TMS320DM36x
Digital Media System-on-Chip (DMSoC)
Serial Peripheral Interface (SPI)

User's Guide

Literature Number: SPRUFH1B
March 2009–Revised March 2010

List of Figures

List of Tables

Read This First

This document describes the serial peripheral interface (SPI) on the TMS320DM36x Digital Media System-on-Chip (DMSoC).

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion.

Related Documentation From Texas Instruments

The following documents describe the TMS320DM36x Digital Media System-on-Chip (DMSoC). Copies of these documents are available on the internet at www.ti.com.

SPRUFG5 — *TMS320DM365 Digital Media System-on-Chip (DMSoC) ARM Subsystem Reference Guide* This document describes the ARM Subsystem in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The ARM subsystem is designed to give the ARM926EJ-S (ARM9) master control of the device. In general, the ARM is responsible for configuration and control of the device; including the components of the ARM Subsystem, the peripherals, and the external memories.

SPRUFG8 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Video Processing Front End (VPFE) Users Guide* This document describes the Video Processing Front End (VPFE) in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFG9 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Video Processing Back End (VPBE) Users Guide* This document describes the Video Processing Back End (VPBE) in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH0 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) 64-bit Timer Users Guide* This document describes the operation of the software-programmable 64-bit timers in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH1 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Serial Peripheral Interface (SPI) Users Guide* This document describes the serial peripheral interface (SPI) in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (1 to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communication between the DMSoC and external peripherals. Typical applications include an interface to external I/O or peripheral expansion via devices such as shift registers, display drivers, SPI EPROMs and analog-to-digital converters.

SPRUFH2 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Universal Asynchronous Receiver/Transmitter (UART) Users Guide* This document describes the universal asynchronous receiver/transmitter (UART) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The UART peripheral performs serial-to-parallel conversion on data received from a peripheral device, and parallel-to-serial conversion on data received from the CPU.

SPRUFH3 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Inter-Integrated Circuit (I2C) Peripheral Users Guide* This document describes the inter-integrated circuit (I2C) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The I2C peripheral provides an interface between the DMSoC and other devices compliant with the I2C-bus specification and connected by way of an I2C-bus.

SPRUFH5 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Multimedia Card (MMC)/Secure Digital (SD) Card Controller Users Guide* This document describes the multimedia card (MMC)/secure digital (SD) card controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH6 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Pulse-Width Modulator (PWM) Users Guide* This document describes the pulse-width modulator (PWM) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH7 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Real-Time Out (RTO) Controller Users Guide* This document describes the Real Time Out (RTO) controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFH8 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) General-Purpose Input/Output (GPIO) Users Guide* This document describes the general-purpose input/output (GPIO) peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs.

SPRUFH9 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Universal Serial Bus (USB) Controller Users Guide* This document describes the universal serial bus (USB) controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The USB controller supports data throughput rates up to 480 Mbps. It provides a mechanism for data transfer between USB devices and also supports host negotiation.

SPRUFI0 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Enhanced Direct Memory Access (EDMA) Controller Users Guide* This document describes the operation of the enhanced direct memory access (EDMA3) controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The EDMA controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the DMSoC.

SPRUFI1 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Asynchronous External Memory Interface (EMIF) Users Guide* This document describes the asynchronous external memory interface (EMIF) in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The EMIF supports a glueless interface to a variety of external devices.

SPRUFI2 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) DDR2/Mobile DDR (DDR2/mDDR) Memory Controller Users Guide* This document describes the DDR2/mDDR memory controller in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The DDR2/mDDR memory controller is used to interface with JESD79D-2A standard compliant DDR2 SDRAM and mobile DDR devices.

SPRUFI3 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Multibuffered Serial Port Interface (McBSP) User's Guide* This document describes the operation of the multibuffered serial host port interface in the TMS320DM36x Digital Media System-on-Chip (DMSoC). The primary audio modes that are supported by the McBSP are the AC97 and IIS modes. In addition to the primary audio modes, the McBSP supports general serial port receive and transmit operation.

SPRUFI4 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Universal Host Port Interface (UHPI) User's Guide* This document describes the operation of the universal host port interface in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI5 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Ethernet Media Access Controller (EMAC) User's Guide* This document describes the operation of the ethernet media access controller interface in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI7 — ***TMS320DM36x Digital Media System-on-Chip (DMSoC) Analog to Digital Converter (ADC) User's Guide*** This document describes the operation of the analog to digital conversion in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI8 — ***TMS320DM36x Digital Media System-on-Chip (DMSoC) Key Scan User's Guide*** This document describes the key scan peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC).

SPRUFI9 — ***TMS320DM36x Digital Media System-on-Chip (DMSoC) Voice Codec User's Guide*** This document describes the voice codec peripheral in the TMS320DM36x Digital Media System-on-Chip (DMSoC). This module can access ADC/DAC data with internal FIFO (Read FIFO/Write FIFO). The CPU communicates to the voice codec module using 32-bit-wide control registers accessible via the internal peripheral bus.

SPRUFJ0 — ***TMS320DM36x Digital Media System-on-Chip (DMSoC) Power Management and Real-Time Clock Subsystem (PRTCSS) User's Guide*** This document provides a functional description of the Power Management and Real-Time Clock Subsystem (PRTCSS) in the TMS320DM36x Digital Media System-on-Chip (DMSoC) and PRTC interface (PRTCIF).

SPRUGG8 — *TMS320DM36x Digital Media System-on-Chip (DMSoC) Face Detection User's Guide*This document describes the face detection capabilities for the TMS320DM36x Digital Media System-on-Chip (DMSoC).

User's Guide
SPRUFH1B–March 2009–Revised March 2010

Serial Peripheral Interface (SPI)

1 Introduction

This document describes the serial peripheral interface (SPI) in the digital media system-on-chip (DMSoC).

1.1 Purpose of the Peripheral

The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (1 to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communication between the device DMSoC and external peripherals. Typical applications include an interface to external I/O or peripheral expansion via devices such as shift registers, display drivers, SPI EPROMs and analog-to-digital converters.

The SPI allows serial communication with other SPI devices through a 3-pin or 4-pin mode interface. The device DMSoC implementation supports multichip-select operation for up to two SPI slave devices. The SPI operates as a master or slave SPI device.

1.2 Features

The SPI has the following features:

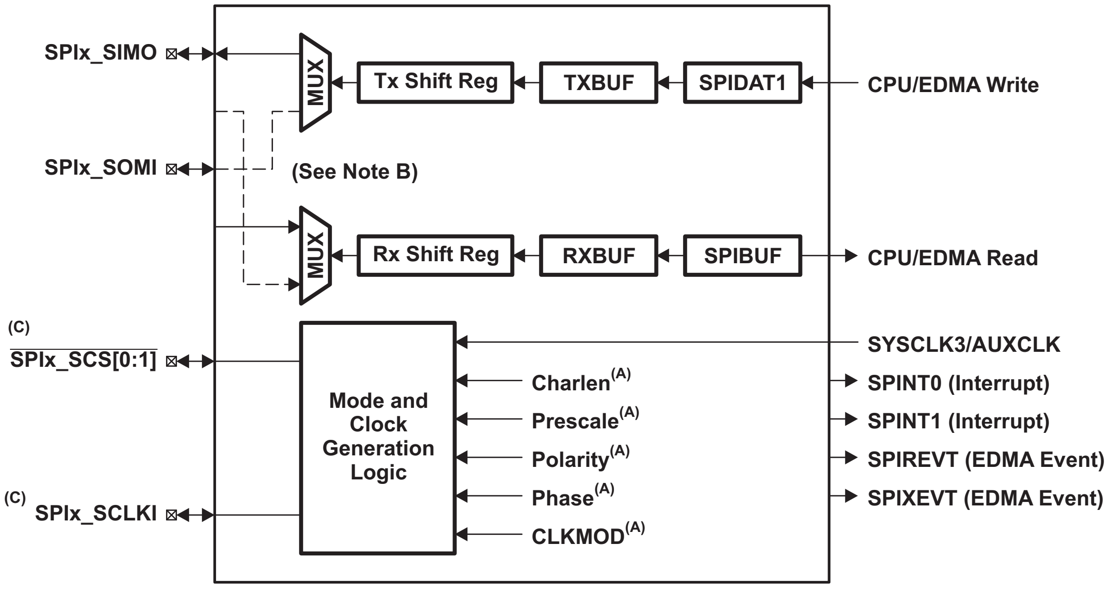
- 16-bit shift register
- Receive buffer register
- 8-bit clock prescaler
- Programmable SPI clock frequency range
- Programmable character length (2 to 16 bits)
- Programmable clock phase (delay or no delay)
- Programmable clock polarity (high or low)
- Two chip select signals ($\overline{\text{SPIx\_SCS[0]}}$ and $\overline{\text{SPIx\_SCS[1]}}$) provide the ability to control two slave devices

1.3 Functional Block Diagram

The SPI operates in a master or slave mode. The MASTER bit in the SPI global control register 1 (SPIGCR1) selects the configuration of the SPIx_SIMO and SPIx_SOMI pins and the CLKMOD bit in SPIGCR1 determines whether an internal or external clock source will be used. The slave chip select ($\overline{\text{SPIx\_SCS[1]}}$ and $\overline{\text{SPIx\_SCS[0]}}$) pins are used when communicating with multiple slave devices. When the SPI (master) writes to SPIDAT1, the $\overline{\text{SPIx\_SCS[n]}}$ pins are automatically driven to select the slave connected to that signal. Refer to Figure 1 that shows the SPI transaction registers. The internal buffers TXBUF and RXBUF are intended to improve the overall throughput of data transfer in SPI.

A block diagram of the major components of the SPI is shown in Figure 1.

Figure 1. Serial Peripheral Interface (SPI) Block Diagram



A Indicates the log controlled by SPI register bits

B Solid line represents data flow for SPI master mode. Dashed line represents data flow for SPI slave mode.

C SPIx_SCS [0:1] are only outputs for SPI4 in slave mode and SPI4 does not have support for chip select pin. Also Interrupts and EDAM events are not supported on SPI4.

1.3.1 Data Sequencing: Write

- If both Tx shift register and TXBUF (internal temporary register) are empty, then the data is directly copied to the Tx shift register. A Transmit EDMA Request or a Transmitter Empty Interrupt Request will be generated at the same time if enabled, so that next data can be fetched.
- If the Tx shift register is already full or in the process of shifting, then the data is copied to the TXBUF irrespective of whether it's already full or not, and the TXFULL flag will be set to 1 at the same time.
- When a shift operation is complete, data from the TXBUF (if it is full) is copied into Tx shift register and the TXFULL flag is cleared to 0 to indicate that next data can be fetched. A Transmit EDMA Request or a Transmitter Empty Interrupt Request will be generated at the same time.

1.3.2 Data Sequencing: Read

- If both SPIBUF and RXBUF are empty, the received data in Rx shift register is directly copied into SPIBUF and Receive EDMA Request and/or Receive Complete Interrupt will be generated if enabled. RXEMPTY flag in the SPIBUF will be cleared at the same time.
- If SPIBUF is already full at the end of receive completion, the Rx shift register contents will be copied to RXBUF (internal temporary register). A Receive EDMA request is generated if enabled. The Receive Complete Interrupt line continues to remain high.
- If SPIBUF is read by ARM/EDMA and if RXBUF is full, then the contents of RXBUF will be copied to SPIBUF as soon as SPIBUF is read. RXEMPTY flag will remain cleared indicating that the SPIBUF is still full.
- If both SPIBUF and RXBUF are full, then RXBUF will be overwritten and RXOVR interrupt flag will be set and an interrupt will be generated if enabled.

1.4 Industry Standard(s) Compliance Statement

The programmable configuration capability of the SPI allows it to gluelessly interface to a variety of SPI format devices. The SPI does not conform to a specific industry standard.

2 Peripheral Architecture

This section describes the architecture of the device DMSoC SPI.

2.1 Clock Control

The output clock generated (SPIx_SCLK) is a derivative of the internal peripheral clock that drives the SPI Module divided by the PRESCALEn bit in the SPI data format register (SPIFMTn). For specific information on the maximum SPI clock rate supported, refer to the SPI timings in the device-specific data manual. The phase and polarity of the SPI clock signal are also programmable, these configurations are explained in Section 2.4.2. The clock rate is set independently for each of the four software-programmable data formats. For more information on the data formats, see Section 2.4.1. The clock rate for a given data format *n* is calculated as:

SPIx_SCLK frequency = [peripheral clock frequency] / [PRESCALEn + 1]

PRESCALE*n* is only supported for values >1, where the SPI clock rate is (peripheral clock frequency)/2.

2.2 Signal Descriptions

Table 1 shows the SPI pins used to interface to external devices. SPIx_SCLK, SPIx_SIMO, and SPIx_SOMI are always used. The $\overline{\text{SPIx\_SCS[1:0]}}$ pins are optional and may be used if the pins are present on the slave device(s). The $\overline{\text{SPIx\_SCS[1:0]}}$ pins are used to selectively enable slaves in a multiple slave system. The SPI can be operated in a 3-pin or 4-pin mode configuration. In the 3-pin mode configuration, the $\overline{\text{SPIx\_SCS[1:0]}}$ pins are not used.

Table 1. Serial Peripheral Interface (SPI) Pins General Configurations

Pin	Type	Function
SPIx_SCLK	I/O	Serial clock
SPIx_SOMI	I/O	Serial data I/O
SPIx_SIMO	I/O	Serial data I/O
$\overline{\text{SPIx\_SCS[0]}}$	I/O	Chip select 0
$\overline{\text{SPIx\_SCS[1]}}$	I/O	Chip select 1

Table 2. Serial Peripheral Interface (SPI) Instance Specific Supported Features

Channel	Function	SPIx_SCLK Direction	SPIx_SCS[n] Chip Select Pin Available	SPIx_SCS[n] Chip Select Pin Available	Reason for Chip Select Support
SPI-0	Master/Slave	In/Out	2 pins (Master mode)	1 pin (Slave mode)	SCS[1] is the same pin for SOMI
SPI-1	Master/Slave	In/Out	2 pins (Master mode)	1 pin (Slave mode)	SCS[1] is the same pin for SOMI
SPI-2	Master/Slave	In/Out	2 pins (Master mode)	1 pin (Slave mode)	SCS[1] is the same pin for SOMI
SPI-3	Master/Slave	In/Out	2 pins (Master mode)	1 pin (Slave mode)	SCS[1] is the same pin for SOMI
SPI-4	Master/Slave	In/Out	2 pins (Master mode)	0 pin (Slave mode)	In Slave mode, Support Only Data Input function with Clock in.

2.3 Pin Multiplexing

Most of the SPI pins are multiplexed with other device functions. For example, when the SPI serial port functions are not selected, the pins may be used as general-purpose input/output (GPIO) pins. For specific information on pin multiplexing, refer to the device-specific data manual.

2.4 SPI Operation

All SPI Instances operate as a master or slave SPI device only. SPI4 can be configured as a slave but no chip select pin is available in slave mode. The MASTER and CLKMOD bits in the SPI global control register 1 (SPIGCR1) must be set to 1 for SPI to function as a master and cleared to 0 for SPI to function as a slave.

2.4.1 Data Formats

The SPI provides the capability to configure four independent data formats. These formats are configured by programming the corresponding SPI data format register (SPIFMT*n*). In each data format, the following characteristics of the SPI operation are selected:

- **Character length from 2 to 16 bits:** The character length is configured by the CHARLEN*n* bit.
- **Shift direction (MSB first or LSB first):** The shift out direction is configured by the SHIFTDIR*n* bit.
- **Clock polarity:** The clock polarity is configured by the POLARITY*n* bit. The clock polarity is explained in Section 2.4.2.
- **Clock phase:** The clock phase is configured by the PHASE*n* bit. The clock phase formats are explained in Section 2.4.2.

The data format is chosen on each transaction, providing the capability to use different formats with different slaves. Transmit data is written to the SPI shift register (SPIDAT1) and in the same write the data word format select (DFSEL) bit in SPIDAT1 indicates which data format is to be used for the next transaction. Alternatively, the data format can be configured once and applies to all transactions that follow until the data format is changed.

2.4.1.1 *Character Length*

The character length is configured by the CHARLEN*n* bit. Legal values are 2 bits (2h) to 16 bits (10h). The character length is independently configured for each of the four data formats and it must be programmed in both master mode and slave mode.

Transmit data is written to SPIDAT1. The transmit data must be written right-justified in the SPIDAT1 field. The SPI automatically sends out the data correctly based on the chosen data format. Figure 2 shows an example of how transmit data should be written for a 14-bit character length.

Figure 2. Right-Aligned Transmit Data in SPIDAT1 Field of the SPI Shift Register (SPIDAT1)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	1	0	1	0	1	0	1	0	1	0	1	0	1	0

When a full word of receive data arrives in SPIDAT1, it is copied to the SPI buffer register (SPIBUF). The received data is read from SPIBUF by the CPU or the EDMA. The received data in SPIBUF is right-justified. If the character length is less than 16 bits, additional bits may be present in SPIBUF left over from the transmitted data. But since the data in SPIBUF is right-justified and the character length is known, the additional bits can be ignored. Figure 3 shows an example of how receive data will be aligned for a 14-bit character length.

Figure 3. Right-Aligned Receive Data in SPIBUF Field of the SPI Buffer Register (SPIBUF)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	1	0	1	0	1	0	1	0	1	0	1	0	1	0

2.4.1.2 *Shift Direction*

The shift out direction is configured as most-significant bit (MSB) first or least significant bit (LSB) first. The shift out direction is selected by the SHIFTDIR*n* bit. The shift out direction is independently configured for each of the four data formats.

- When SHIFTDIR*n* is 0, the transmit data is shifted out MSB first.
- When SHIFTDIR*n* is 1, the transmit data is shifted out LSB first.

2.4.2 Clock Polarity and Phase

The SPI provides the flexibility to program four different clock mode combinations that SPIx_SCLK may operate, enabling a choice of the clock phase (delay or no delay) and the clock polarity (rising edge or falling edge). When operating with PHASE active, the SPI makes the first bit of data available after SPIDAT1 is written and before the first edge of SPIx_SCLK. The data input and output edges depend on the values of both the POLARITY and PHASE bits as shown in Table 3.

Table 3. SPI Clocking Modes

SPIFMT*n* Bit		
POLARITY	**PHASE**	**Action**
0	0	Data is output on the rising edge of SPIx_SCLK . Input data is latched on the falling edge.
0	1	Data is output one half-cycle before the first rising edge of SPIx_SCLK and on subsequent falling edges. Input data is latched on the rising edge of . SPIx_SCLK.
1	0	Data is output on the falling edge of SPIx_SCLK . Input data is latched on the rising edge.
1	1	Data is output one half-cycle before the first falling edge of SPIx_SCLK and on subsequent rising edges. Input data is latched on the falling edge of SPIx_SCLK .

Figure 4 through Figure 7 show the four possible signals of SPIx_SCLK corresponding to each mode. Having four signal options allows the SPI to interface with different types of serial devices. Also shown on the footnotes in each figure is the SPIx_SCLK control bit polarity and phase values corresponding to each signal.

Figure 4. Clock Mode with POLARITY=0 and PHASE=0

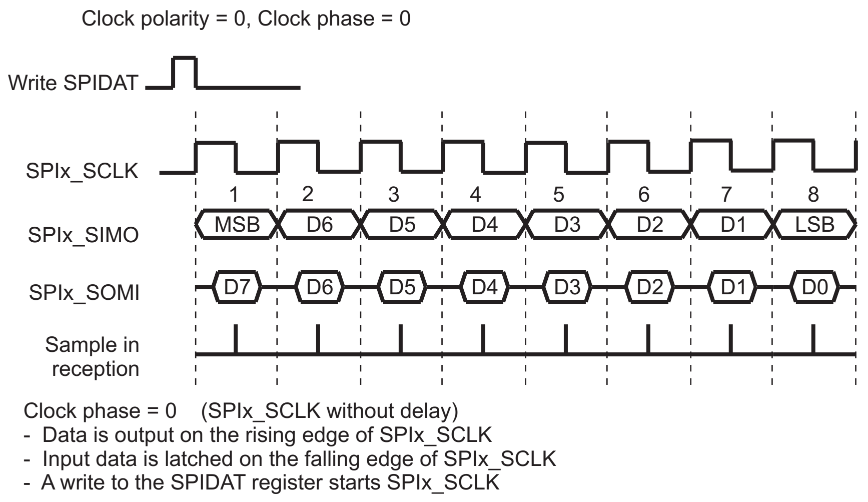
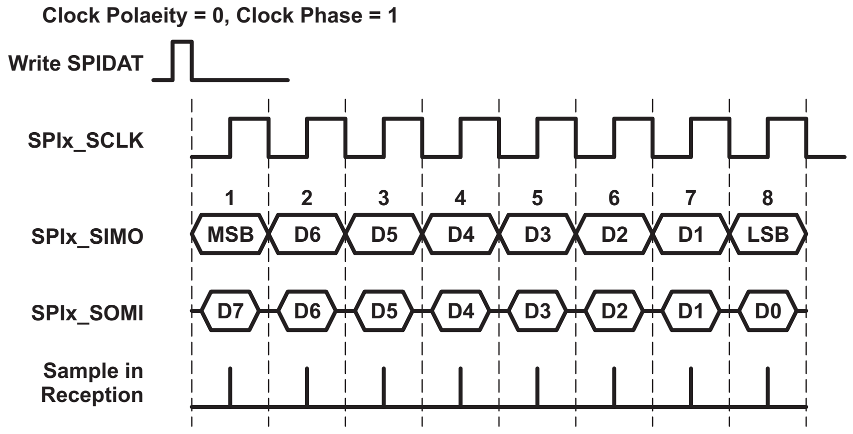
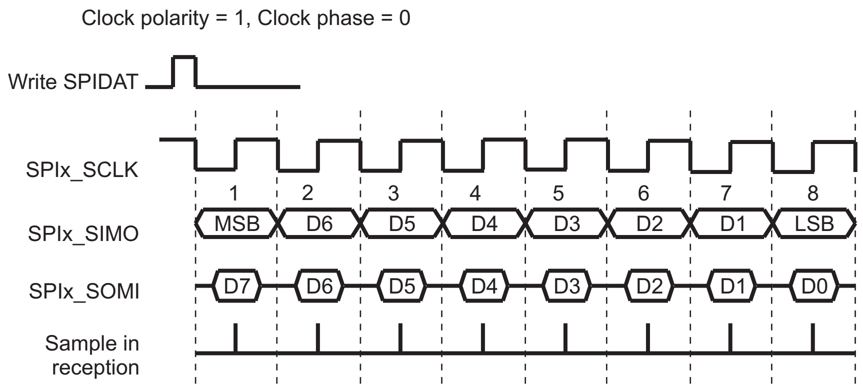

Clock polarity = 0, Clock phase = 0
Write SPIDAT
SPIx_SCLK
1
2
3
4
5
6
7
8
SPIx_SIMO
MSB
D6
D5
D4
D3
D2
D1
LSB
SPIx_SOMI
D7
D6
D5
D4
D3
D2
D1
D0
Sample in reception
Clock phase = 0 (SPIx_SCLK without delay)
- Data is output on the rising edge of SPIx_SCLK
- Input data is latched on the falling edge of SPIx_SCLK
- A write to the SPIDAT register starts SPIx_SCLK

Figure 5. Clock Mode with POLARITY=0 and PHASE=1



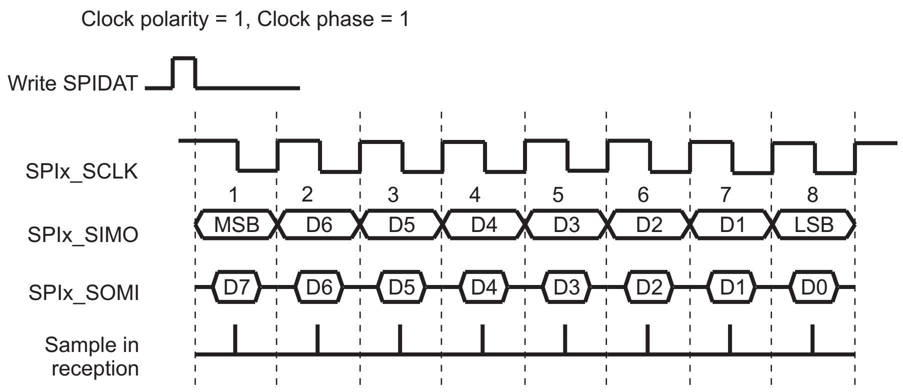
Clock phase = 1 (SPIx_SCLK with delay)
- Data is output one-half cycle before the first rising of SPIx_SCLK and on subsequent falling edges of SPIx_SCLK
- Input data is latched on the rising edge of SPIx_SCLK

Figure 6. Clock Mode with POLARITY=1 and PHASE=0



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

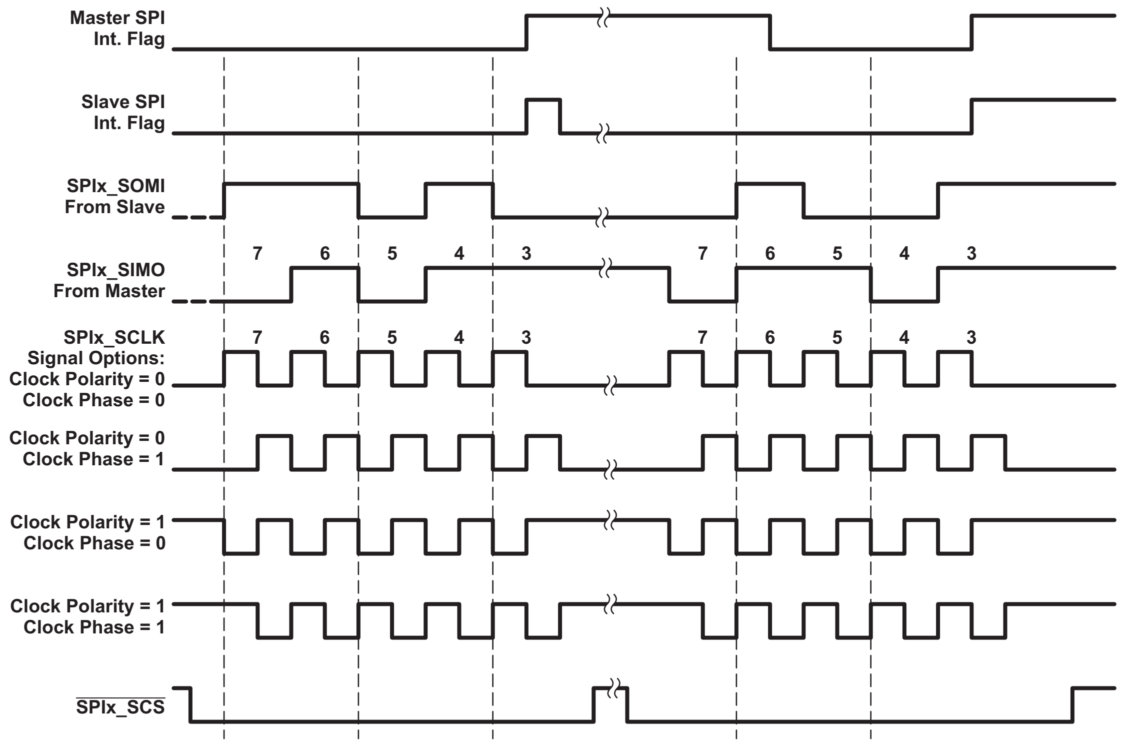
Figure 7. Clock Mode with POLARITY=1 and PHASE=1



Clock phase = 1 (SPIx_SCLK with delay)
- Data is output one-half cycle before the first falling edge
  of SPIx_SCLK and on subsequent rising edges of SPIx_SCLK
- Input data is latched on the falling edge of SPIx_SCLK

Figure 8 shows an example of an SPI data transfer between two devices using a character length of 5 bits and the different clock mode scenarios in 4-pin mode.

Figure 8. Five Bits per Character (Four-Pin Option)


Master SPI Int. Flag
Slave SPI Int. Flag
SPIx_SOMI From Slave
SPIx_SIMO From Master
7 6 5 4 3 7 6 5 4 3
SPIx_SCLK Signal Options: Clock Polarity = 0 Clock Phase = 0
7 6 5 4 3 7 6 5 4 3
Clock Polarity = 0 Clock Phase = 1
Clock Polarity = 1 Clock Phase = 0
Clock Polarity = 1 Clock Phase = 1
SPIx_SCS

2.4.3 Chip Select Control

The SPI provides two chip select signals ($\overline{\text{SPIx\_SCS[0]}}$ and $\overline{\text{SPIx\_SCS[1]}}$) that are used to selectively enable multiple slaves. The behavior of the chip selects is controlled by the CSNR and CSHOLD bits in SPIDAT1.

2.4.3.1 Enabling Chip Selects

The CSNR bit controls which chip selects are active during the transactions that follow. This bit is used to enable either or both chip selects. To use the chip selects, the SCS *n*FUN bit in the SPI pin control register 0(SPIPC0) must be set to 1 for each chip select.

2.4.3.2 Holding Chip Selects Active Between Transactions

Some SPI slave devices require that chip selects remain active between transactions, such as serial EEPROMs that use internal address counters, as long as the chip select is active. The CSHOLD bit controls whether chip selects remain asserted between transactions or not.

- When CSHOLD = 0, the chip selects are de-asserted between transactions.
- When CSHOLD = 1, the chip selects remain asserted between transactions as long as the chip select information (controlled by the CSNR bits in SPIDAT1) has not changed since the last transaction. If the chip select information is altered between transactions, the chip select is de-asserted even if CSHOLD = 1.

2.4.3.3 Programming Chip Select Setup and Hold Timing

The setup time between when the chip select signal goes active and the beginning of the transaction is programmable using the C2TDELAY bit in the SPI delay register (SPIDELAY). The setup time is [C2TDELAY + 2] cycles of the SPI peripheral clock (not the SPIx_SCLK).

The hold time between the end of the transaction and when the chip select signal goes inactive is programmable using the T2CDELAY bit in SPIDELAY. The hold time is [T2CDELAY + 1] cycles of the SPI peripheral clock (not the SPIx_SCLK).

If the CSHOLD function is active, the setup and hold delays are not applied between transactions where the chip select remains asserted.

2.4.3.4 Inactive Chip Select State Control

The driven state of the chip select pins when no transaction is in progress is controlled by the SCS *n*DEF bits in the SPI default chip select register (SPIDEF).

- When SCS *n*DEF = 0, the corresponding chip select is driven to logic 0 when no transaction is in progress.
- When SCS *n*DEF = 1, the corresponding chip select is driven to logic 1 when no transaction is in progress.

2.4.4 SPI Operation: 3-Pin Mode

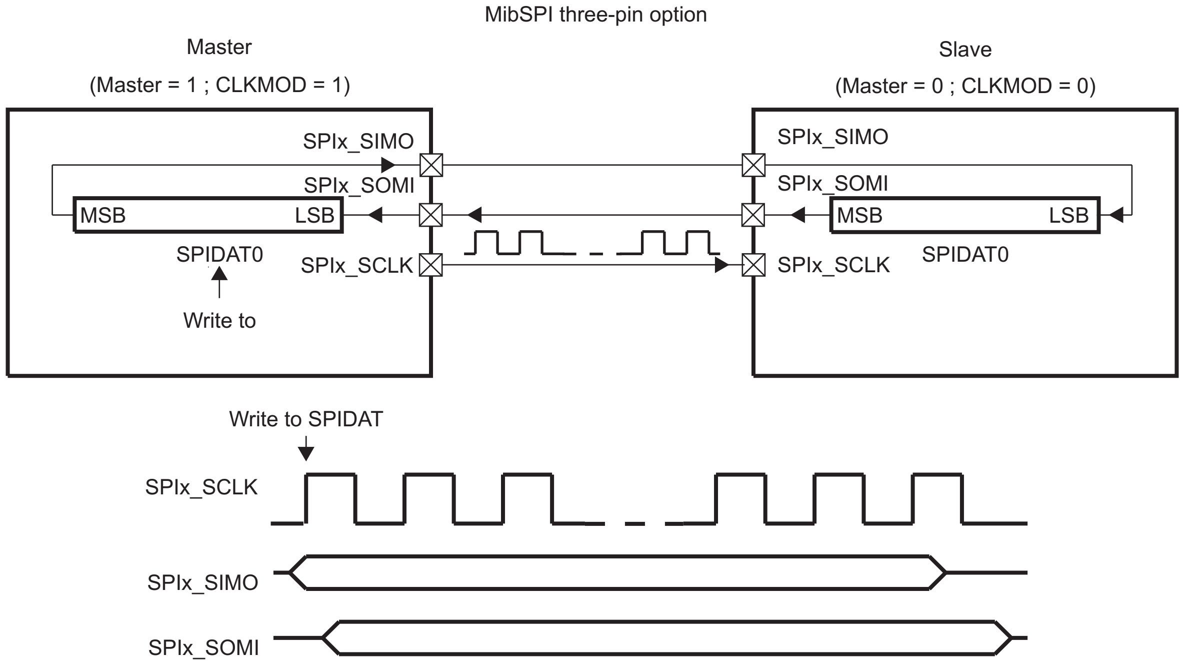
The minimum required number of signal connections for SPI communications is three pins: SPIx_SCLK, SPIx_SOMI, and SPIx_SIMO. The chip select pins ($\overline{\text{SPIx\_SCS[0]}}$ and $\overline{\text{SPIx\_SCS[1]}}$) are not used in 3-pin mode. This connection could be used when a single slave is present and, therefore, no chip selects are required. The SPI operates as a SPI master and provides the serial clock on the SPIx_SCLK pin during the current word transfer and stops between word transfers. Data is transmitted on the SPIx_SIMO pin and received from the SPIx_SOMI pin, as shown in Figure 9.

Right-aligned data written to the SPI shift register (SPIDAT1) initiates data transmission on the SPIx_SIMO pin. Simultaneously, received data is shifted through the SPIx_SOMI pin into the least-significant bit (LSB) of SPIDAT1. The SPI applies data format selected in the DFSEL bit of SPIDAT1 as the format for the transaction. When the selected number of bits have been transferred, the received data is copied to the SPI buffer register (SPIBUF) for the CPU or EDMA to read. Data is stored right-justified in SPIBUF.

When the specified number of bits is shifted through SPIDAT1, the following events occur:

- The receive interrupt flag (RXINTFLAG) bit in the SPI flag register (SPIFLG) is set to 1.
- The newly received SPIDAT1 contents transfer to SPIBUF.
- An interrupt is asserted, if the receive interrupt enable (RXINTEN) bit in the SPI interrupt register (SPIINT) is set to 1.

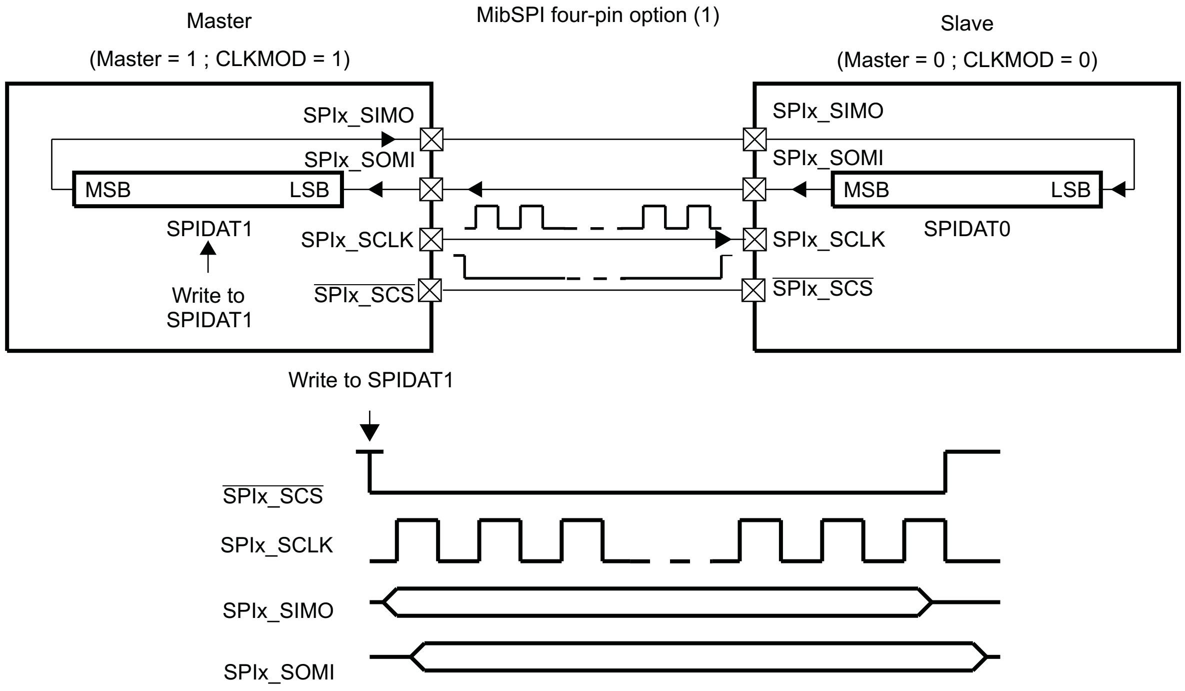
Figure 9. SPI Operation (3-Pin Option)



2.4.5 SPI Operation: 4-Pin Mode

The 3-pin mode and the 4-pin mode of the SPI are similar, except that the 4-pin mode uses the chip select pins ($\overline{\text{SPIx\_SCS[0]}}$ and $\overline{\text{SPIx\_SCS[1]}}$) to enable communication with multiple chip selects. Figure 10 shows how the 4-pin mode is connected. For detailed information on how the chip selects are controlled and used, see Section 2.4.3.

Figure 10. SPI Operation (4-Pin Option)



The $\overline{\text{SPIx\_SCS[1:0]}}$ pins that are going to be used must be configured as functional in the SPI pin control register 0 (SPIPC0). The default pattern to be put on the $\overline{\text{SPIx\_SCS[1:0]}}$ pins when all the slaves are deactivated is set in the SPI default chip select register (SPIDEF). This pattern allows different slaves with different chip-select polarity to be activated by the SPI. During transmission, the CSNR field of the SPI shift register 1 (SPIDAT1) is applied on the pins, this pattern will select the slave to which the transmission is dedicated. A Slave mode SPI can be selected only by an active value of 0 on any of its selected $\overline{\text{SPIx\_SCS[1:0]}}$ pins. However, Master SPI is capable of driving both 0 or 1 as the active value of any of $\overline{\text{SPIx\_SCS[1:0]}}$ pins. In master mode SPI, this is fully controlled by the CSNR field of SPIDAT1.

Note: During an SPI transfer, if the slave mode SPI, detects a de-assertion of its chip select even before its internal Character Length counter overflows, then it tristates its SPIx_SOMI pin. Once this condition has occurred, if a SPIx_SCLK edge is detected while the chip select is de-asserted, the SPI stops the transfer and sets an error flag DLENERR (data length) and generates an interrupt if enabled.

2.5 Reset Considerations

This section provides the software and hardware reset considerations.

2.5.1 Software Reset Considerations

In the event of an emulator software reset, the SPI module register values are not affected.

The SPI module contains a software reset (RESET) bit in the SPI global control register 0 (SPIGCR0) that is used to reset the SPI module. As a result of a reset, the SPI module register values go to their reset state. The RESET bit must be set before any operation on the SPI is done.

2.5.2 Hardware Reset Considerations

In the event of a hardware reset, the SPI module register values go to their reset state and the application software needs to reprogram the registers to the desired values.

There are two different methods to perform hardware reset that affects the SPI module register values. One is a full device hardware reset that resets all the device modules and the second is an individual peripheral hardware reset initiated by the Power and Sleep Controller (PSC) module. For information about the operation of the PSC, see the *TMS320DM365 Digital Media System-on-Chip (DMSoC) ARM Subsystem Reference Guide* (SPRUFG5).

2.6 Initialization

The following section provides procedures for initializing the SPI in 3-pin or 4-pin mode.

2.6.1 3-Pin Mode Initialization

1. Make sure the SPI module is in reset by clearing the RESET bit in the SPI global control register 0 (SPIGCR0) to 0.
2. Remove the SPI peripheral from reset by setting the RESET bit in SPIGCR0 to 1.
3. Enable the CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1) to set the SPI function in master or slave mode.
4. Enable the SPIx_SOMI, SPIx_SIMO, and SPIx_SCLK pins by setting the corresponding bits in the SPI pin control register (SPIPC0).
5. Configure the desired data format in the SPI data format register (SPIFMT*n*).
   (a) Program the clock prescale value in the PRESCALE*n* bit.
   (b) Program the character size in the CHARLEN*n* bit.
   (c) Set the SPI clock PHASE*n* and POLARITY*n* bits.
   (d) Set the shift direction in the SHIFTDIR*n* bit.
6. Select the pre-configured data format using the DFSEL bit in the SPI shift register (SPIDAT1).
7. Enable the desired interrupts, if any, in the SPI interrupt register (SPIINT).
8. Select whether you want the interrupt events mapped to SPINT0 or SPINT1 using the RXINTLVL bit in the SPI interrupt level register (SPILVL).
9. If using the EDMA to perform the transfers, setup and enable the EDMA channels for transmit or

receive.

10. Enable the SPIENA bit in SPIGCR1.
11. If using the EDMA, set the DMAREQEN bit in SPIINT to 1 initiating the EDMA to start writing to SPIDAT1; therefore, initiating the data transfer.
12. Data is ready to be transferred using the CPU by writing to SPIDAT1.

2.6.2 4-Pin Mode Initialization

1. Make sure the SPI module is in reset by clearing the RESET bit in the SPI global control register 0 (SPIGCR0) to 0.
2. Remove the SPI peripheral from reset by setting the RESET bit in SPIGCR0 to 1.
3. Enable the CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1) to set the SPI function in master or slave mode.
4. Enable the SPIx_SIMO, SPIx_SIMO, and SPIx_SCLK pins and the necessary chip select pins ($\overline{\text{SPIx\_SCS[0]}}$ and $\overline{\text{SPIx\_SCS[1]}}$) by setting the corresponding bits in the SPI pin control register (SPIPC0).
5. Configure the desired data format in the SPI data format register (SPIFMT*n*).
   (a) Program the clock prescale value in the PRESCALE*n* bit.
   (b) Program the character size in the CHARLEN*n* bit.
   (c) Set the SPI clock PHASE*n* and POLARITY*n* bits.
   (d) Set the shift direction in the SHIFTDIR*n* bit.
6. Select the pre-configured data format using the DFSEL bit in the SPI shift register (SPIDAT1).
7. If needed, configure the setup or hold time for the chip select lines using the C2TDELAY or T2CDELAY bits in the SPI delay register (SPIDELAY).
8. Select the desired chip select number. The CSNR field in SPIDAT1 defines the chip select that shall be activated during the data transfer. Note that the $\overline{\text{SPIx\_SCS[n]}}$ signals are active low.
9. Setup the default chip select pin value when chip select lines are inactive using the SCS *n*DEF bits in the SPI default chip select register (SPIDEF).
10. Enable the desired interrupts, if any, in the SPI interrupt register (SPIINT).
11. Select whether you want the interrupt events mapped to SPINT0 or SPINT1 using the RXINTLVL bit in the SPI interrupt level register (SPILVL).
12. If using the EDMA to perform the transfers, setup and enable the EDMA channels for transmit or receive.
13. Enable the SPIENA bit in SPIGCR1.
14. If using the EDMA, set the DMAREQEN bit in SPIINT to 1 initiating the EDMA to start writing to SPIDAT1; therefore, initiating the data transfer.
15. Data is ready to be transferred using the CPU by writing to SPIDAT1.

2.7 Interrupt Support

The SPI module outputs two interrupts that are routed to the ARM CPU interrupt controller, SPIINT0 and SPIINT1. Each of the interrupt events causes a CPU interrupt on SPIINT0 or SPIINT1. The SPI interrupt system is controlled by three registers:

- The SPI interrupt level register (SPILVL) controls which events (SPIINT0 or SPIINT1) are assigned to each interrupt.
- The SPI interrupt register (SPIINT) contains bits to selectively enable/disable each interrupt event.
- The SPI flag register (SPIFLG) contains flags indicating when each of the interrupt conditions have occurred.

Multiple interrupt sources can be assigned to the same CPU interrupt. To identify the interrupt source in the SPI peripheral, the CPU reads the SPI flag register (SPIFLG) or the INTVECT*n* code in the SPI interrupt vector register *n* (IINTVECT*n*).

2.7.1 Interrupt Events and Requests

2.7.1.1 Transmit Interrupt

The transmit interrupt occurs when a the transmit buffer (TXBUF) is empty and a new data can be written to it.

- To enable the SPI transmit interrupt, set the TXINTENA bit in the SPI interrupt register (SPIINT) to 1.
- To assign the transmit interrupt to occur on SPIINT0, clear the TXINTLVL bit in the SPI interrupt level register (SPILVL) to 0; to assign the transmit interrupt to occur on SPIINT1, set the TXINTLVL bit in SPILVL to 1

.

The occurrence of the receive interrupt is recorded in the TXINTFLAG bit in the SPI flag register (SPIFLG). This flag is cleared by writing new data to SPIDAT1, writing a 0 to the SPIEN bit in SPIGCR1.

2.7.1.2 Receive Interrupt

The receive interrupt occurs when a data character has been received and copied in the SPI buffer register (SPIBUF).

- To enable the SPI receive interrupt, set the RXINTEN bit in the SPI interrupt register (SPIINT) to 1.
- To assign the receive interrupt to occur on SPIINT0, clear the RXINTLVL bit in the SPI interrupt level register (SPILVL) to 0; to assign the receive interrupt to occur on SPIINT1, set the RXINTLVL bit in SPILVL to 1.

The occurrence of the receive interrupt is recorded in the RXINTFLAG bit in the SPI flag register (SPIFLG). This flag is cleared by reading SPIBUF, writing a 1 to the RXINTFLAG bit, disabling the receive interrupt, or a system reset.

2.7.1.3 Receive Overrun Interrupt

The receive overrun interrupt occurs when a data character has been received in the shift register before the previous character has been read from the SPI buffer register (SPIBUF). To enable the SPI receive overrun interrupt, set the OVRNINTEN bit in the SPI interrupt register (SPIINT) to 1. To assign the receive overrun interrupt to occur on SPIINT0, clear the OVRNINTLVL bit in the SPI interrupt level register (SPILVL) to 0; to assign the receive overrun interrupt to occur on SPIINT1, set the OVRNINTLVL bit in SPILVL to 1.

The occurrence of the receive overrun interrupt is recorded in the OVRNINTFLAG bit in the SPI flag register (SPIFLG). This flag is cleared by writing a 1 to the OVRNINTFLAG bit, disabling the receive overrun interrupt, or a system reset. Reading SPIBUF does not clear the OVRNINTFLAG bit.

2.7.1.4 Transmit Error Interrupt

The transmit error interrupt occurs when the internal data transmitted does not match the external data bits sensed on the SPIx_SIMO signal. This error is used as an indication of a fault on the SPIx_SIMO line. To enable the SPI transmit error interrupt, set the BITERRENA bit in the SPI interrupt register (SPIINT) to 1. To assign the transmit error interrupt to occur on SPIINT0, clear the BITERRLVL bit in the SPI interrupt level register (SPILVL) to 0; to assign the transmit error interrupt to occur on SPIINT1, set the BITERRLVL bit in SPILVL to 1.

The occurrence of the transmit error interrupt is recorded in the BITERRFLG bit in the SPI flag register (SPIFLG).

2.7.2 Interrupt Multiplexing

The total number of interrupts in the device exceeds 64, which is the maximum number of interrupts supported by the ARM Interrupt Controller (AINTC) module. Therefore, several interrupts are multiplexed, and you must use the register ARM_INTMUX in the System Control Module to select the interrupt source for multiplexed interrupts. Some of the SPI peripheral interrupts are multiplexed. For more information on the System Control Module and ARM Interrupt Controller, refer to the *TMS320DM365 Digital Media System-on-Chip (DMSoC) ARM Subsystem Reference Guide* (SPRUFG5).

2.8 EDMA Event Support

If handling the SPI message traffic on a character-by-character basis requires too much CPU overhead, the SPI may use the system EDMA to receive or transmit data directly to or from memory.

The SPI module has two EDMA synchronization event outputs that go to the system EDMA, allowing EDMA transfers to be triggered by SPI read receive or write transmit events. SPIXEVT is a transmit sync event; SPIREVT is a receive sync event. The SPI module enables EDMA requests by enabling the DMA request enable (DMAREQEN) bit in the SPI interrupt register (SPIINT).

When a character is being transmitted or received, the SPI signals the EDMA via the EDMA synchronization event signal. The EDMA controller then performs the needed data manipulation. EDMA transfers the data from the source programmed into the SPI shift register (SPIDAT1). Data is then read from the SPI buffer register (SPIBUF), which automatically clears the RXINTFLAG bit in the SPI flag register (SPIFLG).

In most cases, if the EDMA is being used to service received data from the SPI, the receive interrupt enable (RXINTEN) bit in SPIINT should be cleared to 0. This prevents the CPU from both responding to the received data in addition to the EDMA. For specific SPI synchronization event number and detailed EDMA features, refer to the *TMS320DM36x DMSoC Enhanced Direct Memory Access (EDMA) Controller Users Guide* (SPRUFI0) .

The total number of EDMA events in DM36x exceeds 64, which is the maximum number of events supported by the EDMA module. Therefore, several events are multiplexed, and you must use the register EDMA_EVTMUX in the System Control Module to select the event source for multiplexed events. Some of the SPI peripheral events are multiplexed. Refer to the *TMS320DM365 DMSoC ARM Subsystem Reference Guide* (SPRUFG5) for more information on the System Control Module

2.9 Power Management

The SPI can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the SPI is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *TMS320DM365 Digital Media System-on-Chip (DMSoC) ARM Subsystem Reference Guide* (SPRUFG5).

Local low-power mode is asserted by setting the POWERDOWN bit in the SPI global control register 1 (SPIGCR1). Setting this bit stops the clocks to the SPI internal logic and the SPI registers. Setting the POWERDOWN bit causes the SPI to enter local low-power mode and clearing the POWERDOWN bit causes SPI to exit from local low-power mode. All the registers are accessible during local power-down mode as any register access enables the clock to SPI for that particular access alone. This may cause activities on the SPI (if in MASTER mode) functional pins if data was written to the SHIFT register before entering the Low power mode.

Since entering a low-power mode has the effect of suspending all state machine activities, care must be taken when entering such modes to ensure that a valid state is entered when low-power mode is active. As a result, application software must ensure that a low-power mode is not entered during a transmission or reception of data.

2.10 SPI Internal Loop-Back Test Mode

CAUTION

The internal loop-back self-test mode should not be entered during a normal data transaction or unpredictable operation may occur.

The internal loop-back self-test mode can be utilized to test the SPI transmit path and receive path. In this mode, the transmit signal is internally fed back to the receiver and the SPIx_SIMO, SPIx_SOMI, and SPIx_SCLK pins are disconnected. For example, the transmitted data is internally transferred to the corresponding receive buffer while external signals remain unchanged. This mode allows the CPU 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. This capability can be useful during code development and debug. The loop-back test mode is enabled by setting the LOOPBACK bit in the SPI global control register 1 (SPIGCR1) to 1.

2.11 Emulation Considerations

The SPI module does not support soft or hard stop during emulation breakpoints. The SPI module will continue to run if an emulation breakpoint is encountered.

During debug, read the SPI emulation register (SPIEMU) if you need to read the received data without otherwise altering the state of the SPI peripheral. Reading the SPI buffer register (SPIBUF) causes the flags in SPIBUF to be cleared; reading SPIEMU will read the receive data without altering the flags in SPIBUF.

3 Registers

Table 4 lists the memory-mapped registers for the SPI. See the device-specific data manual for the memory address of these registers. All other register offset addresses not listed in Table 4 should be considered as reserved locations and the register contents should not be modified.

Table 4. SPI Registers

Offset	Acronym	Register Description	Section
00h	SPIGCR0	SPI global control register 0	Section 3.1
04h	SPIGCR1	SPI global control register 1	Section 3.2
08h	SPIINT	SPI interrupt register	Section 3.3
0Ch	SPILVL	SPI interrupt level register	Section 3.4
10h	SPIFLG	SPI flag register	Section 3.5
14h	SPIPC0	SPI pin control register	Section 3.6
1Ch	SPIPC2	SPI pin control register 2	Section 3.7
3Ch	SPIDAT1	SPI shift register	Section 3.8
40h	SPIBUF	SPI buffer register	Section 3.9
44h	SPIEMU	SPI emulation register	Section 3.10
48h	SPIDELAY	SPI delay register	Section 3.11
4Ch	SPIDEF	SPI default chip select register	Section 3.12
50h-5Ch	SPIFMT0	SPI data format register 0	Section 3.13
60h	INTVECT0	SPI interrupt vector register 0	Section 3.14
64h	INTVECT1	SPI interrupt vector register 1	Section 3.15

3.1 SPI Global Control Register 0 (SPIGCR0)

The SPI global control register 0 (SPIGCR0) is shown in Figure 11 and described in Table 5.

Figure 11. SPI Global Control Register 0 (SPIGCR0)

31–16
Reserved
R-0

15–1	0
Reserved	RESET
R-0	R/W-0

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

Table 5. SPI Global Control Register 0 (SPIGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
0	RESET		Reset bit for the SPI module. RESET must be set before any operation on SPI is done.
		0	SPI is in reset state.
		1	SPI is out of reset state.

3.2 SPI Global Control Register 1 (SPIGCR1)

The SPI global control register 1 (SPIGCR1) is shown in Figure 12 and described in Table 6.

Figure 12. SPI Global Control Register 1 (SPIGCR1)

31 – 25	24	23 – 17	16
Reserved	SPIEN	Reserved	LOOPBACK
R-0	R/W-0	R-0	R/WP-0

15 – 9	8	7 – 2	1	0
Reserved	POWERDOWN	Reserved	CLKMOD	MASTER
R-0	R/W-0	R-0	R/W-0	R/W-0

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

Table 6. SPI Global Control Register 1 (SPIGCR1) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Any writes to these bit(s) must always have a value of 0.
24	SPIEN		SPI enable. Enables the SPI transfers. This bit must be set to 1 after all other SPI configuration bits have been written. When SPIEN bit is 0 or cleared to 0, the following SPI registers get forced to their default states (0s except for the RXEMPTY bit in SPIBUF). • Both Tx and Rx shift registers • The TXDATA field of SPIDAT1 • All the fields of SPIFLG • Contents of SPIBUF and the internal RXBUF
		0	SPI is not activated for transfers
		1	Activates SPI.
23-17	Reserved	0	Any writes to these bit(s) must always have a value of 0.
16	LOOPBACK		Internal loop-back test mode. The internal self-test option is 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, the SPIx_SCLK pin outputs an inactive value and SPIx_SOMI remains in a high-impedance state. The SPI must be initialized in master mode before the loop-back is selected. If the SPI is initialized in slave mode or a data transfer is ongoing, errors may result. **Note:** This loopback mode can be used only in master mode. This automatically selects digital loopback path. When the loopback mode is selected, CLKMOD bit should be set to 1, (i.e., SPIx_SCLK can only be internal).
		0	Internal loop-back test mode is disabled.
		1	Internal loop-back test mode is enabled.
15-9	Reserved		Any writes to these bit(s) must always have a value of 0.
8	POWERDOWN		When active, the SPI state machines enter a power-down state.
		0	SPI is in active mode.
		1	SPI is in power-down mode.
7-2	Reserved		Any writes to these bit(s) must always have a value of 0.
1	CLKMOD		Clock mode. This bit selects either an internal or external clock source. This bit also determines the I/O direction of the $\overline{\text{SPIx\_SCS[1:0]}}$ pins in functional mode.
		0	Clock is external
		1	Clock is internal

Table 6. SPI Global Control Register 1 (SPIGCR1) Field Descriptions (continued)

Bit	Field	Value	Description
0	MASTER		SPIx_SIMO pin direction. Determines the direction of the SPIx_SIMO and SPIx_SOMI pins. **Note:** For master mode operation of SPI, MASTER bit should be set to 1 and CLKMOD bit can be set either to 1 or 0. This means that even master mode SPI can run on external clock on SPIx_SCLK. However, for slave mode operation, both MASTER and CLKMOD bits must be cleared to 0. Any other combinations could result in unpredictable behavior of the SPI. In slave mode, SPIx_SCLK will not be generated internally.
		0	SPIx_SIMO pin is an input; SPIx_SOMI pin is an output
		1	SPIx_SOMI is an input; SPIx_SIMO pin is an output

3.3 SPI Interrupt Register (SPIINT)

The SPI interrupt register (SPIINT) is shown in Figure 13 and described in Table 7.

Figure 13. SPI Interrupt Register (SPIINT)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved							DMAREQEN
R-0							R/W-0

15	14	13	12	11	10	9	8
Reserved						TXINTENA	RXINTENA
R-0						R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved	OVRNINTENA	Reserved	BITERRENA	Reserved	PARERRENA	Reserved	
R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	

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

Table 7. SPI Interrupt Register (SPIINT) Field Descriptions

Bit	Field	Value	Description
31-17	Reserved	0	Any writes to these bit(s) must always have a value of 0.
16	DMAREQEN		DMA request enable. Enables the DMA request signal to be generated for both receive and transmit channels. When using the SPI module with the EDMA, it is important that the related EDMA channels are configured and enabled before enabling this bit.
		0	DMA is not used.
		1	DMA is used.
15-10	Reserved	0	Any writes to these bit(s) must always have a value of 0.
9	TXINTENA		An interrupt is to be generated every time data is written to the shift register, so that a new data can be written to TXBUF. Setting this bit will generate an interrupt if the TXINTFLG bit in SPIFLG is set to 1.
		0	No interrupt will be generated upon TXINTFLG being set to 1.
		1	Interrupt will be generated upon TXINTFLG being set to 1. Note: An interrupt request will be generated as soon as this bit is set to 1. By default it will be generated on SPINT0 line. SPILVL can be programmed beforehand to change this default.
8	RXINTENA		Receive interrupt enable. An interrupt is generated when the RXINTFLAG bit in the SPI flag register (SPIFLG) is set by hardware; otherwise, no interrupt is generated.
		0	Interrupt is not generated.
		1	Interrupt is generated.
7	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
6	OVRNINTENA		Overrun interrupt enable. An interrupt is generated when the OVRNINTFLG bit in the SPI flag register (SPIFLG) is set by hardware; otherwise, no interrupt is generated.
		0	Overrun interrupt is not generated.
		1	Overrun interrupt is generated.
5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4	BITERRENA		Enables interrupt on bit error.
		0	No interrupt asserted upon bit error.
		1	Enables an interrupt on a bit error (BITERR = 1).
3	Reserved		Any writes to these bit(s) must always have a value of 0.
2	PARERRENA		Enables interrupt on parity error.
		0	No interrupt asserted upon parity error.
		1	Enables an interrupt on a parity error (PARITYERR = 1).

Table 7. SPI Interrupt Register (SPIINT) Field Descriptions (continued)

Bit	Field	Value	Description
1-0	Reserved		Any writes to these bit(s) must always have a value of 0.

3.4 SPI Interrupt Level Register (SPILVL)

The SPI interrupt level register (SPILVL) is shown in Figure 14 and described in Table 8.

Figure 14. SPI Interrupt Level Register (SPILVL)

31–16
Reserved
R-0

15–10	9	8	7	6	5	4	3	2	1–0
Reserved	TXIN TLVL	RXINTLVL	Rsvd	OVRNINTLVL	Rsvd	BITERRLVL	Rese rved	PAR ERR LVL	Reserved
R-0	R/W-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0	R/W-0	R-0

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

Table 8. SPI Interrupt Level Register (SPILVL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Any writes to these bit(s) must always have a value of 0.
9	TXINTLVL		Transmit interrupt level.
		0	Transmit interrupt is mapped to interrupt line INT0.
		1	Transmit interrupt is mapped to interrupt line INT1.
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	0	Any writes to these bit(s) must always have a value of 0.
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	0	Aby writes to these bit(s) must always have a value of 0.
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	Reserved	0	Any writes to these bit(s) must always have a value of 0.
2	PARERRLVL		Parity error interrupt level.
		0	A parity error interrupt (PARITYERR = 1) is mapped to interrupt line SPINT0.
		1	A parity error interrupt (PARITYERR = 1) is mapped to interrupt line SPINT1.
1-0	Reserved		Any writes to these bit(s) must always have a value of 0.

3.5 SPI Flag Register (SPIFLG)

The SPI flag register (SPIFLG) is shown in Figure 15 and described in Table 9.

Figure 15. SPI Flag Register (SPIFLG)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved							
R-0							

15	14	13	12	11	10	9	8
Reserved						TXINTFLAG	RXINTFLAG
R-0						R-0	R/W1C-0

7	6	5	4	3	2	1	0
Reserved	OVRNINTFLG	Reserved	BITERRFLG	Reserved	PARERRFLG	Reserved	
R-0	R/W1C-0	R-0	RC-0	R-0	R/W1C-0	R-0	

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

Table 9. SPI Flag Register (SPIFLG) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved		Any writes to these bit(s) must always have a value of 0.
9	TXINTFLAG		Transmitter empty interrupt flag. Serves as an interrupt flag indicating that the transmit buffer (TXBUF) is empty and a new data can be written to it. This flag is set when a data is copied to the shift register either directly or from TXBUF. This bit is cleared by one of following ways: Writing a new data to SPIDAT1. = Writing a 0 to SPIEN bit in SPIGCR1.
		0	Transmit buffer is now full. No interrupt pending for transmitter empty.
		1	Transmit buffer is empty. An interrupt is pending to fill the transmitter.
8	RXINTFLAG		Receive interrupt flag. RXINTFLAG is set when a word is received and copied into the SPI buffer register (SPIBUF). If the RXINTEN bit in the SPI interrupt register (SPIINT) is set to 1 (enabled), an interrupt is also generated. During emulation mode, a read of the SPI emulation register (SPIEMU) does not clear RXINTFLAG. This bit is cleared by: • Reading SPIBUF • Reading INTVEC0 or INTVEC1 when there is a receive buffer full interrupt. • Writing a 1 to this bit • Writing a 0 to the SPIENA bit in the SPI global control register 1 (SPIGCR1) • System reset
		0	Interrupt condition did not occur.
		1	Interrupt condition did occur. **Note:** Clearing RXINTFLG bit by writing a 1 before reading SPIBUF sets the RXEMPTY bit in SPIBUF. This way, one can ignore a received data. However, if the internal RXBUF is already full, the data from RXBUF will be copied to SPIBUF and the RXEMPTY bit will be cleared again. The SPIBUF contents should be read first if this situation needs to be avoided.
7	Reserved	0	Any writes to these bit(s) must always have a value of 0.

Table 9. SPI Flag Register (SPIFLG) Field Descriptions (continued)

Bit	Field	Value	Description
6	OVRNINTFLG		Receiver overrun flag. OVRNINTFLG is set by the SPI hardware when an operation completes before the previous character has been read from the buffer. OVRNINTFLG bit indicates that the last received character has been overwritten and therefore lost. If the OVRNINTEN bit in the SPI interrupt register (SPIINT) is set to 1 (enabled), an interrupt is also generated. This bit is cleared by: • Reading INTVECT0 or INTVECT1 • Writing a 1 to this bit Reading the SPIBUF register does not clear this bit.
		0	Overrun condition did not occur.
		1	Overrun condition did occur. **Note:** Reading SPIBUF does not clear this OVRNINTFLG bit. If an RXOVRN interrupt is detected, then SPIBUF may need to be read twice to get to the overrun buffer. This is due to the fact that the overrun will always occur to the internal RXBUF. Each read to the SPIBUF will result in RXBUF contents (if it is full) getting copied to SPIBUF. A special condition under which OVRNINTFLG flag gets set. If both SPIBUF and RXBUF are already full and while another buffer receive is underway, if any errors like BITERR occur, then RXOVR in RXBUF and OVRNINTFLG in SPIFLG will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receiver overrun.
5	Reserved	0	Any writes to these bit(s) must always have a value of 0.
4	BITERRFLG		Bit error flag. The SPI samples the signal of the transmit pin (master: SPIx_SIMO) at the receive point (half clock cycle after transmit point). If the sampled value differs from the transmitted value, a bit error is detected and the BITERRFLG bit is set and the BITERR bit in the SPI buffer register (SPIBUF) is set. Possible reasons for a bit error are a high bit rate/capacitive load or another master/slave trying to transmit at the same time. If the BITERRENA bit in the SPI interrupt register (SPIINT) is set to 1 (enabled) and the BITERR bit (in SPIBUF) is set to 1, an interrupt is also generated. This bit is cleared by reading the bit.
		0	Bit error did not occur.
		1	Bit error did occur.
3	Reserved		Any writes to these bit(s) must always have a value of 0.
2	PARERRFLG		Calculated parity differs from received parity bit.
		0	No parity error detected. This flag can be cleared by one of the following ways: • Write a 1 to this bit. • Clear SPIEN bit in SPIGCR1 to 0
		1	A parity error occurred. If the parity generator is enabled (can be selected individually for each buffer) an even or odd parity bit is added at the end of a data word. During reception of the data word the parity generator calculates the reference parity and compares it to the received parity bit. In the event of a mismatch the PARITYERR flag is set and an interrupt is asserted if PARERRENA is set.
1-0	Reserved		Any writes to these bit(s) must always have a value of 0.

3.6 SPI Pin Control Register (SPIPC0)

The SPI pin control register (SPIPC0) is shown in Figure 16 and described in Table 10.

Figure 16. SPI Pin Control Register (SPIPC0)

31–16
Reserved
R-0

15–12	11	10	9	8–2	1	0
Reserved	SOMIFUN	SIMOFUN	SCLKFUN	Reserved	SCS1FUN	SCS0FUN
R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0

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

Table 10. SPI Pin Control Register (SPIPC0) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
11	SOMIFUN		SPI data input (SPIx_SOMI) functional pin. This bit must be set for the SPI module to operate.
		0	Reserved
		1	SPIx_SOMI is a functional pin.
10	SIMOFUN		SPI data output (SPIx_SIMO) functional pin. This bit must be set for the SPI module to operate.
		0	Reserved
		1	SPIx_SIMO is a functional pin.
9	SCLKFUN		SPI clock (SPIx_SCLK) functional pin. This bit must be set for the SPI module to operate.
		0	Reserved
		1	SPIx_SCLK is a functional pin.
8-2	Reserved	0	Reserved. The reserved bit location is always read as 0. This field must be written with zeroes.
1	SCS1FUN		SPI slave 1 ($\overline{\text{SPIx\_SCS[1]}}$) functional pin. This bit must be set for the SPI module to communicate with slave SPI devices.
		0	Reserved
		1	$\overline{\text{SPIx\_SCS[1]}}$ is a functional pin.
0	SCSFUN		SPI slave 0 ($\overline{\text{SPIx\_SCS[0]}}$) functional pin. This bit must be set for the SPI module to communicate with slave SPI devices.
		0	Reserved
		1	$\overline{\text{SPIx\_SCS[0]}}$ is a functional pin.

3.7 SPI Pin Control Register 2 (SPIPC2)

The SPI pin control register 2 (SPIPC2) is shown in Figure 17 and described in Table 11.

Figure 17. SPI Pin Control Register 2 (SPIPC2)

31–16
Reserved
R-0

15–12	11	10	9	8–2	1	0
Reserved	SOMIDIN	SIMODIN	SCLKDIN	Reserved	SCS1DIN	SCS0DIN
R-0	R-x	R-x	R-x	R-0	R-x	R-x

LEGEND: R = Read only; -*n* = value after reset; -x = value is indeterminate after reset

Table 11. SPI Pin Control Register 2 (SPIPC2) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Any writes to these bit(s) must always have a value of 0.
11	SOMIDIN		SPI data input (SPIx_SOMI) pin value. This bit reflects the value on the SPIx_SOMI pin.
		0	Current value on pin is SPIx_SOMI logic 0.
		1	Current value on SPIx_SOMI pin is logic 1.
10	SIMODIN		SPI data output (SPIx_SIMO) pin value. This bit reflects the value 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	SCLKDIN		SPI clock (SPIx_SCLK) pin value. This bit reflects the value on the SPIx_SCLK pin.
		0	Current value on SPIx_SCLK pin is logic 0.
		1	Current value on SPIx_SCLK pin is logic 1.
8-2	Reserved	0	Any writes to these bit(s) must always have a value of 0.
1	SCS1DIN		SPI slave 1 ($\overline{\text{SPIx\_SCS[1]}}$) pin value. This bit reflects the value on the $\overline{\text{SPIx\_SCS[1]}}$ pin.
		0	Current value on $\overline{\text{SPIx\_SCS[1]}}$ pin is logic 0.
		1	Current value on $\overline{\text{SPIx\_SCS[1]}}$ pin is logic 1.
0	SCS0DIN		SPI slave 0 ($\overline{\text{SPIx\_SCS[0]}}$) pin value. This bit reflects the value on the $\overline{\text{SPIx\_SCS[0]}}$ pin.
		0	Current value on $\overline{\text{SPIx\_SCS[0]}}$ pin is logic 0.
		1	Current value on $\overline{\text{SPIx\_SCS[0]}}$ pin is logic 1.

3.8 SPI Shift Register (SPIDAT1)

The SPI shift register (SPIDAT1) is shown in Figure 18 and described in Table 12.

Figure 18. SPI Shift Register (SPIDAT1)

31–29	28	27	26	25–24	23–18	17–16
Reserved	CSHOLD	Reserved	WDEL	DFSEL	Reserved	CSNR
R-0	R/W-0	R-0	R/W-0	R/W-0	R-0	R/W-0

15–0
SPIDAT1
R/W-0

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

Table 12. SPI Shift Register (SPIDAT1) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
28	CSHOLD		Chip select hold mode. CSHOLD is considered in 4-pin mode only. CSHOLD defines the behavior of the chip select line at the end of a transfer.
		0	Chip select signal is deactivated at the end of a transfer after the T2CDELAY time has passed. If two consecutive transfers are dedicated to the same chip select, this chip select signal is shortly deactivated before it is activated again.
		1	Chip select signal is held active at the end of a transfer until a control field with new data and control information is loaded into the SPIDAT1 bit. If the new chip select information equals the previous information, the active chip select signal is extended until the end of transfer with CSHOLD cleared or until the chip select information changes.
27	Reserved	0	Reserved. The reserved bit location is always read as 0. This field must be written with zeroes.
26	WDEL		Enable the delay counter at the end of the current transaction. **Note:** The WDEL bit is supported in master mode only. In slave mode, this bit will be ignored.
		0	No delay will be inserted. However, $\overline{\text{SPIx\_SCS[1:0]}}$ pins will still be de-activated for at least for two SPI peripheral clock cycles if CSHOLD = 0. **Note:** In SPI, the duration for which the $\overline{\text{SPIx\_SCS[1:0]}}$ pins remaining de-activated will also depend upon time taken to supply a new data after completing the shifting operation. If the internal buffer TXBUF is already full, then the $\overline{\text{SPIx\_SCS[1:0]}}$ pins will be de-asserted for at least two SPI peripheral clock cycles (if WDEL = 0).
		1	After a transaction, WDELAY of the corresponding data format will be loaded into the delay counter. No transaction will be performed until the WDELAY counter overflows. The $\overline{\text{SPIx\_SCS}}$ pins will be de-activated for at least (WDELAY + 2) SPI peripheral clock_Period duration.
25-24	DFSEL	0-3h	Data word format select.
		0	Data format 0 is selected.
		1h	Data format 1 is selected.
		2h	Data format 2 is selected.
		3h	Data format 3 is selected.
23-18	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
17-16	CSNR	0-3h	Chip select number. CSNR defines the chip select that shall be activated during the data transfer. Note that the $\overline{\text{SPIx\_SCS}}$ signals are active low.
		0	Both chip select $\overline{\text{SPIx\_SCS[0]}}$ and $\overline{\text{SPIx\_SCS[1]}}$ are selected.
		1h	Chip select $\overline{\text{SPIx\_SCS[1]}}$ is selected only.
		2h	Chip select $\overline{\text{SPIx\_SCS[0]}}$ is selected only.
		3h	No chip select is used.

Table 12. SPI Shift Register (SPIDAT1) Field Descriptions (continued)

Bit	Field	Value	Description
15-0	SPIDAT1	0-FFFFh	Transfer data. When written, these bits will be copied to the shift register if it is empty. If the shift register is not empty, the TXBUF will hold the written values. SPIEN bit in SPIGCR1 must be set to 1 before this register can be written to. Writing a 0 to the SPIEN bit forces the lower 16 bits of SPIDAT1 to 0. A write to this field drives the $\overline{\text{SPIx\_SCS [1:0]}}$ signal low if the $\overline{\text{SPIx\_SCS [1:0]}}$ signals are enabled in the SPI pin control register (SPIPC0). When this register is read, the contents of internal buffer register TXBUF which holds the latest written data will be returned. **Note:** Irrespective of the character length, the transmit data should be right-justified before writing to SPIDAT1.

3.9 SPI Buffer Register (SPIBUF)

The SPI buffer register (SPIBUF) is shown in Figure 19 and described in Table 13. Reading SPIBUF clears the OVRNINTFLG and RXINTFLAG bits in the SPI flag register (SPIFLG).

Figure 19. SPI Buffer Register (SPIBUF)

31	30	29	28	27	26	25	24
RXEMPTY	RXOVR	TXFULL	BITERR	Reserved	PARITYERR	Reserved	
R-1	RC-0	R-0	RC-0	R-0	RC-0	R-0	

23	22	21	20	19	18	17	16
Reserved						LCSNR	
R-0						R-0	

15	14	13	12	11	10	9	8
SPIBUF							
R-x							

7	6	5	4	3	2	1	0
SPIBUF							
R-x							

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

Table 13. SPI Buffer Register (SPIBUF) Field Descriptions

Bit	Field	Value	Description
31	RXEMPTY		Receive data buffer empty. This is a read-only flag. When the host reads an SPIBUF field or the entire SPIBUF, this automatically sets the RXEMPTY bit. When a data transfer has been finished and the received data is copied into SPIBUF, the RXEMPTY bit is cleared.
		0	Data is received and copied into an SPIBUF field.
		1	No data received since last reading SPIBUF. This flag gets set to 1 under following conditions: • Reading the RXDATA field of SPIBUF. • Writing 1 to clear the RXINTFLG bit in SPIFLG. Write-Clearing the RXINTFLG bit before reading SPIBUF indicates the received data is being ignored. Conversely, RXINTFLG can be cleared by reading the RXDATA field of SPIBUF. So, reading the entire SPIBUF clears the receiver full interrupt flag.
30	RXOVR		Receive data buffer overrun. When a data transfer is completed and the received data is copied into the RXBUF while it is already full, RXOVR is set. An overrun always occurs to the RXBUF, and SPIBUF contents never get overwritten until after it is read by CPU/EDMA. If enabled, RXOVRN interrupt gets generated when RXBUF is overwritten, and reading SPIFLG or INTVECn shows the RXOVRN condition. However, two read operations to SPIBUF are required to reach the overrun buffer. This flag is cleared to 0 when the RXDATA field of SPIBUF is read. **Note:** A special condition under which RXOVR flag gets set. If both SPIBUF and RXBUF are already full and while another buffer receive is underway, if any errors like TIMEOUT, BITERR, and DLEN_ERR occur, then RXOVR in RXBUF and SPIFLG will be set to indicate that the status flags are getting overwritten by the new transfer. This overrun should be treated like a normal receiver overrun.
		0	No receive data overrun condition occurred since last time reading the data field.
		1	A receive data overrun condition occurred since last time reading the data field.
29	TXFULL		Transmit data buffer full. This flag is a read-only flag. Writing into SPIDAT1 field while the Tx shift register is full will automatically set the TXFULL flag. Once the data is copied to the shift register, the TXFULL flag will be cleared. Writing to SPIDAT1 when both TXBUF and the Tx shift register are empty does not set the TXFULL flag.
		0	The transmit buffer is empty; SPIDAT1 is ready to accept a new data.
		1	The transmit buffer is full; SPIDAT1 is not ready to accept new data.

Table 13. SPI Buffer Register (SPIBUF) Field Descriptions (continued)

Bit	Field	Value	Description
28	BITERR		Bit error. There was a mismatch of internal transmit data and transmitted data.
		0	No bit error occurred. **Note:** This flag is cleared to 0 when RXDATA field of SPIBUF is read.
		1	1 A bit error occurred. The SPI samples the signal of the transmit pin (master: SPIx_SIMO, slave: SPIx_SOMI) at the receive point (half clock cycle after transmit point). If the sampled value differs from the transmitted value, a bit error is detected and the flag BITERR is set. A possible reason for a bit error can be noise, a too-high bit rate/capacitive load, or another master/slave trying to transmit at the same time.
27	Reserved		Any writes to these bit(s) must always have a value of 0.
26	PARITYERR		Parity error. The calculated parity differs from received parity bit.
		0	No parity error detected. Note: This flag is cleared to 0 when the RXDATA field of SPIBUF is read.
		1	A parity error occurred. If the parity generator is enabled (can be selected individually for each buffer) an even or odd parity bit is added at the end of a data word. During reception of the data word, the parity generator calculates the reference parity and compares it to the received parity bit. If a mismatch is detected, the PARITYERR flag is set.
25-18	Reserved		Any writes to these bit(s) must always have a value of 0.
17-16	LCSNR	0-3h	Last chip select number. LCSNR in the status field is a copy of the CSNR bit in the corresponding control field. It defines the chip select that has been activated during the last data transfer from the corresponding buffer. LCSNR is copied after transmission during write back of received data.
15-0	SPIBUF	0-FFFFh	SPI receive buffer. This field makes up the SPI receive buffer. Data in this field is the data received via the SPIx_SOMI pin and transferred from the SPIDAT1 bits in the SPI shift register (SPIDAT1). Since the data is shifted into the SPI with the most-significant bit first, for word lengths less than 16, the data is stored right-justified in the register.

3.10 SPI Emulation Register (SPIEMU)

The SPI emulation register (SPIEMU) is shown in Figure 20 and described in Table 14.

Figure 20. SPI Emulation Register (SPIEMU)

31 — 16
Reserved
R-0

15 — 0
SPIEMU
R-x

LEGEND: R = Read only; -*n* = value after reset; -x = value is indeterminate after reset

Table 14. SPI Emulation Register (SPIEMU) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
15-0	SPIEMU	0-FFFFh	SPI emulation. SPI emulation is a mirror of the SPI buffer register (SPIBUF). The only difference between SPIEMU and SPIBUF is that a read from SPIEMU does not clear the OVRNINTFLG and RXINTFLAG bits in the SPI flag register (SPIFLG).

3.11 SPI Delay Register (SPIDELAY)

The SPI delay register (SPIDELAY) is shown in Figure 21 and described in Table 15.

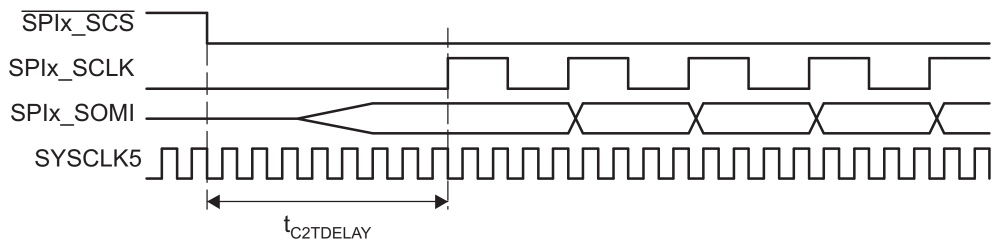
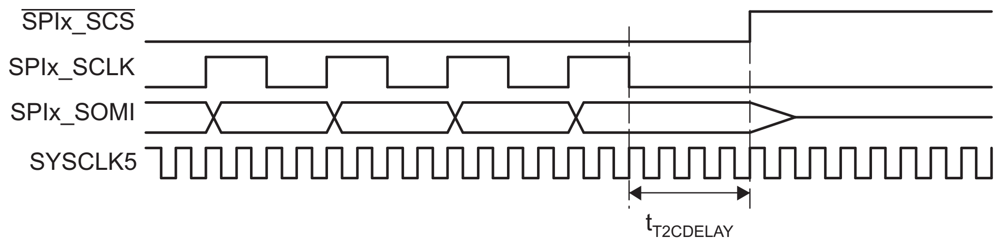
Figure 21. SPI Delay Register (SPIDELAY)

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

15 0
Reserved
R/W-0

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

Table 15. SPI Delay Register (SPIDELAY) Field Descriptions

Bit	Field	Value	Description
31-24	C2TDELAY	0-FFh	Chip-select-active-to-transmit-start-delay. C2TDELAY defines a setup time for the slave device that delays the data transmission from the chip select active edge by a multiple of SPI peripheral clock cycles. C2TDELAY is configured between 2 and 256 SPI peripheral clock cycles.  $\overline{\text{SPIx\_SCS}}$ SPIx_SCLK SPIx_SOMI SYSCLK5 $t_{C2TDELAY}$ The setup time value is calculated as: $$t_{C2TDELAY} = \frac{C2TDELAY}{SPI\ peripheral\ clock} + 2$$ Example: SPI peripheral clock = 25 MHz and C2TDELAY = 06h; therefore, $t_{C2TDELAY}$ = 320 ns. When the chip select signal becomes active, the slave has to prepare data transfer within 320 ns. **Note:** If C2TDELAY = 0, then tC2TDELAY = 0. When the chip select signal becomes active, the slave has to prepare data transfer within 360 ns. **Note:** If phase = 1, the delay between $\overline{\text{SPIx\_SCS}}$ fall-edge to the first edge of SPIx_SCLK will have an additional 0.5 SPIx_SCLK period delay. This delay is as per the SPI protocol.
23-16	T2CDELAY	0-FFh	Transmit-end-to-chip-select-inactive-delay. T2CDELAY defines a hold time for the slave device that delays the chip select deactivation by a multiple of SPI peripheral clock cycles after the last bit is transferred. T2CDELAY is configured between 1 and 255 SPI peripheral clock cycles.  $\overline{\text{SPIx\_SCS}}$ SPIx_SCLK SPIx_SOMI SYSCLK5 $t_{T2CDELAY}$ The hold time value is calculated as: $$t_{T2CDELAY} = \frac{T2CDELAY}{SPI\ peripheral\ clock} + 1$$ Example: SPI peripheral clock = 25 MHz and T2CDELAY = 02h; therefore, $t_{T2CDELAY}$ = 120 ns. After the last data bit is being transferred, the chip select signal is held active for 120 ns. **Note:** If T2CDELAY = 0, then tT2CDELAY = 0 Note: If phase = 0, then between the last edge of SPIx_SCLK and rise-edge of $\overline{\text{SPIx\_SCS}}$ there will be an additional delay of 0.5 SPIx_SCLK period. This is as per the SPI protocol.
15-0	Reserved	0	Reserved. The reserved bit location is always read as 0. This field must be written with zeroes.

3.12 SPI Default Chip Select Register (SPIDEF)

The SPI default chip select register (SPIDEF) is shown in Figure 22 and described in Table 16.

Figure 22. SPI Default Chip Select Register (SPIDEF)

31–16
Reserved
R-0

15–2	1	0
Reserved	SCS1DEF	SCS0DEF
R-0	R/W-1	R/W-1

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

Table 16. SPI Default Chip Select Register (SPIDEF) Field Descriptions

Bit	Field	Value	Description
31-2	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
1	SCS1DEF		Chip select default pattern. Selects the output to the slave 1 chip select pin ($\overline{\text{SPIx\_SCS[1]}}$) when no transmission is currently in progress. SCS1DEF allows you to set a chip select pattern that deselects all the SPI slaves.
		0	$\overline{\text{SPIx\_SCS[1]}}$ is driven to logic 0 when no transaction is in progress.
		1	$\overline{\text{SPIx\_SCS[1]}}$ is driven to logic 1 when no transaction is in progress.
0	SCS0DEF		Chip select default pattern. Selects the output to the slave 0 chip select pin ($\overline{\text{SPIx\_SCS[0]}}$) when no transmission is currently in progress. EN0DEF allows you to set a chip select pattern that deselects all the SPI slaves.
		0	$\overline{\text{SPIx\_SCS[0]}}$ is driven to logic 0 when no transaction is in progress.
		1	$\overline{\text{SPIx\_SCS[0]}}$ is driven to logic 1 when no transaction is in progress.

3.13 SPI Data Format Registers (SPIFMTn)

The SPI data format register (SPIFMT0, SPIFMT1, SPIFMT2, and SPIFMT3) is shown in Figure 23 and described in Table 17.

Figure 23. SPI Data Format Register (SPIFMT*n*)

31 30	29 28 27 26 24	23	22	21	20	19	18	17	16
Reserved	WDELAY*n*	PARPOL*n*	PARITYENA*n*	Reserved	SHIFTDIR*n*	Reserved	DISCSTIMERS*n*	POLARITY*n*	PHASE*n*
R-0	R/WP-0	R/WP-0	R/WP-0	R-0	R/WP-0	R-0	R/WP-0	R/WP-0	R/WP-0

15 8	7 5	4 0
PRESCALE*n*	Reserved	CHARLEN*n*
R/W-0	R-0	R/W-0

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

Table 17. SPI Data Format Register (SPIFMT*n*) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	The reserved bit location is always read as 0. This field must be written with zeroes.
29-24	WDELAY*n*	0-3Fh	Delay in between transmissions for data format n (n = 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 x PCLK + 2 x PCLK PCLK = Period of SPI Peripheral clock
23	PARPOL*n*		Parity polarity: even or odd. PARPOLn can be modified in privilege mode only. It can be used for data format n (n = 0,1,2,3).
		0	An even parity flag is added at the end of the transmit data stream.
		1	An odd parity flag is added at the end of the transmit data stream.
22	PARITYENA*n*		Parity enable for data format *n*.
		0	No parity generation/ verification is performed for this data format.
		1	A parity is transmitted at the end of each transmit data stream. At the end of a transfer the parity generator compares the received parity bit with the locally calculated parity flag. If the parity bits do not match the RXERR flag is set in the corresponding control field. The parity type (even or odd) can be selected via the PARPOL bit.
21	Reserved		The reserved bit location is always read as 0. This field must be written with zeroes.
20	SHIFTDIR*n*		Shift direction for data format *n*. SHIFTDIR*n* selects the shift direction for data format *n* (*n* = 1,2,3).
		0	Most-significant bit is shifted out first.
		1	Least-significant bit is shifted out first.
19	Reserved	0	The reserved bit location is always read as 0. A value written to this field has no effect.
18	DISCSTIMERS*n*		Disable chip select timers for this format register. The C2TDELAY and T2CDELAY timers are by default enabled for all the data format registers. By using this bit, these timers can be disabled for a particular data format if not required. When a master is handling multiple slaves, with varied set-up hold requirement, the application can selectively choose to include or not include the chip select delay timers for any slaves.
		0	Both C2TDELAY and T2CDELAY counts are inserted for the chip selects.
		1	No C2TDELAY or T2CDELAY is inserted in the chip select timings.
17	POLARITY*n*		Clock polarity for data format *n*. POLARITY*n* defines the clock polarity for data format *n* (*n* = 1,2,3).
		0	SPI clock signal is low-inactive, that is, before and after data transfer the clock signal is low.
		1	SPI clock signal is high-inactive, that is, before and after data transfer the clock signal is high.

Table 17. SPI Data Format Register (SPIFMT*n*) Field Descriptions (continued)

Bit	Field	Value	Description
16	PHASE*n*		Clock delay for data format *n*. PHASE*n* defines the clock delay for data format *n* (*n* = 1,2,3).
		0	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	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*n*	1-FFh	Prescaler for data format *n*. PRESCALE*n* defines the bit transfer rate for data format *n* (*n* = 1,2,3). PRESCALE*n* is directly derived from SPI peripheral clock. The clock rate is calculated as: $$SPIx\_SCLK = \frac{SPI\ periperal\ clock}{(PRESCALEn + 1)}$$ PRESCALEn is only supported for values >1. For specific information on the maximum SPI clock rate supported, refer to the SPI timings in the device-specific data manual.
7-5	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
4-0	CHARLEN*n*	0-1Fh	Data word length for data format *n*. CHARLEN*n* defines the word length for data format *n* (*n* = 1,2,3). Legal values are 2h (data word length = 2 bits) to 10h (data word length = 16 bits). Other values, such as 0 or 1Fh, are not detected and their effect is indeterminate.
		0-1h	Not detected and their effect is indeterminate.
		2h-10h	Data word length is 2 bits to 16 bits.
		11h-1Fh	Not detected and their effect is indeterminate.

3.14 SPI Interrupt Vector Register 0 (INTVECT0)

The SPI interrupt vector register 0 (INTVECT0) is shown in Figure 24 and described in Table 18.

Figure 24. SPI Interrupt Vector Register 0 (INTVECT0)

31–16
Reserved
R-0

15–6	5–1	0
Reserved	INTVECT0	SUSPEND0
R-0	R-0	R-0

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

Table 18. SPI Interrupt Vector Register 0 (INTVECT0) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
5-1	INTVECT0	0-1Fh	Interrupt vector for interrupt line INT0. INTVECT0 returns the vector of the pending interrupt at interrupt line INT0. If more than one interrupt is pending, INTVECT0 always references the highest prior interrupt source first. The INTVECT0 field reflects the status of SPIFLG in a vectorized format. So, any updates to SPIFLG will automatically reflect in the vector value in this register. The interrupts available for SPI, in the descending order of their priorities are as given below. Transmission error Interrupt Receive buffer overrun interrupt Receive buffer full interrupt Transmit buffer empty interrupt Vectors for each of these interrupts will be reflected on the INTVECT0 bits, when they occur. Reading the vectors for the receive buffer overrun and receive buffer Full interrupts will automatically clear the respective flags in SPIFLG. On reading the INTVECT0 bits, the vector of the next highest priority interrupt (if any) will be then reflected on the INTVECT0 bits. If two or more interrupts occur simultaneously, the vector for the highest priority interrupt will be reflected on the INTVECT0 bits. Reading the vector register when transmitter empty is indicated does not clear the TXINTFLG in SPIFLG. Writing a new data to SPIDAT1 clears the transmitter empty interrupt. The following are the SPI interrupt vectors for line INT0 in SPI.
		0	No interrupt is pending.
		1h-10h	Reserved
		11h	Error interrupt is pending. Refer to lower halfword of IINT) to obtain more details about the type of error and the concerned buffer.
		12h	The pending interrupt is receive buffer full interrupt.
		13h	The pending interrupt is receive buffer overrun interrupt.
		14h	The pending interrupt is transmit buffer empty interrupt.
		15h-1Fh	Reserved
0	SUSPEND0		Transfer suspended or transfer finished interrupt.(SPI only). Each time INTVEC0 is read by the host, the corresponding interrupt flag of the referenced transfer group is cleared and INTVEC0 is updated with the vector coming next in the priority chain. In parallel, the SUSPEND0 flag is updated depending on the type of interrupt. **Note:** The SUSPEND0 bit always returns value 0 in SPI. Even if there is a RXOVRN interrupt in multi-buffer mode, SUSPEND0 bit stays 0.
		0	The interrupt type is a transfer finished interrupt, that is, the buffer array referenced by INTVECT0 has asserted an interrupt, because all data from the whole transfer group has been transferred. Note: Reading Error Vector. Reading an error vector in INTVEC0 will update the INTVECT0 bits but will not clear the error flags in SPIFLG.
		1	The interrupt type is a transfer suspended interrupt, that is, the transfer group referenced by INTVECT0 has asserted an interrupt, because the buffer to be transferred next is in suspend to wait mode.

3.15 SPI Interrupt Vector Register 1 (INTVECT1)

The SPI interrupt vector register 1 (INTVECT1) is shown in Figure 25 and described in Table 19.

Figure 25. SPI Interrupt Vector Register 1 (INTVECT1)

31–16
Reserved
R-0

15–6	5–1	0
Reserved	INTVECT1	SUSPEND1
R-0	R-0	R-0

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

Table 19. SPI Interrupt Vector Register 1 (INTVECT1) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.
5-1	INTVECT1	0-1Fh	Interrupt vector for interrupt line INT1. INTVECT1 returns the vector of the pending interrupt at interrupt line INT1. If more than one interrupt is pending, INTVECT1 always references the highest prior interrupt source first. The INTVECT1 field reflects the status of SPIFLG in a vectorized format. So, any updates to SPIFLG will automatically reflect in the vector value in this register. The interrupts available for SPI, in the descending order of their priorities are as given below. The interrupts available for SPI, in the descending order of their priorities: • Transmission error Interrupt • Receive buffer overrun interrupt • Receive buffer full interrupt • Transmit buffer empty Interrupt Receive overrun interrupt Vectors for each of these interrupts will be reflected on the INTVECT1 bits, when they occur. Reading the vectors for the receive buffer overrun and receive buffer full interrupts will automatically clear the respective flags in SPIFLG. On reading the INTVECT1 bits, the vector of the next highest priority interrupt (if any) will be then reflected on the INTVECT1 bits. If two or more interrupts occur simultaneously, the vector for the highest priority interrupt will be reflected on the INTVECT1 bits. Reading the vector register when transmitter empty is indicated does not clear the TXINTFLG in SPIFLG. Writing a new data to SPIDAT1 clears the transmitter empty interrupt. The following are the SEL interrupt vectors for line INT1 in SPI.
		0	No interrupt is pending.
		1h-10h	Reserved
		11h	Error interrupt is pending. Refer to lower halfword of the IINT to obtain more details about the type of error and the concerned buffer.
		12h	The pending interrupt is receive buffer full interrupt.
		13h	The pending interrupt is receive buffer overrun interrupt.
		14h	The pending interrupt is transmit buffer empty interrupt.
		15h-1Fh	Reserved
0	SUSPEND1	0	Transfer suspended or transfer finished interrupt. SEL mode only. Every time INTVEC1 is read by the host, the corresponding interrupt flag of the referenced transfer group is cleared and INTVEC1 is updated with the vector coming next in the priority chain. In parallel, the SUSPEND1 flag is updated depending on the type of interrupt. Note: The SUSPEND1 bit always returns value 0 in SPI. Even if there is a RXOVRN interrupt in multi-buffer mode, the SUSPEND1 bit stays 0.
		0	The interrupt type is a transfer finished interrupt, that is, the buffer array referenced by INTVECT0 has asserted an interrupt, because all data from the whole transfer group has been transferred. Note: Reading Error Vector. Reading an error vector in INTVEC0 will update the INTVECT0 bits but will not clear the error flags in SPIFLG.
		1	The interrupt type is a transfer suspended interrupt; that is, the transfer group referenced by INTVECT0 has asserted an interrupt, because the buffer to be transferred next is in suspend to wait mode.

Appendix A Revision History

This document has been revised from SPRUFH1A to SPRUFH1B because of the following technical change(s).

Table 20. Changes Made in This Revision

Location	Additions, Deletions, Changes
Figure 1	Changed figure note C.
Table 1	Changed table title.
Table 2	Added table.
Section 2.4	Changed section.

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Communications and Telecom	www.ti.com/communications
DSP	dsp.ti.com	Computers and Peripherals	www.ti.com/computers
Clocks and Timers	www.ti.com/clocks	Consumer Electronics	www.ti.com/consumer-apps
Interface	interface.ti.com	Energy	www.ti.com/energy
Logic	logic.ti.com	Industrial	www.ti.com/industrial
Power Mgmt	power.ti.com	Medical	www.ti.com/medical
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Space, Avionics & Defense	www.ti.com/space-avionics-defense
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video and Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless-apps