TMS320DM646x DMSoC
Serial Peripheral Interface (SPI)

User's Guide

Literature Number: SPRUER4B
March 2011

List of Figures

List of Tables

Read This First

About This Manual

This document describes the serial peripheral interface (SPI) in the TMS320DM646x 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 TMS320DM646x Digital Media System-on-Chip (DMSoC). 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.

The current documentation that describes the DM646x DMSoC, related peripherals, and other technical collateral, is available in the C6000 DSP product folder at: www.ti.com/c6000.

SPRUEP8 — ***TMS320DM646x DMSoC DSP Subsystem Reference Guide.*** Describes the digital signal processor (DSP) subsystem in the TMS320DM646x Digital Media System-on-Chip (DMSoC).

SPRUEP9 — ***TMS320DM646x DMSoC ARM Subsystem Reference Guide.*** Describes the ARM subsystem in the TMS320DM646x 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 DSP subsystem and a majority of the peripherals and external memories.

SPRUEQ0 — ***TMS320DM646x DMSoC Peripherals Overview Reference Guide.*** Provides an overview and briefly describes the peripherals available on the TMS320DM646x Digital Media System-on-Chip (DMSoC).

SPRAA84 — ***TMS320C64x to TMS320C64x+ CPU Migration Guide.*** Describes migrating from the Texas Instruments TMS320C64x digital signal processor (DSP) to the TMS320C64x+ DSP. The objective of this document is to indicate differences between the two cores. Functionality in the devices that is identical is not included.

SPRU732 — ***TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide.*** Describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C64x and TMS320C64x+ digital signal processors (DSPs) of the TMS320C6000 DSP family. The C64x/C64x+ DSP generation comprises fixed-point devices in the C6000 DSP platform. The C64x+ DSP is an enhancement of the C64x DSP with added functionality and an expanded instruction set.

SPRU871 — ***TMS320C64x+ DSP Megamodule Reference Guide.*** Describes the TMS320C64x+ digital signal processor (DSP) megamodule. Included is a discussion on the internal direct memory access (IDMA) controller, the interrupt controller, the power-down controller, memory protection, bandwidth management, and the memory and cache.

Serial Peripheral Interface (SPI)

1 Introduction

This document describes the serial peripheral interface (SPI) in the TMS320DM646x 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 (2 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 DM646x 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, 4-pin, or 5-pin mode interface. The DM646x DMSoC implementation supports multichip-select operation for up to two SPI slave devices. The SPI can operate as both a master device and a slave device.

1.2 Features

The SPI has the following features:

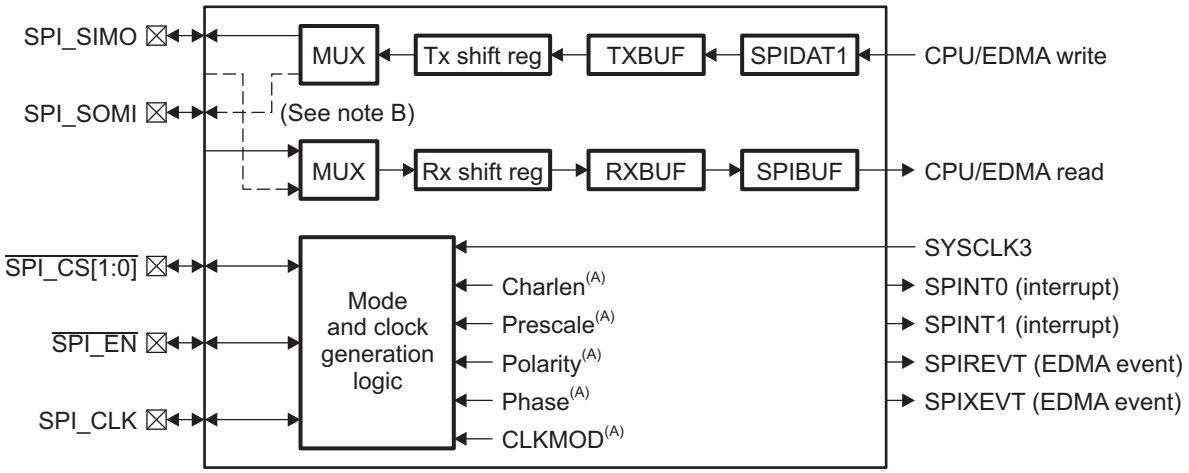
- 16-bit shift register
- Receive buffer register
- 8-bit clock prescaler
- Serial clock I/O pin (SPI_CLK)
- Programmable SPI clock frequency range
- Programmable character length (2 to 16 bits)
- Programmable clock phase (delay or no delay)
- Programmable clock polarity (high or low)
- Two chip select signals ($\overline{\text{SPI\_CS}}$ 1 and $\overline{\text{SPI\_CS}}$ 0) provide the ability to control two slave devices
- One slave in, master out (SPI_SIMO) pin and one slave out, master in (SPI_SOMI) pin
- One enable signal (input to SPI) controlled by the slave device indicating that the slave device is ready to receive

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 SPI_SIMO and SPI_SOMI pins and the CLKMOD bit in SPIGCR1 determines whether an internal or external clock source will be used. The slave chip select ($\overline{\text{SPI\_CS1}}$ and $\overline{\text{SPI\_CS0}}$) pins are used when communicating with multiple slave devices. When the SPI (master) writes to SPIDAT1, the $\overline{\text{SPI\_CS}}$ pins are automatically driven to select the slave connected to that signal. In addition, a handshaking mechanism, provided by the $\overline{\text{SPI\_EN}}$ pin, enables the slave to delay the generation of the clock signal supplied by the master as long as it is not prepared for the next exchange of data.

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.

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.

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 CPU/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 Architecture

This section describes the architecture of the SPI.

2.1 Clock Control

The SPI internal system clock is derived from SYSCLK3. For detailed information on the PLLs and clock distribution on the processor, see the *TMS320DM646x DMSoC ARM Subsystem Reference Guide* (SPRUEP9).

The output clock generated (SPI_CLK) is a derivative of the internal SYSCLK3 clock. The maximum clock bit rate supported by the SPI peripheral is SYSCLK3/4, as determined by the PRESCALE*n* bit in the SPI data format register (SPIFMT*n*). The phase and polarity of the SPI clock signal are also programmable, these configurations are explained in Section 2.3.2. The clock rate is set independently for each of the four software-programmable data formats. For more information on the data formats, see Section 2.3.1. The clock rate for a given data format *n* is calculated as:

SPI_CLK frequency = [SYSCLK3 frequency] / [PRESCALEn + 1]

PRESCALE*n* is only supported for values >2, where the maximum SPI clock rate is (SYSCLK3)/4.

2.2 Signal Descriptions

Table 1 shows the SPI pins used to interface to external devices. The SPI can be operated in a 3-pin, 4-pin, or 5-pin mode

Table 1. Serial Peripheral Interface (SPI) Pins

Pin	Type	Function
SPI_CLK	Input/Output	Serial clock input in slave mode, serial clock output in master mode
$\overline{\text{SPI\_EN}}$	Input/Output	Input in master mode, output in slave mode indicating slave is ready to be served
SPI_SIMO	Input/Output	Serial data input in slave mode, serial data output in master mode
SPI_SOMI	Input/Output	Serial data output in slave mode, serial data input in master mode
$\overline{\text{SPI\_CS0}}$	Input/Output	Slave 0 chip select output in master mode, input in slave mode
$\overline{\text{SPI\_CS1}}$	Input/Output	Slave 1 chip select output in master mode, input in slave mode

2.3 SPI Operation

The SPI can operate as both a master and a slave. 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.

2.3.1 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 2.3.2.
- **Clock phase:** The clock phase is configured by the PHASE*n* bit. The clock phase formats are explained in Section 2.3.2.

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 1 (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.

2.3.1.1 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. 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 TXDATA Field of SPI Shift Register 1 (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 element 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 RXDATA Field of 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

2.3.1.2 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.

2.3.2 Clock Polarity and Phase

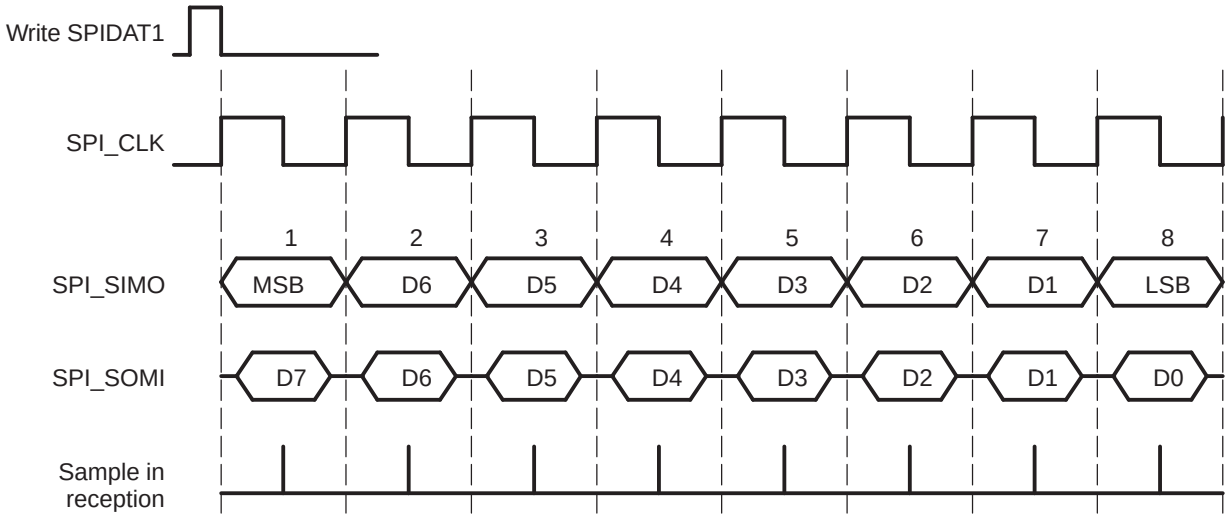
The SPI provides the flexibility to program four different clock mode combinations that SPI_CLK may operate, enabling a choice of the clock phase (delay or no delay) and the clock polarity (rising edge or falling edge). When operating with PHASE active, the SPI makes the first bit of data available after SPIDAT1 is written and before the first edge of SPI_CLK. The data input and output edges depend on the values of both the POLARITY and PHASE bits as shown in Table 2.

Table 2. SPI Clocking Modes

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

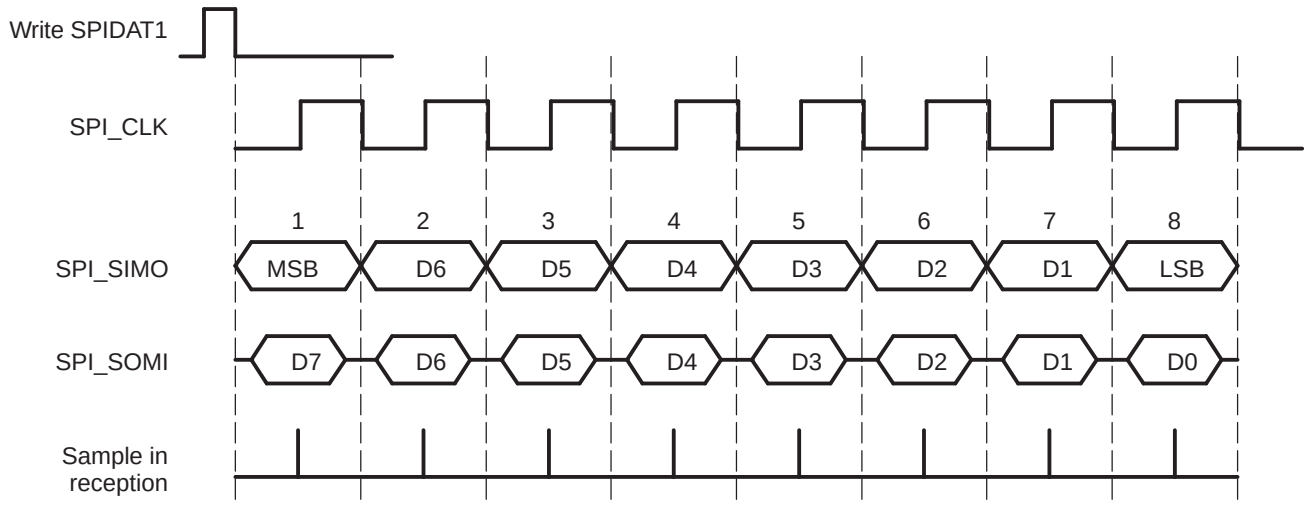
Figure 4 through Figure 7 show the four possible signals of SPI_CLK corresponding to each mode. Having four signal options allows the SPI to interface with different types of serial devices. Also shown on the footnotes in each figure is the SPI_CLK control bit polarity and phase values corresponding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

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



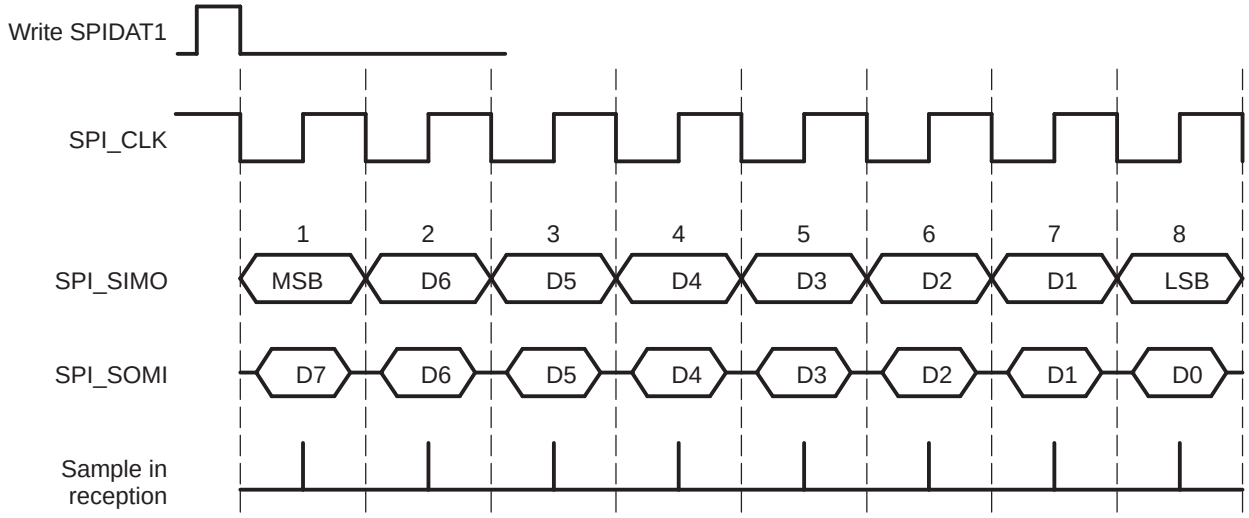
Clock phase = 0 (SPI_CLK without delay)
- Data is output on the rising edge of SPI_CLK
- Input data is latched on the falling edge of SPI_CLK
- A write to SPIDAT1 starts SPI_CLK

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



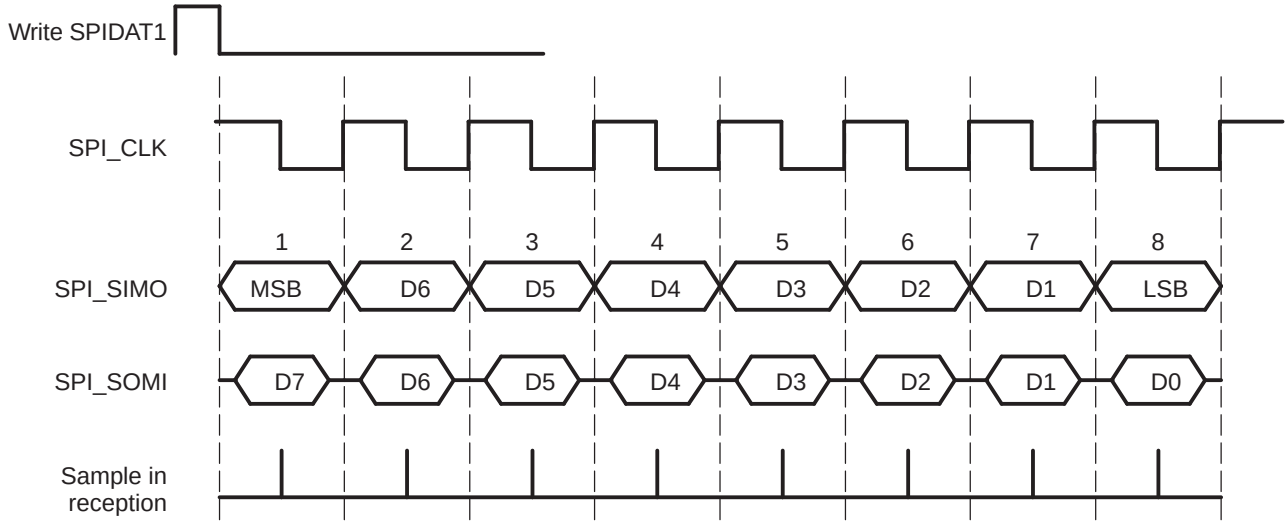
Clock phase = 1 (SPI_CLK with delay)
- Data is output one-half cycle before the first rising of SPI_CLK and on subsequent falling edges of SPI_CLK
- Input data is latched on the rising edge of SPI_CLK

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



Clock phase = 0 (SPI_CLK without delay)
- Data is output on the falling edge of SPI_CLK
- Input data is latched on the rising edge of SPI_CLK
- A write to SPIDAT1 starts SPI_CLK

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



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

2.3.3 Chip Select Control

The SPI provides two chip select signals ($\overline{\text{SPI\_CS0}}$ and $\overline{\text{SPI\_CS1}}$) that are used to selectively enable multiple slaves. The behavior of the chip selects is controlled by the CSNR and CSHOLD bits in SPIDAT1.

2.3.3.1 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 SCSFUN*n* bit in the SPI pin control register 0 (SPIPC0) must be set to 1 for each chip select.

2.3.3.2 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; and it is programmable only in the master mode.

- When CSHOLD = 0, the chip selects are deasserted 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 deasserted even if CSHOLD = 1.

2.3.3.3 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 SYSCLK3 clock (not the SPI_CLK).

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 SYSCLK3 clock (not the SPI_CLK).

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

2.3.3.4 Inactive Chip Select State Control

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

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

2.3.4 SPI Operation

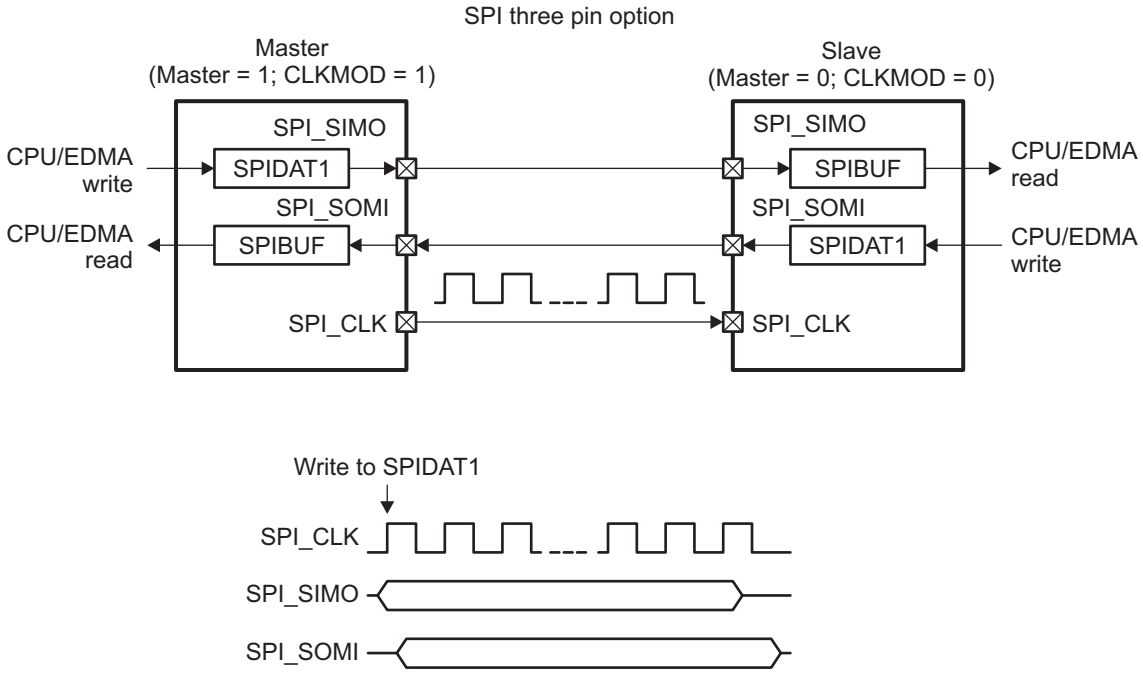
2.3.4.1 SPI Operation: 3-Pin Mode

In master mode configuration (MASTER = 1 and CLKMOD = 1 in SPIGCR1), the SPI provides the serial clock on the SPI_CLK pin for the entire serial communications network. Data is output on the SPI_SIMO pin and latched in from the SPI_SOMI pin, as shown in Figure 8.

Data written to the SPI shift register 1 (SPIDAT1) initiates data transmission on the SPI_SIMO pin, most significant bit (MSB) first. Simultaneously, received data is shifted through the SPI_SOMI pin into the least significant bit (LSB) of SPIDAT1. When the selected number of bits has been transmitted, the received data in the Shift Register is transferred to the SPI buffer register (SPIBUF) for the CPU to read. Data is stored right-justified in SPIBUF.

Refer to Section 1.3.1 and Section 1.3.2 for details about the data handling for transmit and receive operations.

Figure 8. SPI 3-Pin Option



In slave mode configuration (MASTER = 0 and CLKMOD = 0 in SPIGCR1), data shifts out on the SPI_SOMI pin and in on the SPI_SIMO pin. The SPI_CLK pin is used as the input for the serial shift clock, which is supplied from the external network master. The transfer rate is defined by this clock.

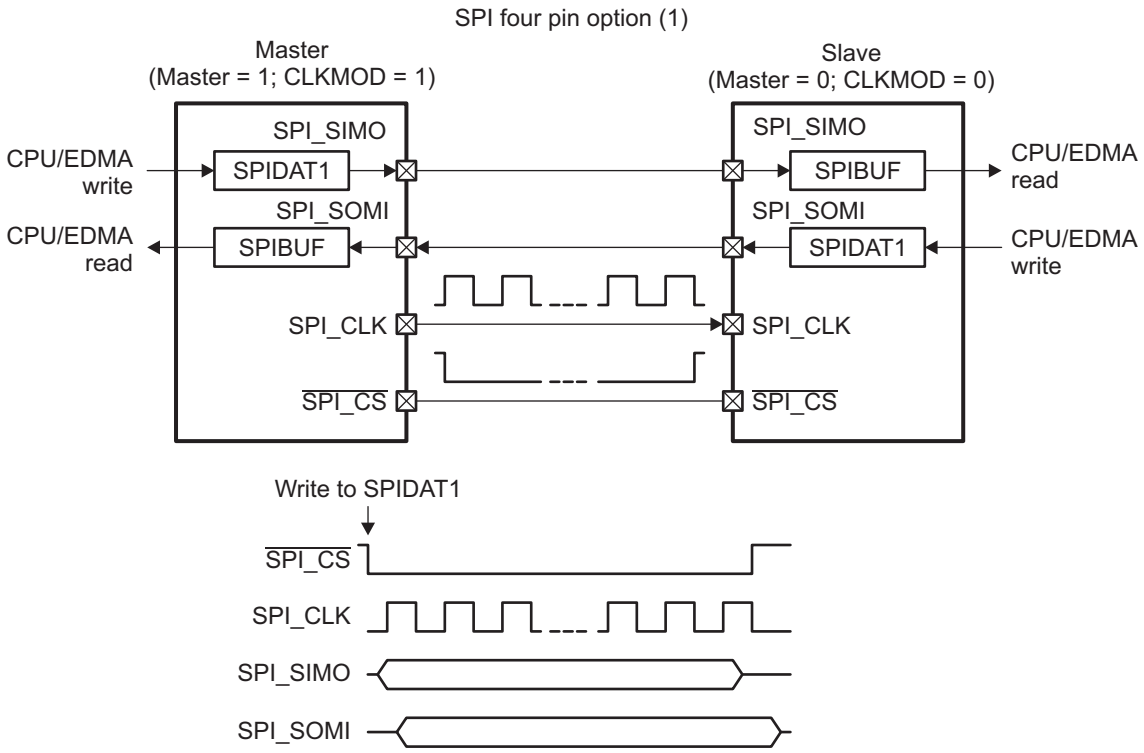
Data written to SPIDAT1 is transmitted to the network when the SPI_CLK signal is received from the network master. To receive data, the SPI waits for the network master to send the SPI_CLK signal and then shifts data on the SPI_SIMO pin into the Rx shift register. If data is to be transmitted by the slave simultaneously, it must be written to SPIDAT1 before the beginning of the SPI_CLK signal.

2.3.4.2 SPI Operation: 4-Pin Mode

The three-pin option and the four-pin options of the SPI are identical in the master mode, except that the four-pin option uses either $\overline{\text{SPI\_EN}}$ or $\overline{\text{SPI\_CS}}$ pins. The I/O directions of these pins are determined by the CLKMOD bit in SPIGCR1.

2.3.4.2.1 4-Pin Mode with $\overline{\text{SPI\_CS}}$ Pins

Figure 9. SPI 4-Pin Option with $\overline{\text{SPI\_CS}}$



The $\overline{\text{SPI\_CS}}$ 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 $\overline{\text{SPI\_CS}}$ 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.

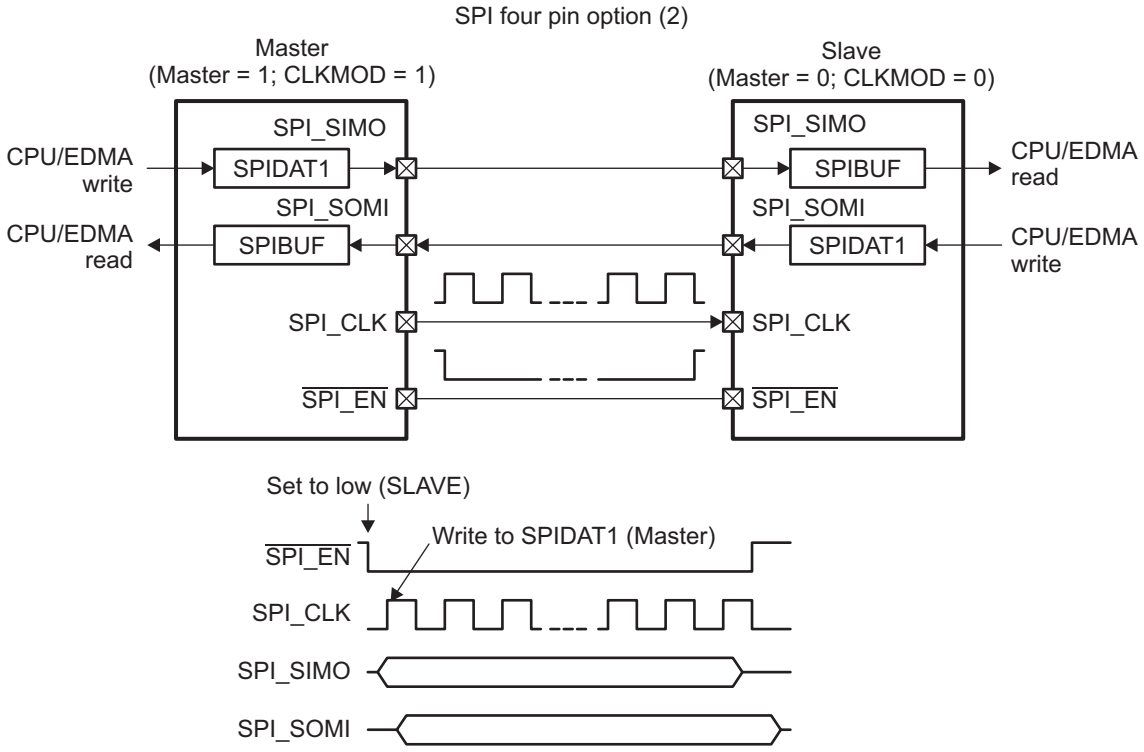
A Slave mode SPI can be selected only by an active value of 0 on any of its selected $\overline{\text{SPI\_CS}}$ pins. However, Master SPI is capable of driving both 0 or 1 as the active value of any of $\overline{\text{SPI\_CS}}$ pins. In Master mode SPI, this is fully controlled by the CSNR field of SPIDAT1.

NOTE: During an SPI transfer, if the Slave mode SPI, detects a de-assertion of its Chipselect even before its internal Character Length counter overflows, then it tristates its SPI_SOMI pin. Once this condition has occurred, if a SPI_CLK edge is detected while the ChipSelect is de-asserted, the SPI stops the transfer and sets an error flag DLENERR (Data Length) and generates an interrupt if enabled.

2.3.4.2.2 4-Pin Mode with $\overline{\text{SPI\_EN}}$ Pin

To use the $\overline{\text{SPI\_EN}}$ as a WAIT signal pin, the $\overline{\text{SPI\_EN}}$ pin must be configured to be functional (ENAFUN = 1 in SPIPC0). In this mode, an active low signal on the $\overline{\text{SPI\_EN}}$ pin will allow the master SPI to drive the clock pulse stream; otherwise, the master will hold the clock signal. The slave will automatically drive $\overline{\text{SPI\_EN}}$ low after a new element is written to the slave Tx shift register. If the $\overline{\text{SPI\_EN}}$ pin is in high-impedance mode (ENABLE_HIGHZ = 1 in SPIINT), the slave will automatically put $\overline{\text{SPI\_EN}}$ into the high-impedance once it completes receiving a new element (Rx shift register is full). If the $\overline{\text{SPI\_EN}}$ pin is in push-pull mode (ENABLE_HIGHZ = 0), the slave will automatically drive $\overline{\text{SPI\_EN}}$ to 1 once it completes receiving a new element (Rx shift register is full). The slave will drive $\overline{\text{SPI\_EN}}$ low again after new data is written to the slave Tx shift register. The 4-pin mode with $\overline{\text{SPI\_EN}}$ mainly applies to slave-transmitting-only and full-duplex cases. For slave-receiving-only cases, each time the slave is ready to receive (Rx shift register is empty), the slave must provide a dummy write to the SPI shift register 1 (SPIDAT1) in order for it to receive the new element from the master in this mode.

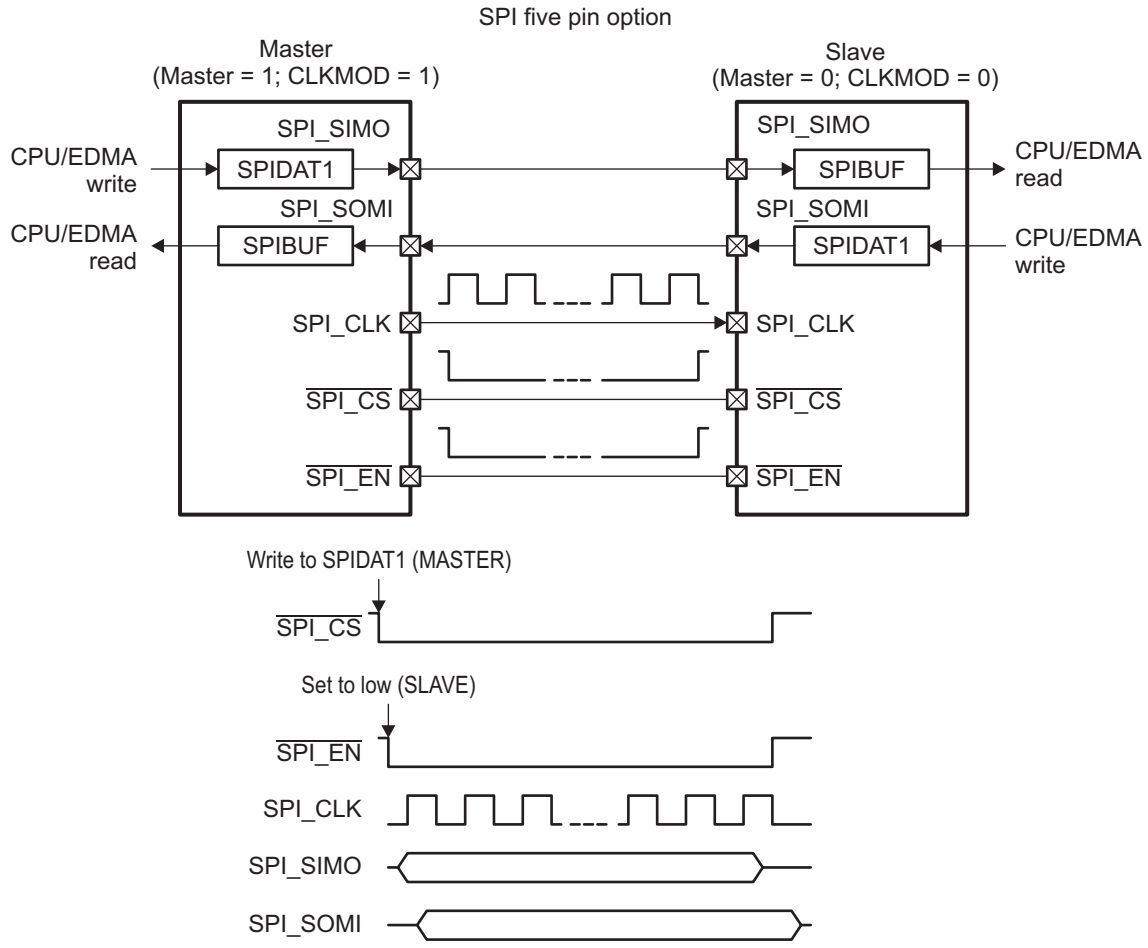
Figure 10. SPI 4-Pin Option with $\overline{\text{SPI\_EN}}$



2.3.4.3 SPI Operation: 5-Pin Mode (Hardware Handshaking)

To use the hardware handshaking mechanism, both $\overline{\text{SPI\_EN}}$ and $\overline{\text{SPI\_CS}}$ pins must be configured as functional pins.

Figure 11. SPI 5-Pin Option with $\overline{\text{SPI\_EN}}$ and $\overline{\text{SPI\_CS}}$



In the master SPI, if the $\overline{\text{SPI\_EN}}$ pin is configured as functional, then the pin will act as an input pin. If configured as a slave SPI, the $\overline{\text{SPI\_EN}}$ pin when configured as functional, acts as an output pin.

If the $\overline{\text{SPI\_EN}}$ pin is in high-z mode (ENABLE_HIGHZ = 1 in SPIINT), the slave SPI will put this signal into the high-impedance state by default. The slave will drive the signal $\overline{\text{SPI\_EN}}$ low when new data is written to the slave Tx shift register and the slave has been selected by the master ($\overline{\text{SPI\_CS}}$ is low).

If the $\overline{\text{SPI\_EN}}$ pin is in push-pull mode (ENABLE_HIGHZ = 0), the slave SPI will drive this pin high by default when it's in functional mode. The slave SPI will drive the $\overline{\text{SPI\_EN}}$ signal low when new data is written to the slave Tx shift register and the slave is selected by the master ($\overline{\text{SPI\_CS}}$ is low). If the slave is de-selected by the master ($\overline{\text{SPI\_CS}}$ goes high), the slave $\overline{\text{SPI\_EN}}$ signal is driven high automatically.

NOTE: Push-Pull mode of $\overline{\text{SPI\_EN}}$ pin can be used only when there's a single slave in the system. When there are multiple SPI slave devices connected to the common $\overline{\text{SPI\_EN}}$ pin, all the Slaves should configure their $\overline{\text{SPI\_EN}}$ pins in high-impedance mode.

In the master SPI (CLKMOD = 1 in SPIGCR1), if the SPI_CS pins are configured as functional pins, then the pin will be in output mode. If configured as a slave SPI (CLKMOD = 0), the SPI_CS pin will act as a functional input. A write to the master's SPI shift register 1 (SPIDAT1) will automatically drive the SPI_CS signal low. The master will drive the SPI_CS signal high again after completing the transfer of the bits of the data.

NOTE: During an SPI transfer, if Slave mode SPI detects a de-assertion of its Chipselect even before its internal Character Length counter overflows, then it tristates its SPI_SOMI and SPI_EN pins (if ENABLE_HIGHZ bit in SPIINT is set to 1). Once this condition has occurred, if a SPI_CLK edge is detected while the ChipSelect is deasserted, then the SPI stops that transfer and sets an error flag DLENERR (Data Length) and generates an interrupt if enabled.

The 5-pin mode mainly applies to slave-transmitting-only and full-duplex cases. For slave-receiving-only cases, each time the slave is ready to receive (Rx shift register is empty), the slave must provide a dummy write to SPIDAT1 in order for it to receive the new element from the master in this mode.

2.4 Reset Considerations

This section provides the software and hardware reset considerations.

2.4.1 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.

2.4.2 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 *TMS320DM646x DMSoC ARM Subsystem Reference Guide* (SPRUEP9).

2.5 Initialization

The following list provides procedures for initializing the SPI.

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. Set 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 SPI_SIMO, SPI_SOMI, and SPI_CLK pins and the necessary chip select pins (SPI_CS0, SPI_CS1, and SPI_EN) by setting the corresponding bits in the SPI pin control register 0 (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 preconfigured data format using the DFSEL bit in the SPI shift register 1 (SPIDAT1).

7. If using SPI in 4-pin mode with $\overline{\text{SPI\_CS}}$, configure the following:
   (a) Configure the setup or hold time for the chip select lines using the C2TDELAY or T2CDELAY bits in the SPI delay register (SPIDELAY).
   (b) Select the desired chip select number. The CSNR field in SPIDAT1 defines the chip select that shall be activated during the data transfer.
   (c) Setup the default chip select pin value when chip select lines are inactive using the CSDEF*n* bits in the SPI default chip select register (SPIDEF).
8. If using SPI in 4-pin mode with $\overline{\text{SPI\_EN}}$, configure the following:
   (a) Configure the $\overline{\text{SPI\_EN}}$ pin to be functional by setting ENAFUN bit in SPIPC0 to 1.
   (b) Configure the ENABLE_HIGHZ bit in the SPI interrupt register (SPIINT) to indicate the behavior of $\overline{\text{SPI\_EN}}$ when SPI is not active.
9. If using SPI in 5-pin mode, do both step 6 and step 7.
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. Enable the SPIEN bit in SPIGCR1.
13. If using the EDMA to perform the transfers, setup and enable the EDMA channels for transmit or receive and then set the DMAREQEN bit in SPIINT.
14. Data is ready to be transferred using the CPU or EDMA by writing to SPIDAT1.

2.6 Interrupt Support

The SPI module outputs two interrupts (Table 3) that are routed to the CPU. The SPI interrupt system is controlled by three registers:

- The SPI interrupt level register (SPILVL) controls which events (SPINT0 or SPINT1) are assigned to each interrupt.
- The SPI interrupt register (SPIINT) contains bits to selectively enable/disable each interrupt event.
- The SPI flag status 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 status register (SPIFLG) or the INTVECT*n* code in the SPI interrupt vector register *n* (INTVEC*n*).

Table 3. SPI Module Interrupts

ARM Event	Acronym	Source
43	SPINT0	SPI
44	SPINT1	SPI

2.7 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 (Table 4) that go to the system EDMA, allowing EDMA transfers to be triggered by SPI read receive or write transmit events. The SPIXEVT is a transmit sync event; the SPIREVT is a receive sync event. The SPI module enables EDMA requests by enabling the EDMA request enable (DMAREQEN) bit in 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 SPIDAT1. Data is then read from the SPI buffer register (SPIBUF), which automatically clears the RXINTFLG bit in the SPI flag status 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 *TMS320DM646x DMSoC Enhanced Direct Memory Access (EDMA) Controller Reference Guide* (SPRUEQ5).

Table 4. EDMA Events

Event	Acronym	Source
16	SPIXEVT	SPI transmit event
17	SPIREVT	SPI receive event

2.8 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 *TMS320DM646x DMSoC ARM Subsystem Reference Guide* (SPRUEP9).

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.

2.9 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 SPI_SIMO, SPI_SOMI, and SPI_CLK 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.

2.10 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 5 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 5 should be considered as reserved locations and the register contents should not be modified.

Table 5. SPI Registers

Offset	Acronym	Register Description	Section
0h	SPIGCR0	SPI Global Control Register 0	Section 3.1
4h	SPIGCR1	SPI Global Control Register 1	Section 3.2
8h	SPIINT	SPI Interrupt Register	Section 3.3
Ch	SPILVL	SPI Interrupt Level Register	Section 3.4
10h	SPIFLG	SPI Flag Status Register	Section 3.5
14h	SPIPC0	SPI Pin Control Register 0	Section 3.6
1Ch	SPIPC2	SPI Pin Control Register 2	Section 3.7
3Ch	SPIDAT1	SPI Shift Register 1	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	SPIFMT0	SPI Data Format Register 0	Section 3.13
54h	SPIFMT1	SPI Data Format Register 1	Section 3.13
58h	SPIFMT2	SPI Data Format Register 2	Section 3.13
5Ch	SPIFMT3	SPI Data Format Register 3	Section 3.13
60h	INTVEC0	SPI Interrupt Vector Register 0	Section 3.14
64h	INTVEC1	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 12 and described in Table 6.

Figure 12. SPI Global Control Register 0 (SPIGCR0)

31	16
Reserved	
R-0	

15	1	0
Reserved		RESET
R-0		R/WP-0

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

Table 6. SPI Global Control Register 0 (SPIGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return zero and writes have no effect.
0	RESET		Reset bit for the module. This bit needs to be set to 1 before any operation on SPI can be 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 13 and described in Table 7.

Figure 13. 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; WP = Write in privilege mode only; -*n* = value after reset

Table 7. SPI Global Control Register 1 (SPIGCR1) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return zero and writes have no effect.
24	SPIEN		SPI enable. This bit 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	Reads return zero 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 SPI_SIMO and SPI_SOMI pins are configured with SPI functionality, then the SPI_SIMO pin is internally connected to the SPI_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 SPI_CLK pin outputs an inactive value and SPI_SOMI remains in high-impedance state. 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. **Note:** This loopback mode can be used only in master mode. This automatically selects digital loopback path. When this Loopback mode is selected, CLKMOD bit should be set to 1, meaning that SPI_CLK can only be internal.
		0	Internal loop-back test mode disabled.
		1	Internal loop-back test mode enabled.
15-9	Reserved	0	Reads return zero and writes have no effect.
8	POWERDOWN		When active, the SPI state machines enter a power-down state.
		0	The SPI is in active mode.
		1	The SPI is in power-down mode.
7-2	Reserved	0	Reads return zero and writes have no effect.
1	CLKMOD		Clock mode. This bit selects either an internal or external clock source. This bit also determines the I/O direction of the $\overline{\text{SPI\_EN}}$ and $\overline{\text{SPI\_CS}}$ pins in functional mode.
		0	Clock is external.
		1	Clock is internal.

Table 7. SPI Global Control Register 1 (SPIGCR1) Field Descriptions (continued)

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

3.3 SPI Interrupt Register (SPIINT)

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

Figure 14. SPI Interrupt Register (SPIINT)

31	25	24	23	17	16
Reserved		ENABLE_HIGHZ	Reserved		DMAREQEN
R-0		R/W-0	R-0		R/W-0

15	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	DESYNCENA	PARERRENA	TIMEOUTENA	DLENERRENA
R-0	R/W-0	R-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

Table 8. SPI Interrupt Register (SPIINT) Field Descriptions

Bit	Field	Value	Description
31-25	Reserved	0	Reads return zero and writes have no effect.
24	ENABLE_HIGHZ		$\overline{\text{SPI\_EN}}$ pin high-impedance enable. When active, the $\overline{\text{SPI\_EN}}$ pin (when it is configured as a WAIT functional output signal in a slave SPI) is forced to place it is output in high-impedance when not driving a low signal. If inactive, then the pin will output both a high and a low signal.
		0	$\overline{\text{SPI\_EN}}$ pin is pulled high when not active.
		1	$\overline{\text{SPI\_EN}}$ pin remains in high-impedance when not active.
23-17	Reserved	0	Reads return zero and writes have no effect.
16	DMAREQEN		EDMA request enable. Enables the EDMA request signal to be generated for both receive and transmit channels. Enable EDMA REQ only after setting the SPIEN bit in SPIGCR1 to 1.
		0	EDMA is not used.
		1	EDMA is requests will be generated.
			Note: An EDMA request will be generated on TX EDMA REQ line each time a transmit data is copied to the shift register either from TXBUF or directly from SPIDAT1 writes.
			An EDMA request will be generated on RX EDMA REQ line each time a received data is copied to SPIBUF either from RXBUF or directly from the shift register.
15-10	Reserved	0	Reads return zero and writes have no effect.
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	RXINTEN		An interrupt is to be generated when the RXINTFLG bit in SPIFLG is set by hardware; otherwise, no interrupt will be generated.
		0	Interrupt will not be generated.
		1	Interrupt will be generated. Both transmitter empty and receiver full interrupts are valid in SPI only.
7	Reserved	0	Reads return zero and writes have no effect.
6	OVRNINTENA		Overrun interrupt enable. An interrupt is to be generated when the RCVROVRN flag bit in SPIFLG is set by hardware; otherwise, no interrupt will be generated.
		0	Overrun interrupt is not generated.
		1	Overrun interrupt is generated.
5	Reserved	0	Reads return zero and writes have no effect.

Table 8. SPI Interrupt Register (SPIINT) Field Descriptions (continued)

Bit	Field	Value	Description
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	DESYNCENA		Enables interrupt on desynchronized slave. DESYNCENA is used in master mode only.
		0	No interrupt asserted upon desynchronization error.
		1	Enables an interrupt on desynchronization of the slave (DESYNC = 1).
2	PARERRENA		Enables interrupt on parity error.
		0	No interrupt asserted upon parity error.
		1	Enables an interrupt on a parity error (PARITYERR = 1).
1	TIMEOUTENA		Enables interrupt on ENA signal time-out.
		0	No interrupt asserted upon ENA signal time-out.
		1	Enables an interrupt on a time-out of the ENA signal (TIMEOUT = 1).
0	DLENERRENA		Data length error interrupt enable. A data length error occurs under the following conditions. Master: In a 4-pin with $\overline{\text{SPI\_EN}}$ mode or 5-pin mode, if the $\overline{\text{SPI\_EN}}$ pin from the slave is deasserted before the master has completed its transfer, the data length error is set. That is, if the character length counter has not overflowed while $\overline{\text{SPI\_EN}}$ deassertion is detected, then it means that the slave has neither received full data from the master nor has it transmitted complete data. Slave: In a 4-pin with chip selects mode or 5-pin mode, if the incoming valid $\overline{\text{SPI\_CS}}$ pin is deactivated before the character length counter overflows, then data length error is set.
		0	No interrupt is generated upon data length error.
		1	Enables an interrupt when data length error occurs.

3.4 SPI Interrupt Level Register (SPILVL)

The SPI interrupt level register (SPILVL) is shown in Figure 15 and described in Table 9.

Figure 15. SPI Interrupt Level Register (SPILVL)

31							16
Reserved							
R-0							

15						10	9	8
Reserved							TXINTLVL	RXINTLVL
R-0							R/W-0	R/W-0

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

Table 9. SPI Interrupt Level Register (SPILVL) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return zero and writes have no effect.
9	TXINTLVL		Transmit interrupt level.
		0	Transmit interrupt is mapped to interrupt line SPINT0.
		1	Transmit interrupt is mapped to interrupt line SPINT1.
8	RXINTLVL		Receive interrupt level.
		0	Receive interrupt is mapped to interrupt line SPINT0.
		1	Receive interrupt is mapped to interrupt line SPINT1.
7	Reserved	0	Reads return zero and writes have no effect.
6	OVRNINTLVL		Receive overrun interrupt level.
		0	Receive overrun interrupt is mapped to interrupt line SPINT0.
		1	Receive overrun interrupt is mapped to interrupt line SPINT1.
5	Reserved	0	Reads return zero and writes have no effect.
4	BITERRLVL		Bit error interrupt level.
		0	Bit error interrupt is mapped to interrupt line SPINT0.
		1	Bit error interrupt is mapped to interrupt line SPINT1.
3	DESYNCLVL		Desynchronized slave interrupt level. DESYNCLVL is used in master mode only.
		0	An interrupt due to desynchronization of the slave (DESYNC = 1) is mapped to interrupt line SPINT0.
		1	An interrupt due to desynchronization of the slave (DESYNC = 1) is mapped to interrupt line SPINT1.
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	TIMEOUTLVL		$\overline{\text{SPI\_EN}}$ signal time-out interrupt level.
		0	An interrupt on a time-out of the ENA signal (TIMEOUT = 1) is mapped to interrupt line SPINT0.
		1	An interrupt on a time-out of the ENA signal (TIMEOUT = 1) is mapped to interrupt line SPINT1.
0	DLENERRLVL		Data length error interrupt enable level.
		0	An interrupt on data length error is mapped to interrupt line SPINT0.
		1	An interrupt on data length error is mapped to interrupt line SPINT1.

3.5 SPI Flag Status Register (SPIFLG)

The SPI flag status register (SPIFLG) is shown in Figure 16 and described in Table 10.

Figure 16. SPI Flag Status Register (SPIFLG)

31							16
Reserved							
R-0							
15					10	9	8
Reserved						TXINTFLG	RXINTFLG
R-0						R-0	R/W1C-0
7	6	5	4	3	2	1	0
Reserved	OVRNINTFLG	Reserved	BITERRFLG	DESYNCFLG	PARERRFLG	TIMEOUTFLG	DLENERRFLG
R-0	R/W1C-0	R-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0

LEGEND: R/W = Read/Write; R = Read only; W1C = Write to clear; *-n* = value after reset

Table 10. SPI Flag Status Register (SPIFLG) Field Descriptions

Bit	Field	Value	Description
31-10	Reserved	0	Reads return zero and writes have no effect.
9	TXINTFLG		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	RXINTFLG		Receiver full interrupt flag. This flag is set when a word is received and copied into the SPI buffer register (SPIBUF). If RXINTEN is enabled, an interrupt is also generated. During emulation mode, however, a read to the emulation register (SPIEMU) does not clear this flag bit. This bit is cleared under the following ways: • Reading SPIBUF. • Reading INTVEC0 or INTVEC1 when there is a receive buffer full interrupt. • Writing a 1 to this bit. • Writing a 0 to SPIEN bit in SPIGCR1. • System reset.
		0	No new received data pending. Receive buffer is empty.
		1	A newly received data is ready to be read. Receive buffer is full. **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	Reads return zero and writes have no effect.

Table 10. SPI Flag Status Register (SPIFLG) Field Descriptions (continued)

Bit	Field	Value	Description
6	OVRNINTFLG		Receiver overrun flag. The SPI hardware sets this bit when a receive operation completes before the previous character has been read from the receive 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 OVRNINTEN bit in SPIINT is set high.
		0	Overrun condition did not occur.
		1	Overrun condition has occurred. In SPI, this bit is cleared under the following conditions: • Reading INTVEC0 or INTVEC1 when there is a receive buffer overrun interrupt. • Writing a 1 to OVRNINTFLG bit in SPIFLG. **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 TIMEOUT, BITERR, and DLEN_ERR 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	Reads return zero and writes have no effect.
4	BITERRFLG		Mismatch of internal transmit data and transmitted data.
		0	No bit error occurred. 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 bit error occurred. The SPI samples the signal of the transmit pin (master: SPI_SIMO, slave: SPI_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. If BITERRENA is set an interrupt is asserted. A possible reason for a bit error can be a too high bit rate/capacitive load or another master/slave trying to transmit at the same time.
3	DESYNCFLG		Desynchronization of slave device. Desynchronization monitor is active in master mode only.
		0	No slave desynchronization 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. **Note:** Inconsistency of Desync flag in SPI. Due to the nature of this error, under some circumstances it is possible for a desync error detected for the previous buffer to be visible in the current buffer. This is due to the fact that receive completion flag/interrupt will be generated when the buffer transfer is completed. But desync will be detected after the buffer transfer is completed. So, if CPU/EDMA reads the received data quickly when an RXINT is detected, then the status flag may not reflect the correct desync condition. This inconsistency in the desync flag is valid only in SPI.
		1	A slave device is desynchronized. The master monitors the ENABLE signal coming from the slave device and sets the DESYNC flag after the last bit is transmitted plus $t_{T2EDELAY}$. If DESYNCENA is set an interrupt is asserted. Desynchronization can occur if a slave device misses a clock edge coming from the master.
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.

Table 10. SPI Flag Status Register (SPIFLG) Field Descriptions (continued)

Bit	Field	Value	Description
1	TIMEOUTFLG		Time-out due to non-activation of $\overline{\text{SPI\_EN}}$ signal.
		0	No ENA signal time-out occurred. 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	An ENA signal time-out occurred. The SPI generates a time-out because the slave hasn't responded in time by activating the ENA signal after the chip select signal has been activated. If a time-out condition is detected the corresponding chip select is deactivated immediately and the TIMEOUT flag is set. In addition the TIMOUT flag in the status field of the corresponding buffer is set. The transmit request of the concerned buffer is cleared, that is, the SPI doesn't re-start a data transfer from this buffer.
0	DLENERRFLG		Data length error flag.
		0	No data length error has occurred. This flag can be cleared by one of the following ways: • Write a 1 to this bit. • Clear SPIEN bit in SPIGCR1 to 0. **Note:** Whenever any transmission errors (TIMEOUT, BITERR, DLENERR, PARITYERR, DESYNC) are detected, and the error flags are cleared by writing to the error bit in SPIFLG, the corresponding error flag in SPIBUF does not get cleared. Software needs to read the SPIBUF until it becomes empty before proceeding. This ensures that all the older status bits in SPIBUF are cleared before starting the next transfer.
		1	A data length error has occurred.

3.6 SPI Pin Control Register 0 (SPIPC0)

The SPI pin control register 0 (SPIPC0) is shown in Figure 17 and described in Table 11.

NOTE: Duplicate Control Bits for SPI_SIMO and SOMI0

Bit 24 is not physically implemented, it is a mirror of bit 11. Any write to bit 24 is reflected on bit 11 and when bit 24 and bit 11 are simultaneously written, the value of bit 11 controls the SPI_SOMI pin. Reading bit 24 always reflects the bit 11 value. This is true for bit 24 and bit 11 of both SPIPC0 and SPIPC2. Same is true for SPI_SIMO pin with bit 16 and bit 10.

Figure 17. SPI Pin Control Register 0 (SPIPC0)

31	16
Reserved	
R-0	

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

7	2	1	0
Reserved		SCSFUN1	SCSFUN0
R-0		R/W-0	R/W-0

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

Table 11. SPI Pin Control Register 0 (SPIPC0) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMIFUN		Slave out, master in function.
		0	Reserved.
		1	SPI_SOMI pin is a SPI functional pin
10	SIMOFUN		Slave in, master out function.
		0	Reserved.
		1	SPI_SIMO pin is a SPI functional pin.
9	CLKFUN		SPI clock function.
		0	Reserved.
		1	The SPI_CLK pin is a SPI functional pin.
8	ENAFUN		$\overline{\text{SPI\_EN}}$ function.
		0	Reserved.
		1	The $\overline{\text{SPI\_EN}}$ pin is a SPI functional pin.
7-2	Reserved	0	Reads return zero and writes have no effect.
1	SCSFUN1		$\overline{\text{SPI\_CS1}}$ function.
		0	Reserved.
		1	The $\overline{\text{SPI\_CS1}}$ pin is a SPI functional pin.
0	SCSFUN0		$\overline{\text{SPI\_CS0}}$ function.
		0	Reserved.
		1	The $\overline{\text{SPI\_CS0}}$ pin is a SPI functional pin.

3.7 SPI Pin Control Register 2 (SPIPC2)

The SPI pin control register 2 (SPIPC2) is shown in Figure 18 and described in Table 12.

Figure 18. SPI Pin Control Register 2 (SPIPC2)

31	16
Reserved	
R-0	

15	12	11	10	9	8
Reserved		SOMIDIN	SIMODIN	CLKDIN	ENADIN
R-0		R/W-0	R/W-0	R/W-0	R/W-0

7	2	1	0
Reserved		SCSDIN1	SCSDIN0
R-0		R/W-0	R/W-0

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

Table 12. SPI Pin Control Register 2 (SPIPC2) Field Descriptions

Bit	Field	Value	Description
31-12	Reserved	0	Reads return zero and writes have no effect.
11	SOMIDIN		SPI_SOMI data in. Reflects the value of the SPI_SOMI pin.
		0	Current value on SPI_SOMI pin is logic 0.
		1	Current value on SPI_SOMI pin is logic 1.
10	SIMODIN		SPI_SIMO data in. Reflects the value of the SPI_SIMO pin.
		0	Current value on SPI_SIMO pin is logic 0.
		1	Current value on SPI_SIMO pin is logic 1.
9	CLKDIN		Clock data in. This bit reflects the value of the SPI_CLK pin.
		0	Current value of SPI_CLK pin is logic 0.
		1	Current value of SPI_CLK pin is logic 1.
8	ENADIN		$\overline{\text{SPI\_EN}}$ data in. This bit reflects the value of the $\overline{\text{SPI\_EN}}$ pin.
		0	Current value of $\overline{\text{SPI\_EN}}$ pin is logic 0.
		1	Current value of $\overline{\text{SPI\_EN}}$ pin is logic 1.
7-2	Reserved	0	Reads return zero and writes have no effect.
1	SCSDIN1		$\overline{\text{SPI\_CS1}}$ data in. This bit reflects the value of the $\overline{\text{SPI\_CS1}}$ pin.
		0	Current value of $\overline{\text{SPI\_CS1}}$ pin is logic 0.
		1	Current value of $\overline{\text{SPI\_CS1}}$ pin is logic 1.
0	SCSDIN0		$\overline{\text{SPI\_CS0}}$ data in. This bit reflects the value of the $\overline{\text{SPI\_CS0}}$ pin.
		0	Current value of $\overline{\text{SPI\_CS0}}$ pin is logic 0.
		1	Current value of $\overline{\text{SPI\_CS0}}$ pin is logic 1.

3.8 SPI Shift Register 1 (SPIDAT1)

The SPI shift register 1 SPIDAT1) is shown in Figure 19 and described in Table 13.

Figure 19. SPI Shift Register 1 (SPIDAT1)

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

15 0
TXDATA
R/W-0

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

Table 13. SPI Shift Register 1 (SPIDAT1) Field Descriptions

Bit	Field	Value	Description
31-29	Reserved	0	Reads return zero and writes have no effect.
28	CSHOLD		Chip select hold mode. In SPI, the CSHOLD bit is supported in master mode only. In slave mode, this bit is ignored. CSHOLD defines the behavior of the chip select line at the end of a data transfer.
		0	The 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 will be deactivated for at least 2 SYSCLK3 cycles before it is activated again.
		1	The 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 SPIDAT1. If the new chip select information equals the previous one, 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	Reads return zero and writes have no effect.
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, $\overline{\text{SPI\_CS}}$ pins will still be de-activated for at least for 2 SYSCLK3 cycles if CSHOLD = 0.
			Note: In SPI, the duration for which the $\overline{\text{SPI\_CS}}$ 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 $\overline{\text{SPI\_CS}}$ pins will be de-asserted for at least 2 SYSCLK3 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 $\overline{\text{SPI\_CS}}$ pins will be de-activated for at least (WDELAY + 2) × SYSCLK3_Period duration.
25-24	DFSEL	0-3h	Data word format select.
		0	Data word format 0 is selected .
		1h	Data word format 1 is selected.
		2h	Data word format 2 is selected.
		3h	Data word 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 $\overline{\text{SPI\_CSn}}$ signals are active low.
		0	Both chip select $\overline{\text{SPI\_CS0}}$ and $\overline{\text{SPI\_CS1}}$ are selected.
		1h	Chip select $\overline{\text{SPI\_CS1}}$ is selected only.
		2h	Chip select $\overline{\text{SPI\_CS0}}$ is selected only.
		3h	No chip is selected.

Table 13. SPI Shift Register 1 (SPIDAT1) Field Descriptions (continued)

Bit	Field	Value	Description
15-0	TXDATA	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 register will drive the contents of CSNR [1:0] into the respective pins in $\overline{\text{SPI\_CS}}$[1:0] if those are configured as functional pins. 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 20 and described in Table 14. Reading SPIBUF clears the OVRNINTFLG and RXINTFLG bits in the SPI flag status register (SPIFLG).

Figure 20. SPI Buffer Register (SPIBUF)

31	30	29	28	27	26	25	24
RXEMPTY	RXOVR	TXFULL	BITERR	DESYNC	PARITYERR	TIMEOUT	DLENERR
RS-1	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	RC-0

23	16
LCSNR	
R-0	

15	0
RXDATA	
R-0	

LEGEND: R/W = Read/Write; R = Read only; RC = Clear on read; S = Set; -*n* = value after reset

Table 14. SPI Buffer Register (SPIBUF) Field Descriptions

Bit	Field	Value	Description
31	RXEMPTY		RXEMPTY. Receive data buffer empty. When host reads the SPIBUF field or the entire SPIBUF, this will automatically set the RXEMPTY flag. When a data transfer is completed, the received data is copied into SPIBUF, the RXEMPTY flag is cleared.
		0	New data has been received and copied into the SPIBUF field.
		1	No data received since last reading of 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 INTVEC*n* 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.

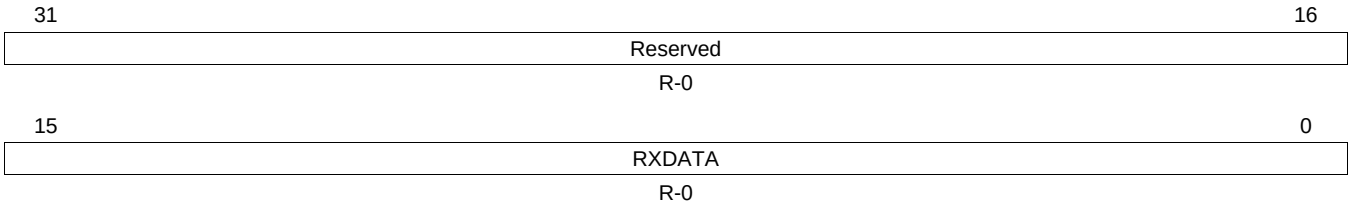
Table 14. SPI Buffer Register (SPIBUF) Field Descriptions (continued)

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	A bit error occurred. The SPI samples the signal of the transmit pin (master: SPI_SIMO, slave: SPI_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	DESYNC		Desynchronization of slave device. This bit is active in master mode only.
			Note: Possible inconsistency of DESYNC flag in SPI. Because of the nature of this error, under some circumstances it is possible for a desync error detected for the previous buffer to be visible in the current buffer. This is because the receive completion flag/interrupt will be generated when the buffer transfer is completed. But desync will be detected after the buffer transfer is completed. So, if CPU/EDMA reads the received data quickly when an RXINT is detected, then the status flag may not reflect the correct desync condition. This inconsistency in the desync flag is valid only in the SPI.
		0	No slave de-synchronization detected.
			Note: This flag is cleared to 0 when the RXDATA field of SPIBUF is read.
		1	A slave device is desynchronized. The master monitors the ENA signal coming from the slave device and sets the DESYNC flag if ENA is deactivated before the last reception point or after the last bit is transmitted plus $t_{T2EDELAY}$ (see Section 3.11). If DESYNCENA is set, an interrupt is asserted. Desynchronization can occur if a slave device misses a clock edge coming from the master.
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 (see Section 3.13). 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	TIMEOUT		Time-out because of non-activation of ENA pin.
			Note: This bit is valid in master mode only.
		0	No ENA-pin time-out occurred.
			Note: This flag is cleared to 0 when RXDATA field of SPIBUF is read.
		1	An ENA signal time-out occurred. The SPI generates a time-out because the slave has not responded in time by activating the ENA signal after the chip select signal has been activated. If a time-out condition is detected, the corresponding chip select is deactivated immediately and the TIMEOUT flag is set. In addition the TIMOUT flag in the status field of the corresponding buffer and in SPIFLG is set.
24	DLENERR		Data length error flag.
		0	No data length error has occurred.
			Note: This flag is cleared to 0 when the RXDATA field of SPIBUF is read.
		1	A data length error has occurred.
23-16	LCSNR	0-FFh	Last chip select number. LCSNR in the status field is a copy of CSNR 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 from the kernel of SPI after transmission during write back of received data.
15-0	RXDATA	0-FFFFh	SPI receive data. This is the received data, transferred from the receive shift-register at the end of a transfer completion. Irrespective of the programmed character length and the direction of shifting, the received data is stored right-justified in the register.

3.10 SPI Emulation Register (SPIEMU)

The SPI emulation register (SPIEMU) is shown in Figure 21 and described in Table 15.

Figure 21. SPI Emulation Register (SPIEMU)



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

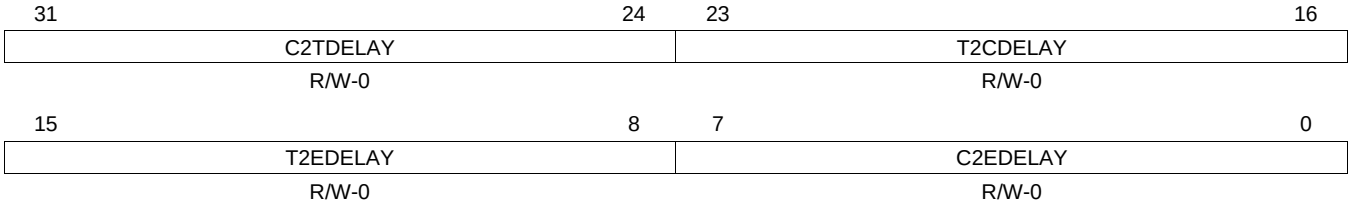
Table 15. SPI Emulation Register (SPIEMU) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reads return zero and writes have no effect.
15-0	RXDATA	0-FFFFh	SPI receive data. 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 any of the status flags.

3.11 SPI Delay Register (SPIDELAY)

The SPI delay register (SPIDELAY) is shown in Figure 22 and described in Table 16.

Figure 22. SPI Delay Register (SPIDELAY)



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

Table 16. SPI Delay Register (SPIDELAY) Field Descriptions

Bit	Field	Value	Description
31-24	C2TDELAY	0-FFh	Chip-select-active-to-transmit-start-delay. See Figure 23 for an example. C2TDELAY is used in master mode only. It defines a setup time for the slave device that delays the data transmission from the chip select active edge by a multiple of SYSCLK3 cycles. C2TDELAY can be configured between 2 and 256 SYSCLK3 cycles. The setup time value is calculated as follows: $t_{C2TDELAY}$ = (C2TDELAY + 2) * SYSCLK3 Period **Note:** If C2TDELAY = 0, then $t_{C2TDELAY}$ = 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 SCS fall-edge to the first edge of SPI_CLK will have an additional 0.5 SPI_CLK period delay. This delay is as per the SPI protocol.

Table 16. SPI Delay Register (SPIDELAY) Field Descriptions (continued)

Bit	Field	Value	Description
23-16	T2CDELAY	0-FFh	Transmit-end-to-chip-select-inactive-delay. See Figure 24 for an example. 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 SYSCLK3 cycles after the last bit is transferred. T2CDELAY can be configured between 2 and 256 SYSCLK3 cycles. The hold time value is calculated as follows: $t_{T2CDELAY}$ = (T2CDELAY + 1) * SYSCLK3 Period **Note:** If T2CDELAY = 0, then $t_{T2CDELAY}$ = 0 **Note:** If phase = 0, then between the last edge of SPI_CLK and rise-edge of SCS there will be an additional delay of 0.5 SPI_CLK period. This is as per the SPI protocol. Both C2TDELAY and T2CDELAY counters will not have any dependancy on the $\overline{\text{SPI\_EN}}$ pin value. Even if the $\overline{\text{SPI\_EN}}$ pin is asserted by the slave, the master will continue to delay the start of SPI_CLK until the C2TDELAY counter overflows. Similarly, even if the $\overline{\text{SPI\_EN}}$ pin is de-asserted by the slave, the master will continue to hold the $\overline{\text{SPI\_CS}}$ pins active until the T2CDELAY counter overflows. This way, it is guaranteed that the setup/hold times of the $\overline{\text{SPI\_CS}}$ pins is determined by the delay timers alone. To achieve better throughput, it should be ensured that these two timers are kept at the minimum possible values.
15-8	T2EDELAY	0-FFh	Transmit-data-finished-to-ENA-pin-inactive-time-out. See Figure 25 for an example. T2EDELAY is used in master mode only. It defines a time-out value as a multiple of SPI clock before the ENABLE signal has to become inactive and after the CS becomes inactive. The SPI clock depends on which data format is selected. If the slave device is missing one or more clock edges, it is becoming de-synchronized. Although the master has finished the data transfer the slave is still waiting for the missed clock pulses and the ENA signal is not disabled. The T2EDELAY defines a time-out value that triggers the DESYNC flag, if the ENA signal is not deactivated in time. DESYNC flag is set to indicate that the slave device did not de-assert its $\overline{\text{SPI\_EN}}$ pin in time to acknowledge that it has received all the bits of the sent character. DESYNC flag is also set if SPI detects a de-assertion of $\overline{\text{SPI\_EN}}$ pin even before the end of the transmission. The time-out value is calculated as follows: $t_{T2EDELAY}$ = T2EDELAY/SPIclock
7-0	C2EDELAY	0-FFh	Chip-select-active-to-ENA-signal-active-time-out. See Figure 26 for an example. C2EDELAY is used only in master mode and it applies only if the addressed slave generates an 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 ENA signal. C2EDELAY defines a time-out value as a multiple of SPI clocks. The SPI clock depends on whether data format 0 or data format 1 is selected. **Note:** If the slave device is not responding with the ENA signal before the time-out value is reached, the TIMEOUT flag in SPIFLG is set and a interrupt is asserted if enabled. If a time-out occurs the SPI clears the transmit request of the timed-out buffer, sets the TIMEOUT flag for the current buffer and continues with the transfer of the next buffer in the sequence that is enabled. The timeout value is calculated as follows: $t_{C2EDELAY}$ = C2EDELAY/SPIclock

Figure 23. Example Waveform: $t_{C2TDELAY}$ = 8 SYSCLK3 Cycle

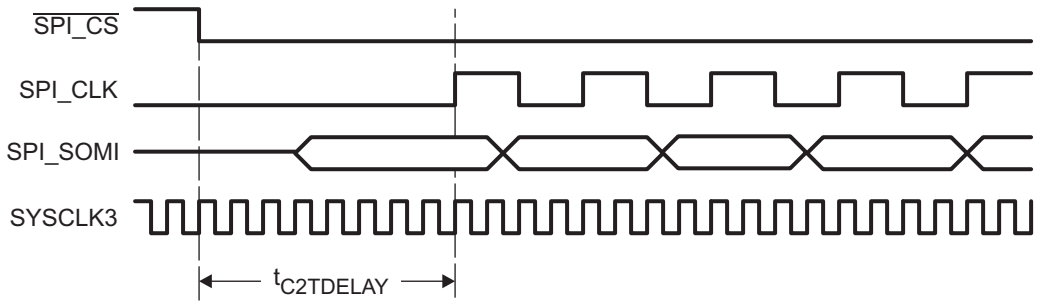


Figure 24. Example Waveform: $t_{T2CDELAY}$ = 4 SYSCLK3 Cycles

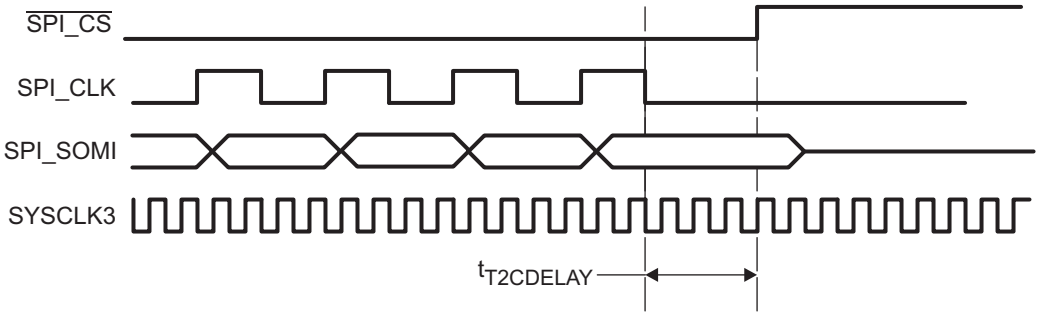


Figure 25. Transmit-Data-Finished-to-ENA-Inactive-Timeout

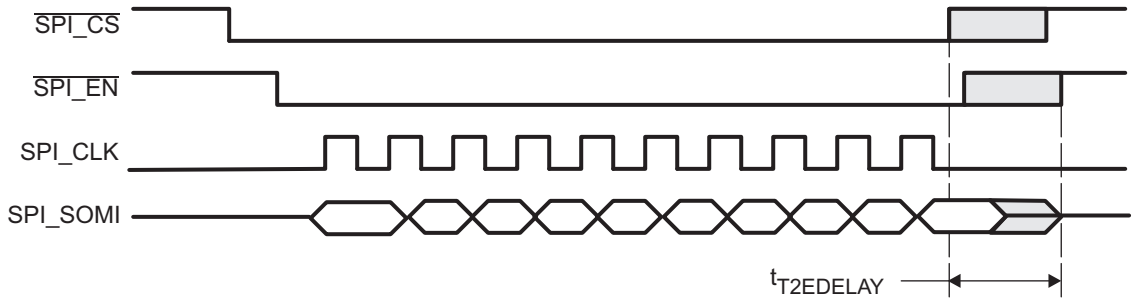
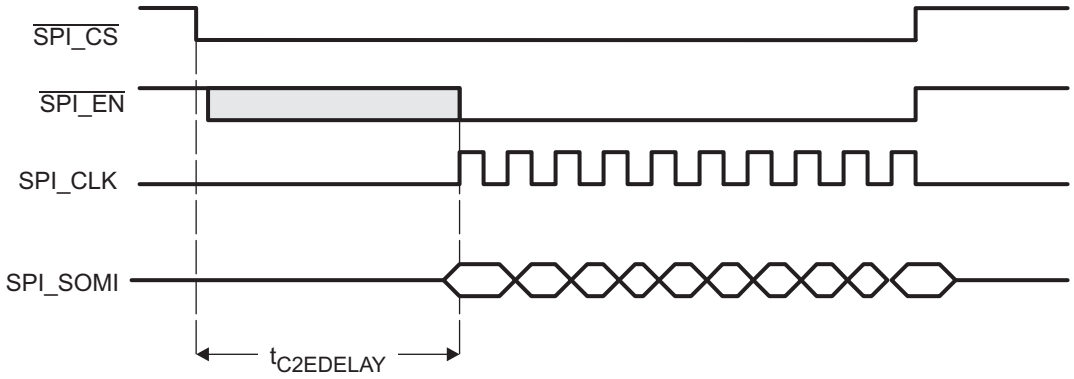


Figure 26. Chip-Select-Active-to-ENA-Signal-Active-Timeout



3.12 SPI Default Chip Select Register (SPIDEF)

The SPI default chip select register (SPIDEF) is shown in Figure 27 and described in Table 17.

Figure 27. SPI Default Chip Select Register (SPIDEF)

31	16
Reserved	
R-0	

15	2	1	0
Reserved		CSDEF1	CSDEF0
R-0		R/W-1	R/W-1

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

Table 17. 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	CSDEF1		Master mode behavior. The CSDEF1 bits are output to the chip select pins when no transmission are currently performed. It allows the user to set a chip select pattern which deselects all the SPI slaves.
		0	If CSDEF1 is cleared to 0 the corresponding chip select is cleared to 0 when no transfer occurs.
		1	If CSDEF1 is set to 1 the corresponding chip select is set to 1 when no transfer occurs.
0	CSDEF0		Master mode behavior. The CSDEF0 bits are output to the chip select pins when no transmission are currently performed. It allows the user to set a chip select pattern which deselects all the SPI slaves.
		0	If CSDEF0 is cleared to 0 the corresponding chip select is cleared to 0 when no transfer occurs.
		1	If CSDEF0 is set to 1 the corresponding chip select is set to 1 when no transfer occurs.

3.13 SPI Data Format Registers (SPIFMT*n*)

The SPI data format register (SPIFMT0, SPIFMT1, SPIFMT2, and SPIFMT3) is shown in Figure 28 and described in Table 18.

Figure 28. SPI Data Format Register (SPIFMT*n*)

31	30	29					24
Reserved		WDELAY*n*					
R-0		R/WP-0					
23	22	21	20	19	18	17	16
PARPOL*n*	PARITYENA*n*	WAITENA*n*	SHIFTDIR*n*	Reserved	DISCSTIMERS*n*	POLARITY*n*	PHASE*n*
R/WP-0	R/WP-0	R/WP-0	R/WP-0	R-0	R/WP-0	R/WP-0	R/WP-0
15							8
PRESCALE*n*							
R/WP-0							
7		5	4				0
Reserved			CHARLEN*n*				
R-0			R/WP-0				

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

Table 18. SPI Data Format Register (SPIFMT*n*) Field Descriptions

Bit	Field	Value	Description
31-30	Reserved	0	Reads return zero and writes have no effect.
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 × $P_{SYSCLK3}$ + 2 × $P_{SYSCLK3}$ $P_{SYSCLK3}$ = Period of SYSCLK3
23	PARPOL*n*		Parity polarity: even or odd. PARPOL*n* 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	WAITENA*n*		The master waits for ENA signal from slave for data format *n*. WAITENA is used in master mode only; in slave mode, this bit has no meaning. WAITENA enables a flexible SPI network where slaves with ENA signal and slaves without ENA signal can be mixed. WAITENA defines for each buffer whether the addressed slave generates the ENA signal or not.
		0	The SPI does not wait for the ENA signal from the slaves and directly starts the transfer.
		1	Before the SPI starts the data transfer it waits for the ENA signal to become low. If the ENA signal is not pulled down by the addressed slave before the internal time-out counter (C2EDELAY) overflows, then the master aborts the transfer and sets the TIMEOUT error flag.
20	SHIFTDIR*n*		Shift direction for data format *n*. With bit SHIFTDIR*n*, the shift direction for data format *n* (n = 0,1,2,3) can be selected.
		0	Data format *n* shift direction: Most-significant bit is shifted out first.
		1	Data format *n* shift direction: Least-significant bit is shifted out first.
19	Reserved	0	Reads return zero and writes have no effect.

Table 18. SPI Data Format Register (SPIFMT*n*) Field Descriptions (continued)

Bit	Field	Value	Description
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. 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*		SPI data format *n* clock polarity. POLARITY*n* defines the clock polarity of data format *n*.
		0	The SPI clock signal is low-inactive, that is, before and after data transfer the clock signal is low.
		1	The SPI clock signal is high-inactive, that is, before and after data transfer the clock signal is high.
16	PHASE*n*		SPI Data format *n* clock delay. PHASEx defines the clock delay of data format *n*. The following restrictions apply when switching clock phase and/or polarity: 1. In 3-pin/4-pin with nENA pin configuration of a slave SPI, the clock phase and polarity cannot be changed on the fly between two transfers. The slave should be reset and re-configured if clock phase/polarity needs to be switched. In summary, SPI format switching is not fully supported in slave mode. 2. Even while using chip select pins, the polarity of SPI_CLK can be switched only while the slave is not selected by a valid chip select. The master SPI should ensure that while switching SPI_CLK polarity, it has deselected all of its slaves. Otherwise, the switching of SPI_CLK polarity could be treated as a valid clock by some slaves.
		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	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. The master and slave receive the first bit with the first edge.
15-8	PRESCALE*n*	3-FFh	SPI data format *n* prescaler. PRESCALE*n* determines the bit transfer rate of data format *n* if the SPI is the network master. PRESCALE*n* is directly derived from SYSCLK3. If the SPI is configured as slave, PRESCALE*n* does not need to be configured. PRESCALE*n* is only supported for values >2.
7-5	Reserved	0	Reads return 0 and writes have no effect.
4-0	CHARLEN*n*	0-1Fh	SPI data format *n* data word length. CHARLEN*n* defines the word length of data format *n*. Legal values are 2h (data word length = 2 bits) to 10h (data word length = 16 bits). Illegal values, such as 0 or 1Fh are not detected and their effect is indeterminate.

3.14 SPI Interrupt Vector Register 0 (INTVEC0)

The SPI interrupt vector register 0 (INTVEC0) is shown in Figure 29 and described in Table 19.

Figure 29. SPI Interrupt Vector Register 0 (INTVEC0)

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 19. SPI Interrupt Vector Register 0 (INTVEC0) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return zero and writes have no effect.
5-1	INTVECT0	0-1Fh	Interrupt vector for interrupt line SPINT0. INTVECT0 returns the vector of the pending interrupt at interrupt line SPINT0. 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 SPINT0 in SPI.
		0	No interrupt pending
		1h-10h	Reserved
		11h	Error interrupt pending. Refer to lower halfword of SPIINT to determine 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). Every 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 (INTVEC1)

The SPI interrupt vector register 1 (INTVEC1) is shown in Figure 30 and described in Table 20.

Figure 30. SPI Interrupt Vector Register 1 (INTVEC1)

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 20. SPI Interrupt Vector Register 1 (INTVEC1) Field Descriptions

Bit	Field	Value	Description
31-6	Reserved	0	Reads return zero and writes have no effect.
5-1	INTVECT1	0-1Fh	Interrupt vector for interrupt line SPINT1. INTVECT1 returns the vector of the pending interrupt at interrupt line SPINT1. 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. • 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 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 SPINT1 in SPI.
		0	No interrupt pending
		1h-10h	Reserved
		11h	Error interrupt pending. Refer to lower halfword of SPIINT to determine 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		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

Table 21 lists the changes made since the previous version of this document.

Table 21. Document Revision History

Reference	Additions/Modifications/Deletions
Section 2.1	Changed fourth paragraph.

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**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com	Wireless	www.ti.com/wireless-apps
RF/IF and ZigBee® Solutions	www.ti.com/lprf		
	TI E2E Community Home Page		e2e.ti.com

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