TMS320C672x DSP Serial Peripheral Interface (SPI)

Reference Guide

Literature Number: SPRU718B
October 2005–Revised July 2007

TEXAS INSTRUMENTS

Contents

List of Figures

List of Tables

Read This First

About This Manual

This reference guide provides the specifications for a 16-bit configurable, synchronous serial peripheral interface. The SPI is a programmable-length shift register, used for high speed communication between external peripherals or other microcontrollers

Notational Conventions

The TMS320C672x™ DSP has two SPI modules. This document describes both modules. Where pins are described, the pin name is listed as SPIx_<suffix>. The 'x' is either '0' for SPI0 or '1' for SPI1.

When describing particular bits, the convention REGISTER.BIT(s) is used. For example, bit 0 of the SPIGCR0 register (the RESET bit) is referred to as SPIGCR0.$\overline{\text{RESET}}$.

Related Documentation From Texas Instruments

The following documents describe the C6000™ devices and related support tools. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

TMS320C672x DSP Peripherals Overview Reference Guide (literature number SPRU723) describes peripherals available on the TMS320C672x™ DSPs.

TMS320C6713 to TMS320C672x Migration Guide (literature number SPRAA78) describes the issues related to migrating from the TMS320C6713 to the TMS320C672x™ floating-point digital signal processors.

TMS320C672x DSP CPU and Instruction Set Reference Guide (literature number SPRU733) describes the TMS320C672x™ CPU architecture, instruction set, pipeline, and interrupts for these digital signal processors.

Trademarks

All other trademarks are the property of their respective owners.

Reference Guide
SPRU718B–October 2005–Revised July 2007

TMS320C672x DSP Serial Peripheral Interface (SPI)

1 Overview

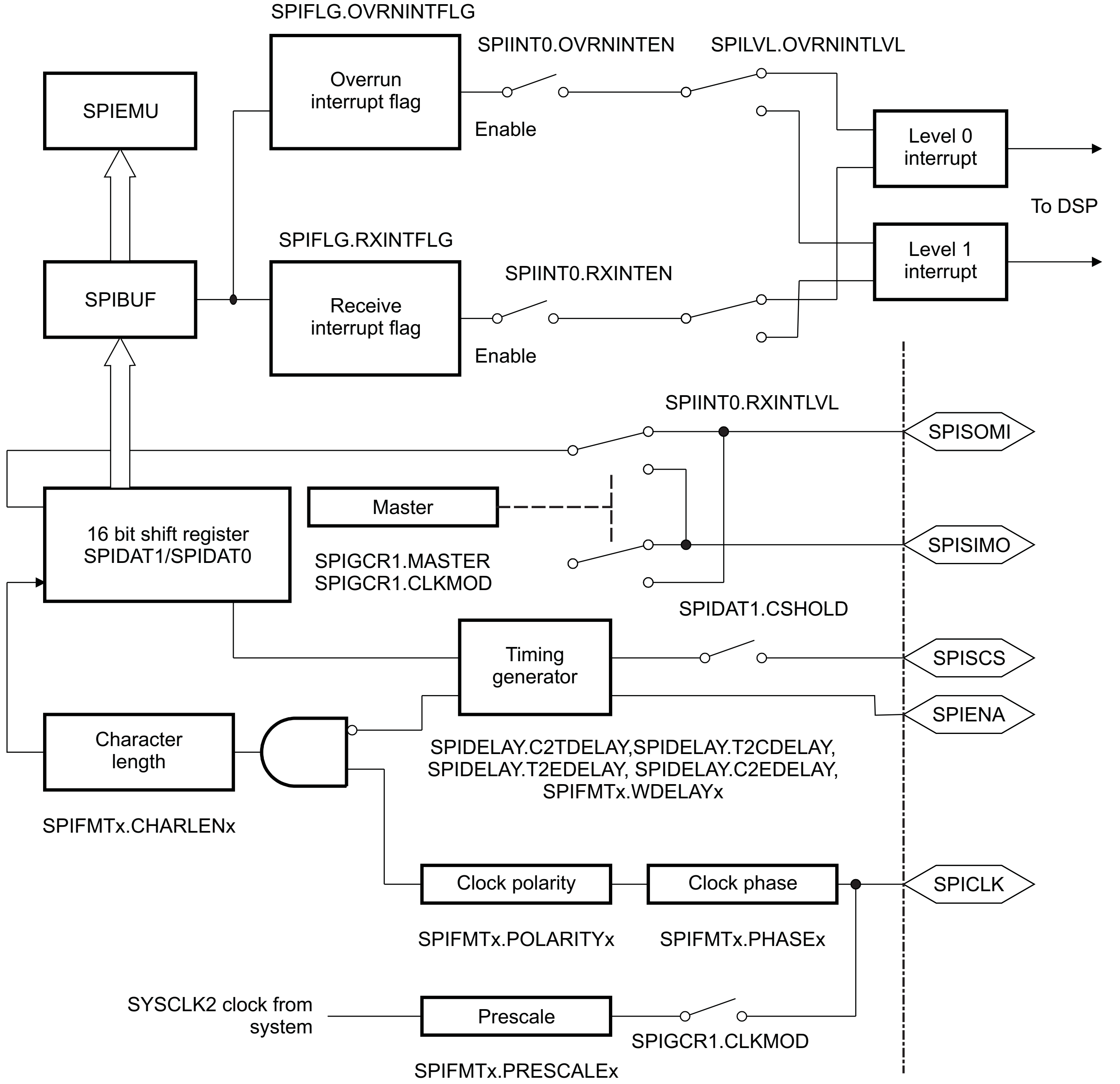
This reference guide describes the TMS320C672x Serial Peripheral Interface Module (SPI). The SPI is a synchronous, full-duplex serial port which supports 3-pin, 4-pin, and 5-pin options. Figure 1 contains a block diagram of the SPI module. The major SPI components are:

- 16-bit shift register (SPIDAT0)
- Combined 16-bit shift register and format selection register (SPIDAT1)
- 16-bit receive buffer register (SPIBUF)
- 16-bit receive buffer emulation 'alias' register (SPIEMU)
- 8-bit baud clock generator
- Serial clock (SPIx_CLK) I/O pin
- Slave in, master out (SPIx_SIMO) I/O pin
- Slave out, master in (SPIx_SOMI) I/O pin
- SPI enable ($\overline{\text{SPIx\_ENA}}$) I/O pin (4-pin or 5-pin mode only)
- Slave chip select ($\overline{\text{SPIx\_SCS}}$) I/O pin (4-pin or 5-pin mode only)
- Programmable clock divider, including polarity and phase options.
- Interrupt capability

The SPI allows software to program the following options:

- Master or Slave Mode
- SPIx_CLK frequency (interface clock [SYSCLK2] /8 through /256)
- SPI pins as functional or digital I/O pins
- Programmable 3-, 4-, and 5-pin options
- Programmable character length (2 to 16 bits) and shift direction (MSB/LSB first)
- Phase (delay/no delay), Polarity (high or low)
- Programmable delay between transmissions in master mode
- Chip select to clock delay insertion in master mode
- Chip select hold mode for master

Figure 1. SPI Module Block Diagram



2 SPI Operation

This section describes the operation modes of the SPI. First it gives an overview of SPI operation and then provides details on the 3-, 4-, and 5-pin options, as well as more specific details on the supported data formats.

2.1 Overview of SPI Operation

The SPI operates in master or slave mode. The SPI bus master is the device which drives the SPIx_CLK, SPIx_SIMO, and optionally the SPIx_SCS signals, and therefore initiates SPI bus transfers. SPIGCR1.CLKMOD and SPIGCR1.MASTER select between master and slave mode.

In both master and slave mode, the SPI supports four options:

- 3-pin option: See Section 2.5.
- 4-pin with chip select option: See Section 2.6.
- 4-pin with enable option: See Section 2.7.
- 5-pin option (with enable and chip select): See Section 2.8.

The 3-pin option is the basic clock, data in, and data out SPI interface and uses the SPIx_CLK, SPIx_SIMO, and SPIx_SOMI pins. The 4-pin with chip select option adds the SPIx_SCS pin which is used to support multiple SPI slave devices on a single SPI bus. The 4-pin with enable option adds the SPIx_ENA pin which is used to increase the overall throughput by adding hardware handshaking. The 5-pin option uses all the SPI pins and is a superset of the different options.

2.2 SPI Registers

A list of the C672x device SPI0 and SPI1 module internal registers is contained in Table 1.

Table 1. SPI Registers

SPI0 Address	SPI1 Address	Mnemonic	Name	Description
0x4700 0000	0x4800 0000	SPIGCR0	Global Control Register 0	Contains the Software Reset bit for the module
0x4700 0004	0x4800 0004	SPIGCR1	Global Control Register 1	Controls basic configurations of the module
0x4700 0008	0x4800 0008	SPIINT0	Interrupt Register	Enables bits for interrupts, error, DMA and other functionality
0x4700 000C	0x4800 000C	SPILVL	Interrupt Level Register	SPI Interrupt Levels are set in this register
0x4700 0010	0x4800 0010	SPIFLG	Interrupt Flag Register	Shows the status of several events during the operation
0x4700 0014	0x4800 0014	SPIPC0	Pin Control Register 0	Determines if pins will operate as general I/O or SPI functional pin
0x4700 0018	0x4800 0018	SPIPC1	Pin Control Register 1	Controls the direction of data on the I/O pins
0x4700 001C	0x4800 001C	SPIPC2	Pin Control Register 2	Reflects the values on the I/O pins
0x4700 0020	0x4800 0020	SPIPC3	Pin Control Register 3	Controls the values sent to the I/O pins
0x4700 0024	0x4800 0024	SPIPC4	Pin Control Register 4	Sets data values in the SPIPC3 register
0x4700 0028	0x4800 0028	SPIPC5	Pin Control Register 5	Clears values in the SPIPC3 register
0x4700 002C	0x4800 002C	Reserved	Reserved	Reserved - Do not write to this register
0x4700 0030	0x4800 0030	Reserved	Reserved	Reserved - Do not write to this register
0x4700 0034	0x4800 0034	Reserved	Reserved	Reserved - Do not write to this register
0x4700 0038	0x4800 0038	SPIDAT0	Shift Register 0	Main shift register
0x4700 003C	0x4800 003C	SPIDAT1	Shift Register 1	Shift register used in automatic slave chip select mode only
0x4700 0040	0x4800 0040	SPIBUF	Buffer Register	Holds received word
0x4700 0044	0x4800 0044	SPIEMU	Emulation Register	Mirror of SPIBUF. Read does not clear flags
0x4700 0048	0x4800 0048	SPIDELAY	Delay Register	Sets $\overline{\text{SPIx\_SCS}}$ mode, $\overline{\text{SPIx\_SCS}}$ pre-/post-transfer delay time and $\overline{\text{SPIx\_ENA}}$ timeout
0x4700 004C	0x4800 004C	SPIDEF	Default Chip Select Register	In $\overline{\text{SPIx\_SCS}}$ decoded mode only: sets high low/active $\overline{\text{SPIx\_SCS}}$ signals
0x4700 0050	0x4800 0050	SPIFMT0	Format 0 Register	Configuration of data word format 0
0x4700 0054	0x4800 0054	SPIFMT1	Format 1 Register	Configuration of data word format 1
0x4700 0058	0x4800 0058	SPIFMT2	Format 2 Register	Configuration of data word format 2
0x4700 005C	0x4800 005C	SPIFMT3	Format 3 Register	Configuration of data word format 3
0x4700 0060	0x4800 0060	TGINTVECT0	TG Interrupt Vector	Transfer group interrupt vector for line SPI INT0
0x4700 0064	0x4800 0064	TGINTVECT1	TG Interrupt Vector	Transfer group interrupt vector for line SPI INT1

2.3 Master Mode Settings

The four master mode options are defined by the configuration bit settings listed in Table 2. Other configuration bits may take any value in the range listed in Table 3. The values listed in Table 2 and Table 3 should not be changed while SPIGCR1.ENABLE='1'. Note that in certain cases the allowed values may still be ignored. For example the Table 3 below indicates that SPIDELAY may take a range of values in Master 3-pin mode, however SPIDELAY has no effect in Master 3-pin mode. For complete details on each mode read the following sections which explain the SPI operation for each of the master modes.

Table 2. SPI Register Settings Defining Master Modes

Register	Bit(s)	Master 3-pin	Master 4-pin Chip Select	Master 4-pin Enable	Master 5-pin
SPIGCR0	$\overline{\text{RESET}}$	1b	1b	1b	1b
SPIGCR1	ENABLE	1b	1b	1b	1b
SPIGCR1	LOOPBACK	0b	0b	0b	0b
SPIGCR1	CLKMOD	1b	1b	1b	1b
SPIGCR1	MASTER	1b	1b	1b	1b
SPIPC0	SOMIFUN	1b	1b	1b	1b
SPIPC0	SIMOFUN	1b	1b	1b	1b
SPIPC0	CLKFUN	1b	1b	1b	1b
SPIPC0	ENAFUN	0b	0b	1b	1b
SPIPC0	SCSFUN0	0b	1b	0b	1b

Table 3. Allowed SPI Register Settings in Master Modes

Register	Bit(s)	Master 3-pin	Master 4-pin Chip Select	Master 4-pin Enable	Master 5-pin
SPIINT0	ENABLEHIGHZ	{0,1}	{0,1}	{0,1}	{0,1}
SPIFMTx[1]	WDELAYx[1]	{0x00,0x01,...,0x3F}	{0x00,0x01,...,0x3F}	{0x00,0x01,...,0x3F}	{0x00,0x01,...,0x3F}
SPIFMTx[1]	SHIFTDIRx[1]	{0,1}	{0,1}	{0,1}	{0,1}
SPIFMTx[1]	WAITENAx[1]	{0}	{0}	{1}	{1}
SPIFMTx[1]	POLARITYx[1]	{0,1}	{0,1}	{0,1}	{0,1}
SPIFMTx[1]	PHASEx[1]	{0,1}	{0,1}	{0,1}	{0,1}
SPIFMTx[1]	PRESCALEx[7:0][1]	{0x07,0x08,...,0xFF}	{0x07,0x08,...,0xFF}	{0x07,0x08,...,0xFF}	{0x07,0x08,...,0xFF}
SPIFMTx[1]	CHARLENx[4:0][1]	{0x02,0x03,...,0x10}	{0x02,0x03,...,0x10}	{0x02,0x03,...,0x10}	{0x02,0x03,...,0x10}
SPIDELAY	C2TDELAY[4:0]	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}
SPIDELAY	T2CDELAY[4:0]	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}
SPIDELAY	T2EDELAY[7:0]	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}
SPIDELAY	C2EDELAY[7:0]	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}

[1] Where x is 0, 1, 2, 3.

2.4 Slave Mode Settings

The four slave mode options are defined by the configuration bit settings listed in Table 4. Other configuration bits may take any value in the range listed in Table 5. The values listed in Table 4 and Table 5 should not be changed while SPIGCR1.ENABLE='1'. Note that in certain cases the allowed values may still be ignored. For complete details on each mode read the following sections which explain the SPI operation for each of the slave modes.

Table 4. SPI Register Settings Defining Slave Modes

Register	Bit(s)	Slave 3-pin	Slave 4-pin Chip Select	Slave 4-pin Enable	Slave 5-pin
SPIGCR0	$\overline{\text{RESET}}$	1b	1b	1b	1b
SPIGCR1	ENABLE	1b	1b	1b	1b
SPIGCR1	LOOPBACK	0b	0b	0b	0b
SPIGCR1	CLKMOD	0b	0b	0b	0b
SPIGCR1	MASTER	0b	0b	0b	0b
SPIPC0	SOMIFUN	1b	1b	1b	1b
SPIPC0	SIMOFUN	1b	1b	1b	1b
SPIPC0	CLKFUN	1b	1b	1b	1b
SPIPC0	ENAFUN	0b	0b	1b	1b
SPIPC0	SCSFUN0	0b	1b	0b	1b

Table 5. Allowed SPI Register Settings in Slave Modes

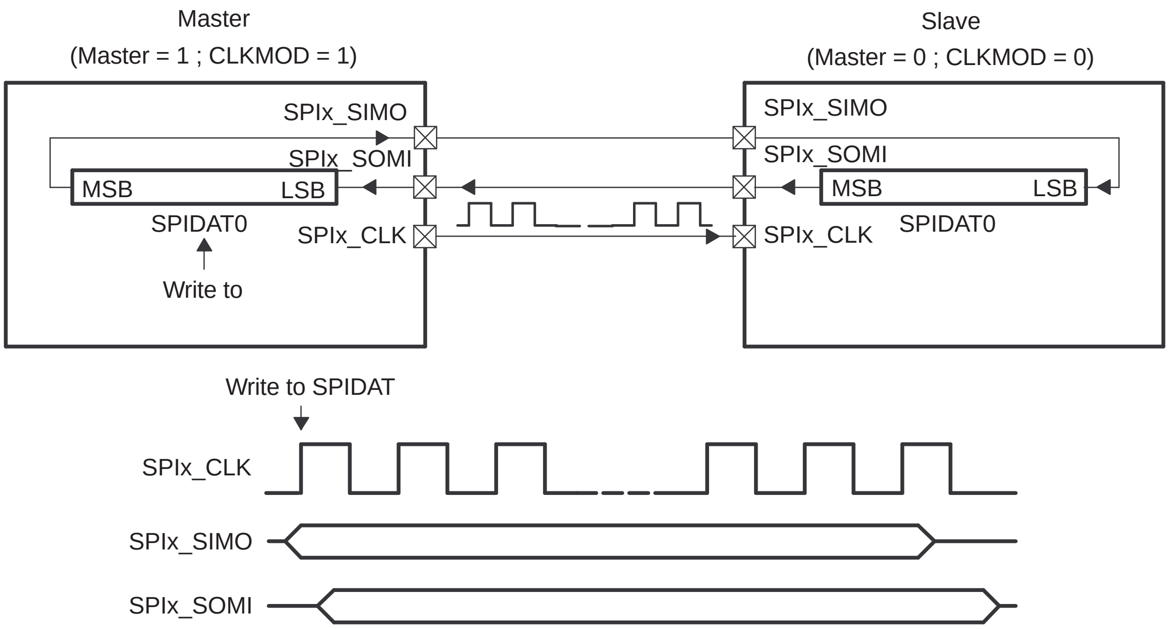
Register	Bit(s)	Slave 3-pin	Slave 4-pin Chip Select	Slave 4-pin Enable	Slave 5-pin
SPIINT0	ENABLEHIGHZ	{0,1}	{0,1}	{0,1}	{0,1}
SPIFMTx[1]	WDELAYx[1]	{0x00,0x01,...,0x3F}	{0x00,0x01,...,0x3F}	{0x00,0x01,...,0x3F}	{0x00,0x01,...,0x3F}
SPIFMTx[1]	SHIFTDIRx[1]	{0,1}	{0,1}	{0,1}	{0,1}
SPIFMTx[1]	WAITENAx[1]	{0,1}	{0,1}	{0,1}	{0,1}
SPIFMTx[1]	POLARITYx[1]	{0,1}	{0,1}	{0,1}	{0,1}
SPIFMTx[1]	PHASEx[1]	{0,1}	{0,1}	{0,1}	{0,1}
SPIFMTx[1]	PRESCALEx[7:0][1]	{0x07,0x08,...,0xFF}	{0x07,0x08,...,0xFF}	{0x07,0x08,...,0xFF}	{0x07,0x08,...,0xFF}
SPIFMTx[1]	CHARLENx[4:0][1]	{0x02,0x03,...,0x10}	{0x02,0x03,...,0x10}	{0x02,0x03,...,0x10}	{0x02,0x03,...,0x10}
SPIDELAY	C2TDELAY[4:0]	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}
SPIDELAY	T2CDELAY[4:0]	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}	{0x00,0x01,...,0x1F}
SPIDELAY	T2EDELAY[7:0]	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}
SPIDELAY	C2EDELAY[7:0]	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}	{0x00,0x01,...,0xFF}

[1] x = 0, 1, 2, 3. However, in slave mode, only SPIFMT0 is used. This means that when SPIDAT1 is written, SPIDAT1.DFSEL[1:0] shall be set to 00 to select SPIFMT0.

2.5 SPI Operation: 3-Pin Option

The 3-pin option for SPI operation uses only the clock (SPIx_CLK), and data (SPIx_SOMI and SPIx_SIMO) pins for bidirectional communication between master and slave devices. Figure 2 illustrates the basic 3-pin SPI option.

Figure 2. SPI 3-Pin Option



Note: If only unidirectional communication is required, the SPICLK pin and the two data pins (SPISOMI and SPISIMO) must all be configured as functional pins. A 2-pin unidirectional mode is not supported.

To select the 3-pin SPI option, the SPIx_CLK, SPIx_SOMI, and SPIx_SIMO pins should be configured as functional pins by writing 0x0E00 to SPIPC0. $\overline{\text{SPIx\_SCS}}$ and $\overline{\text{SPIx\_ENA}}$ can be used as general purpose I/O pins by configuring the SPIPC1 through SPIPC5 registers.

The SPI can operate in either master or slave mode. The SPIGCR1.CLKMOD and SPIGCR1.MASTER bits select between master and slave mode; both must be programmed to '1' to configure the SPI for master mode or '0' to configure the SPI for slave mode. The SPI bus master is the device which drives the SPIx_CLK signal and therefore initiates SPI bus transfers. In SPI master mode the SPIx_SOMI pin output buffer is tristated and SPIx_CLK and SPIx_SIMO pin output buffer is enabled. In SPI slave mode the SPIx_SIMO and SPIx_CLK pin output buffer is tristated and SPIx_SOMI pin output buffer is enabled.

In master mode with the 3-pin option, the DSP writes transmit data to the SPIDAT0[15:0] or the SPIDAT1[15:0] register. This initiates a transfer. A series of clocks pulses will be driven out on the SPIx_CLK pin to complete the transfer. Each clock pulse on the SPIx_CLK pin causes the simultaneous transfer (in both directions) of one bit by both the master and slave SPI devices. The DSP writes to the configuration bits in the SPIDAT1[31:16] alone (not writing to SPIDAT1[15:0]) do not result in a new transfer.

In slave mode with 3-pin option, the DSP writes to the SPIDAT0[15:0] or the SPIDAT1[15:0] makes the slave ready to transmit. The DSP writes to the configuration bits in SPIDAT1[31:16] alone (not writing to SPIDAT1[15:0]) do not make the slave ready to transmit.

Whether in master or slave mode, the word length must be configured in the SPIFMTx.CHARLEN[4:0] field before the first transfer is initiated. When the character length SPIFMTx.CHARLEN[4:0] is greater than 8 bits the two bytes of SPIDAT0[15:0] or SPIDAT1[15:0] must be written in a single write operation. The word length must match the number of clock pulses per transfer expected on the SPIx_CLK pin. Note that there are four different format registers: SPIFMT0, SPIFMT1, SPIFMT2, and SPIFMT3. The format register that applies is selected by SPIDAT1.DFSEL[1:0]. The default format register is SPIFMT0 since SPIDAT1.DFSEL[1:0] defaults to '00'.In slave mode only SPIFMT0 is supported.

Examples of clock polarity and phase options are given in Section 2.10.

Data should be written to the shift register (either SPIDAT0[15:0] or SPIDAT1[15:0]) right aligned. (If there is receive data from a previous transfer, this should be read out of the SPIBUF register first, since the write to the shift register will cause the next transfer to begin). In addition, if the SPI is operating in slave mode, then the write to the shift register must occur before the first clock pulse from the master occurs.

The SPI automatically selects the appropriate bits from the right aligned data in the shift register and shifts these out on the transmit data pin (see Section 2.9 *SPI Data Alignment in Transmit and Receive Buffers* for more details). At the same time, data is shifted in from the receive pin. In master mode the transmit data pin is SPIx_SIMO and the receive data pin is SPIx_SOMI. In slave mode the roles are reversed, SPIx_SIMO is the receive data pin and SPIx_SOMI is the transmit data pin.

When all of the bits have been transferred, the received data is copied to the SPIBUF register for the DSP or dMAX to read. Data is stored right-justified in SPIBUF and realigned automatically if needed by the SPI module.

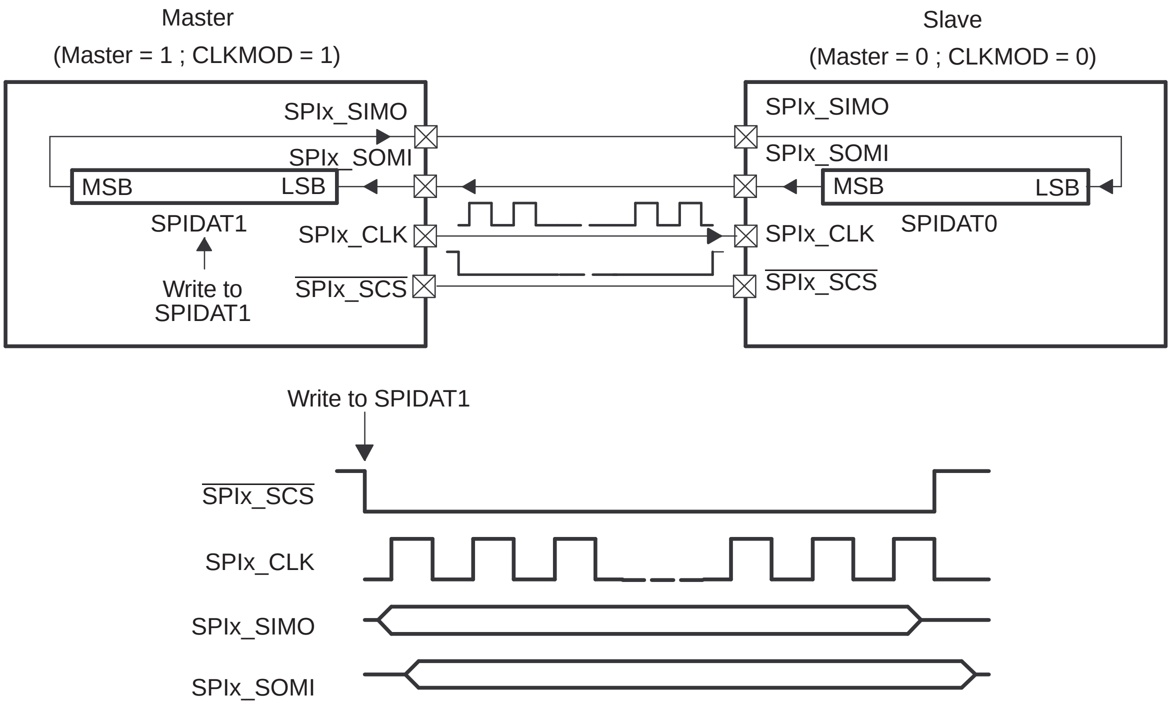
In addition, the following actions occur:

- The interrupt flag SPIFLG.RXINTFLAG bit is set to 1
- An interrupt is asserted if the SPIINT0.RXINTEN bit is set to 1
- A dMAX event is generated if the SPINT0.DMAREQEN is set to 1

2.6 SPI Operation: 4-Pin With Chip Select Option

The 4-pin with chip select option is a superset of the 3-pin option and uses the chip select (SPIx_SCS) pin in addition to the clock (SPIx_CLK) and data (SPIx_SOMI and SPIx_SIMO) pins. Figure 3 illustrates the SPI used with this option.

Figure 3. SPI 4-Pin With Chip Select Option



To select the 4-pin with chip select option, the SPIx_CLK, SPIx_SOMI, SPIx_SIMO, and SPIx_SCS pins should be configured as functional pins by writing 0x0E01 to SPIPC0. SPIx_ENA can be used as general purpose input/output pin by configuring the SPIPC1 through SPIPC5 registers.

In SPI master mode the SPIx_SOMI pin output buffer is tristated and SPIx_CLK, SPIx_SIMO and SPIx_SCS pin output buffer is enabled. In SPI slave mode the SPIx_CLK, SPIx_SIMO and SPIx_SCS pin output buffer is tristated, and SPIx_SOMI pin output buffer is enabled when SPIx_SCS is asserted and tristated when SPIx_SCS is deasserted.

In slave mode with the chip select option enabled, the SPI ignores all transactions on the bus unless SPIx_SCS is asserted by the bus master. It also 3-states its output pin when SPIx_SCS is deasserted by the master to avoid conflicting with the active slave device on the bus.

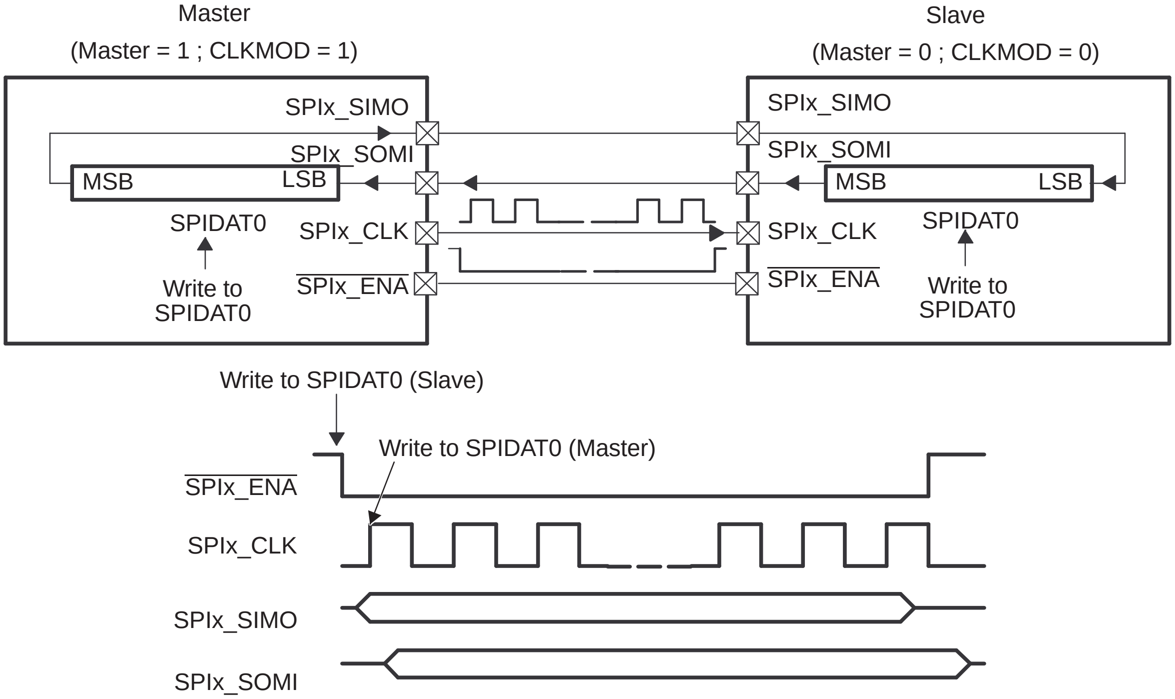
In master mode, the SPIx_SCS pin functions as an output, and toggles when a specific slave device is selected. However, this is most useful on devices which support multiple SPIx_SCS pins. The C672x DSP only supports a single SPIx_SCS and so the usefulness of this pin in master mode is limited. In practice, general purpose I/O pins will be needed to support multiple slave device chip selects.

However, one reason to use the SPIx_SCS pin as a functional pin for the SPI master is to take advantage of the timing parameters which can be set using the SPIDELAY register. This register allows delays to be added automatically so that the slave timing requirements between clock and chip select may be more easily met. See Section A.1.17 for additional details. Another reason would be to make use of the error detection built into the SPI (seeSection 2.13).

2.7 SPI Operation: 4-Pin With Enable Option

The 4-pin with enable option is also a superset of the 3-pin option and uses the enable (SPIx_ENA) pin in addition to the clock (SPIx_CLK) and data (SPIx_SOMI and SPIx_SIMO) pins. Figure 4 illustrates the SPI used with this option.

Figure 4. SPI 4-Pin With Enable Option



To select the 4-pin with enable option, the SPIx_CLK, SPIx_SOMI, SPIx_SIMO, and SPIx_ENA pins should be configured as functional pins by writing 0x0F00 to SPIPC0. SPIx_SCS can be used as general purpose input/output pin by configuring the SPIPC1 through SPIPC5 registers.

In SPI master mode the SPIx_SOMI and SPIx_ENA pin output buffer is tristated and SPIx_CLK and SPIx_SIMO pin output buffer is enabled. In SPI slave mode the SPIx_CLK and SPIx_SIMO pin output buffer is tristated, SPIx_SOMI pin output buffer is enabled. In SPI slave mode the SPIx_ENA pin output buffer enable depends upon the status of the transmit buffer and the configuration of the SPIINT0.ENABLEHIGHZ bit.

The handshake operation works this way:

- After a transfer completes, both the master and slave SPI modules need to be serviced.
- The slave SPI deasserts SPIx_ENA after the transfer, indicating it requires servicing and is not ready.
- The slave should begin servicing its SPI by reading receive data from SPIBUF first.
- Next, the slave device should write transmit data to SPIDAT0 or SPIDAT1. This causes the slave SPI to assert SPIx_ENA indicating it is ready for the next transmission.
- In parallel, the master device can service its SPI at any time. It does not need to insert a delay before writing to its SPIDAT0 or SPIDAT1 registers in order to avoid overrunning the slave device. Instead, the master SPI module will automatically delay the next transfer until the slave has asserted SPIx_ENA again to indicate it is ready for the transmission.

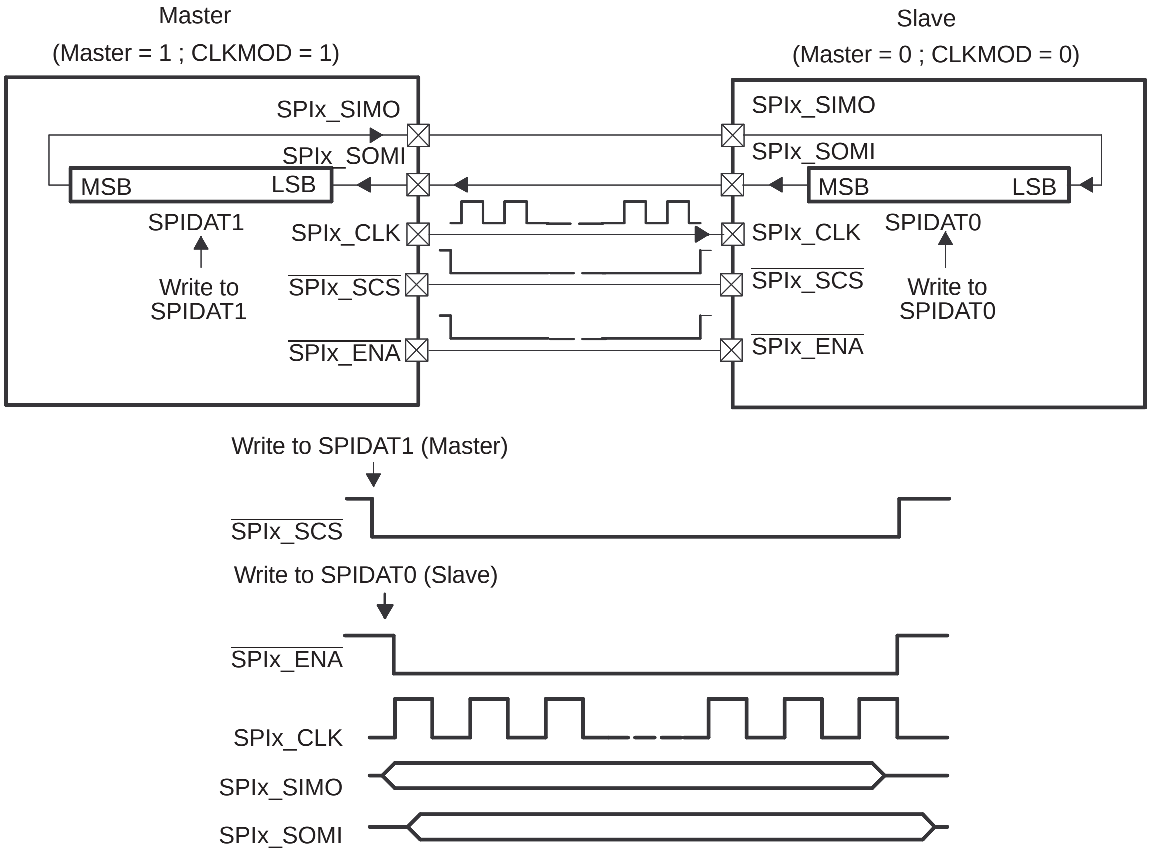
This handshake allows the two SPIs to communicate at the maximum rate possible. Without the handshake pin, the master must insert a delay between each transfer long enough to support the worst case response time of the slave servicing its SPI or risk an overrun condition. With the handshake, the throughput is determined by the average response time of the two devices servicing their SPI ports.

The SPIx_ENA pin can be driven in a push-pull or open drain mode, depending upon the setting of the SPIINT0.ENABLE HIGHZ bit.

2.8 SPI Operation: 5-Pin Option (Enable and Chip Select)

The 5-pin mode is a superset of both 4-pin modes. To use the 5-pin mode, both the SPIx_ENA pin and SPIx_SCS pin must be configured as functional pins, in addition to the SPIx_CLK, SPIx_SIMO, and SPIx_SOMI pins by writing 0x0F01 to SPIPC0. Figure 5 illustrates the SPI used with the 5-pin option.

Figure 5. SPI 5-Pin Option With Chip Select and Enable



In SPI master mode the SPIx_SOMI and SPIx_ENA pin output buffer is tristated and SPIx_CLK,SPIx_SIMO and SPIx_SCS pin output buffer is enabled. In SPI slave mode the SPIx_CLK, SPIx_SIMO and SPIx_SCS pin output buffer is tristated, SPIx_SOMI pin output buffer is enabled and disabled asynchronously by the SPIx_SCS input and SPIx_ENA pin output buffer enable depends upon the status of the transmit buffer and the state of the SPIx_SCS input. In SPI slave mode the assertion of the SPIx_ENA pin by the slave is delayed until the master asserts SPIx_SCS thereby allowing multiple SPI slaves on a single SPI bus, each slave with its own enable pin.

In 5-pin mode the SPI master must see the SPIx_ENA pin deasserted practically immediately after it asserts SPIx_SCS. The only way to meet this requirement is to have the SPIx_ENA deasserted before the SPIx_SCS pin is asserted. To meet this requirement the SPIINT0.ENABLEHIGHZ must be set to '1' and an external pullup resistor of appropriate strength should be used.

CAUTION

In 5-pin slave mode if SPIINT0.ENABLEHIGHZ is ‘0’ then the slave is expected to deassert SPIx_ENA when the slave is deasserted. However if the slave writes to the SPIDAT and asserts SPIx_ENA before the SPIx_SCS is deasserted then SPIx_ENA remains asserted even after the SPIx_SCS is deasserted. If the slave writes to SPIDAT after SPIx_SCS is deasserted then SPIx_ENA remains deasserted until the master asserts SPIx_SCS again.

2.9 SPI Data Alignment in Transmit and Receive Buffers

The SPI module supports several data formatting options:

- Shift direction: MSB or LSB first (SPIFMTx.SHIFTDIR)
- Character Length: 2 to 16 bits (SPIFMTx.CHARLEN

For example, you might configure the SPI to transmit MSB first, with a character length of 7 bits, and wish to transmit the binary sequence '100 0000b'. Since the shift register size is 16 bits and the MSB first option is selected, it is natural to think that you might need to shift the binary sequence to the MSB position using the CPU, before writing data to the SPI module. For example, writing "1000 0000 0000 0000b" to SPIDAT0 or SPIDAT1. However, the C672x SPI module automatically performs this shift based upon the settings in SPIFMTx.SHIFTDIR and SPIFMTx.CHARLEN. Therefore, the SPIDAT0 or SPIDAT1 register should be written with "0000 0000 0100 0000" (no shift required).

The SPI also realigns incoming receive data automatically. For example, if the SPI is configured for LSB first bit order (so that receive data is shifted into MSB of the shift register) then if the same seven bits are shifted into the SPI in the order of "100 0000b" where the first bit to shift in is the '1b', then the shift register would normally contain "0000 001x xxxx xxxxb" after the transfer, where 'x' means bits that are not part of the current transfer.

However, before copying this data to the SPIBUF register, the SPI automatically realigns the data so that it is right-justified. In addition, it replaces the 'x' bits with '0'. So in the receive example, the SPIBUF register will read "0000 0000 0000 0001b."

To summarize: data is always written to SPIDAT0/SPIDAT1 and read from SPIBUF by the DSP or dMAX right aligned. The term right aligned means that the LSB of the data word is always aligned to bit zero of the buffer register. This is true regardless of character length or shift direction. The SPI automatically corrects for the character length and shift direction settings.

2.10 SPI Clocking Modes

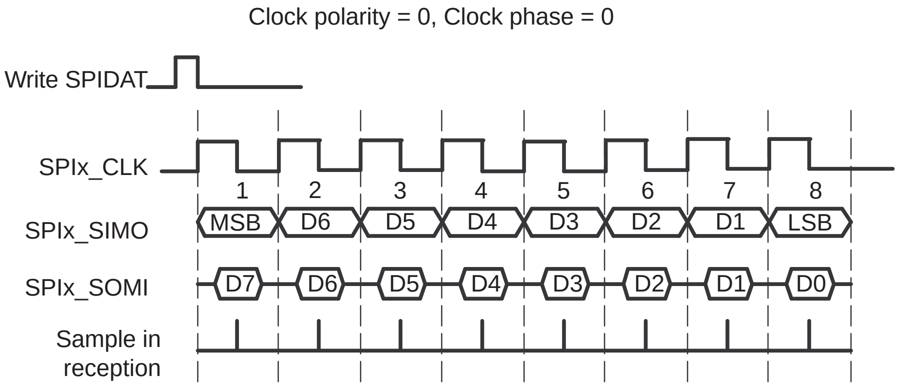
There are four clock modes in which SPIx_CLK may operate, depending on the choice of the phase (delay/no delay) and the polarity (rising edge/falling edge) of the clock. When operating with PHASE active, the SPI makes the first bit of data available after the SPIDAT0 register is written and before the first edge of SPIx_CLK. The data input and output edges depend on the values of both POLARITY and PHASE as shown in Table 6.

Table 6. Clocking Modes

Polarity	Phase	Action
0	0	Data is output on the rising edge of SPIx_CLK. Input data is latched on the falling edge.
0	1	Data is output one half-cycle before the first rising edge of SPIx_CLK and on subsequent falling edges. Input data is latched on the rising edge of SPIx_CLK.
1	0	Data is output on the falling edge of SPIx_CLK. Input data is latched on the rising edge.
1	1	Data is output one half-cycle before the first falling edge of SPIx_CLK and on subsequent rising edges. Input data is latched on the falling edge of SPIx_CLK.

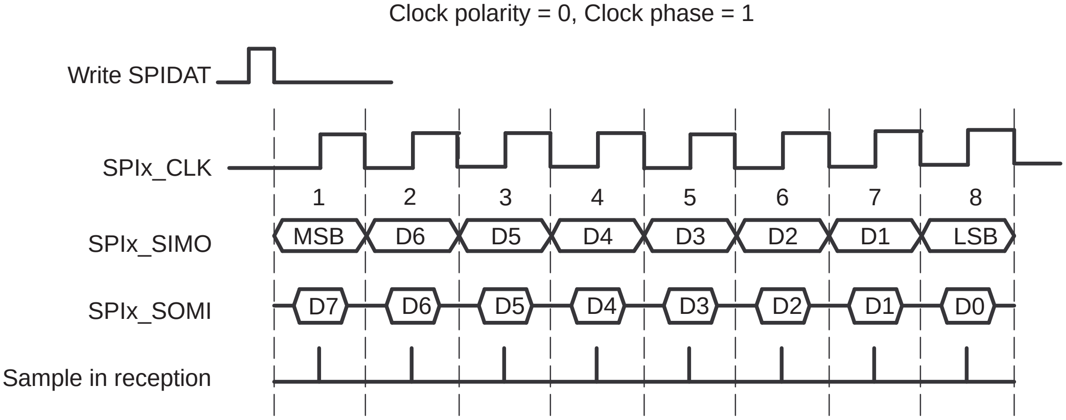
Figure 6 through Figure 9 illustrate the four possible signals of SPIx_CLK corresponding to each mode. Having four signal options allows the SPI to interface with different types of serial devices. Also shown are the SPIx_CLK control bit polarity and phase values corresponding to each signal.

Figure 6. Clock Mode With POLARITY = 0 and PHASE = 0



Clock phase = 0 (SPIx_CLK without delay)
- Data is output on the rising edge of SPIx_CLK
- Input data is latched on the falling edge of SPIx_CLK
- A write to the SPIx_DAT register starts SPIx_CLK

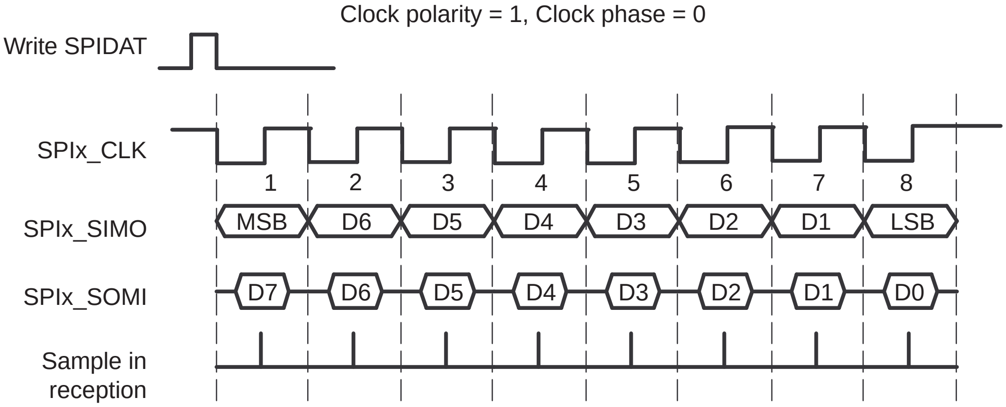
Figure 7. Clock Mode With POLARITY = 0 and PHASE = 1



Clock phase = 1 (SPIx_CLK with delay)

- Data is output one-half cycle before the first rising of SPIx_CLK and on subsequent falling edges of SPIx_CLK
- Input data is latched on the rising edge of SPIx_CLK

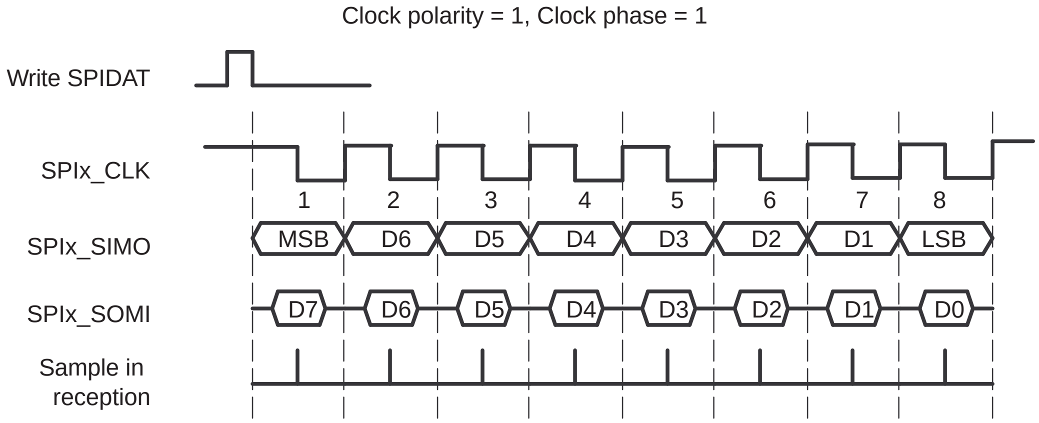
Figure 8. Clock Mode With POLARITY = 1 and PHASE = 0



Clock phase = 0 (SPIx_CLK without delay)

- Data is output on the falling edge of SPIx_CLK
- Input data is latched on the rising edge of SPIx_CLK
- A write to the SPIDAT register starts SPIx_CLK

Figure 9. Clock Mode WIth POLARITY = 1 and PHASE = 1



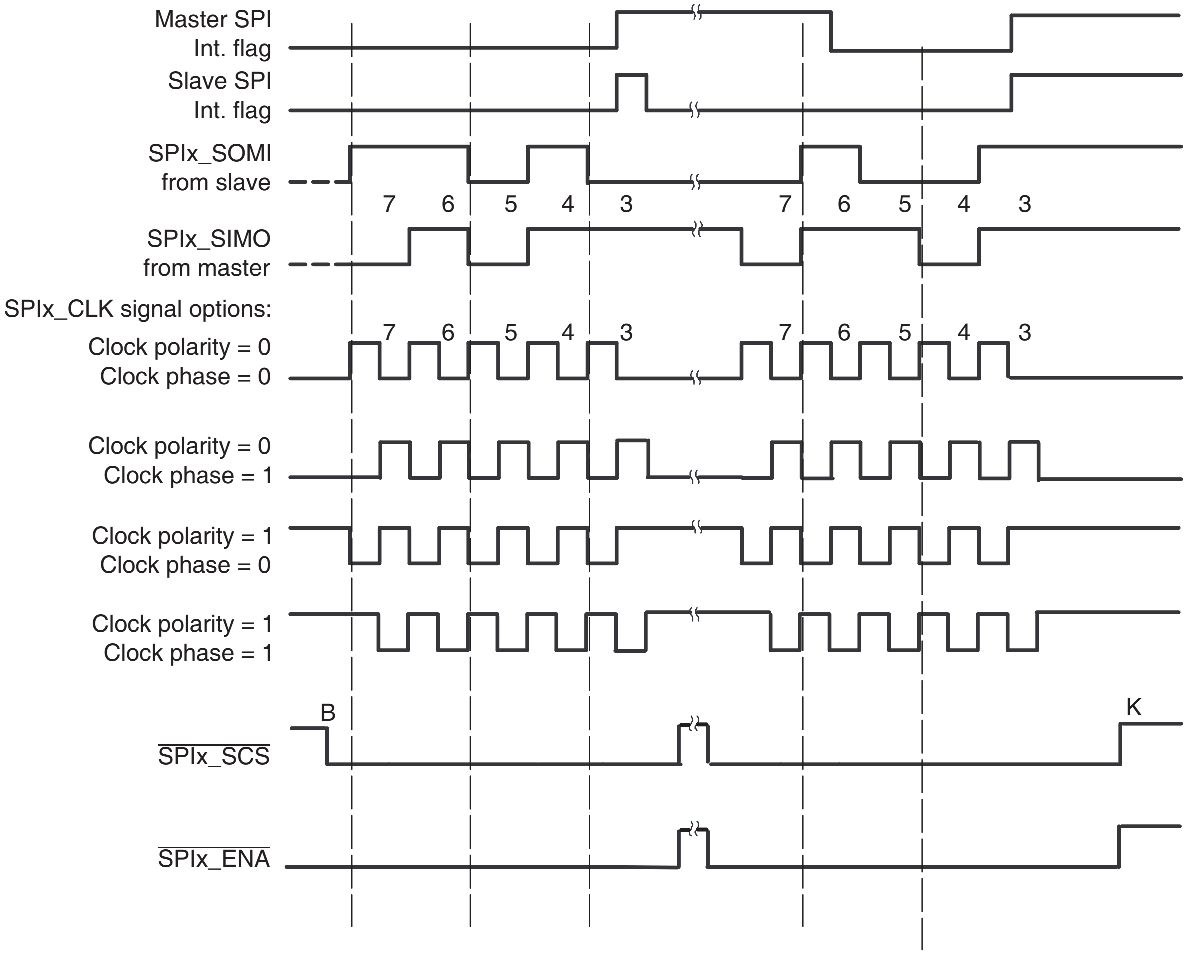
Clock phase 1 (SPIx_CLK with delay)

- Data is output one-half cycle before the first falling edge of SPIx_CLK and on the subsequent rising edges of SPIx_CLK
- Input data is latched on the falling edge of SPIx_CLK

2.11 SPI Data Transfer Example

Figure 10 illustrates an SPI data transfer between two devices using a character length of five bits.

Figure 10. Five Bits per Character (5-Pin Option)



2.12 SPI Master Mode Timing Options

The SPI in master mode supports several options to modify the timing of its generation of the chip select signal (SPIx_SCS). This allows the SPI to support the timing requirements of various slave devices without adding additional overhead to the DSP by generating the appropriate delays automatically.

2.12.1 Chip Select Setup Time

The master can be configured to provide a (slow) slave device a certain chip select setup time to the first edge on SPIx_CLK. This delay is controlled by the SPIDELAY.C2TDELAY[28:24] bit field and can be configured between for 2 and 33 SYSCLK2 cycles. The C2TDELAY is applicable only in 4-pin with chip select and 5-pin SPI master modes. The C2TDELAY begins when the SPI master asserts SPIx_SCS. The duration of C2T delay period does not always complete, sometimes it is skipped or terminated early. In 5-pin SPI master mode the C2T delay period terminates immediately when the SPIx_ENA input is sampled (using SYSCLK2) as asserted. However assuming the C2T delay period completes its duration is specified by:

Max duration of C2TDELAY period = SPIDELAY.C2TDELAY+2 (SYSCLK2 cycles)

The previous value of SPIDAT1.CSHOLD must be ‘0’ for C2T delay to be enabled. In 5-pin mode the SPIx_ENA must also be deasserted at the beginning of the C2TDELAY period for C2T delay to be enabled. See Section A.1.17 for more details. See Section A.1.17

Note: If the SPIDAT1.CSHOLD bit is set within the control field, the current hold time and the following setup time will not be applied in between transaction.

2.12.2 Chip Select Hold Time

The master can also be configured to provide a (slow) slave device a certain chip select hold time after the last edge on SPIx_CLK. This delay is controlled by the SPIDELAY.T2CDELAY[20:16] bit field and can be configured between for 1 and 32 SYSCLK2 cycles. The T2CDELAY is applicable only in 4-pin with chip select and 5-pin SPI master modes. The T2CDELAY begins after the data shifting period ends. The duration of T2C delay period does not always complete, sometimes it is skipped or terminated early. In 5-pin SPI master mode the T2C delay period terminates immediately when the SPIx_ENA input is sampled (using SYSCLK2) as deasserted. However assuming the T2C delay period completes its duration is specified by:

Max duration of T2CDELAY period = SPIDELAY.T2CDELAY+1 (SYSCLK2 cycles)

If SPIFMTx.PHASE is '0' then T2CDELAY period lasts for an additional SPICLK time over and above that specified by the above equation.

The current value of SPIDAT1.CSHOLD must be ‘0’ for T2C delay to be enabled. In 5-pin mode the SPIx_ENA must also be asserted at the beginning of the T2CDELAY period for T2C delay to be enabled. See Section A.1.17

Note: If the SPIDAT1.CSHOLD bit is set within the control field, the current hold time and the following setup time will not be applied in between transaction.

2.12.3 Automatic Delay Between Transfers

The SPI master can automatically insert a delay of between 2 and 65 SYSCLK2 cycles between transmissions. This delay is configured in the SPIFMTx.WDELAY[29:24] bit field and is enabled by setting SPIDAT1.WDEL. The WDELAY period begins when the T2EDELAY period terminates (if T2E delay period is enabled) or when the T2CDELAY period terminates (if T2E delay period was disabled and T2C delay period was enabled) or when the master deasserts SPIx_SCS (if T2E and T2C delay period are disabled). If a transfer is initiated by writing a 32-bit value to SPIDAT1 then the new values of SPIDAT1.WDEL and SPIFMTx.WDELAY are used, otherwise the old values of SPIDAT1.WDEL and SPIFMTx.WDELAY are used. The WDELAY delay period is specified by:

Max duration of WDELAY period = SPIDELAY.WDELAY+2 (SYSCLK2 cycles)

2.12.4 SPI Chip Select Hold Option

There are slave devices available that require the chip select signal to be held continuously active during several consecutive data word transfers. Other slave devices require the chip select signal to be deactivated between consecutive data word transfers.

The SPI can support both types of slave devices. The SPIDAT1.CSHOLD bit selects between the two options.

If the chip select hold option is enabled, the chip select will not toggle between two consecutive accesses. Therefore the SPIDELAY.T2CDELAY of the first transfer and the SPIDELAY.C2TDELAY of the second transfer will not be applied.

However, the wait delay could be still applied between the two transactions, if the bit SPIDAT1.WDEL is set.

The current and previous values of CSHOLD are retained. Though the current value of CSHOLD is initialized to '0' when SPIGCR0.$\overline{\text{RESET}}$ is set to '0', the previous value of CSHOLD is not initialized. The previous value of CSHOLD must be explicitly initialized by writing CSHOLD field twice.

CAUTION

When data is written to the SPIDAT1 register with the CSHOLD bit set to '1', the master keeps the SPIx_SCS asserted at the end of the transfer. When data is written to the SPIDAT1 register with CSHOLD set to '0' the master de asserts the SPIx_SCS pin at the end of the transfer. However it is found that when SPIx_SCS is asserted and data is written to the SPIDAT1 register with the CSHOLD set to '0' the SPIx_SCS gets momentarily deasserted and asserted back resulting in a glitch. To avoid such a glitch all transfers during which SPIx_SCS should be active the user should write to SPIDAT1 register with CSHOLD set to '1'. This should be followed by a write to only the CSHOLD field with a value '0'. Alternatively the user can write to the CSHOLD fields only before and after the set of transfers to toggle the SPIx_SCS and write only to the SPIDAT1[15:0] during the transfers.

2.13 Robustness Features

The C672x SPI includes many features to make the SPI communication link robust. A loopback test mode can be used to facilitate a power on self test routine. Additionally, the SPI master continually monitors the bus for faults on its data line. The handshaking between master and slave can be monitored as well, and appropriate actions can be taken (interrupt, timeout) when the handshake breaks down. The following sections describe these robustness features in more detail.

2.13.1 SPI Internal Loop-Back Test Mode (Master Only)

To select the loopback mode the SPIx_CLK, SPIx_SOMI, SPIx_SIMO pins should be configured as functional pins by writing 0x0E00 to SPIPC0. SPIx_ENA and SPIx_SCS can be used as general purpose input/output pins by configuring the SPIPC1 through SPIPC5 registers. The internal loop-back self-test mode can be utilized to test the SPI transmit path and receive path including the transmit and receive buffers. In this mode the transmit signal is internally fed back to the receiver and the SPIx_SIMO, SPIx_SOMI and SPIx_CLK pins are tristated. This mode allows the DSP to write into the transmit buffer, and check that the receive buffer contains the correct transmit data. If an error occurs the corresponding error is set within the status field.

CAUTION

This mode cannot be changed during a transmission.

2.13.2 SPI Transmission Continuous Self-Test

During a data transfer, the SPI inputs the value from its data output pin on the appropriate SPIx_CLK edge. This value is compared against the expected value and any difference indicates a fault on the SPI bus. If a fault is detected, then the SPIBUF.BITERR and SPIFLG.BITERRFLG bits are set and an error interrupt is generated if enabled. The SPI continuous self test mode is not available in SPI loopback mode.

2.13.3 SPI Detection of Slave Desynchronization

In the 4-pin with enable and 5-pin modes, the SPI master can monitor the slave SPIx_ENA activity to detect a desynchronization event.

Some conditions which may cause a desynchronization event are:

- Master or slave device being reset during a transmission
- Asserting a software reset of the SPI module during transmission
- Having an incorrect SPI pin configuration, causing the SPIx_ENA pin to behave incorrectly
- Signal integrity problem causing additional clocks to be recognized by the slave

The master can detect two desynchronization error conditions on the SPIx_ENA pin:

1. Slave deasserts SPIx_ENA after a transmission has begun, but before it completes.
2. Slave fails to deassert SPIx_ENA within a certain time period after the completion of the last bit of the transmission.

The first error condition is straightforward to detect. To detect the second error condition, the SPI module includes an eight bit counter with a timeout count that can be configured through the SPIDELAY.T2EDELAY bit field.

When a desynchronization event is detected, the master SPIBUF.DESYNC and SPIFLG.DESYNCFLG flags are set and a desynchronization error interrupt is asserted if enabled.

Note: Remember that even though the desynchronization is detected by the master device, the problem causing the desynchronization event can be on either the master or the slave device.

The T2EDELAY period begins once the T2CDELAY period terminates or after the data shifting period in case the T2CDELAY is disabled. It defines the maximum time for the slave to deassert the SPIx_ENA signal. If the slave device does not deassert SPIx_ENA signal before the T2EDELAY timeout value expires, the SPIFLG.DESYNC flag is set and a desynchronization interrupt is asserted if enabled. The T2E delay period does not always complete, sometimes it is skipped or terminated early. The T2E delay period terminates immediately after the SPIx_ENA input is sampled (using SYSCLK2 at intervals of SPIFMTx.PRESCALEx+2) as deasserted. However assuming the T2E period completes its duration is specified by:

Max duration of T2EDELAY period =

$$\left\{\begin{array}{ll} (\text{SPIDELAY.T2EDELAY})\ .\ (\text{SPIFMTx.PRESCALE} + 2) & \{\text{If SPIFMTx.PRESCALEx} < 127\} \\ (\text{SPIDELAY.T2EDELAY}) & \{\text{if SPIFMTx.PRESCALEx} >= 128\} \end{array}\right.$$

SYSCLK2 cycles

The T2EDELAY period is enabled only when the SPIx_ENA is asserted at the beginning of the T2E delay period, the SPIDELAY.T2EDELAY field has a non-zero value, and SPIFMTx.WAITENA field is set to '1'.

Note: In SPI master mode during the T2EDELAY period, the slave must deassert SPIx_ENA with a minimum duration of (SYSCLK2*(SPIFMTx.PRESCALEx+1)) of the master. This constraint is applicable only in SPI master mode during T2EDELAY period. The SPI does not guarantee this timing in slave mode.

2.13.4 $\overline{\text{SPIx\_ENA}}$ Signal Timeout

In 5-pin mode, in addition to the desynchronizations event described in Section 2.13.3, the master can also detect whether the slave fails to respond to the $\overline{\text{SPIx\_SCS}}$ signal by asserting $\overline{\text{SPIx\_ENA}}$ in a timely manner.

This condition could be the result of a serious error, or it could simply be the result of the slave device taking too long to service its SPI.

To detect this condition, the SPIDELAY.C2EDELAY[7:0] field is used. C2EDELAY period begins once the C2TDELAY period terminates or when the master asserts SPIx_SCS (if C2TDELAY is disabled). It defines the maximum time for the addressed slave to respond by activating the SPIx_ENA signal. If the slave does not respond with the SPIx_ENA signal before the timeout value expires then the SPIFLG.TIMEOUTFLG and SPIBUF.TIMEOUT flags are set, a interrupt is asserted if enabled, and the current transfer is terminated. The C2E delay period does not always complete, sometimes it is skipped or terminated early. The C2E delay period terminates immediately after the SPIx_ENA input is sampled (using SYSCLK2 at intervals of SPIFMTx.PRESCALEx+2) as asserted. However assuming the C2E period completes its duration is specified by:

Max duration of C2EDELAY period =

$$\left\{\begin{array}{ll} (\text{SPIDELAY.C2EDELAY})\ .\ (\text{SPIFMTx.PRESCALE} + 2) & \{\text{If SPIFMTx.PRESCALEx} < 127\} \\ (\text{SPIDELAY.C2EDELAY}) & \{\text{if SPIFMTx.PRESCALEx} >= 128\} \end{array}\right.$$

SYSCLK2 cycles

The C2EDELAY period is enabled only when the SPIx_ENA is deasserted at the beginning of the C2E delay period and SPIFMTx.WAITENA is set to '1'. If SPIFMTx.WAITENA is set to '1' and C2EDELAY is '0' then the master waits indefinitely for the slave to assert SPIx_ENA.

3 General Purpose I/O

Each of the SPI pins may be programmed via the SPI Pin Control Registers (SPIPC0, SPIPC1, SPIPC2, SPIPC3, SPIPC4, SPIPC5) to be a general-purpose I/O pin.

Whenever one of the the SPI module pins is not used for chosen SPI bus protocol, the unused pins should be programmed to be either general-input or general-output pins by setting the corresponding bit in SPIPC0 to '0'.

For example, in 3-pin mode, SPIx_SOMI, SPIx_SIMO, and SPIx_CLK must be configured as SPI pins, while the SPIx_SCS and SPIx_ENA pins should be configured as GPIO.

The direction of the pin is controlled in the SPIPC1 register. If configured as a general purpose output, then the SPIPC3 register controls the output value.

There is also a write '1' to set (SPIPC4) and write '1' to clear (SPIPC5) register for the data out value. These registers allow different tasks running on the DSP to manipulate the SPI I/O pins without read-modify-write hazards.

The SPIPC2 register reflects the current value on the pin when the particular pin is configured as functional or general purpose input pin. When the pin is configured as functional or general purpose output pin SPIPC2 register indicates the value that is attempted to be driven on the pin..

4 Low-Power Mode (Not Supported)

The SPI module does not support low power mode on C672x DSPs.

Do not program the SPIGCR1.POWERDOWN bit to '1'. The behavior of the SPI is undefined if this bit is set.

5 Emulation Access and Emulation Suspend (Not Supported)

The C672x DSP does not support emulation accesses or emulation suspend for its peripherals. This means that the SPI will continue to operate while the DSP has been halted through the use of a JTAG emulator.

In addition, any status registers that are cleared after reading will be affected if viewed in a memory or watch window of the debugger; since the emulator will read these registers to update the value displayed in the window.

CAUTION

Viewing or otherwise reading the following SPI registers: SPIBUF, SPIFLG, TGINTVECT0, TGINTVECT1 through the JTAG debugger will cause their contents to change, possibly invalidating the results of the debug session. Be sure to set up the debugger to avoid reading these registers.

6 Interrupts

The SPI module generates interrupts and provides separate interrupt vectors on two levels; however these are combined into a single interrupt request along with interrupts from the I2C modules and the other SPI module to create INT14 on the C672x DSP.

It is STRONGLY RECOMMENDED to use CPU INT14 only for SPI error interrupts, and to use the SPI module DMA interface to generate data transfer requests. Interrupts are enabled using the SPIINT0 register (Section A.1.4).

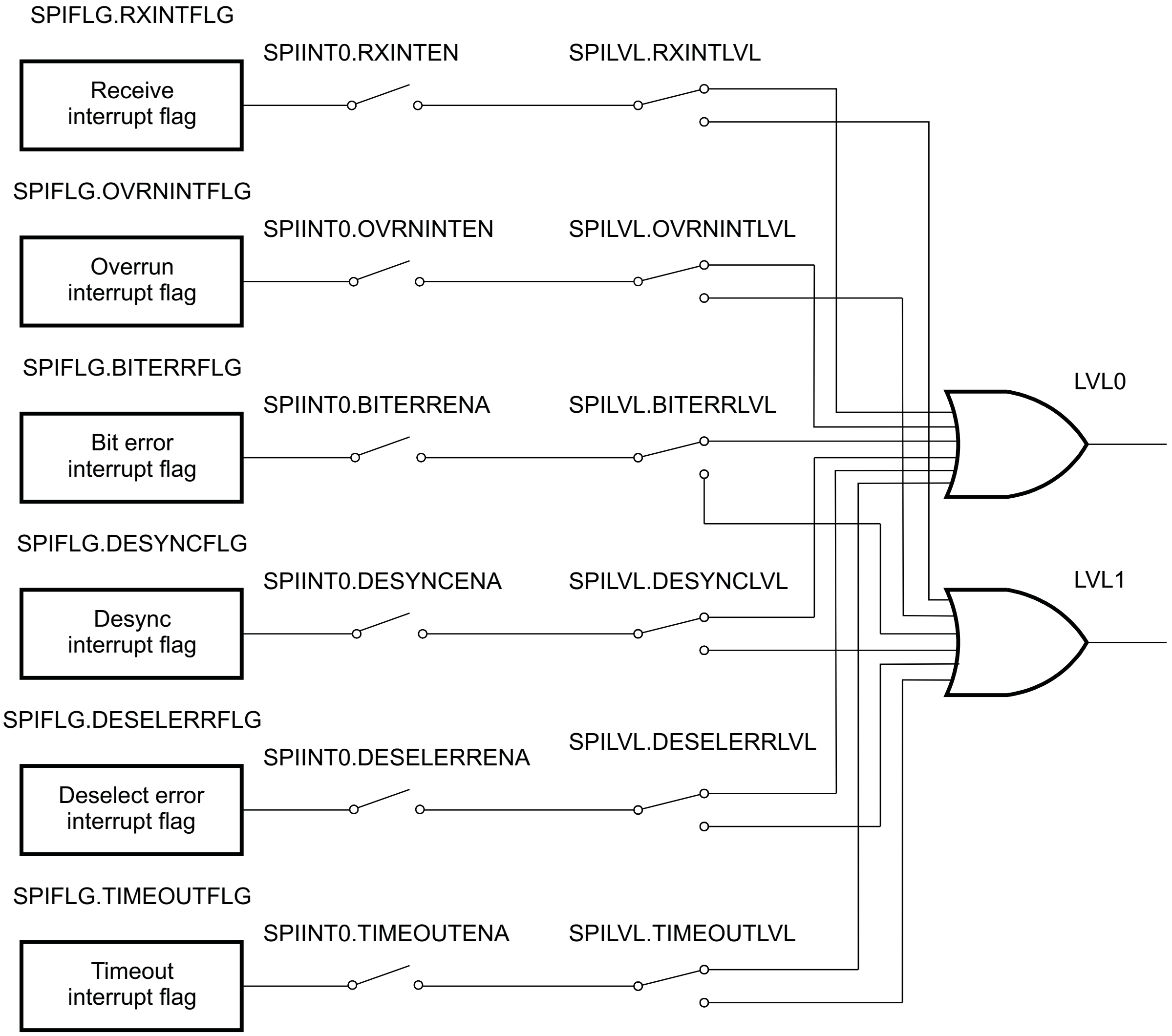
Also, since the two interrupt levels of the SPI are combined to generate a single interrupt request to the CPU, it is also recommended to make use of only interrupt level 0 and leave interrupt level 1 unused. Interrupt level assignments are configurable in the SPILVL register (Section A.1.5).

When an interrupt condition occurs, the bit indicating this condition is set in the SPIFLG register (Section A.1.6). If this condition is configured to generate an interrupt (if the SPIINT0 register enable bit for this condition is set) then the interrupt is routed to one of the two levels. The level for each interrupt is controlled by the SPILVL register.

For each level, the SPI combines with a logical 'OR' all of the SPIFLG bits that are both enabled and configured for this level. This is illustrated in Figure 11.

Then at the chip level, two interrupt requests from SPI0, the two interrupt requests from SPI1, the I2C0, and the I2C1 interrupt request lines are combined with a logical 'OR' to generate CPU INT14.

Figure 11. SPIFLG Interrupt Structure



CAUTION

The CPU interrupt request line INT14 is rising edge sensitive. This means that if any one of the interrupt flags enabled to generate INT14 remains set, the INT14 line will remain at a logic '1' and all other interrupts will be blocked. Therefore any interrupt enabled for CPU INT14 must be cleared promptly or it will block other interrupts on the same line.

CAUTION

In addition, before exiting the INT14 ISR, the DSP must make sure that the (enabled) interrupt flags in all peripherals assigned to use INT14 have been cleared. Otherwise, further interrupts may be blocked by the peripheral with the unserviced interrupt request.

CAUTION

In addition to the above caution, it should be noted that after a CPU INT14 is generated, all of the modules which have been configured to generate INT14 should be checked for interrupt status. It is quite possible that more than one module has generated an interrupt request simultaneously, but that only one CPU interrupt results.

7 DMA Interface

The SPI supports a DMA interface for transmit and receive data transfers.

It is STRONGLY RECOMMENDED to use the SPI DMA interface for data transfers, and limit the use of interrupts to error detection. This is especially true since the same interrupt request is shared at the device level between the two SPI modules and the two I2C modules (See Section 6)

To use the SPI DMA interface, the SPIINT0.DMAREQEN bit should be set to '1' and SPIINT0.RXINTEN should be set to '0'. The SPIINT0.DMAREQEN must be set to ‘1’ only after the SPIGCR1.ENABLE is set. If the SPIINT0.DMAREQEN is set before SPIGCR1.ENABLE is enabled then the first DMA request may be dropped.

The SPI will generate an active high request pulse when it is time to service the transmit and receive buffers.

On the C672x device, this request is routed to the dMAX unit:

- SPI0 request is routed to dMAX Event Input 13
- SPI1 request is routed to dMAX Event Input 14

Note: Remember, only a single DMA request is generated, and this should trigger a read of SPIBUF followed by a write to SPIDAT0 or SPIDAT1 through dMAX configuration.

Appendix A

A.1 SPI Registers

A.1.1 Introduction

Table A-1 lists the memory-mapped registers for the serial peripheral interface (SPI). See the device-specific data manual for the memory address of these registers.

Table A-1. SPI Registers

SPI0 Address	SPI1 Address	Mnemonic	Name	Description
0x4700 0000	0x4800 0000	SPIGCR0	Global Control Register 0	Contains the Software Reset bit for the Module
0x4700 0004	0x4800 0004	SPIGCR1	Global Control Register 1	Controls Basic configurations of the module
0x4700 0008	0x4800 0008	SPIINT0	Interrupt Register	Enable bits for interrupts, error, DMA and other functionality
0x4700 000C	0x4800 000C	SPILVL	Interrupt Level Register	SPI Interrupt Levels are set in this register
0x4700 0010	0x4800 0010	SPIFLG	Interrupt Flag Register	Shows the status of several events during the operation
0x4700 0014	0x4800 0014	SPIPC0	Pin Control Register 0	Determines if pins will operate as general I/O or SPI functional pin
0x4700 0018	0x4800 0018	SPIPC1	Pin Control Register 1	Controls the direction of data on the I/O pins
0x4700 001C	0x4800 001C	SPIPC2	Pin Control Register 2	Reflects the values on the I/O pins
0x4700 0020	0x4800 0020	SPIPC3	Pin Control Register 3	Controls the values sent to the I/O pins
0x4700 0024	0x4800 0024	SPIPC4	Pin Control Register 4	Sets data values in the SPIPC3 register
0x4700 0028	0x4800 0028	SPIPC5	Pin Control Register 5	Clears values in the SPIPC3 register
0x4700 002C	0x4800 002C	reserved	Reserved	Reserved - Do not write to this register
0x4700 0030	0x4800 0030	reserved	Reserved	Reserved - Do not write to this register
0x4700 0034	0x4800 0034	reserved	Reserved	Reserved - Do not write to this register
0x4700 0038	0x4800 0038	SPIDAT0	Shift Register 0	Main shift register
0x4700 003C	0x4800 003C	SPIDAT1	Shift Register 1	Shift register used in automatic slave chip select mode only
0x4700 0040	0x4800 0040	SPIBUF	Buffer Register	Holds received word
0x4700 0044	0x4800 0044	SPIEMU	Emulation Register	Mirror of SPIBUF. Read does not clear flags
0x4700 0048	0x4800 0048	SPIDELAY	Delay Register	Sets $\overline{\text{SPIx\_SCS}}$ mode, $\overline{\text{SPIx\_SCS}}$ pre-/post-transfer delay time and $\overline{\text{SPIx\_ENA}}$ timeout
0x4700 004C	0x4800 004C	SPIDEF	Default chip select register	In $\overline{\text{SPIx\_SCS}}$ decoded mode only: sets high low/active $\overline{\text{SPIx\_SCS}}$ signals
0x4700 0050	0x4800 0050	SPIFMT0	Format 0 register	Configuration of data word format 0
0x4700 0054	0x4800 0054	SPIFMT1	Format 1 register	Configuration of data word format 1
0x4700 0058	0x4800 0058	SPIFMT2	Format 2 register	Configuration of data word format 2
0x4700 005C	0x4800 005C	SPIFMT3	Format 3 register	Configuration of data word format 3
0x4700 0060	0x4800 0060	TGINTVECT0	TG interrupt vector	Transfer group interrupt vector for line SPI INT0
0x4700 0064	0x4800 0064	TGINTVECT1	TG interrupt vector	Transfer group interrupt vector for line SPI INT1

A.1.2 SPI Global Control Register 0 (SPIGCR0)

The SPI global control register 0 (SPIGCR0) is shown in Figure A-1 and described in Table A-2.

Figure A-1. SPI Global Control Register 0 (SPIGCR0)

31 … 8
Reserved
R-0

7 … 1	0
Reserved	$\overline{\text{RESET}}$
R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0000 **SPI1 Address: 0x4800_0000**

Table A-2. SPI Global Control Register 0 (SPIGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved		Reads return 0 and writes have no effect.
0	$\overline{\text{RESET}}$		Software reset. This bit should be written to a '1' before any other operations are done on the SPI module.
		0	SPI is held in a software reset.
		1	SPI is released from software reset.

A.1.3 SPI Global Control Register 1 (SPIGCR1)

The SPI global control register 1 (SPIGCR1) is shown in Figure A-2 and described in Table A-3.

Figure A-2. SPI Global Control Register 1 (SPIGCR1)

31	25	24
Reserved		ENABLE
R-0		R/W-0

23	17	16
Reserved		LOOPBACK
R-0		R/W-0

15	9	8
Reserved		POWERDOWN
R-0		R/W-0

7	2	1	0
Reserved		CLKMOD	MASTER
R-0		R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0004 **SPI1 Address: 0x4800_0004**

Table A-3. SPI Global Control Register 1 (SPIGCR1) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved		Reads return 0 and writes have no effect.
24	ENABLE		SPI Enable. By default, the SPI is disabled after the software reset bit (SPIGCR0.$\overline{\text{RESET}}$) is released . The other SPI configuration registers, except SPIINT0.DMAREQEN should be setup before writing a '1' to this bit. This will prevent the SPI from responding to bus operations erroneously while it is in the process of being configured. The SPIINT0.DMAREQEN should be enabled after setting ENABLE. If SPIINT0.DMAREQEN is enabled before setting ENABLE then the first DMA request which occurs before the SPI is ready for data transfer may get dropped In particular, the SPIFLG.RXINTFLAG and SPIFLG.OVRNINTFLG bits are forced to 0 and the SPIx_CLK pin will not clock until this bit is set.
		0	SPI is disabled, and can be safely reconfigured
		1	SPI is enabled and will respond to bus transactions
23-17	Reserved		Reads return 0 and writes have no effect.
16	LOOPBACK		Internal loop-back test mode. The internal self-test option can be enabled by setting this bit. If the SPIx_SIMO and SPIx_SOMI pins are configured with SPI functionality, then the SPIx_SIMO pin is internally connected to the SPIx_SOMI pin. The transmit data is looped back as receive data and is stored in the receive field of the concerned buffer. Externally, during loop-back operation, SPIx_SIMO, SPIx_SOMI, and SPIx_CLK pins are tri-stated. The SPI has to be initialized in master mode before the loop-back can be selected. If the SPI is initialized in slave mode or a data transfer is ongoing, errors may result.
		0	Internal loop-back test mode disabled
		1	Internal loop-back test mode enabled
15-9	Reserved		Reads return 0 and writes have no effect.
8	POWERDOWN		SPI Powerdown mode is not supported on C672x devices.
		0	SPI in active mode
		1	Illegal. Do not write a '1' to this bit.
7-2	Reserved		Reads return 0 and writes have no effect.

Table A-3. SPI Global Control Register 1 (SPIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
1-0	CLKMOD, MASTER		These two bits (CLKMOD, MASTER) determine whether the SPI operates in master or slave mode.
		00	SLAVE MODE SPIx_CLK is an input from the master who initiates the transfers. Data is transmitted on the SPIx_SOMI pin and received on the SPIx_SIMO pin. The SPIx_SCS pin is an input if configured as SPI slave chip select. The SPIx_ENA pin is an output if configured as the SPI enable pin.
		11	MASTER MODE SPIx_CLK is an output and the SPI initiates transfers. Data is transmitted on the SPIx_SIMO pin and received on the SPIx_SOMI pin. The SPIx_SCS pin is an output if configured as SPI slave chip select. The SPIx_ENA pin is an input if configured as the SPI enable pin.
		01	RESERVED - The SPI will not function properly with this setting
		10	RESERVED - The SPI will not function properly with this setting

A.1.4 SPI Interrupt register (SPIINT0)

The SPI interrupt register (SPIINT0) is shown in Figure A-3 and described in Table A-4.

Figure A-3. SPI Interrupt register (SPIINT0)

31–25	24
Reserved	ENABLE HIGHZ
R-0	R/W-0

23–17	16
Reserved	DMAREQEN
R-0	R/W-0

15–9	8
Reserved	RXINTEN
R-0	R/W-0

7	6	5	4	3	2	1	0
Reserved	OVRNINTEN	Reserved	BITERRENA	DESYNCENA	DESELERRENA	TIMEOUTENA	Reserved
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0008 **SPI1 Address: 0x4800_0008**

Table A-4. SPI Interrupt register (SPIINT0) Field Descriptions

Bit	Field	Value	Description
31-15	Reserved		Reads return 0 and writes have no effect.
24	ENABLE HIGHZ		Selects between push pull and open drain modes for the SPIx_ENA pin in functional mode. . In 5-pin mode the master must see the SPIx_ENA pin deasserted immediately after it asserts SPIx_SCS. To meet this requirement the ENABLE_HIGHZ must be set to '1' and an external pullup resistor of appropriate strength should be used.
		0	SPIx_ENA pin is driven when both active and inactive (push pull)
		1	SPIx_ENA pin is 3-stated when inactive and driven low when active (open drain)
23-17	Reserved		Reads return 0 and writes have no effect.
16	DMAREQEN		DMA request enable. Enables the DMA request signal to be generated for both receive and transmit channels.
		0	DMA is not used
		1	DMA is used
15-9	Reserved		Reads return 0 and writes have no effect.
8	RXINTEN		Receive interrupt enable. An interrupt is to be generated when the SPIFLG.RXINTFLAG is set.
		0	Interrupt will not be generated
		1	Interrupt will be generated
7	Reserved		Reads return 0 and writes have no effect.
6	OVRNINTEN		Overrun interrupt enable. An interrupt is to be generated when the SPIFLG.OVRINTFLG is set.
		0	Overrun interrupt will not be generated
		1	Overrun interrupt will be generated
5	Reserved		Reserved
4	BITERRENA		Enables interrupt on bit error. An interrupt is to be generated when the SPIFLG.BITERRFLG is set.
		0	No interrupt asserted upon bit error.
		1	Enables an interrupt on a bit error
3	DESYNCENA		Enables interrupt on desynchronized slave. The desynchronization monitor is active in master mode for the 4-pin with enable and 5-pin options. An interrupt is to be generated when the SPIFLG.DESYNCFLG is set.
		0	No interrupt asserted upon desynchronization error.
		1	Enables an interrupt on desynchronization of the slave

Table A-4. SPI Interrupt register (SPIINT0) Field Descriptions (continued)

Bit	Field	Value	Description
2	DESELERRENA		Enables interrupt on slave deselection. The slave deselection monitor is active in slave mode for 4-pin with chip select and 5-pin options. An interrupt is to be generated when the SPIFLG.DESELFLG is set.
		0	DESEL interrupt will not be generated
		1	DESEL interrupt will be generated
1	TIMEOUTENA		Enables interrupt on $\overline{\text{SPIx\_ENA}}$ signal timeout. An interrupt is to be generated when SPIFLG.TIMEOUTFLG is set. No interrupt asserted upon SPIx_ENA signal timeout. Enables an interrupt on a timeout of the SPIx_ENA signal
		0	No interrupt asserted upon $\overline{\text{SPIx\_ENA}}$ signal timeout.
		1	Enables an interrupt on a timeout of the $\overline{\text{SPIx\_ENA}}$ signal
0	Reserved		Reads return 0 and writes have no effect.

CAUTION

As described in Section 6 it is STRONGLY RECOMMENDED to use the DMA mode for data transfers and reserve interrupts for error conditions for the C672x SPI implementation. This means leaving the RXINTEN bit of this register at '0' and setting the DMAREQEN bit of this register to '1'.

A.1.5 SPI Interrupt Level Register (SPILVL)

The SPI interrupt level register (SPILVL) is shown in Figure A-4 and described in Table A-5.

Figure A-4. SPI Interrupt Level Register (SPILVL)

31-16
Reserved
R-0

15-9	8
Reserved	RXINTLVL
R-0	R/W-0

7	6	5	4	3	2	1	0
Reserved	OVRNINTLVL	Reserved	BITERRLVL	DESYNCLVL	DESELERRLVL	TIMEOUTLVL	Reserved
R-0	R/W-0	R-0	R/W-0	0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_000C **SPI1 Address: 0x4800_000C**

Table A-5. SPI Interrupt Level Register (SPILVL) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reads return 0 and writes have no effect.
8	RXINTLVL		Receive interrupt level.
		0	Receive interrupt is mapped to interrupt line INT0.
		1	Receive interrupt is mapped to interrupt line INT1.
7	Reserved		Reads return 0 and writes have no effect.
6	OVRNINTLVL		Receive Overrun interrupt level.
		0	Receive Overrun interrupt is mapped to interrupt line INT0.
		1	Receive Overrun interrupt is mapped to interrupt line INT1.
5	Reserved		Reads return 0 and writes have no effect.
4	BITERRLVL		Bit error interrupt level.
		0	Bit error interrupt is mapped to interrupt line INT0.
		1	Bit error interrupt is mapped to interrupt line INT1.
3	DESYNCLVL		Desynchronized slave interrupt level. DESYNCLVL is used in master mode only.
		0	An interrupt due to desynchronization of the slave is mapped to interrupt line INT0.
		1	An interrupt due to desynchronization of the slave is mapped to interrupt line INT1.
2	DESELERRLVL		Deselected slave interrupt level. DESELERRLVL is use in slave mode only.
		0	An interrupt due to deselection of the slave is mapped to interrupt line INTO.
		1	An interrupt due to deselection of the slave is mapped to interrupt line INT1.
1	TIMEOUTLVL		$\overline{\text{SPIx\_ENA}}$ signal timeout interrupt level.
		0	An interrupt on a timeout of the $\overline{\text{SPIx\_ENA}}$ signal is mapped to interrupt line INT0.
		1	An interrupt on a timeout of the $\overline{\text{SPIx\_ENA}}$ signal is mapped to interrupt line INT1.
0	Reserved		Reads return 0 and writes have no effect.

CAUTION

In most cases, it is recommended to only use interrupt level 0 on a C672x device. This is because both SPI interrupt levels are combined again outside the SPI to assert the same CPU interrupt request. So making use of both interrupt levels inside the SPI will simply create additional overhead when servicing the SPI interrupt. For additional explanation, see Section 6.

A.1.6 SPI Flag Register (SPIFLG)

The SPI flag register (SPIFLG) is shown in Figure A-5 and described in Table A-6.

Figure A-5. SPI Flag Register (SPIFLG)

31-16
Reserved
R-0

15-9	8
Reserved	RXINTFLAG
R-0	RC-0

7	6	5	4	3	2	1	0
Reserved	OVRNINTFLG	Reserved	BITERRFLG	DESYNCFLG	DESELERRFLG	TIMEOUTFLG	Reserved
R-0	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	R-0

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -*n* = value after reset

SPI0 Address: 0x4700_0010 **SPI1 Address: 0x4800_0010**

Table A-6. SPI Flag Register (SPIFLG) Field Descriptions

Bit	Field	Value	Description
31-9	Reserved		Reads return 0 and writes have no effect.
8	RXINTFLAG		Serves as the interrupt flag. This flag is set when a word is received and copied into the buffer register (SPIBUF). If SPIINT0.RXINTEN is enabled, an interrupt is also generated. A read to the emulation register (SPIEMU) does not clear this flag bit. This bit is cleared in one of five ways: • Reading the SPIBUF register (Read must include SPIBUF[15:0]) • Writing a 1 to this bit • Writing a 0 to SPICGR1.ENABLE • Writing a 0 to SGIGCR0.RESET • TGINTVECT0/TGINTVECT1 is read and returns the value ‘100100b’ indicating receive interrupt The set condition takes priority over clear condition.
		0	Interrupt condition did not occur
		1	Interrupt condition did occur
7	Reserved		Reads return 0 and writes have no effect.
6	OVRNINTFLG		Receiver overrun flag. This bit is a read/clear only flag. The SPI hardware sets this bit when an operation completes before the previous character has been read from the buffer. The bit indicates that the last received character has been overwritten and therefore lost. The SPI will generate an interrupt request if this bit is set and the SPIINT0.OVRNINTEN bit is set. This bit is cleared in one of five ways: • Writing a 1 to this bit • Writing a 0 to SPICGR1.ENABLE • Writing a 0 to SPIGCR0.$\overline{\text{RESET}}$ • TGINTVECT0/TGINTVECT1 is read and returns the value '100110b' indicating overrun error. The set condition takes priority over clear condition.
		0	Overrun condition did not occur
		1	Overrun condition has occurred
5	Reserved		Reads return 0 and writes have no effect.

Table A-6. SPI Flag Register (SPIFLG) Field Descriptions (continued)

Bit	Field	Value	Description
4	BITERRFLG		Mismatch of internal transmit data and transmitted data. This flag is read/clear only flag, i.e. reading the flag will automatically clear it. No bit error occurred. A bit error occurred. The SPI samples the signal of the transmit pin at the time that the slave device samples receive data. If the sampled value differs from the transmitted value a bit error is detected then SPIFLG.BITERRFLG and SPIBUF.BITERR are set. If SPIINT0.BITERRENA is set an interrupt is asserted. A possible reason for a bit error can be a to high bit rate/capacitive load or another master/slave trying to transmit at the same time. This bit is cleared in one of the following ways: • Reading the SPIBUF register [1] • Reading the SPIFLG register [2] • Writing a 1 to this bit • Writing a 0 to SPIGCR1.ENABLE • Writing a 0 to SPIGCR0.$\overline{\text{RESET}}$ Set condition takes priority over clear condition.
		0	No bit error occurred.
		1	A bit error occurred.
3	DESYNCFLG		Desynchronization of slave device. This flag is read/clear only flag, i.e. reading the flag will automatically clear it. This flag is applicable only for the master mode. The master monitors the SPIx_ENA signal coming from the slave device and sets the SPIBUF.DESYNC and SPIFLG.DESYNCFLG bits if SPIx_ENA is deactivated before the last reception point or after the last bit is transmitted plus tT2EDELAY (see Section 8.21). If SPIINT0.DESYNCENA is set an interrupt is asserted. desynchronization can occur if a slave device misses a clock edge coming from the master or is detecting an additional clock edge due to perturbation. This bit is cleared in one of the following ways: • Reading the SPIBUF register [1] • Reading the SPIFLG register [2] • Writing a 1 to this bit • Writing a 0 to SPIGCR1.ENABLE • Writing a 0 to SPIGCR0.$\overline{\text{RESET}}$ Set condition takes priority over clear condition.
		0	No slave synchronization detected.
		1	A slave device is desynchronized.
2	DESELERRFLG		Deselection of the slave device. This flag is read/clear flag i.e., reading the flag will automatically clear it. This flag is applicable only for slave mode.
		0	No slave deselect detected
		1	The SPI slave device is detected. The slave monitors the SPIx_SCS signal coming from the master and sets the SPIFLG.DESELERRFLG bit if SPIx_SCS is deasserted before the transmission is completed. If SPINT0.DESELENA is set, an interrupt is asserted. This is an indication of an error on the master side, but the slave SPI may require a software reset to regain synchronization with the master. This bit is cleared in one of the following ways: • Reading the SPIBUF register[1] • Reading the SPIFLG register[2] • Writing a 1 to this bit • Writing a 0 to SPIGCR1.ENABLE • Writing a 0 to SPIGCR0.$\overline{\text{RESET}}$ Set condition takes priority over clear condition.

[1] Read must include SPIBUF[31:24]

[2] Read must include SPIFLG[7:0]

Table A-6. SPI Flag Register (SPIFLG) Field Descriptions (continued)

Bit	Field	Value	Description
1	TIMEOUTFLG		Timeout due to non-activation of SPIx_ENA signal. This flag is read/clear only flag, i.e. reading the flag will automatically clear it. This flag is applicable for the master mode. The SPI generates a time out error because the slave hasn't responded in time by activating the SPIx_ENA signal after the chip select signal has been activated. If a time out condition is detected the corresponding chip select is deasserted immediately and the SPIFLG.TIMEOUTFLG and SPIBUF.TIMEOUT bits are set. This bit is cleared in one of the following ways: • Reading the SPIBUF register [(1)] • Reading the SPIFLG register [(2)] • Writing a 1 to this bit • Writing a 0 to SPIGCR1.ENABLE • Writing a 0 to SPIGCR0.$\overline{\text{RESET}}$ Set condition takes priority over clear condition.
		0	No SPIx_ENA signal timeout occurred
		1	A SPIx_ENA signal timeout occurred
0	Reserved		Reads return 0 and writes have no effect.

CAUTION

The set condition takes priority over clear condition for SPIFLG.RXINTFLG. This leads to a possibility that the SPIFLG.RXINTFLG may not get cleared when SPIBUF is read within a specific timing window resulting in a missed interrupt. Hence it is STRONGLY RECOMMENDED to use interrupt only for error conditions and use SPI DMA events for servicing the SPI data transfers.

CAUTION

The set condition takes priority over clear condition for SPIFLG.DESYNCFLG. As a result the set condition may remain valid for multiple cycles, until the master completes the shifting state or the slave asserts the SPIx_ENA again. During this time it is not possible to clear the SPIFLG.DESYNCFLG since the set conditions take priority over the clear conditions.

A.1.7 SPI Pin Control Register (SPIPC0)

The SPI pin control register (SPIPC0) is shown in Figure A-6 and described in Table A-7.

Figure A-6. SPI Pin Control Register (SPIPC0)

31-16
Reserved
R-0

15-12	11	10	9	8
Reserved	SOMIFUN	SIMOFUN	CLKFUN	ENABLEFUN
R-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
reserved	reserved	reserved	reserved	reserved	reserved	reserved	SCSFUN0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0014 **SPI1 Address: 0x4800_0014**

Table A-7. SPI Pin Control Register (SPIPC0) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved		Reads return 0 and writes have no effect.
11	SOMIFUN		Slave out, master in function. Determines whether the SPIx_SOMI pin is to be used as a general-purpose I/O pin or as a SPI functional pin.
		0	SPIx_SOMI pin is a GPIO
		1	SPIx_SOMI pin is a SPI functional pin
10	SIMOFUN		Slave in, master out function. Determines whether the SPIx_SIMO pin is to be used as a general-purpose I/O pin, or as a SPI functional pin.
		0	SPIx_SIMO pin is a GPIO
		1	SPIx_SIMO pin is a SPI functional pin
9	CLKFUN		SPI clock function. Determines whether the SPIx_CLK pin is to be used as a general-purpose I/O pin, or as a SPI functional pin.
		0	SPIx_CLK pin is a GPIO
		1	SPIx_CLK pin is a SPI functional pin
8	ENABLEFUN		$\overline{\text{SPIx\_ENA}}$ function. Determines whether the $\overline{\text{SPIx\_ENA}}$ pin is to be used as a general-purpose I/O pin, or as a SPI functional pin.
		0	$\overline{\text{SPIx\_ENA}}$ pin is a GPIO
		1	$\overline{\text{SPIx\_ENA}}$ pin is a SPI functional pin
7-1	Reserved		These bits must be written as '0'. Reads are indeterminate.
0	SCSFUN0		$\overline{\text{SPIx\_SCS}}$ function. Determines whether the $\overline{\text{SPIx\_SCS}}$ pin is a general-purpose I/O pins or an SPI functional pin. If the slave $\overline{\text{SPIx\_SCS}}$ pin is in functional mode and receives an inactive high signal, the slave SPI will place its output in high-z and disable shifting.
		0	$\overline{\text{SPIx\_SCS}}$ pin is a GPIO
		1	$\overline{\text{SPIx\_SCS}}$ pin is a SPI functional pin

A.1.8 SPI Pin Control Register 1 (SPIPC1)

The SPI pin control register 1 (SPIPC1) is shown in Figure A-7 and described in Table A-8.

Figure A-7. SPI Pin Control Register 1 (SPIPC1)

31–16
Reserved
R-0

15–12	11	10	9	8
Reserved	SOMIDIR	SIMODIR	CLKDIR	ENABLEDIR
R-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
reserved	reserved	reserved	reserved	reserved	reserved	reserved	SCSDIR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0018 **SPI1 Address: 0x4800_0018**

Table A-8. SPI Pin Control Register 1 (SPIPC1) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved		Reads return 0 and writes have no effect.
11	SOMIDIR		SPIx_SOMI direction. Controls the direction of the SPIx_SOMI pin when it is used as a general-purpose I/O pin. If the SPIx_SOMI pin is used as a SPI functional pin, the I/O direction is determined by the whether the SPI is configured as a master or slave.
		0	SPIx_SOMI pin is an input when configured as GPIO
		1	SPIx_SOMI pin is an output when configured as GPIO
10	SIMODIR		SPIx_SIMO direction. Controls the direction of the SPIx_SIMO pin when it is used as a general-purpose I/O pin. If the SPIx_SIMO pin is used as a SPI functional pin, the I/O direction is determined by whether the SPI is configured as a master or slave.
		0	SPIx_SIMO pin is an input when configured as GPIO
		1	SPIx_SIMO pin is an output when configured as GPIO
9	CLKDIR		SPIx_CLK direction. Controls the direction of the SPIx_CLK pin when it is used as a general-purpose I/O pin. In functional mode, the I/O direction is determined by whether the SPI is configured as master or slave.
		0	SPIx_CLK pin is an input when configured as GPIO
		1	SPIx_CLK pin is an output when configured as GPIO
8	ENABLEDIR		$\overline{\text{SPIx\_ENA}}$ direction. Controls the direction of the $\overline{\text{SPIx\_ENA}}$ pin when it is used as a general-purpose I/O. In functional mode, the I/O direction is determined by whether the SPI is configured as master or slave.
		0	$\overline{\text{SPIx\_ENA}}$ pin is an input when configured as GPIO
		1	$\overline{\text{SPIx\_ENA}}$ pin is an output when configured as GPIO
7-1	Reserved		These bits must be written as '0'. Reads are indeterminate.
0	SCSDIR0		$\overline{\text{SPIx\_SCS}}$ direction. Controls the direction of the $\overline{\text{SPIx\_SCS}}$ pins when it is used as a general-purpose I/O pin. In functional mode, the I/O direction is determined by whether the SPI is configured as master or slave.
		0	$\overline{\text{SPIx\_SCS}}$ pin is an input when configured as GPIO
		1	$\overline{\text{SPIx\_SCS}}$ pin is an output when configured as GPIO

A.1.9 SPI Pin Control Register 2 (SPIPC2)

The SPI pin control register 2 (SPIPC2) is shown in Figure A-8 and described in Table A-9.

Figure A-8. SPI Pin Control Register 2 (SPIPC2)

31–16
Reserved
R-0

15–12	11	10	9	8
Reserved	SOMIDIN	SIMODIN	CLKDIN	ENABLEDIN
R-0	R-U	R-U	R-U	R-U

7	6	5	4	3	2	1	0
reserved	reserved	reserved	reserved	reserved	reserved	reserved	SCSDIN0
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U

LEGEND: R/W = Read/Write; R = Read only; U = Undefined; -*n* = value after reset

SPI0 Address: 0x4700_001C **SPI1 Address: 0x4800_001C**

Table A-9. SPI Pin Control Register 2 (SPIPC2) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved		Reads return 0 and writes have no effect.
11	SOMIDIN		SPIx_SOMI data in. Reflects the value of the SPIx_SOMI pin if configured for input. If SPIx_SOMI is configured for output then this bit indicates the value that is attempted to be driven on the SPIx_SOMI pin.
		0	Current value on SPIx_SOMI pin is logic 0
		1	Current value on SPIx_SOMI pin is logic 1
10	SIMODIN		SPIx_SIMO data in. Reflects the value of the SPIx_SIMO pin if configured for input. If SPIx_SIMO is configured for output then this bit indicates the value that is attempted to be driven on the SPIx_SIMO pin.
		0	Current value on SPIx_SIMO pin is logic 0
		1	Current value on SPIx_SIMO pin is logic 1
9	CLKDIN		Clock data in. Reflects the value of the SPIx_CLK pin if configured for input. If SPIx_CLK is configured for output then this bit indicates the value that is attempted to be driven on the SPIx_CLK pin.
		0	Current value on SPIx_CLK pin is logic 0
		1	Current value on SPIx_CLK pin is logic 1
8	ENABLEDIN		$\overline{\text{SPIx\_ENA}}$ data in. Reflects the value of the $\overline{\text{SPIx\_ENA}}$ pin if configured for input. If SPIx_ENA is configured for output then this bit indicates the value that is attempted to be driven on the SPIx_ENA pin.
		0	Current value on $\overline{\text{SPIx\_ENA}}$ pin is logic 0
		1	Current value on $\overline{\text{SPIx\_ENA}}$ pin is logic 1
7-1	Reserved		Reads are indeterminate.
0	SCSDIN0		$\overline{\text{SPIx\_SCS}}$ data in. Reflects the value of the $\overline{\text{SPIx\_SCS}}$ pin if configured for input. If SPIx_SCS is configured for output then this bit indicates the value that is attempted to be driven on the SPIx_SCS pin.
		0	Current value on $\overline{\text{SPIx\_SCS}}$ pin is logic 0
		1	Current value on $\overline{\text{SPIx\_SCS}}$ pin is logic 1

A.1.10 SPI Pin Control Register 3 (SPIPC3)

The SPI pin control register 3 (SPIPC3) is shown in Figure A-9 and described in Table A-10.

Figure A-9. SPI Pin Control Register 3 (SPIPC3)

31–16
Reserved
R-0

15–12	11	10	9	8
Reserved	SOMIDOUT	SIMODOUT	CLKDOUT	ENABLEDOUT
R-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
reserved	reserved	reserved	reserved	reserved	reserved	reserved	SCSDOUT0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0020 SPI1 Address: 0x4800_0020

Table A-10. SPI Pin Control Register 3 (SPIPC3) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved		Reads return 0 and writes have no effect.
11	SOMIDOUT		SPIx_SOMI dataout write. Only active when the SPIx_SOMI pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	SPIx_SOMI pin is driven to logic '0' if configured as a general purpose output pin.
		1	SPIx_SOMI pin is driven to logic '1' if configured as a general purpose output pin.
10	SIMODOUT		SPIx_SIMO dataout write. Only active when the SPIx_SIMO pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	SPIx_SIMO pin is driven to logic '0' if configured as a general purpose output pin.
		1	SPIx_SIMO pin is driven to logic '1' if configured as a general purpose output pin.
9	CLKDOUT		SPIx_CLK dataout write. Only active when the SPIx_CLK pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	SPIx_CLK pin is driven to logic '0' if configured as a general purpose output pin.
		1	SPIx_CLK pin is driven to logic '1' if configured as a general purpose output pin.
8	ENABLEDOUT		$\overline{\text{SPIx\_ENA}}$ dataout write. Only active when the $\overline{\text{SPIx\_ENA}}$ pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.
		0	$\overline{\text{SPIx\_ENA}}$ pin is driven to logic '0' if configured as a general purpose output pin.
		1	$\overline{\text{SPIx\_ENA}}$ pin is driven to logic '1' if configured as a general purpose output pin.
7-1	Reserved		These bits should be written as '0'. Reads are indeterminate.
0	SCSDOUT0		$\overline{\text{SPIx\_SCS}}$ data out write. Only active when the $\overline{\text{SPIx\_SCS}}$ pin is configured as a general-purpose I/O pins and configured as an output pins. The value of these bit indicates the value sent to the pins.
		0	$\overline{\text{SPIx\_SCS}}$ pin is driven to logic '0' if configured as a general purpose output pin.
		1	$\overline{\text{SPIx\_SCS}}$ pin is driven to logic '1' if configured as a general purpose output pin.

A.1.11 SPI Pin Control Register 4 (SPIPC4)

The SPI pin control register 4 (SPIPC4) is shown in Figure A-10 and described in Table A-11.

Figure A-10. SPI Pin Control Register 4 (SPIPC4)

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		SOMISET	SIMOSET	CLKSET	ENABLESET
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
reserved	reserved	reserved	reserved	reserved	reserved	reserved	SCSSET0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0024 **SPI1 Address: 0x4800_0024**

Table A-11. SPI Pin Control Register 4 (SPIPC4) Field Descriptions[1]

Bit	Field	Value	Description
31-12	Reserved		Reads return 0 and writes have no effect.
11	SOMISET		SPIx_SOMI data out set bit on write of '1'.
		Write: 0	No Effect
		Write: 1	SPIPC3.SOMIDOUT is set to '1'.
10	SIMOSET		SPIx_SIMO data out set bit on write of '1'.
		Write: 0	No Effect
		Write: 1	SPIPC3.SIMODOUT is set to '1'.
9	CLKSET		SPIx_CLK data out set bit on write of '1'.
		Write: 0	No Effect
		Write: 1	SPIPC3.CLKDOUT is set to '1'.
8	ENABLESET		$\overline{\mathrm{SPIx\_ENA}}$ data out set bit on write of '1'.
		Write: 0	No Effect
		Write: 1	SPIPC3.ENABLEDOUT is set to '1'.
7-1	Reserved		Reserved. Write '0' to these bits always. Reads are indeterminate.
0	SCSSET0		$\overline{\mathrm{SPIx\_SCS}}$ data out set bit on write of '1'.
		Write: 0	No Effect
		Write: 1	SPIPC3.SCSDOUT0 is set to '1'.

[1] Reading this register returns the value currently in the SPIPC3 register.

A.1.12 SPI Pin Control Register (SPIPC5)

The SPI pin control register (SPIPC5) is shown in Figure A-11 and described in Table A-12.

Figure A-11. SPI Pin Control Register (SPIPC5)

31–16
Reserved
R-0

15–12	11	10	9	8
Reserved	SOMICLR	SIMOCLR	CLKCLR	ENABLECLR
R-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
reserved	reserved	reserved	reserved	reserved	reserved	reserved	SCSCLR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0028 **SPI1 Address: 0x4800_0028**

Table A-12. SPI Pin Control Register (SPIPC5) Field Descriptions[1]

Bit	Field	Value	Description
31-12	Reserved		Reads return 0 and writes have no effect.
11	SOMICLR		SPIx_SOMI data out clear bit on write of '1'.
		Write: 0	No Effect
		Write: 1	SPIPC3.SOMIDOUT is cleared to '0'.
10	SIMOCLR		SPIx_SIMO data out clear bit on write of '1'.
		Write: 0	No Effect
		Write: 1	SPIPC3.SIMODOUT is cleared to '0'.
9	CLKCLR		SPIx_CLK data out clear bit on write of '1'.
		Write: 0	No Effect
		Write: 1	SPIPC3.CLKDOUT is cleared to '0'.
8	ENABLECLR		$\overline{\text{SPIx\_ENA}}$ data out clear bit on write of '1'.
		Write: 0	No Effect
		Write: 1	SPIPC3.ENABLEDOUT is cleared to '0'.
7-1	Reserved		Reserved. Write '0' to these bits always. Reads are indeterminate.
0	SCSCLR0		$\overline{\text{SPIx\_SCS}}$ data out clear bit on write of '1'.
		Write: 0	No Effect
		Write: 1	SPIPC3.SCSDOUT0 is cleared to '0'.

[1] Reading this register returns the value currently in the SPIPC3 register.

A.1.13 SPI Shift Register (SPIDAT0)

The SPI shift register (SPIDAT0) is shown in Figure A-12 and described in Table A-13.

Figure A-12. SPI Shift Register (SPIDAT0)

31	16
Reserved	
R-0	

15	0
SPIDAT0	
R/W-0	

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0038 **SPI1 Address: 0x4800_0038**

Table A-13. SPI Shift Register (SPIDAT0) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reads return 0 and writes have no effect.
15-0	SPIDAT0		These bits make up the SPI shift register 0. Writing a 0 to the SPICGR1.ENABLE register forces the lower 16 bits of the SPIDAT0 register to 0x00. SPICGR1.ENABLE must be set to 1 before this register can be written to.

A.1.14 SPI Shift register (SPIDAT1)

The SPI shift register (SPIDAT1) is shown in Figure A-13 and described in Table A-14.

Figure A-13. SPI Shift register (SPIDAT1)

31 29	28	27	26	25 24
Reserved	CSHOLD	Reserved	WDEL	DFSEL
R-0	R/W-0	R-0	R/W-0	R/W-0

23 16
CSNR
R/W-0

15 0
SPIDAT1
R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_003C SPI1 Address: 0x4800_003C

Table A-14. SPI Shift register (SPIDAT1) Field Descriptions[(1)]

Bit	Field	Value	Description
31-29	Reserved		Reads return 0 and writes have no effect.
28	CSHOLD		Chip select hold mode. Only applies in 4-pin chip select and 5-pin master modes. The current and previous values of CSHOLD are retained. Though the current value of CSHOLD is initialized to '0' when SPIGCR0.$\overline{\text{RESET}}$ is set to '0', the previous value of CSHOLD is not initialized. The previous value of CSHOLD must be explicitly initialized by writing CSHOLD field twice.
		0	The chip select is toggled between two consecutive accesses.
		1	The chip select is kept asserted between two consecutive accesses.
27	Reserved		Reads return 0 and writes have no effect.
26	WDEL		Enable the delay counter at the end of the current transaction.
		0	No delay inserted.
		1	After the transaction SPIFMTx.WDELAY[5:0] of the selected SPIFMTx register is loaded into the delay counter. The next transaction is delayed until the delay counter counts down to zero.
25-24	DFSEL	0-3h	Selects the SPIFMTx register to use. In slave mode only SPIFMT0 is supported.
		0	SPIFMT0 is selected
		1h	SPIFMT1 is selected
		2h	SPIFMT2 is selected
		3h	SPIFMT3 is selected
23-16	CSNR		Chip select number.
		00h	This field must be written as 00h.
		all others	Illegal.
15-0	SPIDAT1		These bits make up the SPI shift register 1. Data must be written to this register right aligned regardless of the settings of SPIFMTx.CHARLEN. SPICGR1.ENABLE must be set to 1 before this register can be written to. Writing a 0 to the SPICGR1.ENABLE register forces the lower 16 bits of the SPIDAT1 register to 0x00.

[(1)] A write to this register in SPI master mode will drive the $\overline{\text{SPIx\_SCS}}$ signal low unless it is configured as general purpose I/O.

CAUTION

The SPIDAT1 write operations that change the control fields CSHOLD, WDEL, DFSEL or CSNR should be done only after the previous transfer is completed.

CAUTION

When data is written to the SPIDAT1 register with the CSHOLD bit set to ‘1’, the master keeps the SPIx_SCS asserted at the end of the transfer. When data is written to the SPIDAT1 register with CSHOLD set to ‘0’ the master de asserts the SPIx_SCS pin at the end of the transfer. However it is found that when SPIx_SCS is asserted and data is written to the SPIDAT1 register with the CSHOLD set to ‘0’ the SPIx_SCS gets momentarily deasserted and asserted back resulting in a glitch. To avoid such a glitch all transfers during which SPIx_SCS should be active the user should write to SPIDAT1 register with CSHOLD set to ‘1’. This should be followed by a write to only the CSHOLD field with a value ‘0’. Alternatively the user can write to the CSHOLD fields only before and after the set of transfers to toggle the SPIx_SCS and write only to the SPIDAT1[15:0] during the transfers.

A.1.15 SPI Buffer Register (SPIBUF)

The SPI buffer register (SPIBUF) is shown in Figure A-14 and described in Table A-15.

Figure A-14. SPI Buffer Register (SPIBUF)

31	30	29	28	27	26	25	24
RXEMPTY	OVRNINT	TXFULL	BITERR	DESYNC	Reserved	TIMEOUT	Reserved
R-1	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	R-0

23	16
Reserved	
R-0	

15	0
SPIBUF	
R-x	

LEGEND: R/W = Read/Write; R = Read only; C = Clear; -*n* = value after reset, x = indeterminate

SPI0 Address: 0x4700_0040 SPI1 Address: 0x4800_0040

Table A-15. SPI Buffer Register (SPIBUF) Field Descriptions

Bit	Field	Value	Description
31	RXEMPTY		Receive data buffer empty. This flag is a read-only flag. When host reads the SPIBUF register this will automatically set the RXEMPTY flag. When a data transfer has been finished and the received data is copied into SPIBUF the RXEMPTY flag is cleared.
		0	Data is received and copied into SPIBUF field.
		1	No data received since last reading of SPIBUF register.
30	OVRNINT		Receive data buffer overrun. This flag is read/clear only flag, i.e. reading the flag will automatically clear it. When a data transfer has been finished and the received data is copied into the SPIBUF while RXEMPTY flag is already cleared OVRNINT as well as SPIFLG.OVRNINTFLG is set. . This bit is cleared in one of five ways: • Reading the SPIBUF register (Read must include SPIBUF[15:0]) • Writing a 1 to SPIFLG.OVRNINTFLG • Writing a 0 to SPICGR1.ENABLE • Writing a 0 to SPIGCR0.$\overline{\text{RESET}}$ • TGINTVECT0/TGINTVECT1 is read and returns the value '100110b' indicating overrun error.
		0	No receive data overrun condition occurred since last time reading the status field.
		1	A receive data overrun condition occurred since last time reading the status field.
29	TXFULL		Transmit data buffer full. This flag is a read-only flag. Writing into SPIDAT0 or SPIDAT1 will automatically set the TXFULL flag. After transfer of the transmit data the TXFULL flag is cleared. . This bit is also cleared by writing a 0 to SPIGCR1.ENABLE or by writing a 0 to SPIGCR0.$\overline{\text{RESET}}$.
		0	No new transmit data from host since previous transfer of transmit data.
		1	Host provided new transmit data to SPIDAT0 or SPIDAT1.
28	BITERR		Mismatch of internal transmit data and transmitted data. This flag is read/clear only flag, i.e. reading the flag will automatically clear it. It represents a copy of the SPIFLG.BITERRFLG.
		0	No bit error occurred.
		1	A bit error occurred. The SPI samples the signal of the transmit pin at the time that the slave device samples receive data. If the sampled value differs from the transmitted value a bit error is detected then SPIFLG.BITERRFLG and SPIBUF.BITERR are set. A possible reason for a bit error can be noise, a to high bit rate/capacitive load or another master/slave trying to transmit at the same time.

Table A-15. SPI Buffer Register (SPIBUF) Field Descriptions (continued)

Bit	Field	Value	Description
27	DESYNC		Desynchronization of slave device. This flag is read/clear only flag, i.e. reading the flag will automatically clear it. The desynchronization monitor is active in master mode for the 4-pin with enable and 5-pin options. DESYNC represents a copy of the SPIFLG.DESYNCFLG bit
		0	No slave desynchronization detected.
		1	A slave device is desynchronized. The master monitors the $\overline{\text{SPIx\_ENA}}$ signal coming from the slave device and sets this bit and the SPIFLG.DESYNCFLG bit if $\overline{\text{SPIx\_ENA}}$ is deactivated before the last reception point or after the last bit is transmitted plus $t_{T2EDELAY}$ (see Section 8.21). If SPIINT0.DESYNCENA is set an interrupt is asserted. Desynchronization can occur if a slave device misses a clock edge coming from the master or is detecting an additional clock edge due to perturbation.
26	DESEL		Deselection of slave device. This flag is read/clear only flag i.e. reading the flag will automatically clear it. DESEL represents a copy of the SPIFLG.DESELERRFLG bit.
		0	No slave deselect detected
		1	The SPI slave device is deselected. The slave monitors the SPIx_SCS signal coming from the master and sets this bit if SPIx_SCS is deasserted before the transmission is completed. If SPIINT0.DESELENA is set an interrupt is asserted. This is an indication of an error on the master side, but the slave SPI may require a software reset to regain synchronization with the master.
25	TIMEOUT		Timeout due to non-activation of $\overline{\text{SPIx\_ENA}}$ signal. This flag is read/clear only flag, i.e. reading the flag will automatically clear it.
		0	No $\overline{\text{SPIx\_ENA}}$ signal timeout occurred.
		1	An $\overline{\text{SPIx\_ENA}}$ signal timeout occurred. The SPI generates a timeout because the slave hasn't responded in time by activating the $\overline{\text{SPIx\_ENA}}$ signal after the chip select signal has been activated. If a timeout condition is detected the corresponding chip select is deactivated immediately and the TIMEOUT flag is set. In addition the SPIFLG.TIMEOUTFLG and SPIBUF.TIMEOUT flags are set.
24	Reserved		Reads return 0 and writes have no effect.
23-16	Reserved		Reserved - the C672x SPI only supports one chip select pin.
15-0	SPIBUF		SPI buffer. The data in this register is the data transferred from the shift-register (SPIDAT). Since the data is shifted into the SPI most significant bit first, for word lengths less than 16, the data is stored right-justified in the register.

CAUTION

The user is strongly recommended to use dMAX to service SPI data requests. In case polling mechanism is used to service SPI data requests then the user is advised to use SPIBUF.RXEMPTY or SPIFLG.RXINTFLG instead of SPIBUF.TXFULL.

Note: A read operation of any portion of SPIBUF qualifies as a receive data read even if this does not include the SPIBUF[15:0]. The SPIBUF should always be accessed with a 32-bit read operation.

A.1.16 SPI Emulation Register (SPIEMU)

The SPI emulation register (SPIEMU) is shown in Figure A-15 and described in Table A-16.

Figure A-15. SPI Emulation Register (SPIEMU)

31	16
Reserved	
R-0	

15	0
SPIEMU	
R-U	

LEGEND: R/W = Read/Write; R = Read only; ; U = Undefined; -*n* = value after reset

SPI0 Address: 0x4700_0044 **SPI1 Address: 0x4800_0044**

Table A-16. SPI Emulation Register (SPIEMU) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved		Reads return 0 and writes have no effect.
15-0	SPIEMU		SPI emulation - an alias of the SPIBUF register. Unlike SPIBUF, a read from SPIEMU does not clear the SPIFLG.OVRNINTFLG or SPIFLG.RXINTFLAG bits. Also, the status portion of the SPIBUF register is not available when reading the SPIEMU register.

A.1.17 SPI Delay Register (SPIDELAY)

The SPI delay register (SPIDELAY) is shown in Figure A-16 and described in Table A-17.

Figure A-16. SPI Delay Register (SPIDELAY)

31 29	28 24	23 21	20 16
Reserved	C2TDELAY	Reserved	T2CDELAY
R-0	R/W-0	R-0	R/W-0

15 8	7 0
T2EDELAY	C2EDELAY
R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0048 **SPI1 Address: 0x4800_0048**

Table A-17. SPI Delay Register (SPIDELAY) Field Descriptions

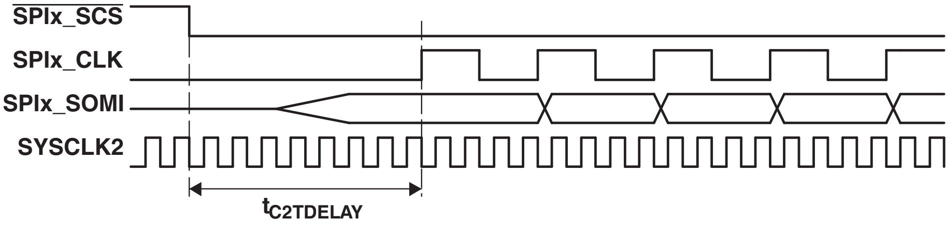
Bit	Field	Value	Description
31-29	Reserved		Reads return 0 and writes have no effect.
28-24	C2TDELAY		Chip-select-active-to-transmit-start-delay. C2TDELAY is used in master mode only. If the slave device indicates the ready-to-transfer status via the $\overline{\text{SPIx\_ENA}}$ signal, C2TDELAY time is not delayed after the $\overline{\text{SPIx\_ENA}}$ signal is activated. It defines a setup time for the slave device that delays the data transmission from the chip select active edge by a multiple of SYSCLK2 cycles. C2TDELAY can be configured between 2 and 33 SYSCLK2 cycles. **Figure A-17. Example: $t_{C2TDELAY}$ = 8 SYSCLK2 Cycles**  $\overline{\text{SPIx\_SCS}}$ SPIx_CLK SPIx_SOMI SYSCLK2 $t_{C2TDELAY}$ The setup time value is calculated as shown in Equation 1. $t_{C2TDELAY} = \frac{(C2TDELAY + 2)}{SYSCLK2}$ tExample: SYSCLK2 = 25 MHz; C2TDELAY = 06h; $t_{C2TDELAY}$ = 320 ns; When the chip select signal becomes active the slave has to prepare data transfer within 320 ns.

Table A-17. SPI Delay Register (SPIDELAY) Field Descriptions (continued)

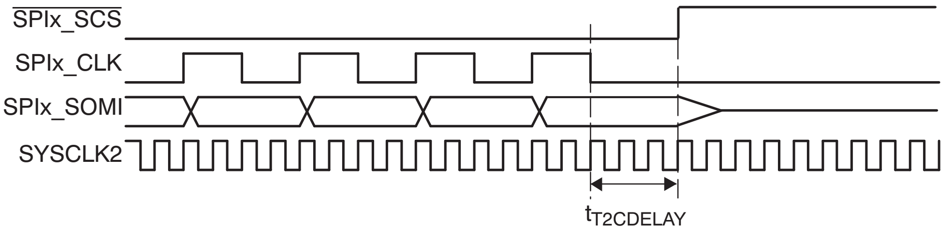
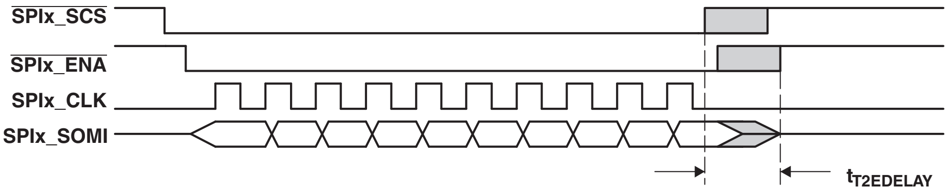
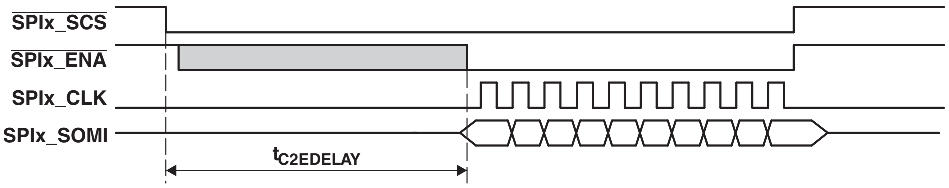
Bit	Field	Value	Description
20-16	T2CDELAY		Transmit-end-to-chip-select-inactive-delay. T2CDELAY is used in master mode only. It defines a hold time for the slave device that delays the chip select deactivation by a multiple of SYSCLK2 cycles after the last bit is transferred. T2CDELAY can be configured between 1 and 32 SYSCLK2 cycles. **Figure A-18. Example: $t_{T2CDELAY}$ = 3 SYSCLK2 Cycles**  SPIx_SCS SPIx_CLK SPIx_SOMI SYSCLK2 $t_{T2CDELAY}$ The hold time value is calculated as shown in Equation 2. $t_{T2CDELAY} = \frac{(T2CDELAY+1)}{SYSCLK2}$ Example: SYSCLK2 = 25 MHz; T2CDELAY = 02h; $t_{T2CDELAY}$ = 120 ns; After the last data bit (or parity bit) is being transferred the chip select signal is held active for 120 ns.
23-21	Reserved		Reads return 0 and writes have no effect.
15-8	T2EDELAY		Transmit data finished to SPIx_ENA pin inactive timeout. T2EDELAY is used in master mode only. It defines a timeout value as a multiple of SPI clock before the SPIx_ENA signal has to become inactive and after the SPIx_SCS becomes inactive. The SPI clock depends on which data format is selected. If the slave device is missing one or more clock edges, it's becoming desynchronized. Although the master has finished the data transfer the slave is still waiting for the missed clock pulses and the SPIx_ENA signal isn't disabled. The T2EDELAY defines a timeout value that triggers the SPIBUF.DESYNC and SPIFLG.DESYNCFLG bits, if the SPIx_ENA signal isn't deactivated in time. **Figure A-19. Transmit Data Finished to SPIx_ENA Inactive Timeout Value**  SPIx_SCS SPIx_ENA SPIx_CLK SPIx_SOMI $t_{T2EDELAY}$ The timeout value is calculated as shown in Equation 3. If PRESCALEx < 127 then $t_{T2EDELAY} = \frac{(T2EDELAY \times (PRESCALE + 2))}{SYSCLK2}$ Else $t_{T2EDELAY} = \frac{T2EDELAY}{SYSCLK2}$ Example: SYSCLK2 = 24MHz; PRESCALEx = 1; T2EDELAY = 10h; > $t_{T2EDELAY}$ = 2s; The slave device has to disable the SPIx_ENA signal within 2s, otherwise the SPIBUF.DESYNC and SPIFLG.DESYNCFLG bits are is set and an interrupt is asserted if enabled.

Table A-17. SPI Delay Register (SPIDELAY) Field Descriptions (continued)

Bit	Field	Value	Description
7-0	C2EDELAY		Chip select active to SPIx_ENA signal active timeout. C2EDELAY is utilized only in master mode and it applies only if the addressed slave generates the SPIx_ENA signal as a hardware handshake response. C2EDELAY defines the maximum time between the SPI activates the chip select signal and the addressed slave has to respond by activating the SPIx_ENA signal. C2EDELAY defines a timeout value as a multiple of SPI clocks. The SPI clock depends on whether data format 0 or data format 1 is selected. If the slave device is not responding with the SPIx_ENA signal before the timeout value is reached, the SPIFLG.TIMEOUTFLG bit is set and a interrupt is asserted if enabled. If SPIFMTx.WAITENA is set to '1' and C2EDELAY is '0' then the master waits indefinitely for the slave to assert SPIx_ENA. **Figure A-20. Chip Select Active to SPIx_ENA Signal Active Timeout Value**  The timeout value is calculated as shown in Equation 4. If PRESCALEx < 127 then $t_{C2EDELAY} = \frac{(C2EDELAY\ x\ (PRESCALEx + 2))}{SYSCLK2}$ Else $t_{C2EDELAY} = \frac{C2EDELAY}{SYSCLK2}$ α Example: SYSCLK2 = 24MHz; PRESCALEx = 1; C2EDELAY = 30h; > $t_{C2EDELAY}$ = 6µs; The slave device has to activate the SPIx_ENA signal within 6 µs after the SPI has activated the chip select signal (SCS), otherwise the TIMEOUT flag is set and a interrupt is asserted if enabled.

CAUTION

The formulas in this section do not account for I/O timings in the C672x device datasheet. The formulas will be updated in a future release of this document.

A.1.18 SPI Default Chip Select Register (SPIDEF)

The SPI default chip select register (SPIDEF) is shown in Figure A-21 and described in Table A-18.

Figure A-21. SPI Default Chip Select Register (SPIDEF)

31 – 8
Reserved
R-0

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	CSDEF0
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_004C **SPI1 Address: 0x4800_004C**

Table A-18. SPI Default Chip Select Register (SPIDEF) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0 and writes have no effect.
7-1	Reserved	1	Write a '1' to these bits always.
0	CSDEF[0]		State of the $\overline{\text{SPIx\_SCS}}$ pin when deasserted.
		0	Illegal. Setting this bit to '0' will cause improper operation of the $\overline{\text{SPIx\_SCS}}$ pin.
		1	Write a '1' to this bit always.

A.1.19 SPI Data Format Registers (SPIFMTx)

The SPI data format registers (SPIFMTx, where x = 0, 1, 2, 3) is shown in Figure A-22 and described in Table A-19.

Figure A-22. SPI Data Format Registers (SPIFMTx)

31 30	29 24
Reserved	WDELAYx
R-0	R/W-0

23	22	21	20	19 18	17	16
PARPOLx	PARITYxENA	WAITENAx	SHIFTDIRx	Reserved	POLARITYx	PHASEx
R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

15 8	7 5	4 0
PRESCALEx	Reserved	CHARLENx
R/W-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPIFMT0	**SPI0 Address: 0x4700_0050**	**SPI1 Address: 0x4800_0050**
SPIFMT1	**SPI0 Address: 0x4700_0054**	**SPI1 Address: 0x4800_0054**
SPIFMT2	**SPI0 Address: 0x4700_0058**	**SPI1 Address: 0x4800_0058**
SPIFMT3	**SPI0 Address: 0x4700_005C**	**SPI1 Address: 0x4800_005C**

Table A-19. SPI Data Format Registers (SPIFMTx) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved		Reads return 0 and writes have no effect.
29-24	WDELAY[5:0]		Delay in between transmissions for data format x (x= 0,1,2,3). Idle time that will be applied at the end of the current transmission if the bit WDEL is set in the current buffer. The delay to be applied is equal to: WDELAY * $P_{SYSCLK2}$ + 2 * $P_{SYSCLK2}$.
23	PARPOL		Parity polarity. Parity is not supported on the C672x DSP SPI.
		0	This bit must be written as '0' on C672x DSP devices.
		1	Illegal.
22	PARITYENA		Parity enable. Parity is not supported on the C672x DSP SPI.
		0	This bit must be written as '0' on C672x DSP devices.
		1	Illegal.
21	WAITENA		Master waits for $\overline{\text{SPIx\_ENA}}$ signal from slave for data format x. WAITENA is considered in master mode only. In slave mode this bit has no meaning. WAITENA enables a flexible SPI network where slaves with $\overline{\text{SPIx\_ENA}}$ signal and slaves without $\overline{\text{SPIx\_ENA}}$ signal can be mixed. WAITENA defines for each buffer whether the addressed slave generates the $\overline{\text{SPIx\_ENA}}$ signal or does not.
		0	The SPI does not wait for the $\overline{\text{SPIx\_ENA}}$ signal from the slaves and directly starts the transfer.
		1	Before the SPI starts the data transfer it waits for the $\overline{\text{SPIx\_ENA}}$ signal to become low. If the SPIx_ENA signal is not pulled down by the addressed slave before the internal timeout counter (C2EDELAY) expires a timeout condition is detected. If C2EDELAY is '0' then the master waits for SPIx_ENA from slave indefinitely.
20	SHIFTDIR		Shift direction for data format x. With bit SHIFTDIRx the shift direction for data format x (x=1,2,3) can be selected.
		0	Data format x shift direction: Most significant bit is shifted out first.
		1	Data format x shift direction: Least significant bit is shifted out first.
19-18	Reserved		Reads return 0 and writes have no effect.
17	POLARITY		SPI data format x clock polarity. POLARITYx defines the clock polarity of data format x.
		0	If POLARITYx is set to 0 the SPI clock signal is low-inactive, i.e., before and after data transfer the clock signal is low.
		1	If POLARITYx is set to 1 the SPI clock signal is high-inactive, i.e., before and after data transfer the clock signal is high.

Table A-19. SPI Data Format Registers (SPIFMTx) Field Descriptions (continued)

Bit	Field	Value	Description
16	PHASE		SPI data format x clock delay. PHASEx defines the clock delay of data format x.
		0	If PHASEx is set to 0 the SPI clock signal is not delayed versus the transmit/receive data stream. The first data bit is transmitted with the first clock edge and the first bit is received with the second (inverse) clock edge.
		1	If PHASEx is set to 1 the SPI clock signal is delayed by a half SPI clock cycle versus the transmit/receive data stream. The first transmit bit has to output prior to the first clock edge. Master and slave receive the first bit with the first edge.
15-8	PRESCALE[7:0]		SPI data format x prescaler. PRESCALEx determines the bit transfer rate of data format x if the SPI is the network master. PRESCALEx is directly derived from SYSCLK2. If the SPI is configured as slave, PRESCALEx DOES NOT NEED to be configured. The clock rate for data format x can be calculated as shown in Equation 5. $BR_{Formatx} = \frac{SYSCLK2}{(PRESCALEx + 1)}$ When PRESCALEx is set to zero (0), the SPI clock rate is SYSCLK2/2.
7-5	Reserved		Reads return 0 and writes have no effect. n
4-0	CHARLEN[4:0]		SPI data format x data word length. CHARLENx defines the word length of data format x. Legal values are 0x02 (data word length = 2 bit) to 0x10 (data word length = 16). Illegal values, such as 0x00 or 0x1F are not detected and their effect is indeterminate.

A.1.20 SPI Interrupt Vector Register 0 Transfer Group Interrupt Vector Register 0 (TGINTVECT0)

The SPI interrupt vector register 0 transfer group interrupt vector register 0 (TGINTVECT0) is shown in Figure A-23 and described in Table A-20.

Figure A-23. SPI Interrupt Vector Register 0 Transfer Group Interrupt Vector Register 0 (TGINTVECT0)

31 16
Reserved
R-0

15 8
Reserved
R-0

7 6	5 1	0
Reserved	INTVECT0	Reserved
R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0060   SPI1 Address: 0x4800_0060

Table A-20. SPI Interrupt Vector Register 0 Transfer Group Interrupt Vector Register 0 (TGINTVECT0) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-1	INTVECT0		This bit field indicates the highest priority interrupt flag configured to interrupt on level 0, based on the priority order. The bit field is then automatically updated to indicate the next highest priority interrupt.
		10011b	Priority 1 (highest): SPIFLG.OVRNINTFLG is pending. Reading INTVECT0 with this value automatically clears the SPIFLG.OVRINTFLG.
		10010b	Priority 2: SPIFLG.RXINTFLAG is pending. Reading INTVECT0 with this value automatically clears the SPIFLG.RXINTFLAG.
		10001b	Priority 3: Error interrupt(s) from the group SPIFLG.BITERRFLG, SPIFLG.DESYNCFLG, SPIFLG.DESELERRFLG, or SPIFLG.TIMEOUTFLG is pending. Read SPIFLG to determine which interrupt(s) are pending.
		00000b	Priority 4 (lowest): no interrupt is pending
0	Reserved		Reserved

A.1.21 SPI Interrupt Vector Register 1 Transfer Group Interrupt Vector Register 1 (TGINTVECT1)

The SPI interrupt vector register 1 transfer group interrupt vector register 1 (TGINTVECT1) is shown in Figure A-24 and described in Table A-21.

Figure A-24. SPI Interrupt Vector Register 1 Transfer Group Interrupt Vector Register 1 (TGINTVECT1)

31	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7 6	5 1	0
Reserved	INTVECT1	Reserved
R-0	R-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

SPI0 Address: 0x4700_0064 **SPI1 Address: 0x4800_0064**

Table A-21. SPI Interrupt Vector Register 1 Transfer Group Interrupt Vector Register 1 (TGINTVECT1) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved		Reserved
5-1	INTVECT1		This bit field indicates the highest priority interrupt flag configured to interrupt on level 1, based on the priority order. The bit field is then automatically updated to indicate the next highest priority interrupt.
		10011b	Priority 1 (highest): SPIFLG.OVRNINTFLG is pending. Reading INTVECT1 with this value automatically clears the SPIFLG.OVRINTFLG.
		10010b	Priority 2: SPIFLG.RXINTFLAG is pending. Reading INTVECT1 with this value automatically clears the SPIFLG.RXINTFLG.
		10001b	Priority 3: Error interrupt(s) from the group SPIFLG.BITERRFLG, SPIFLG.DESYNCFLG, SPIFLG.DESELERRFLG, or SPIFLG.TIMEOUTFLG is pending. Read SPIFLG to determine which interrupt(s) are pending.
		00000b	Priority 4 (lowest): no interrupt is pending
0	Reserved		Reserved

CAUTION

In most cases, it is recommended to only use interrupt level 0 on a C672x device. This is because both SPI interrupt levels are combined again outside the SPI to assert the same CPU interrupt request. So making use of both interrupt levels inside the SPI will simply create additional overhead when servicing the SPI interrupt. For additional explanation, see Section 6.

Appendix B

B.1 Programming SPI

1. Take the SPI out of reset by setting SPIGCR0.$\overline{\text{RESET}}$ to '1'
2. Configure the SPI for master or slave mode by configuring the SPIGCR1.CLKMOD and SPIGCR1.MASTER.
3. Configure the SPI for 3-pin, 4-pin with chip select, 4-pin with enable or 5-pin mode by configuring the SPIPC0 register
4. Chose the SPIFMT register to be used by configuring SPIDAT1.DFSEL. In slave mode only SPIFMT0 is supported.
5. Configure the SPI data rate, character length, shift direction, phase, polarity and other format options using the SPIFMTx register selected in step 4.
6. If SPI master then configure the master delay options using SPIDELAY register. In slave mode SPIDELAY is not relevant.
7. Select the error interrupt notifications by configuring SPIINT0 and SPILVL registers.
8. Enable the SPI communication by setting the SPIGCR1.ENABLE to '1'
9. Enable the DMA servicing for the SPI data requests by setting the SPIINT0.DMAREQEN to '1'.
10. Handle SPI data transfer requests using dMAX and service any SPI error conditions using the interrupt service routine.

B.1.1 Example Code Showing SPI Configured as Master with dMAX Servicing the SPI Data Requests

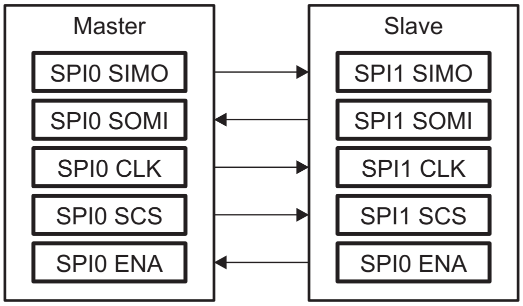
```
#define SPIGCR0       0x0000
#define SPIGCR1       0x0001
#define SPIINT0       0x0002
#define SPILVL        0x0003
#define SPIFLG        0x0004
#define SPIPC0        0x0005
#define SPIPC1        0x0006
#define SPIPC2        0x0007
#define SPIPC3        0x0008
#define SPIPC4        0x0009
#define SPIPC5        0x000A
#define SPIRES1       0x000B
#define SPIRES2       0x000C
#define SPIRES3       0x000D
#define SPIDAT0       0x000E
#define SPIDAT1       0x000F
#define SPIBUF        0x0010
#define SPIEMU        0x0011
#define SPIDELAY      0x0012
#define SPIDEF        0x0013
#define SPIFMT0       0x0014
#define SPIFMT1       0x0015
#define SPIFMT2       0x0016
#define SPIFMT3       0x0017
#define TGINTVECT0    0x0018
#define TGINTVECT1    0x0019

void spi_master()
{
    volatile unsigned int * spi0_ptr = (unsigned int *) 0x47000000;
    unsigned int pinmsk;

    /* Configure dMAX channel to service SPI data transfers */
    dMAX_spi();

    /* Chose the SPI pins to be configured as functional pins */
    pinmsk=0x0E00;     /* SPI 3-pin mode */
    pinmsk=0x0E01;     /* SPI 4-pin with chip select mode */
    pinmsk=0x0F00;     /* SPI 4-pin with enable mode */
    pinmsk=0x0F01;     /* SPI 5-pin mode */

    /* 1. Take the SPI0 out of reset */
    spi0_ptr[SPIGCR0]=0x01;

    /* 2. Configure SPI0 for master */
    spi0_ptr[SPIGCR1]=0x03;

    /* 3. Configure SPI0 mode */
    spi0_ptr[SPIPC0]=pinmsk;

    /* 4. Chose SPI0 SPIFMT0 */
    spi0_ptr[SPIDAT1]=0x00000000;

    /* 5. Configure SPI0 for SHIFTDIR=0,POLARITY=1,PHASE=0,
          CHARLEN=16. Use WAITENA=1 for 4-pin with enable and 5-pin mode */
    if(pinmsk&0x0100)
        spi0_ptr[SPIFMT0]=0x00221810;
    else
        spi0_ptr[SPIFMT0]=0x00021810;

    /* 6. Configure SPI0 for C2TDELAY=2,T2CDELAY=2,T2EDELAY=4,C2EDELAY=8 */
    spi0_ptr[SPIDELAY]=0x02020408;

    /* 7. Configure SPI0 for error notifications for BITERR,DESYNC and TIMEOUT */
    spi0_ptr[SPIINT0]=0x0000001A;
    spi0_ptr[SPILVL]=0x00;

    /* 8. Enable SPI0 communication */
    spi0_ptr[SPIGCR1]|=0x01000000;

    /* 9. Configure SPI0 for dmax servicing */
    spi0_ptr[SPIINT0]|=0x00010000;

    /* 10. Let dMAX handle SPI data transfer and interrupt service routine handle
       SPI error cases */

    /* Perform any other processing */
}
```

B.1.2 Example Code Showing SPI Configured as Slave with dMAX Servicing the SPI Data requests

```
void spi_slave()
{
    volatile unsigned int * spi0_ptr = (unsigned int *) 0x47000000;
    unsigned int pinmsk;

    /* Configure dMAX channel to service SPI data transfers */
    dMAX_spi();

    /* Chose the SPI pins to be configured as functional pins */
    pinmsk=0x0E00;     /* SPI 3-pin mode */
    pinmsk=0x0E01;     /* SPI 4-pin with chip select mode */
    pinmsk=0x0F00;     /* SPI 4-pin with enable mode */
    pinmsk=0x0F01;     /* SPI 5-pin mode */

    /* 1. Take the SPI0 out of reset */
    spi0_ptr[SPIGCR0]=0x01;

    /* 2. Configure SPI0 for slave */
    spi0_ptr[SPIGCR1]=0x00;

    /* 3. Configure SPI0 mode */
    spi0_ptr[SPIPC0]=pinmsk;
    /* 4. Chose SPI0 SPIFMT0 */
    spi0_ptr[SPIDAT1]=0x00000000;

    /* 5. Configure SPI0 for SHIFTDIR=0,POLARITY=1,PHASE=0,
          CHARLEN=16 */
    spi0_ptr[SPIFMT0]=0x00020010;

    /* 6. SPIDELAY for SPI0 not relevant in slave mode */
    spi0_ptr[SPIDELAY]=0x00;

    /* 7. Configure SPI0 for error notifications for OVR,BITERR and DESEL */
    spi0_ptr[SPIINT0]=0x00000054;
    spi0_ptr[SPILVL]=0x00;

    /* 8. Enable SPI0 communication */
    spi0_ptr[SPIGCR1]|=0x01000000;

    /* 9. Configure SPI0 for dmax servicing of spi events*/
    spi0_ptr[SPIINT0]|=0x00010000;

    /* 10. Let dMAX handle SPI data transfer and interrupt service routine handle
       SPI error cases */

    /* Perform any other processing */
}
```

B.2 Example SPI code using polling

Figure B-1. SPI Polling Example Connections



The following code demonstrates SPI data transfer between SPI0 configured as master in 5-pin mode and SPI1 configured as slave in 5-pin mode using polling mechanism. Figure B-1 shows the connections used for this polling example. This code is provided here only for demonstration purpose and the user is strongly recommended to use dMAX for SPI data transfers.

```
#include <stdio.h>

void main()
{
    volatile unsigned int * spi0_ptr = (unsigned int *) 0x47000000;
    volatile unsigned int * spi1_ptr = (unsigned int *) 0x48000000;
    unsigned int i,flg0,flg1,rx0,rx1;

    /* Configure SPI0 for master */
    /* 1. Take the SPI0 out of reset */
    spi0_ptr[SPIGCR0]=0x01;
    /* 2. Configure SPI0 for master */
    spi0_ptr[SPIGCR1]=0x03;
    /* 3. Configure SPI0 for 5-pin mode */
    spi0_ptr[SPIPC0]=0x0F01;
    /* 4. Chose SPI0 SPIFMT1 */
    spi0_ptr[SPIDAT1]=0x01000000;
```

```
/* 5. Configure SPI0 for WAITENA=1,SHIFTDIR=0,POLARITY=1,PHASE=0,
      CHARLEN=16 */
spi0_ptr[SPIFMT1]=0x00221810;
/* 6. Configure SPI0 for C2TDELAY=2,T2CDELAY=2,T2EDELAY=4,C2EDELAY=8 */
spi0_ptr[SPIDELAY]=0x02020408;
/* 7. Configure SPI0 for polling */
spi0_ptr[SPIINT0]=0x00;
/* 8. Enable SPI0 communication */
spi0_ptr[SPIGCR1]|=0x01000000;

/* Configure SPI1 for slave */
/* 1. Take the SPI1 out of reset */
spi1_ptr[SPIGCR0]=0x01;
/* 2. Configure SPI1 for slave */
spi1_ptr[SPIGCR1]=0x00;
/* 3. Configure SPI1 for 5-pin mode */
spi1_ptr[SPIPC0]=0x0F01;
/* 4. Chose SPI1 SPIFMT0 */
spi1_ptr[SPIDAT1]=0x00000000;
/* 5. Configure SPI1 for SHIFTDIR=0,POLARITY=1,PHASE=0,
      CHARLEN=16 */
spi1_ptr[SPIFMT0]=0x00020010;
/* 6. SPIDELAY for SPI1 not relevant in slave mode */
spi1_ptr[SPIDELAY]=0x00;
/* 7. Configure SPI1 for polling */
spi1_ptr[SPIINT0]=0x00;
/* 8. Enable SPI1 communication */
spi1_ptr[SPIGCR1]|=0x01000000;

/* 9. Handle data transfer and error checking using polling */
for(i=1;i<0x10000;++i){
    /* Write Data for TX */
    spi1_ptr[SPIDAT0]=(unsigned int)i-1;
    spi0_ptr[SPIDAT0]=(unsigned int)i;

    /* Wait for slave receive data */
    while(1){
        flg1=spi1_ptr[SPIFLG];
        if(flg1&0x40){
            printf("Slave overrun error\n");
            exit(1);
        }
        if(flg1&0x10){
            printf("Slave bit transmission error\n");
            exit(2);
        }
        if(flg1&0x04){
            printf("Slave deselection error\n");
            exit(3);
        }
        if(flg1&0x100)
            break;
        else if(flg1){
            printf("Unknown slave error\n");
            exit(4);
        }
    }
    rx1=spi1_ptr[SPIBUF]&0xFFFF;

    /* Wait for master receive data */
    while(1){
        flg0=spi0_ptr[SPIFLG];
        if(flg0&0x10){
            printf("Master bit transmission error\n");
            exit(5);
        }
```

```
                if(flg0&0x08){
                    printf("Master detected slave desynchronization error\n");
                    exit(6);
                }
                if(flg0&0x02){
                    printf("Master detected slave timeout error\n");
                    exit(7);
                }
                if(flg0&0x100)
                    break;
                else if(flg0){
                    printf("Unknown master error\n");
                    exit(8);
                }
            }
            rx0=spi0_ptr[SPIBUF]&0xFFFF;

            /* Check the data received */
            if(rx1!=i||rx0!=i-1){
                printf("SPI data error\n");
                exit(3);
            }
        }
        printf("SPI data transfer succeeded\n");
    }
```

Appendix C

C.1 Timing

This section contains timing diagrams illustrating the C2TDELAY, C2EDELAY, T2CDELAY, T2EDELAY and WDELAY delays and their interaction with SPIx_SCS and SPIx_ENA pins for all SPI modes.

C.1.1 SPI 3-Pin Mode

Figure C-1. SPI 3-Pin Master Mode with WDELAY

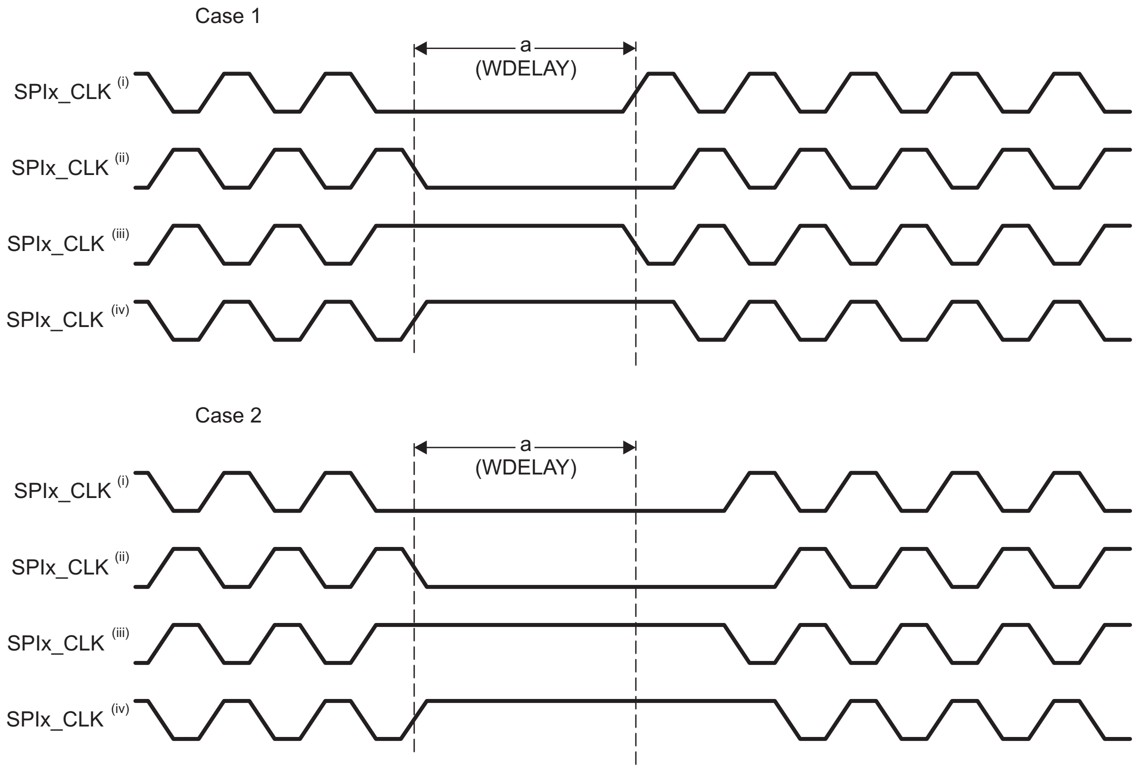


Figure C-1 illustrates the WDELAY option in SPI 3-pin master mode. This is the only delay available in this mode. In CASE1, a new transfer is initiated during the WDELAY period and the transfer begins immediately after the WDELAY period ends. In CASE2, while WDELAY has completed, a new transfer will not begin until SPIDAT0/SPIDAT1 have been written with new data.

C.1.2 SPI 4-Pin with SCS Mode

Figure C-2. SPI 4-Pin with SCS Mode with T2CDELAY, WDELAY, and C2TDELAY

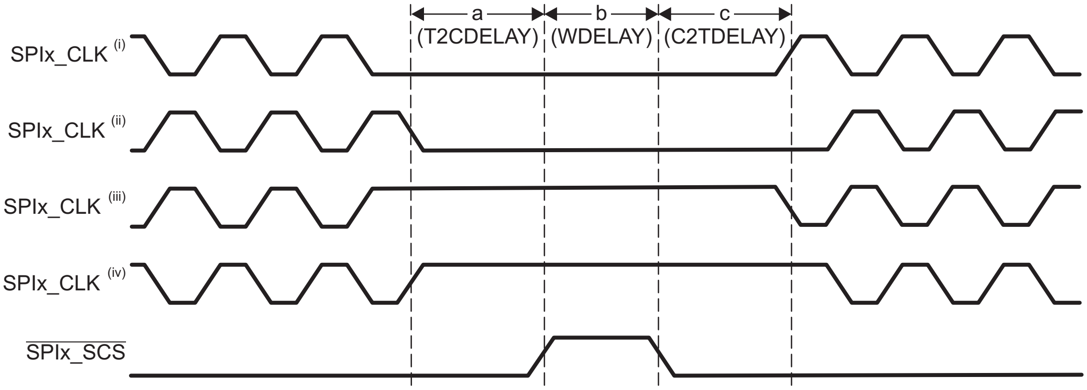


Figure C-2 illustrates the T2CDELAY, WDELAY and C2TDELAY delays in SPI 4-pin with SCS master mode. C2EDELAY and T2EDELAY are not available in this mode. All the three delay periods T2CDELAY, WDELAY and C2TDELAY proceed to completion when enabled.

C.1.3 SPI 4-Pin with ENA Mode

Figure C-3. SPI 4-Pin with ENA Mode Demonstrating T2EDELAY and WDELAY

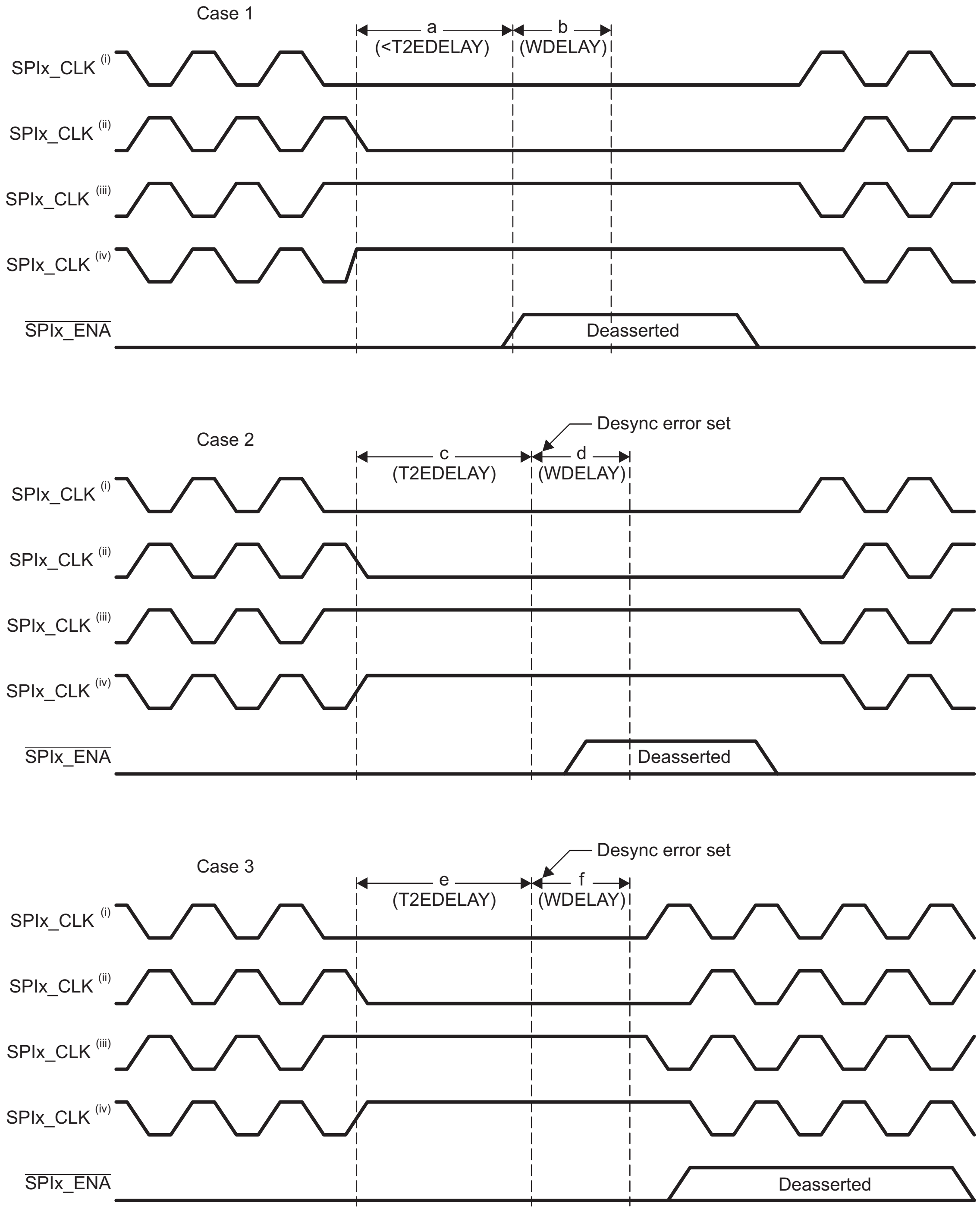


Figure C-3 shows the T2EDELAY and WDELAY delays in SPI 4-pin with ENA master mode. T2CDELAY, C2TDELAY and C2EDELAY are not available in this mode.

In CASE1, the SPIx_ENA is deasserted during the T2EDELAY period. Consequently the T2EDELAY period is terminated early (a) and the WDELAY period begins immediately (b) if enabled. The next transfer is initiated as soon as the slave asserts SPIx_ENA gain.

In CASE2, the T2EDELAY period (c) completes before the SPIx_ENA is deasserted. As a result the DESYNC error is set. However since the SPIx_ENA is deasserted during the WDELAY period (d), the master delays the next transfer until the SPIx_ENA is asserted again.

In CASE3, the T2EDELAY (e) and WDELAY (f) period (if enabled) both expire before the SPIx_ENA input is deasserted. The DESYNC error is set at the end of the T2EDELAY period (e). However in this case the master begins the next transfer immediately after it is initiated and ignores the SPIx_ENA during the transfer even if it is subsequently deasserted.

If the T2EDELAY delay period is disabled then the DESYNC error is not set. The SPI master behavior in this case depends on whether the SPIx_ENA gets deasserted during the WDELAY period (CASE2) or SPIx_ENA gets deasserted after the WDELAY period completes (CASE3).

C.1.4 SPI 5-Pin Mode

Figure C-4. SPI 5-Pin Mode Demonstrating T2CDELAY, T2EDELAY and WDELAY

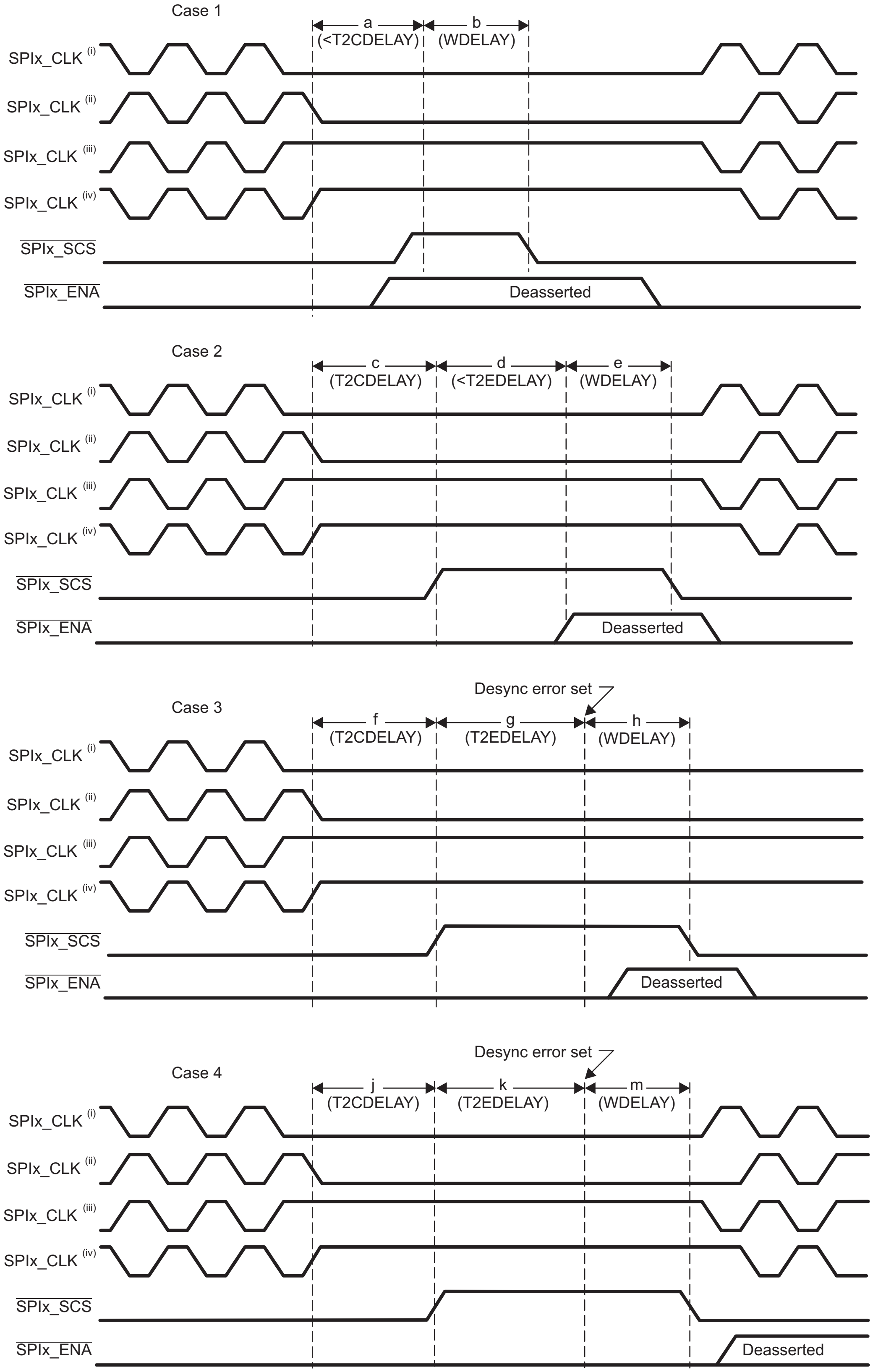


Figure C-4 shows the T2CDELAY, T2EDELAY and WDELAY delays in SPI 5-pin master mode.

In CASE1, the SPIx_ENA is deasserted during the T2CDELAY period. Consequently the T2CDELAY period is terminated early (a), the T2EDELAY period is skipped (if enabled) and the WDELAY period begins immediately (b) (if enabled). The next transfer is initiated as soon as the slave asserts SPIx_ENA gain.

In CASE2, the SPIx_ENA signal is deasserted by the slave during the T2EDELAY period (d) which begins upon the completion of the T2CDELAY period (c). The deassertion of the SPIx_ENA causes the T2EDELAY period to terminate early and the WDELAY period (e) begins immediately (if enabled) after the T2EDELAY period terminates. The next transfer is initiated as soon as the slave asserts SPIx_ENA gain.

In CASE3, the SPIx_ENA signal is deasserted by the slave during the WDELAY period (h) which begins upon the completion of the T2CDELAY period (f) and T2EDELAY period (g). As a result the DESYNC error is set at the end of the T2EDELAY period (g). However since the SPIx_ENA is deasserted during the WDELAY period (h), the master delays the next transfer until the SPIx_ENA is asserted again.

In CASE4, the SPIx_ENA signal is not deasserted until after the completion of the T2CDELAY (j), T2EDELAY (k) and WDELAY (m) (if enabled) periods. The DESYNC error is set at the end of the T2EDELAY period (k). However in this case the master begins the next transfer immediately after it is initiated and ignores the SPIx_ENA during the transfer even if it is subsequently deasserted.

If the T2EDELAY delay period is disabled then the DESYNC error is not set. The SPI master behavior in this case depends on whether the SPIx_ENA gets deasserted during the T2CDELAY period (CASE1), WDELAY period (CASE3) or after the WDELAY period completes (CASE4).

If the slave deasserts the SPIx_ENA signal before the completion of the configured master delays (T2CDELAY, T2EDELAY, WDELAY) then the master delays the next transfer until the slave asserts the SPIx_ENA again. However if the slave delays the SPIx_ENA deassertion until after the completion of the configured master delays then the master begins the next transfer immediately after it is initiated and ignores the SPIx_ENA during the transfer even if it is subsequently deasserted.

Figure C-5. SPI 5-Pin Mode Demonstrating C2TDELAY and C2EDELAY

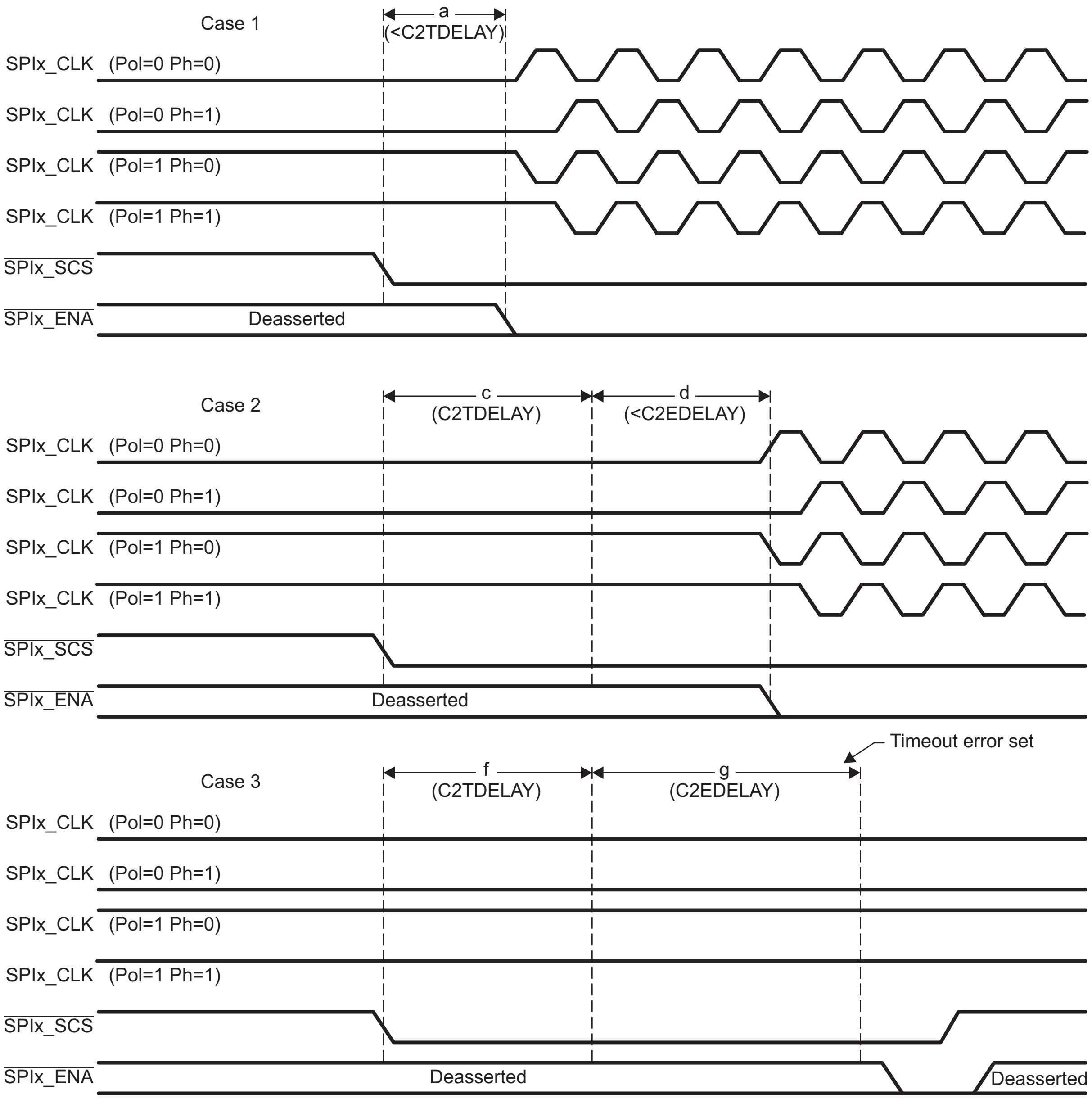


Figure C-5 shows the C2TDELAY and C2EDELAY in SPI 5-pin master mode.

In CASE1, the SPIx_ENA signal is asserted during the C2TDELAY period (a). Consequently the C2TDELAY period is terminated early (a), the C2EDELAY period is skipped (if enabled) and the master begins generating the SPI clock for transmission.

In CASE2, the SPIx_ENA signal is asserted during the C2EDELAY period (d) which begins upon the completion of C2TDELAY period (c). The assertion of the SPIx_ENA causes the C2EDELAY period to terminate early and the master begins generating the SPI clock for transmission.

In CASE3, the SPIx_ENA signal is not asserted until after the completion of the C2TDELAY (f) and C2EDELAY (g) periods. The TIMEOUT error is set at the end of the C2EDELAY period (g). The master deasserts the SCS signal immediately and clears the current transmit request.

If the C2EDELAY delay period is disabled then the SPI master behavior depends on whether the SPIx_ENA gets asserted during the C2TDELAY period (CASE1) or after the C2TDELAY period completes (CASE2). In latter case there is no limit on how long the master will wait for the slave to respond with SPIx_ENA asserted and hence there is no limit on period 'd' shown in CASE2. Thus when C2EDELAY period is disabled the TIMEOUT error is not set.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
Low Power Wireless	www.ti.com/lpw	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265