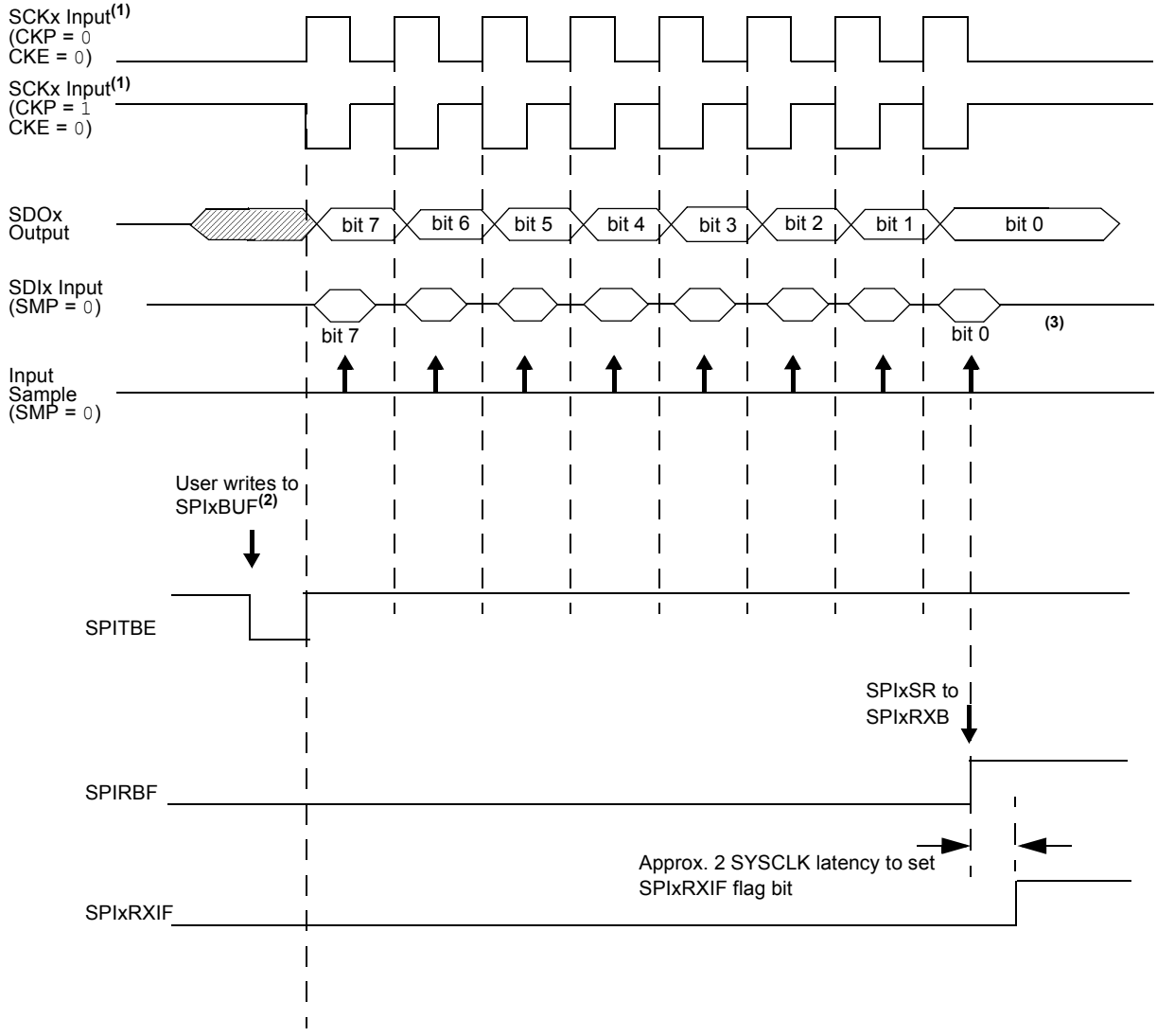
```
/*
    The following code example will initialize the SPI1 in Slave mode.
    It assumes that none of the SPI1 input pins are shared with an analog input. If so, the
    AD1PCFG and corresponding TRIS registers have to be properly configured.
*/
    int     rData;

    IEC0CLR=0x03800000;              // disable all interrupts
    SPI1CON = 0;                     // Stops and resets the SPI1.
    rData=SPI1BUF;                   // clears the receive buffer
    IFS0CLR=0x03800000;              // clear any existing event
    IPC5CLR=0x1f000000;              // clear the priority
    IPC5SET=0x0d000000;              // Set IPL=3, Subpriority 1
    IEC0SET=0x03800000;              // Enable Rx, Tx and Error interrupts

    SPI1STATCLR=0x40;                // clear the Overflow
    SPI1CON=0x8000;                  // SPI ON, 8 bits transfer, Slave mode

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


Figure 23-8: SPI Slave Mode Operation in 8-Bit Mode with Slave Select Pin Disabled (MODE32 = 0, MODE16 = 0, SSEN = 0)

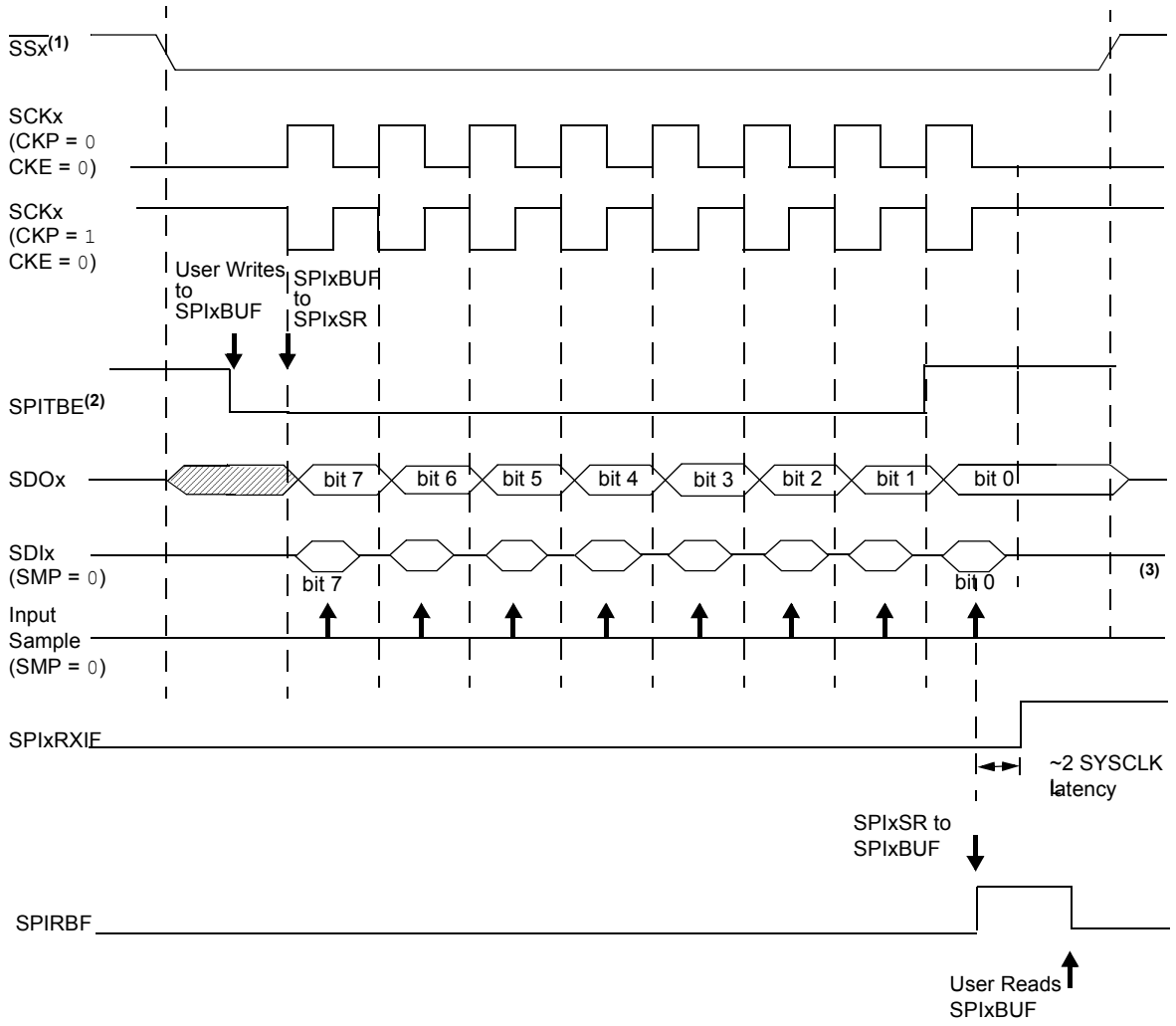


Note 1: Two SPI Clock modes are shown here only to demonstrate the functionality of bits CKP (SPIxCON<6>) and CKE (SPIxCON<8>). Any combination of CKP and CKE bits can be chosen for module operation.

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

3: Operation for 8-bit mode is shown. 16-bit and 32-bit modes are similar.

Figure 23-9: SPI Slave Mode Operation in 8-Bit Mode with Slave Select Pin Enabled (MODE32 = 0, MODE16 = 0, SSEN = 1)



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

2: Transmit data is held in SPIxTXB and SPITBE (SPIxSTAT<2>) remains clear until all bits are transmitted.

3: Operation for 8-bit mode is shown. 16-bit and 32-bit modes are similar.

23.3.3 SPI Error Handling

When a new data word has been shifted into shift register SPIxSR and the previous contents of receive register SPIxRXB have not been read by the user software, the SPIROV bit (SPIx-STAT<6>) will be set. The module will not transfer the received data from SPIxSR 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.

23.3.4 SPI Receive-Only Operation

Setting the control bit DISSDO (SPIxCON<12>) disables transmission at the SDOx pin. This allows the SPIx 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.

23.3.5 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 control bit FRMEN (SPIxCON<31>) 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 SSEN (SPIxCON<7>) is ignored.
- The control bit FRMSYNC (SPIxCON<30>) 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 (SPIxCON<29>) determines the frame synchronization pulse polarity for a single SPI clock cycle.

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

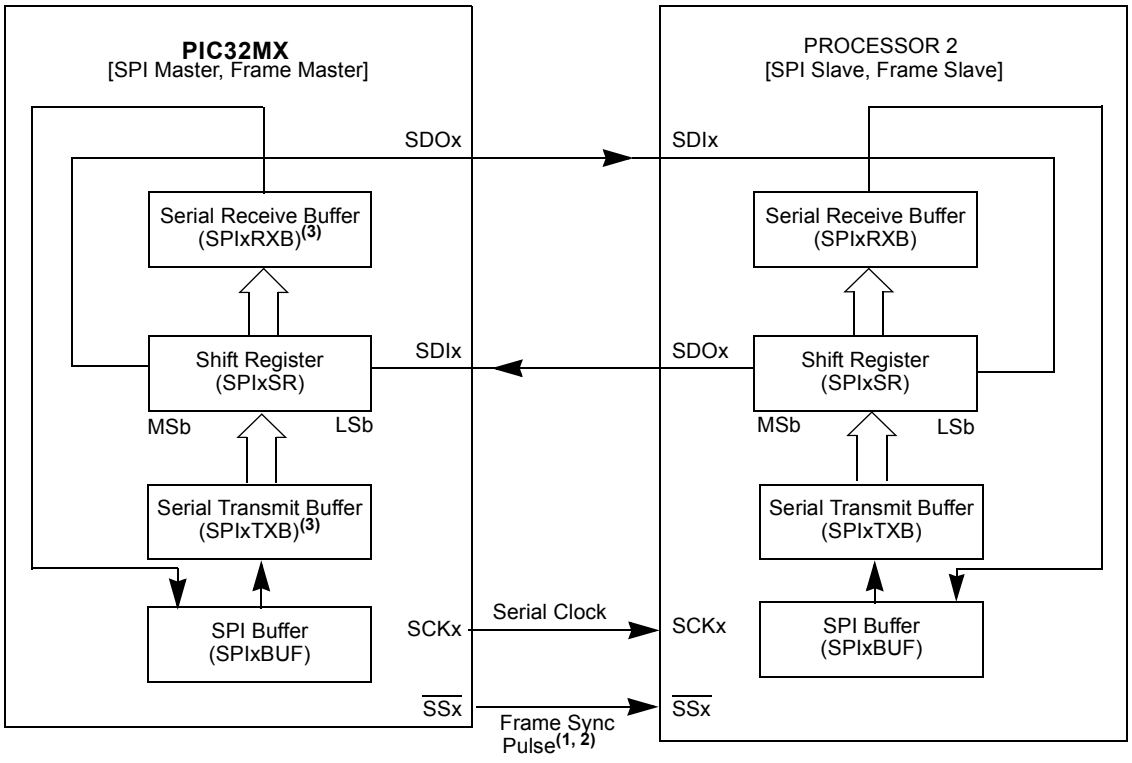
  The SPI module uses a frame synchronization pulse received at the $\overline{SSx}$ pin.

The Framed SPI modes are supported in conjunction with the Master and Slave modes. Thus, the following Framed SPI 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 SPIx module generates the serial clock and the frame synchronization pulse.

Figure 23-10: SPI Master, Frame Master Connection Diagram



Note 1: In Framed SPI modes, the $\overline{SSx}$ pin is used to transmit/receive the frame synchronization pulse.
2: Framed SPI modes require the use of all four pins, i.e., using the $\overline{SSx}$ pin is not optional.
3: The SPIxTXB and SPIxRXB registers are memory mapped to the SPIxBUF register.

23.3.5.1 SCKx in Framed SPI Modes

When FRMEN (SPIxCON<31>) = 1 and MSTEN (SPIxCON<5>) = 1, the SCKx pin becomes an output and the SPI clock at SCKx becomes a free-running clock.

When FRMEN = 1 and MSTEN = 0, the SCKx pin becomes an input. The source clock provided to the SCKx pin is assumed to be a free-running clock.

The polarity of the clock is selected by bit CKP (SPIxCON<6>). Bit CKE (SPIxCON<8>) is not used for the Framed SPI modes.

When CKP = 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 = 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.

23.3.5.2 SPIx Buffers in Framed SPI Modes

When FRMSYNC (SPIxCON<30>) = 0, the SPIx 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 SPIxBUF location (thus writing the SPIxTXB register with transmit data). At the end of the frame sync pulse, SPIxTXB is transferred to SPIxSR and data transmission/reception begins.

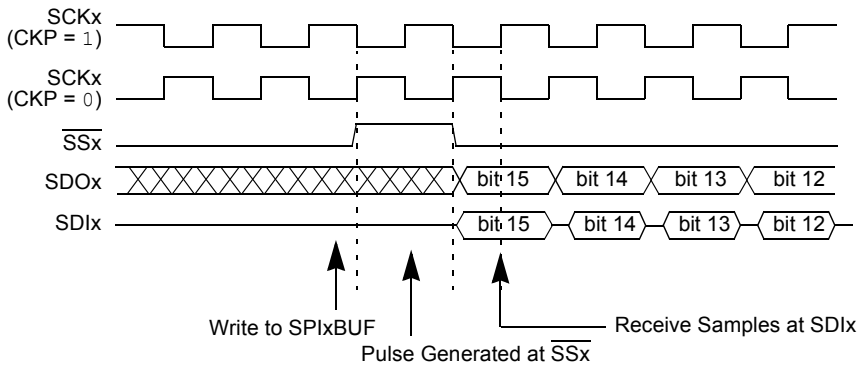
When FRMSYNC = 1, the module is in Frame Slave mode. In this mode, the frame sync pulse is generated by an external source. When the module samples the frame sync pulse, it will transfer the contents of the SPIxTXB register to SPIxSR, and data transmission/ reception begins. The user must make sure that 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.

23.3.5.3 SPI Master Mode and Frame Master Mode

This Framed SPI mode is enabled by setting bits MSTEN (SPIxCON<5>) and FRMEN (SPIxCON<31>) to '1', and bit FRMSYNC (SPIxCON<30>) 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 $\overline{SSx}$ pin will be driven active, high or low depending on bit FRMPOL (SPIxCON<29>), on the next transmit edge of the SCKx clock. The $\overline{SSx}$ pin will be high for one SCKx clock cycle. The module will start transmitting data on the next transmit edge of the SCKx, as shown in Figure 23-11. A connection diagram indicating signal directions for this operating mode is shown in Figure 23.9.

Figure 23-11: SPI Master, Frame Master (MODE32 = 0, MODE16 = 1, SPIFE = 0, FRMPOL = 1)



23.3.5.4 SPI Master Mode and Frame Slave Mode

This Framed SPI mode is enabled by setting bits MSTEN (SPIxCON<5>), FRMEN (SPIxCON<31>), and bits FRMSYNC (SPIxCON<30>) 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 low depending on bit FRMPOL (SPIxCON<29>), data will be transmitted on the subsequent transmit edge of the SPI clock, as shown in Figure 23-12. The 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 23-13.

Figure 23-12: SPI Master, Frame Slave (MODE32 = 0, MODE16 = 1, SPIFE = 0, FRMPOL = 1)

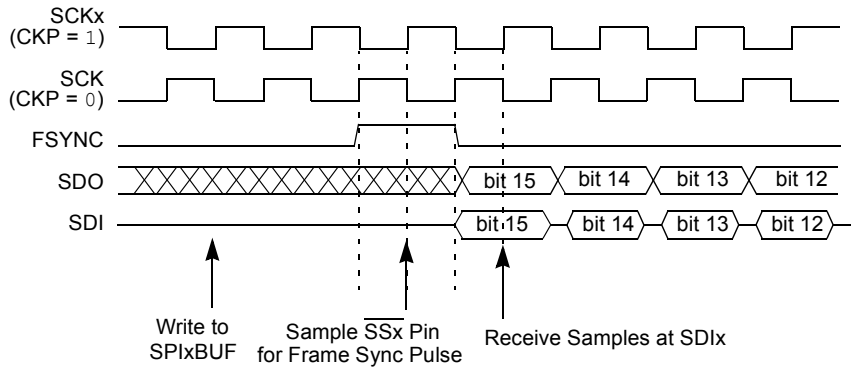
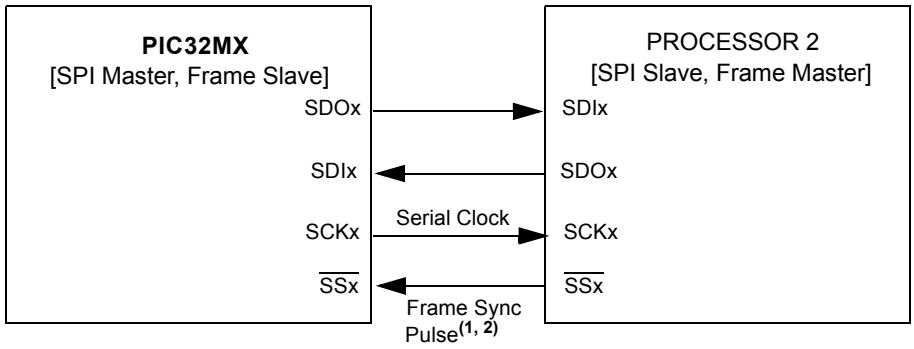


Figure 23-13: SPI Master, Frame Slave Connection Diagram

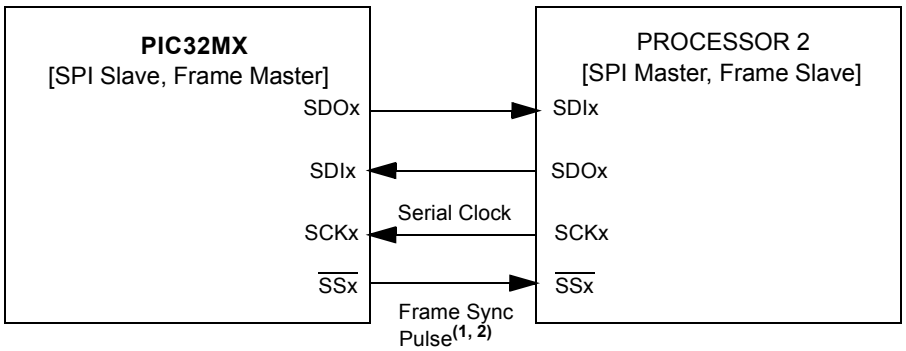


Note 1: In Framed SPI modes, the $\overline{SSx}$ pin is used to transmit/receive the frame synchronization pulse.
2: Framed SPI modes require the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).

23.3.5.5 SPI Slave Mode and Frame Master Mode

This Framed SPI mode is enabled by setting bit MSTEN (SPIxCON<5>) to '0', bit FRMEN (SPIxCON<31>) to '1' and bit FRMSYNC (SPIxCON<30>) to '0'. The input SPI clock will be continuous in Slave mode. The $\overline{SSx}$ pin will be an output when bit FRMSYNC is low. Therefore, when SPIBUF is written, the module will drive the $\overline{SSx}$ pin active, high or low depending on bit FRMPOL (SPIxCON<29>), on the next transmit edge of the SPI clock. The $\overline{SSx}$ 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 23-14.

Figure 23-14: SPI Slave, Frame Master Connection Diagram

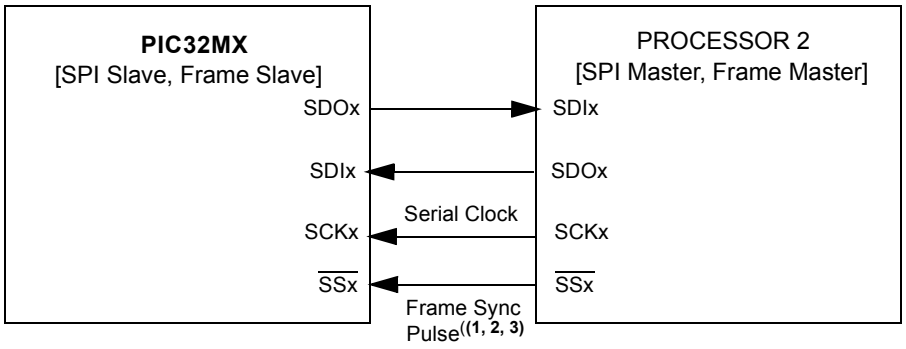


Note 1: In Framed SPI modes, the $\overline{SSx}$ pin is used to transmit/receive the frame sync pulse.
2: Framed SPI modes require the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).

23.3.5.6 SPI Slave Mode and Frame Slave Mode

This Framed SPI mode is enabled by setting bits MSTEN (SPIxCON<5>) to '0', FRMEN (SPIxCON<31>) to '1', and FRMSYNC (SPIxCON<30>) to '1'. Therefore, both the SCKx and $\overline{SSx}$ pins will be inputs. The $\overline{SSx}$ pin will be sampled on the sample edge of the SPI clock. When $\overline{SSx}$ is sampled active, high or low depending on bit FRMPOL (SPIxCON<29>), 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 23-15.

Figure 23-15: SPI Slave, Frame Slave Connection Diagram



Note 1: In Framed SPI modes, the $\overline{SSx}$ pin is used to transmit/receive the frame sync pulse.
2: Framed SPI modes require the use of all four pins (i.e., using the $\overline{SSx}$ pin is not optional).
3: Slave Select is not available when using Frame mode as a Slave device.

23.3.6 SPI Master Mode Clock Frequency

The SPI module allows flexibility in baud rate generation through the 9-bit SPIxBRG register. SPIxBRG 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 SPIxBRG. The SCKx clock obtained by dividing PBCLK is of 50% duty cycle and it is provided to the external devices via the SCKx pin.

Note:	The SCKx clock is not free running for nonframed 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 23-1 defines the SCKx clock frequency as a function of SPIxBRG settings.

Equation 23-1:

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

Therefore, the maximum baud rate possible is $F_{PB}/2$ (SPIXBRG = 0), and the minimum baud rate possible is $F_{PB}/1024$.

Some sample SPI clock frequencies (in kHz) are shown in the table below:

Table 23-3: Sample SCKx Frequencies

SPIxBRG Setting	0	15	31	63	85	127
F_{PB} = 50 MHz	25.00 MHz	1.56 MHz	781.25 kHz	390.63 kHz	290.7 kHz	195.31 kHz
F_{PB} = 40 MHz	20.00 MHz	1.25 MHz	625.00 kHz	312.50 kHz	232.56 kHz	156.25 kHz
F_{PB} = 25 MHz	12.50 MHz	781.25 kHz	390.63 kHz	195.31 kHz	145.35 kHz	97.66 kHz
F_{PB} = 20 MHz	10.00 MHz	625.00 kHz	312.50 kHz	156.25 kHz	116.28 kHz	78.13 kHz
F_{PB} = 10 MHZ	5.00 MHz	312.50 kHz	156.25 kHz	78.13 kHz	58.14 kHz	39.06 kHz

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

23.4 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, signalled by SPI1RXIF (IFS0<25>), SPI2RXIF(IFS1<7>). This event occurs when there is new data assembled in the SPIxBUF receive buffer.
- Transmit buffer empty interrupts, signalled by SPI1TXIF (IFS0<24>), SPI2TXIF (IFS1<6>). This event occurs when there is space available in the SPIxBUF transmit buffer and new data can be written.
- Receive buffer overflow interrupts, signalled by SPI1EIF (IFS0<23>), SPI2EIF(IFS1<5>). This event occurs when there is an overflow condition for the SPIxBUF receive buffer, i.e., new receive data assembled but the previous one not read.

All these interrupt flags must be cleared in software.

To enable the SPI interrupts, use the respective SPI interrupt enable bits:

- SPI1RXIE (IEC0<25>) and SPI2RXIE (IEC1<7>)
- SPI1TXIE (IEC0<24>) and SPI2TXIE (IEC1<6>)
- SPI1FIE (IEC0<23>) and SPI2FIE (IEC1<5>)

The interrupt priority level bits and interrupt subpriority level bits must be also be configured:

- SPI1IP (IPC5<28:26>), SPI1IS (IPC5<25:24>)
- SPI2IP (IPC7<28:26>), SPI2IS (IPC7<25:24>)

Refer to **Section 8. "Interrupts"** for further details.

23.4.1 Interrupt Configuration

Each SPI module has 3 dedicated interrupt flag bits: SPIxEIF, SPIxRXIF, and SPIxTXIF, and corresponding interrupt enable/mask bits SPIxEIE, 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.

SPIxTXIF is set when the SPI transmit buffer is empty and another character can be written to the SPIxBUF register. SPIxRXIF is set when there is a received character available in SPIxBUF. SPIxEIF is set when a Receive Overflow condition occurs.

Note that bits SPIxTXIF, SPIxRXIF, and SPIxEIF will be set without regard to the state of the corresponding enable bit. IF bits can be polled by software if desired.

Bits SPIxEIE, SPIxTXIE, SPIxRXIE are used to define the behavior of the Interrupt Controller (INT) when a corresponding SPIxEIF, SPIxTXIF, or SPIxRXIF bit is set. When the corresponding IE bit is clear, the INT module does not generate a CPU interrupt for the event. If the IE bit is set, the INT module will generate an interrupt to the CPU when the corresponding IF bit is set (subject to the priority and subpriority 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 subpriority bits allow setting the priority of an interrupt source within a priority group. The values of the subpriority SPIxIS<1:0> range from 3 (the highest priority) to 0, the lowest priority. An interrupt within the same priority group but having a higher subpriority value will not preempt a lower subpriority interrupt that is in progress.

The priority group and subpriority bits allow more than one interrupt source to share the same priority and subpriority. If simultaneous interrupts occur in this configuration the natural order of the interrupt sources within a Priority/subpriority group pair determine 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, subpriority, 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 (refer to Table 23-4) 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 SPIxEIF, SPIxTXIF, or SPIxRXIF, and then exit. Refer to the vector address table details in the **Section 8. "Interrupts"** for more information on interrupts.

Table 23-4: SPI Interrupt Vectors for Various Offsets with EBASE = 0x8000:0000

Interrupt	Vector/Natural Order	IRQ Number	Vector Address IntCtl.VS = 0x01	Vector Address IntCtl.VS = 0x02	Vector Address IntCtl.VS = 0x04	Vector Address IntCtl.VS = 0x08	Vector Address IntCtl.VS = 0x10
SPI1E	23	23	8000 04E0	8000 07C0	8000 0D80	8000 1900	8000 3000
SPI1TX	23	24	8000 04E0	8000 07C0	8000 0D80	8000 1900	8000 3000
SPI1RX	23	25	8000 04E0	8000 07C0	8000 0D80	8000 1900	8000 3000
SPI2IE	31	37	8000 05C0	8000 0980	8000 1100	8000 2000	8000 3E00
SPI2TX	31	38	8000 05C0	8000 0980	8000 1100	8000 2000	8000 3E00
SPI2RX	31	39	8000 05C0	8000 0980	8000 1100	8000 2000	8000 3E00

Example 23-3: SPI Initialization with Interrupts Enabled Code Example

```
/*
    The following code example illustrates an SPI1 interrupt configuration.
    When the SPI1 interrupt is generated, the cpu will jump to the vector assigned to SPI1
    interrupt.
    It assumes that none of the SPI1 input pins are shared with an analog input. If so, the
    AD1PCFG and corresponding TRIS registers have to be properly configured.
*/

    int rData;

    IEC0CLR=0x03800000;             // disable all SPI interrupts
    SPI1CON = 0;                    // Stops and resets the SPI1.
    rData=SPI1BUF;                  // clears the receive buffer
    IFS0CLR=0x03800000;             // clear any existing event
    IPC5CLR=0x1f000000;             // clear the priority
    IPC5SET=0x0d000000;             // Set IPL=3, Subpriority 1
    IEC0SET=0x03800000;             // Enable Rx, Tx and Error interrupts

    SPI1BRG=0x1;                    // use FPB/4 clock frequency
    SPI1STATCLR=0x40;               // clear the Overflow
    SPI1CON=0x8220;                 // SPI ON, 8 bits transfer, SMP=1, Master mode
```


Example 23-4: SPI1 ISR Code Example

```
/*
    The following code example demonstrates a simple interrupt service routine for SPI1
    interrupts. The user's code at this vector should perform any application specific operation
    and must clear the SPI1 interrupt flags before exiting.
*/

void __ISR(_SPI1_VECTOR, IPL7)__SPI1Interrupt(void)
{
                                // ... perform application specific operations in response to the
                                // interrupt

    IFS0CLR = 0x03800000;       // Be sure to clear the SPI1 interrupt flags
                                // before exiting the service routine.
}
```

Note: The SPI1 ISR code example shows MPLAB® C32 C compiler specific syntax. Refer to your compiler manual regarding support for ISRs.

23.5 OPERATION IN POWER-SAVING AND DEBUG MODES

Note: In this manual, a distinction is made between a power mode as it is used in a specific module, and a power mode as it is used by the device, e.g., Sleep mode of the Comparator and SLEEP mode of the CPU. To indicate which type of power mode is intended, uppercase and lowercase letters (Sleep, Idle, Debug) signify a module power mode, and all uppercase letters (SLEEP, IDLE, DEBUG) signify a device power mode.

23.5.1 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 subsections describe mode-specific behavior.

23.5.1.1 Master Mode in SLEEP Mode

The following items should be noted in SLEEP mode:

- The Baud Rate Generator is stopped and reset.
- On-going transmission and reception sequences are aborted. The module will not resume aborted sequences when SLEEP mode is exited.
- Once in SLEEP mode, the module will not transmit or receive any new data.

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

23.5.1.2 Slave Mode in SLEEP Mode

In the Slave mode, the SPI module operates from the SCK 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, bit SPIxRXIF will be set. If SPI 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 SPIx module is going into or coming out of SLEEP mode.

23.5.2 IDLE Mode

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

23.5.2.1 Master Mode in IDLE Mode

Bit SIDL (SPIxCON<13>) selects whether the module will stop or continue functioning in IDLE mode.

- If SIDL = 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 SIDL = 0, the module will continue operation in IDLE mode.

23.5.2.2 Slave Mode in IDLE Mode

The module will continue operation in IDLE mode irrespective of the SIDL setting. The behavior is identical to the one in SLEEP mode.

23.5.3 DEBUG Mode

Bit FRZ (SPIxCON<14>) determines whether the SPI module will run or stop while the CPU is executing DEBUG exception code (i.e., application is halted) in DEBUG mode. When FRZ = 0, the SPI module continues to run, even when the application is halted in DEBUG mode. When FRZ = 1 and the application is halted in DEBUG mode, the behavior is different from Master-to-Slave mode.

23.5.3.1 Freeze in Master Mode

When FRZ = 1 and the application is halted in DEBUG mode, the module will freeze its operations and make no changes to the state of the SPI module, such that it will continue exactly as it left off. In other words, the transmission/reception **is not aborted** during this halt.

23.5.3.2 Freeze in Slave Mode

In Slave mode with an externally provided SCK, the module will continue to operate, even though it is frozen (FRZ = 1), i.e., the shift register is functional. However, when data is received in the shift register before DEBUG mode is exited, the data that has been received is ignored, i.e., not transferred to SPIxBUF.

23.5.3.3 Operation of SPIxBUF

23.5.3.3.1 Reads During DEBUG Mode

During DEBUG mode, SPIxBUF can be read; but the read operation does not affect any Status bits. For example, if bit SPIRBF (SPIxSTAT<0>) is set when DEBUG mode is entered, it will remain set on EXIT From DEBUG mode, even though the SPIxBUF register was read in DEBUG mode.

23.5.3.3.2 Writes During DEBUG Mode

When FRZ is set, write functionality depends on whether the SPI is in Master or Slave mode.

In Master mode: the write operation will place the data in the buffer, but the transmission will not start until the DEBUG mode is exited.

In Slave mode: the write operation will place the data in the buffer, and the data will be sent out whenever the Master initiates a new transaction, even if the device is still in DEBUG mode.

Note:	The FRZ bit is readable and writable only when the CPU is executing in Debug Exception mode. In all other modes, the FRZ bit reads as '0'. If FRZ bit is changed during DEBUG mode, the new value does not take effect until the current Debug Exception mode is exited and re-entered. During the Debug Exception mode, the FRZ bit reads the state of the peripheral when entering DEBUG mode.

23.6 EFFECTS OF VARIOUS RESETS

23.6.1 Device Reset

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

- resets all fields in SPIxCON and SPIxSTAT
- resets the transmit and receive buffers (SPIx-BUF) to the empty state
- resets the Baud Rate Generator

23.6.2 Power-on Reset

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

23.6.3 Watchdog Timer Reset

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

23.7 PERIPHERALS USING SPI MODULES

There are no other peripherals using the SPI module.

23.8 I/O PIN CONTROL

Enabling the SPI modules will configure the I/O pin direction as defined by the SPI control bits (see Table 23-5). The port TRIS and LATCH registers will be overridden.

Table 23-5: I/O Pin Configuration for Use with SPI Modules

Required Settings for Module Pin Control							
I/O Pin Name	Required	Module Control(3)	Bit Field(3)	TRIS(4)	Pin Type	Buffer Type	Description
SCK1, SCK2	Yes	ON and MSTEN	—	X	O	CMOS	SPI1, SPI2 module Clock Output in Master mode.
SCK1, SCK2	Yes	ON and $\overline{\text{MSTEN}}$	—	X(5)	I	CMOS	SPI1, SPI2 module Clock Input in Slave mode.
SDI1, SDI2	Yes	ON	—	X(5)	I	CMOS	SPI1, SPI2 module Data Receive pin
SDO1, SDO2	Yes(1)	ON	DISSDO	X	O	CMOS	SPI1, SPI2 module Data Transmit pin
$\overline{\text{SS1}}$, $\overline{\text{SS2}}$	Yes(2)	ON and $\overline{\text{FRMEN}}$ and $\overline{\text{MSTEN}}$	SSEN	X(5)	I	CMOS	SPI1, SPI2 module Slave Select Control pin.
$\overline{\text{SS1}}$, $\overline{\text{SS2}}$	Yes	ON and FRMEN and FRMSYNC	—	X(5)	I	CMOS	SPI1, SPI2 Frame Sync Pulse input in Frame mode.
$\overline{\text{SS1}}$, $\overline{\text{SS2}}$	Yes	ON and FRMEN and $\overline{\text{FRMSYNC}}$	—	X	O	CMOS	SPI1,SPI2 Frame Sync Pulse output in Frame mode.

Legend: CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels
I = Input
O = Output

Note 1: The SDO pins are only required when SPI data output is needed. Otherwise, these pins can be used for general purpose IO and require the user to set the corresponding TRIS control register bits.

2: The Slave Select pins are only required when a select signal to the slave device is needed. Otherwise, these pins can be used for general purpose IO and require the user to set the corresponding TRIS control register bits.

3: These bits are contained in the SPIxCON register.

4: The setting of the TRIS bit is irrelevant.

5: If the input pin is shared with an analog input, then the AD1PCFG and corresponding TRIS register has to be properly set to configure this input as digital.

23.9 DESIGN TIPS

Question 1: ***Can I use the SSx pin as an output to a slave device when the PIC32MX SPI module is configured in Master mode?***

Answer: Yes, you can. Notice, however, that the SSx pin is not driven by the SPI Master. You have to drive the bit yourself and pulse it before the SPI transmission takes place. You can use any other I/O pin for that purpose.

Question 2: ***If I do not use the SDO output for my SPI module, is this I/O pin available as a general purpose I/O pin?***

Answer: Yes. If you are not interested in transmitting data, only receiving, you can use the SDO pin as a general I/O pin. This is mainly useful for SPI modules that are configured as SPI slave devices. Note that when used as a general purpose I/O pin, the user is responsible for configuring the respective data direction register (TRIS) for input or output.

23.10 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 PIC32MX 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/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 PIC32MX family of devices.

23.11 REVISION HISTORY

Revision A (July 2007)

This is the initial released version of this document.

Revision B (October 2007)

Revised Examples 23-1, 23-2, 23-3; Table 23-5.

Revision C (October 2007)

Updated document to remove Confidential status.

Revision D (April 2008)

Revised status to Preliminary; Revised U-0 to r-x.

