

# TMS320DM357 DMSoC Serial Peripheral Interface (SPI)

## User's Guide



Literature Number: SPRUG29

November 2008



|   |           |
|---|-----------|
| <b>Preface</b> .....                                  | <b>6</b>  |
| <b>1 Introduction</b> .....                           | <b>9</b>  |
| 1.1 Purpose of the Peripheral .....                   | 9         |
| 1.2 Features .....                                    | 9         |
| 1.3 Functional Block Diagram .....                    | 10        |
| 1.4 Industry Standard(s) Compliance Statement .....   | 10        |
| <b>2 Peripheral Architecture</b> .....                | <b>10</b> |
| 2.1 Clock Control .....                               | 10        |
| 2.2 Signal Descriptions .....                         | 11        |
| 2.3 Pin Multiplexing .....                            | 11        |
| 2.4 SPI Operation .....                               | 11        |
| 2.5 Reset Considerations .....                        | 18        |
| 2.6 Initialization .....                              | 18        |
| 2.7 Interrupt Support .....                           | 19        |
| 2.8 EDMA Event Support .....                          | 20        |
| 2.9 Power Management .....                            | 21        |
| 2.10 SPI Internal Loop-Back Test Mode.....            | 21        |
| 2.11 Emulation Considerations .....                   | 21        |
| <b>3 Registers</b> .....                              | <b>22</b> |
| 3.1 SPI Global Control Register 0 (SPIGCR0) .....     | 22        |
| 3.2 SPI Global Control Register 1 (SPIGCR1) .....     | 23        |
| 3.3 SPI Interrupt Register (SPIINT).....              | 24        |
| 3.4 SPI Interrupt Level Register (SPILVL) .....       | 25        |
| 3.5 SPI Flag Register (SPIFLG).....                   | 26        |
| 3.6 SPI Pin Control Register (SPIPC0) .....           | 27        |
| 3.7 SPI Pin Control Register 2 (SPIPC2) .....         | 28        |
| 3.8 SPI Shift Register (SPIDAT1) .....                | 29        |
| 3.9 SPI Buffer Register (SPIBUF).....                 | 30        |
| 3.10 SPI Emulation Register (SPIEMU) .....            | 31        |
| 3.11 SPI Delay Register (SPIDELAY) .....              | 32        |
| 3.12 SPI Default Chip Select Register (SPIDEF) .....  | 33        |
| 3.13 SPI Data Format Registers (SPIFMTn).....         | 34        |
| 3.14 SPI Interrupt Vector Register 0 (INTVECT0) ..... | 35        |
| 3.15 SPI Interrupt Vector Register 1 (INTVECT1) ..... | 36        |

## List of Figures

|    |  |    |
|----|--|----|
| 1  | Serial Peripheral Interface (SPI) Block Diagram .....                                  | 10 |
| 2  | Right-Aligned Transmit Data in SPIDAT1 Field of the SPI Shift Register (SPIDAT1) ..... | 12 |
| 3  | Right-Aligned Receive Data in SPIBUF Field of the SPI Buffer Register (SPIBUF) .....   | 12 |
| 4  | Clock Mode with POLARITY = 0 and PHASE = 0 .....                                       | 13 |
| 5  | Clock Mode with POLARITY = 0 and PHASE = 1 .....                                       | 13 |
| 6  | Clock Mode with POLARITY = 1 and PHASE = 0 .....                                       | 14 |
| 7  | Clock Mode with POLARITY = 1 and PHASE = 1 .....                                       | 14 |
| 8  | Five Bits per Character (Four-Pin Option) .....  | 15 |
| 9  | SPI Operation (3-Pin Option) .....   | 17 |
| 10 | SPI Operation (4-Pin Option) .....   | 17 |
| 11 | SPI Global Control Register 0 (SPIGCR0).....   | 22 |
| 12 | SPI Global Control Register 1 (SPIGCR1).....   | 23 |
| 13 | SPI Interrupt Register (SPIINT) .....  | 24 |
| 14 | SPI Interrupt Level Register (SPILVL).....   | 25 |
| 15 | SPI Flag Register (SPIFLG) .....   | 26 |
| 16 | SPI Pin Control Register (SIPIC0) .....  | 27 |
| 17 | SPI Pin Control Register 2 (SIPIC2).....   | 28 |
| 18 | SPI Shift Register (SPIDAT1).....  | 29 |
| 19 | SPI Buffer Register (SPIBUF) .....   | 30 |
| 20 | SPI Emulation Register (SPIEMU).....   | 31 |
| 21 | SPI Delay Register (SPIDELAY) .....  | 32 |
| 22 | SPI Default Chip Select Register (SPIDEF) .....  | 33 |
| 23 | SPI Data Format Register (SPIFMT $n$ ).....  | 34 |
| 24 | SPI Interrupt Vector Register 0 (INTVECT0) .....                                       | 35 |
| 25 | SPI Interrupt Vector Register 1 (INTVECT1) .....                                       | 36 |

## List of Tables

|    |  |    |
|----|--|----|
| 1  | Serial Peripheral Interface (SPI) Pins .....                             | 11 |
| 2  | SPI Clocking Modes .....   | 12 |
| 3  | SPI Registers .....  | 22 |
| 4  | SPI Global Control Register 0 (SPIGCR0) Field Descriptions .....         | 22 |
| 5  | SPI Global Control Register 1 (SPIGCR1) Field Descriptions .....         | 23 |
| 6  | SPI Interrupt Register (SPIINT) Field Descriptions .....                 | 24 |
| 7  | SPI Interrupt Level Register (SPILVL) Field Descriptions .....           | 25 |
| 8  | SPI Flag Register (SPIFLG) Field Descriptions .....                      | 26 |
| 9  | SPI Pin Control Register (SPIPC0) Field Descriptions .....               | 27 |
| 10 | SPI Pin Control Register 2 (SPIPC2) Field Descriptions.....              | 28 |
| 11 | SPI Shift Register (SPIDAT1) Field Descriptions .....                    | 29 |
| 12 | SPI Buffer Register (SPIBUF) Field Descriptions .....                    | 30 |
| 13 | SPI Emulation Register (SPIEMU) Field Descriptions.....                  | 31 |
| 14 | SPI Delay Register (SPIDELAY) Field Descriptions.....                    | 32 |
| 15 | SPI Default Chip Select Register (SPIDEF) Field Descriptions .....       | 33 |
| 16 | SPI Data Format Register (SPIFMT <sub>n</sub> ) Field Descriptions ..... | 34 |
| 17 | SPI Interrupt Vector Register 0 (INTVECT0) Field Descriptions .....      | 35 |
| 18 | SPI Interrupt Vector Register 1 (INTVECT1) Field Descriptions .....      | 36 |

## Read This First

---

---

---

### About This Manual

This document describes the serial peripheral interface (SPI) in the TMS320DM357 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 TMS320DM357 Digital Media System-on-Chip (DMSoC). Copies of these documents are available on the Internet at [www.ti.com](http://www.ti.com). *Tip:* Enter the literature number in the search box provided at [www.ti.com](http://www.ti.com).

**[SPRUG06](#)** — ***TMS320DM357 DMSoC Video Processing Back End (VPBE) User's Guide***. Describes the video processing back end (VPBE) in the TMS320DM357 Digital Media System-on-Chip (DMSoC) video processing subsystem. Included in the VPBE is the video encoder, on-screen display, and digital LCD controller.

**[SPRUG25](#)** — ***TMS320DM357 DMSoC ARM Subsystem Reference Guide***. Describes the ARM subsystem in the TMS320DM357 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 video processing subsystem, and a majority of the peripherals and external memories.

**[SPRUG26](#)** — ***TMS320DM357 DMSoC Universal Asynchronous Receiver/Transmitter (UART) User's Guide***. This document describes the universal asynchronous receiver/transmitter (UART) peripheral in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The UART peripheral performs serial-to-parallel conversion on data received from a peripheral device, and parallel-to-serial conversion on data received from the CPU.

**[SPRUG27](#)** — ***TMS320DM357 DMSoC Inter-Integrated Circuit (I2C) Peripheral User's Guide***. Describes the inter-integrated circuit (I2C) peripheral in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The I2C peripheral provides an interface between the DMSoC and other devices compliant with the I2C-bus specification and connected by way of an I2C-bus. External components attached to this 2-wire serial bus can transmit and receive up to 8-bit wide data to and from the DMSoC through the I2C peripheral. This document assumes the reader is familiar with the I2C-bus specification.

- [SPRUG28](#)** — ***TMS320DM357 DMSoC 64-Bit Timer User's Guide***. Describes the operation of the software-programmable 64-bit timer in the TMS320DM357 Digital Media System-on-Chip (DMSoC). Timer 0 and Timer 1 are used as general-purpose (GP) timers and can be programmed in 64-bit mode, dual 32-bit unchained mode, or dual 32-bit chained mode; Timer 2 is used only as a watchdog timer. The GP timer modes can be used to generate periodic interrupts or enhanced direct memory access (EDMA) synchronization events. The watchdog timer mode is used to provide a recovery mechanism for the device in the event of a fault condition, such as a non-exiting code loop.
- [SPRUG29](#)** — ***TMS320DM357 DMSoC Serial Peripheral Interface (SPI) User's Guide***. Describes the serial peripheral interface (SPI) in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The SPI is a high-speed synchronous serial input/output port that allows a serial bit stream of programmed length (1 to 16 bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communication between the DMSoC and external peripherals. Typical applications include an interface to external I/O or peripheral expansion via devices such as shift registers, display drivers, SPI EPROMs and analog-to-digital converters.
- [SPRUG30](#)** — ***TMS320DM357 DMSoC Host Port Interface (HPI) Reference Guide***. This document describes the host port interface in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The HPI provides a parallel port interface through which an external host processor can directly access the TMS320DM357 DMSoC processor's resources (configuration and program/data memories).
- [SPRUG31](#)** — ***TMS320DM357 DMSoC General-Purpose Input/Output (GPIO) User's Guide***. Describes the general-purpose input/output (GPIO) peripheral in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an input, you can detect the state of the input by reading the state of an internal register. When configured as an output, you can write to an internal register to control the state driven on the output pin.
- [SPRUG32](#)** — ***TMS320DM357 DMSoC Multimedia Card (MMC)/Secure Digital (SD) Card Controller User's Guide***. Describes the multimedia card (MMC)/secure digital (SD) card controller in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The MMC/SD card is used in a number of applications to provide removable data storage. The MMC/SD controller provides an interface to external MMC and SD cards. The communication between the MMC/SD controller and MMC/SD card(s) is performed by the MMC/SD protocol.
- [SPRUG33](#)** — ***TMS320DM357 DMSoC Asynchronous External Memory Interface (EMIF) User's Guide***. Describes the asynchronous external memory interface (EMIF) in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The EMIF supports a glueless interface to a variety of external devices.
- [SPRUG34](#)** — ***TMS320DM357 DMSoC Enhanced Direct Memory Access (EDMA) Controller User's Guide***. Describes the operation of the enhanced direct memory access (EDMA3) controller in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The EDMA3 controller's primary purpose is to service user-programmed data transfers between two memory-mapped slave endpoints on the DMSoC.
- [SPRUG35](#)** — ***TMS320DM357 DMSoC Audio Serial Port (ASP) User's Guide***. Describes the operation of the audio serial port (ASP) audio interface in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The primary audio modes that are supported by the ASP are the AC97 and IIS modes. In addition to the primary audio modes, the ASP supports general serial port receive and transmit operation, but is not intended to be used as a high-speed interface.
- [SPRUG36](#)** — ***TMS320DM357 DMSoC Ethernet Media Access Controller (EMAC)/Management Data Input/Output (MDIO) Module User's Guide***. Discusses the ethernet media access controller (EMAC) and physical layer (PHY) device management data input/output (MDIO) module in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The EMAC controls the flow of packet data from the DMSoC to the PHY. The MDIO module controls PHY configuration and status monitoring.

**[SPRUG37](#) — TMS320DM357 DMSoC Pulse-Width Modulator (PWM) Peripheral User's Guide.**

Describes the pulse-width modulator (PWM) peripheral in the TMS320DM357 Digital Media System-on-Chip (DMSoC).

**[SPRUG38](#) — TMS320DM357 DMSoC DDR2 Memory Controller User's Guide.** Describes the DDR2 memory controller in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The DDR2 memory controller is used to interface with JESD79D-2A standard compliant DDR2 SDRAM devices.

**[SPRUG39](#) — TMS320DM357 DMSoC Video Processing Front End (VPFE) User's Guide.** Describes the video processing front end (VPFE) in the TMS320DM357 Digital Media System-on-Chip (DMSoC) video processing subsystem. Included in the VPFE is the preview engine, CCD controller, resizer, histogram, and hardware 3A (H3A) statistic generator.

**[SPRUGH2](#) — TMS320DM357 DMSoC Peripherals Overview Reference Guide.** This document provides an overview of the peripherals in the TMS320DM357 Digital Media System-on-Chip (DMSoC).

**[SPRUGH3](#) — TMS320DM357 DMSoC Universal Serial Bus Controller User's Guide.** This document describes the universal serial bus (USB) controller in the TMS320DM357 Digital Media System-on-Chip (DMSoC). The USB controller supports data throughput rates up to 480 Mbps. It provides a mechanism for data transfer between USB devices and also supports host negotiation.



# **Serial Peripheral Interface (SPI)**

---

---

---

## **1 Introduction**

This document describes the serial peripheral interface (SPI) in the TMS320DM357 Digital Media System-on-Chip (DMSoC).

### **1.1 Purpose of the Peripheral**

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

The SPI allows serial communication with other SPI devices through a 3-pin or 4-pin mode interface. The DM357 DMSoC implementation supports multichip-select operation for up to two SPI slave devices. The SPI operates as a master SPI device only.

### **1.2 Features**

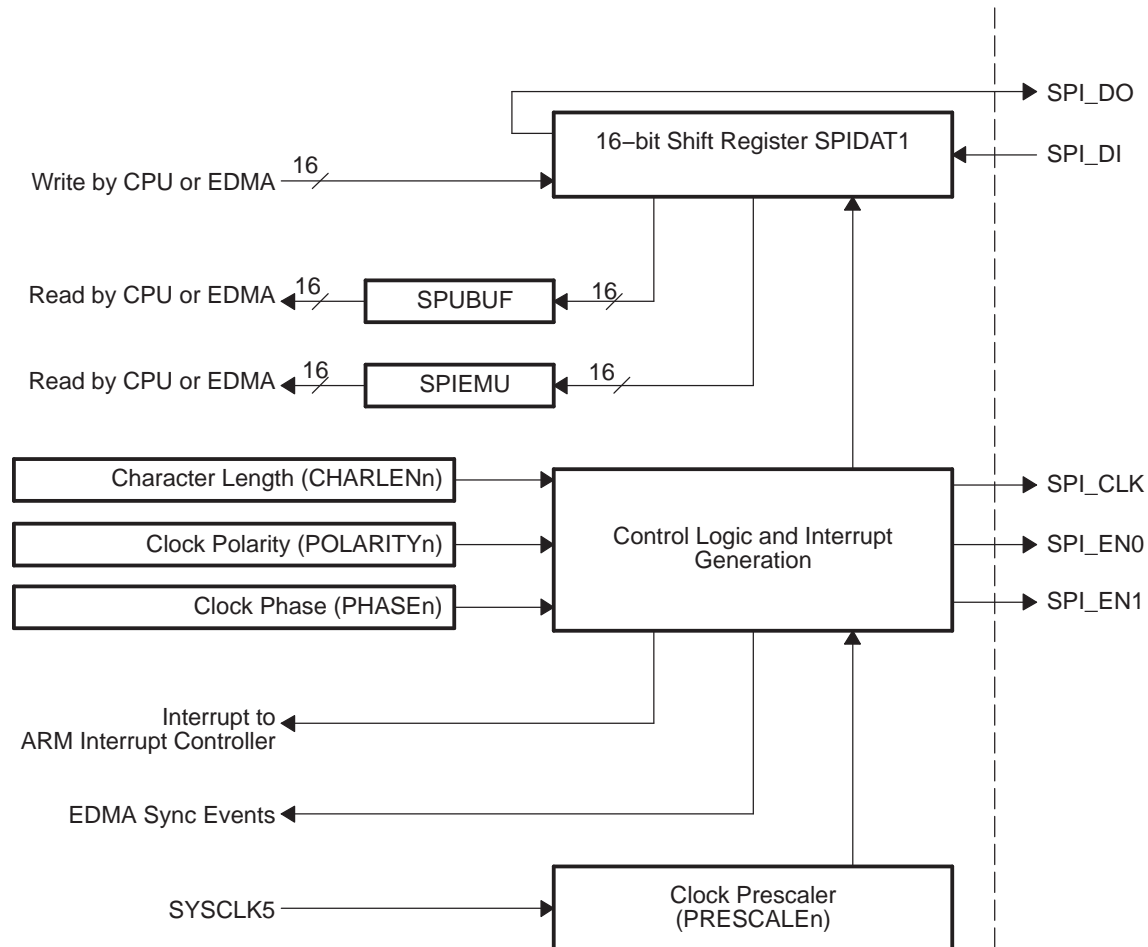
The SPI has the following features:

- 16-bit shift register
- Receive buffer register
- 8-bit clock prescaler
- Programmable SPI clock frequency range
- Programmable character length (2 to 16 bits)
- Programmable clock phase (delay or no delay)
- Programmable clock polarity (high or low)
- Two chip select signals ( $\overline{\text{SPI\_EN0}}$  and  $\overline{\text{SPI\_EN1}}$ ) provide the ability to control two slave devices

### 1.3 Functional Block Diagram

A block diagram of the major components of the SPI is shown in [Figure 1](#).

**Figure 1. Serial Peripheral Interface (SPI) Block Diagram**



### 1.4 Industry Standard(s) Compliance Statement

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

## 2 Peripheral Architecture

This section describes the architecture of the SPI.

### 2.1 Clock Control

The output clock generated (SPI\_CLK) is a derivative of the internal SYSCLK5 DMSoC clock. The maximum clock bit rate supported by the SPI peripheral is  $\text{SYSCLK5}/3$ , 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 2.4.2](#). The clock rate is set independently for each of the four software-programmable data formats. For more information on the data formats, see [Section 2.4.1](#). The clock rate for a given data format  $n$  is calculated as:

$$\text{SPI\_CLK frequency} = [\text{SYSCLK5 frequency}] / [\text{PRESCALE}n + 1]$$

PRESCALE $n$  is only supported for values  $>1$ , where the maximum SPI clock rate is  $(\text{SYSCLK5})/3$ .

## 2.2 Signal Descriptions

Table 1 shows the SPI pins used to interface to external devices. SPI\_CLK, SPI\_DO, and SPI\_DI are always used. The SPI\_EN[1:0] pins are optional and may be used if the pins are present on the slave device(s). The SPI\_EN[1:0] pins are used to selectively enable slaves in a multiple slave system. The SPI can be operated in a 3-pin or 4-pin mode configuration. In the 3-pin mode configuration, the SPI\_EN[1:0] pins are not used.

**Table 1. Serial Peripheral Interface (SPI) Pins**

| Pin                          | Type   | Function            |
|------------------------------|--------|---------------------|
| SPI_CLK                      | Output | Serial clock        |
| SPI_DI                       | Input  | Serial data input   |
| SPI_DO                       | Output | Serial data output  |
| $\overline{\text{SPI\_EN0}}$ | Output | Slave 0 chip select |
| $\overline{\text{SPI\_EN1}}$ | Output | Slave 1 chip select |

## 2.3 Pin Multiplexing

The SPI pins are multiplexed with other device functions on the DMSoC. When the SPI serial port functions are not selected, the pins may be used as general-purpose input/output (GPIO) as described in the pin multiplexing section of the device-specific data manual. In addition, the SPI\_EN1 pin is also multiplexed with an ATA controller pin function.

## 2.4 SPI Operation

The SPI operates as a master SPI device only. The MASTER and CLKMOD bits in the SPI global control register 1 (SPIGCR1) must be set to 1 for SPI module proper operation.

### 2.4.1 Data Formats

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

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

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

### 2.4.1.1 Character Length

The character length is configured by the CHARLEN $n$  bit. Legal values are 2 bits (2h) to 16 bits (10h). The character length is independently configured for each of the four data formats.

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

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

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

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

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

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

### 2.4.1.2 Shift Direction

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

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

### 2.4.2 Clock Polarity and Phase

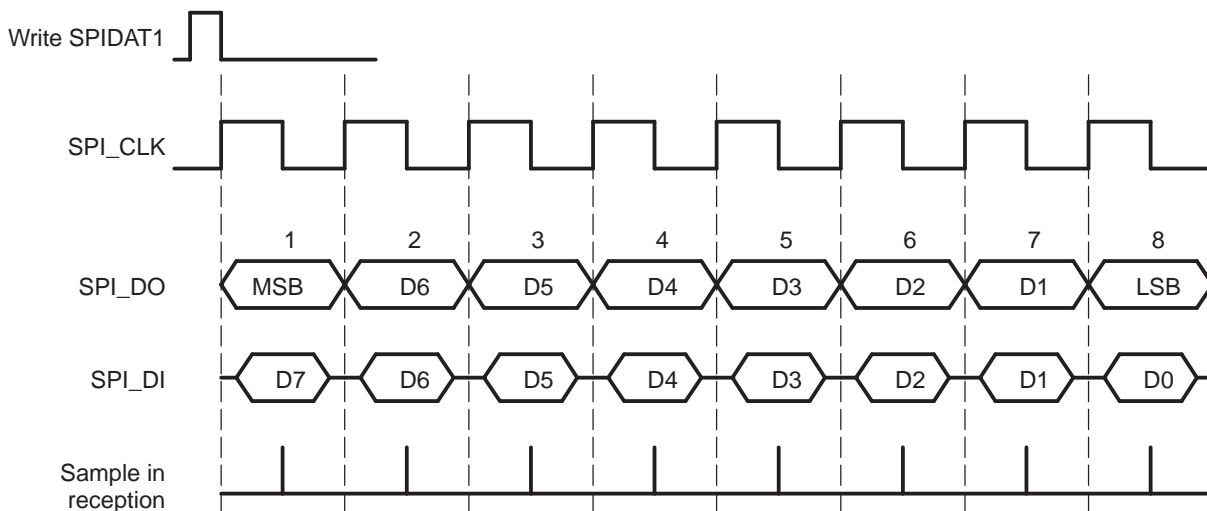
The SPI provides the flexibility to program four different clock mode combinations that SPI\_CLK may operate, enabling a choice of the clock phase (delay or no delay) and the clock polarity (rising edge or falling edge). When operating with PHASE active, the SPI makes the first bit of data available 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 |       | Action   |
|----------------|-------|--|
| POLARITY       | PHASE |  |
| 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.

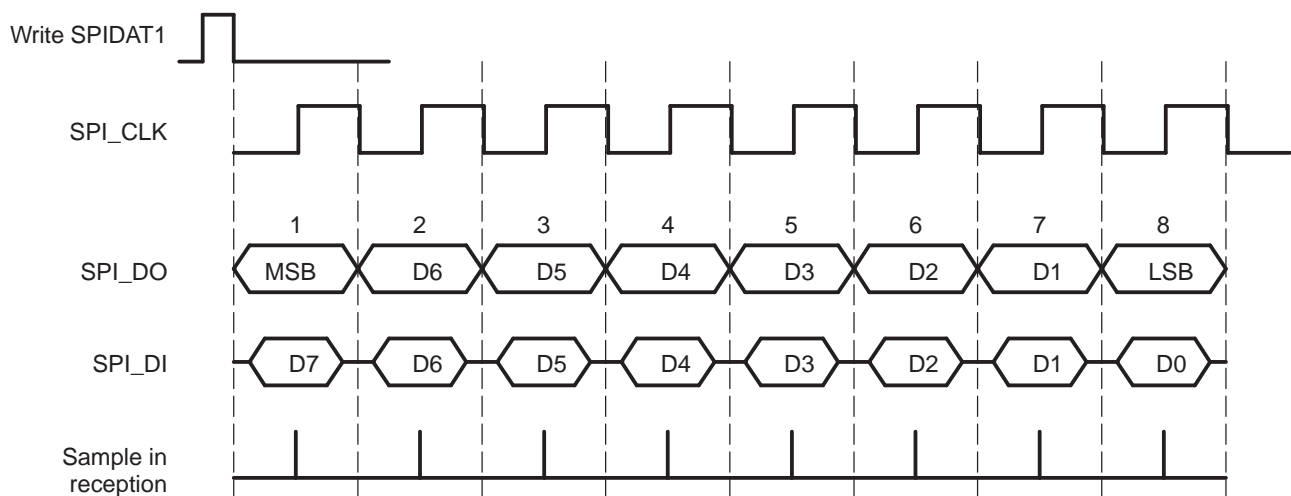
**Figure 4. Clock Mode with POLARITY = 0 and PHASE = 0**



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

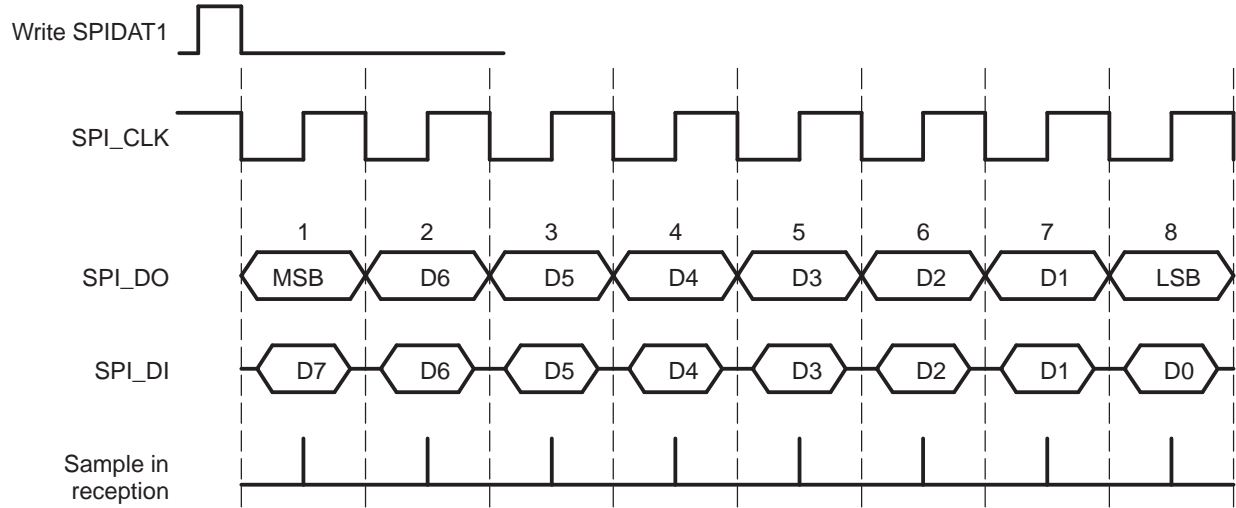
**Figure 5. Clock Mode with POLARITY = 0 and PHASE = 1**



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

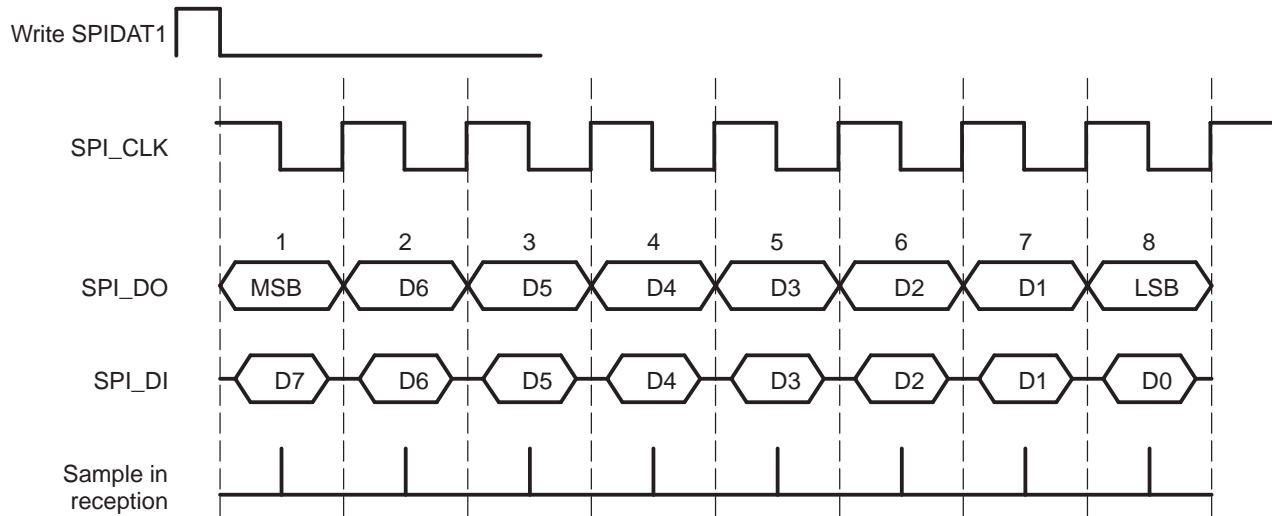
**Figure 6. Clock Mode with POLARITY = 1 and PHASE = 0**



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**

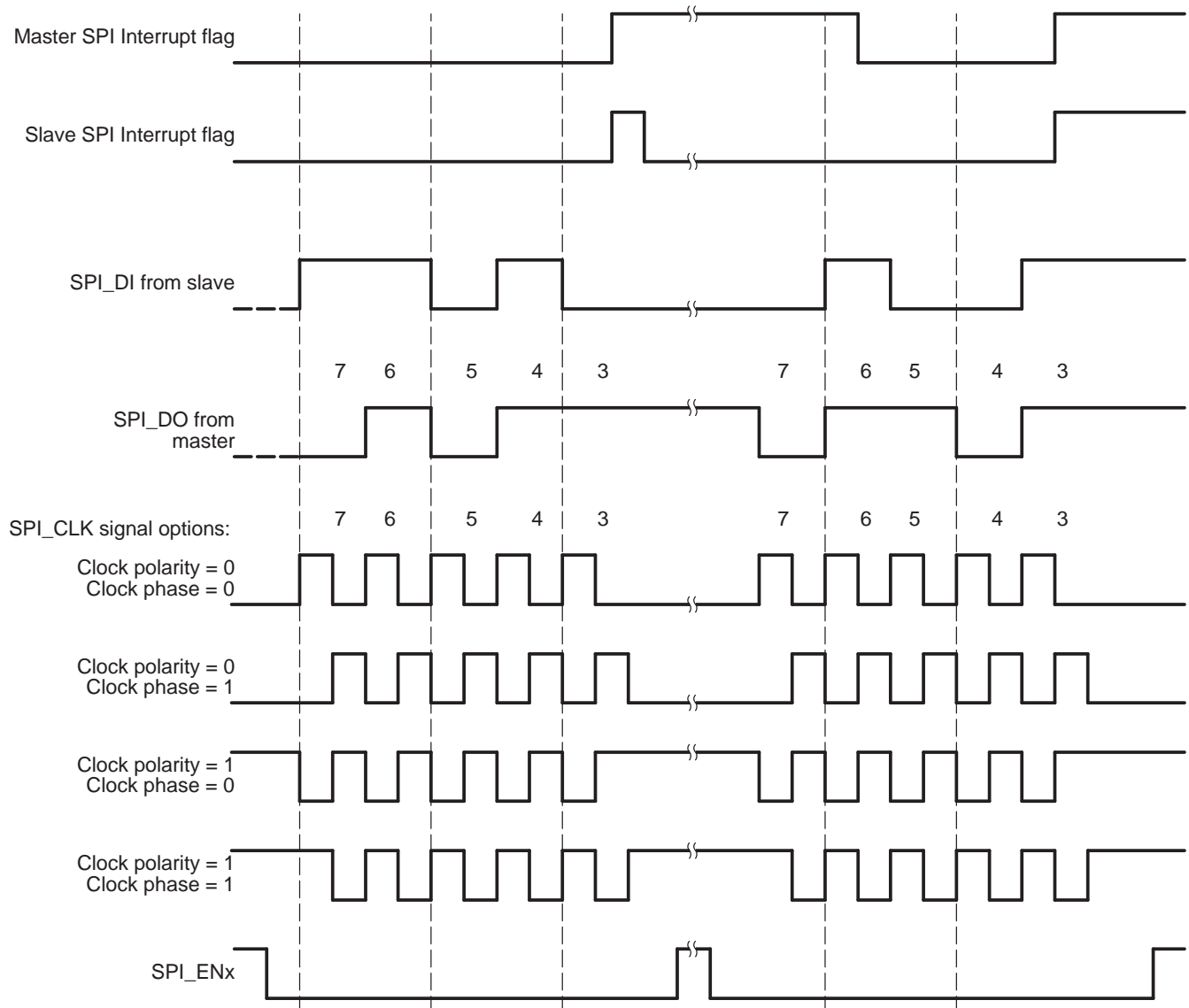


Clock phase = 1 (SPI\_CLK with delay)

- 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

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

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



### 2.4.3 Chip Select Control

The SPI provides two chip select signals ( $\overline{\text{SPI\_EN0}}$  and  $\overline{\text{SPI\_EN1}}$ ) that are used to selectively enable multiple slaves. The behavior of the chip selects is controlled by the CSNR and CSHOLD bits in SPIDAT1.

#### 2.4.3.1 Enabling Chip Selects

The CSNR bit controls which chip selects are active during the transactions that follow. This bit is used to enable either or both chip selects. To use the chip selects, the  $\text{ENnFUN}$  bit in the SPI pin control register (SPIPC0) must be set to 1 for each chip select.

### 2.4.3.2 Holding Chip Selects Active Between Transactions

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

- When CSHOLD = 0, the chip selects are 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.

### 2.4.3.3 Programming Chip Select Setup and Hold Timing

The setup time between when the chip select signal goes active and the beginning of the transaction is programmable using the C2TDELAY bit in the SPI delay register (SPIDELAY). The setup time is  $[C2TDELAY + 2]$  cycles of the SYSCLK5 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 SYSCLK5 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.

### 2.4.3.4 Inactive Chip Select State Control

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

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

## 2.4.4 SPI Operation: 3-Pin Mode

The minimum required number of signal connections for SPI communications is 3 pins: SPI\_CLK, SPI\_DI, and SPI\_DO. The chip select pins (SPI\_EN0 and SPI\_EN1) are not used in 3-pin mode. This connection could be used when a single slave is present and, therefore, no chip selects are required. The SPI operates as a SPI master and provides the serial clock on the SPI\_CLK pin during the current word transfer and stops between word transfers. Data is transmitted on the SPI\_DO pin and received from the SPI\_DI pin, as shown in [Figure 9](#).

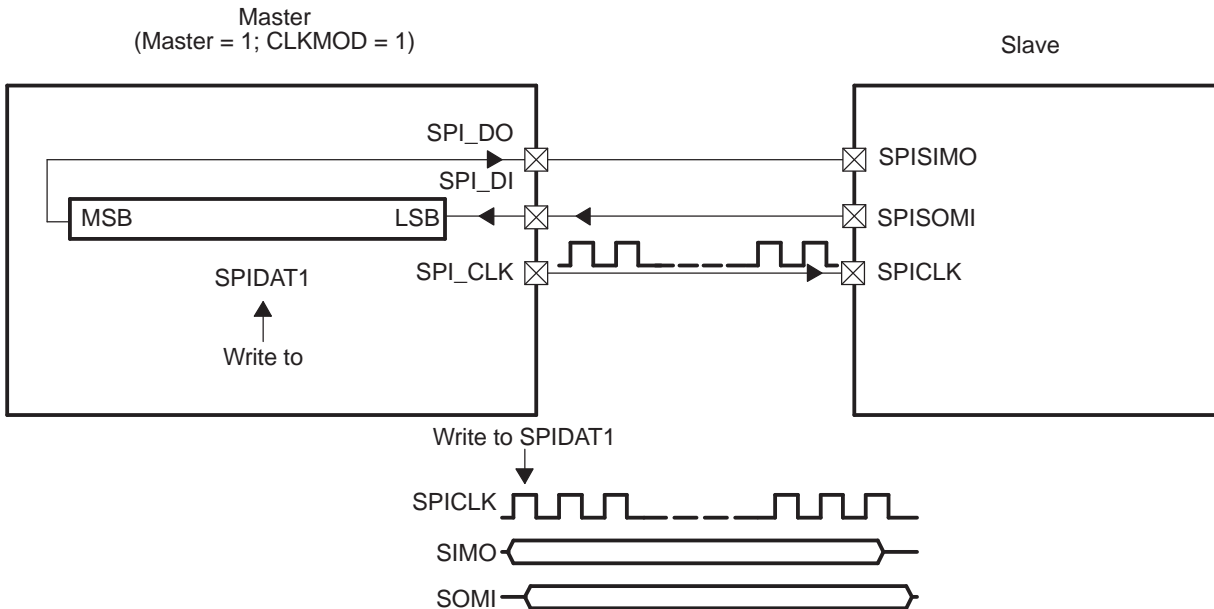
Right-aligned data written to the SPI shift register (SPIDAT1) initiates data transmission on the SPI\_DO pin. Simultaneously, received data is shifted through the SPI\_DI pin into the least-significant bit (LSB) of SPIDAT1. The SPI applies data format selected in the DFSEL bit of SPIDAT1 as the format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

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

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



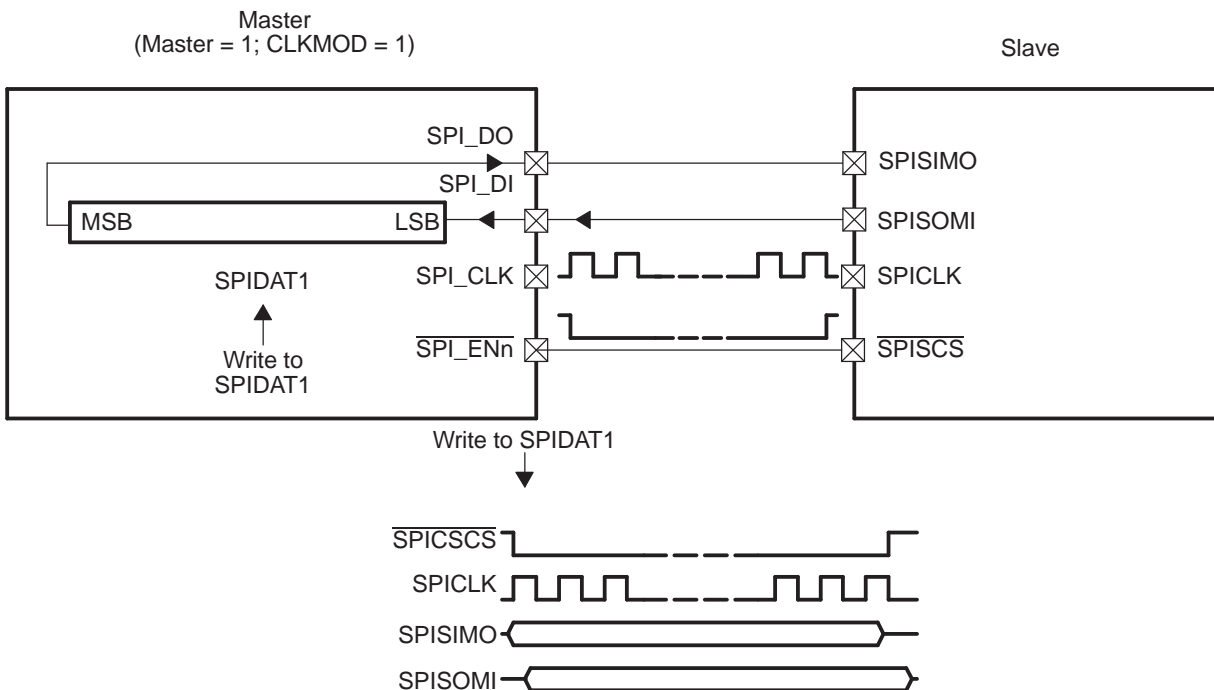
**Figure 9. SPI Operation (3-Pin Option)**



**2.4.5 SPI Operation: 4-Pin Mode**

The 3-pin mode and the 4-pin mode of the SPI are similar, except that the 4-pin mode uses the chip select pins (SPI\_EN0 and SPI\_EN1) to enable communication with multiple chip selects. Figure 10 shows how the 4-pin mode is connected. For detailed information on how the chip selects are controlled and used, see Section 2.4.3.

**Figure 10. SPI Operation (4-Pin Option)**



## 2.5 Reset Considerations

This section provides the software and hardware reset considerations.

### 2.5.1 Software Reset Considerations

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

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

### 2.5.2 Hardware Reset Considerations

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

There are two different methods to perform hardware reset that affects the SPI module register values. One is a full device hardware reset that resets all the device modules and the second is an individual peripheral hardware reset initiated by the Power and Sleep Controller (PSC) module. For information about the operation of the PSC, see the [TMS320DM357 DMSoC ARM Subsystem Reference Guide \(SPRUG25\)](#).

## 2.6 Initialization

The following section provides the initialization steps to quickly get the SPI module configured for the two different pin modes supported.

### 2.6.1 3-Pin Mode Initialization

1. Make sure the SPI module is in reset by clearing the RESET bit in the SPI global control register 0 (SPIGCR0) to 0.
2. Remove the SPI peripheral from reset by setting the RESET bit in SPIGCR0 to 1.
3. Enable the CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1).
4. Enable the SPI\_DI, SPI\_DO, and SPI\_CLK pins by setting the corresponding bits in the SPI pin control register (SPIPC0).
5. Configure the desired data format in the SPI data format register (SPIFMT $n$ ).
  - a. Program the clock prescale value in the PRESCALE $n$  bit.
  - b. Program the character size in the CHARLEN $n$  bit.
  - c. Set the SPI clock PHASE $n$  and POLARITY $n$  bits.
  - d. Set the shift direction in the SHIFTDIR $n$  bit.
6. Select the preconfigured data format using the DFSEL bit in the SPI shift register (SPIDAT1).
7. Enable the desired interrupts, if any, in the SPI interrupt register (SPIINT).
8. Select whether you want the interrupt events mapped to INT0 or INT1 using the selection bits in the SPI interrupt level register (SPILVL).
9. If using the EDMA to perform the transfers, setup and enable the EDMA channels for transmit or receive.
10. Enable the SPIENA bit in SPIGCR1.
11. If using the EDMA, set the DMAREQEN bit in SPIINT to 1 initiating the EDMA to start writing to SPIDAT1; therefore, initiating the data transfer.
12. Data is ready to be transferred using the CPU by writing to SPIDAT1.

## 2.6.2 4-Pin Mode Initialization

1. Make sure the SPI module is in reset by clearing the RESET bit in the SPI global control register 0 (SPIGCR0) to 0.
2. Remove the SPI peripheral from reset by setting the RESET bit in SPIGCR0 to 1.
3. Enable the CLKMOD and MASTER bits in the SPI global control register 1 (SPIGCR1).
4. Enable the SPI\_DI, SPI\_DO, and SPI\_CLK pins and the necessary chip select pins ( $\overline{\text{SPI\_EN0}}$  and  $\overline{\text{SPI\_EN1}}$ ) by setting the corresponding bits in the SPI pin control register (SPIPC0).
5. Configure the desired data format in the SPI data format register (SPIFMT $n$ ).
  - a. Program the clock prescale value in the PRESCALE $n$  bit.
  - b. Program the character size in the CHARLEN $n$  bit.
  - c. Set the SPI clock PHASE $n$  and POLARITY $n$  bits.
  - d. Set the shift direction in the SHIFTDIR $n$  bit.
6. Select the preconfigured data format using the DFSEL bit in the SPI shift register (SPIDAT1).
7. If needed, configure the setup or hold time for the chip select lines using the C2TDELAY or T2CDELAY bits in the SPI delay register (SPIDELAY).
8. Select the desired chip select number. The CSNR field in SPIDAT1 defines the chip select that shall be activated during the data transfer. Note that the SPI\_EN $n$  signals are active low.
9. Setup the default chip select pin value when chip select lines are inactive using the EN $n$ DEF bits in the SPI default chip select register (SPIDEF).
10. Enable the desired interrupts, if any, in the SPI interrupt register (SPIINT).
11. Select whether you want the interrupt events mapped to INT0 or INT1 using the selection bits in the SPI interrupt level register (SPILVL).
12. If using the EDMA to perform the transfers, setup and enable the EDMA channels for transmit or receive.
13. Enable the SPIENA bit in SPIGCR1.
14. If using the EDMA, set the DMAREQEN bit in SPIINT to 1 initiating the EDMA to start writing to SPIDAT1; therefore, initiating the data transfer.
15. Data is ready to be transferred using the CPU by writing to SPIDAT1.

## 2.7 Interrupt Support

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

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

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

### 2.7.1 Interrupt Events and Requests

#### 2.7.1.1 Receive Interrupt

The receive interrupt occurs when a data character has been received and copied in the SPI buffer register (SPIBUF). To enable the SPI receive interrupt, set the RXINTEN bit in the SPI interrupt register (SPIINT) to 1. To assign the receive interrupt to occur on SPIINT0, clear the RXINTLVL bit in the SPI interrupt level register (SPILVL) to 0; to assign the receive interrupt to occur on SPIINT1, set the RXINTLVL bit in SPILVL to 1.

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

### 2.7.1.2 Receive Overrun Interrupt

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

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

### 2.7.1.3 Transmit Error Interrupt

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

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

## 2.7.2 Interrupt Multiplexing

The interrupts generated by the SPI peripheral to the CPU are not multiplexed with any other interrupt sources on the DMSoC.

## 2.8 EDMA Event Support

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

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

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

In most cases, if the EDMA is being used to service received data from the SPI, the receive interrupt enable (RXINTEN) bit in SPIINT should be cleared to 0. This prevents the CPU from both responding to the received data in addition to the EDMA. For specific SPI synchronization event number and detailed EDMA features, refer to the *TMS320DM357 DMSoC Enhanced Direct Memory Access (EDMA) Controller Reference Guide* ([SPRUG34](#)).

## 2.9 Power Management

The SPI can be placed in reduced-power modes to conserve power during periods of low activity. The power management of the SPI is controlled by the processor Power and Sleep Controller (PSC). The PSC acts as a master controller for power management for all of the peripherals on the device. For detailed information on power management procedures using the PSC, see the *TMS320DM357 DMSoC ARM Subsystem Reference Guide* ([SPRUG25](#)).

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

## 2.10 SPI Internal Loop-Back Test Mode

### CAUTION

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

The internal loop-back self-test mode can be utilized to test the SPI transmit path and receive path. In this mode, the transmit signal is internally fed back to the receiver and the SPI\_DO, SPI\_DI, 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.

## 2.11 Emulation Considerations

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

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

### 3 Registers

Table 3 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 3 should be considered as reserved locations and the register contents should not be modified.

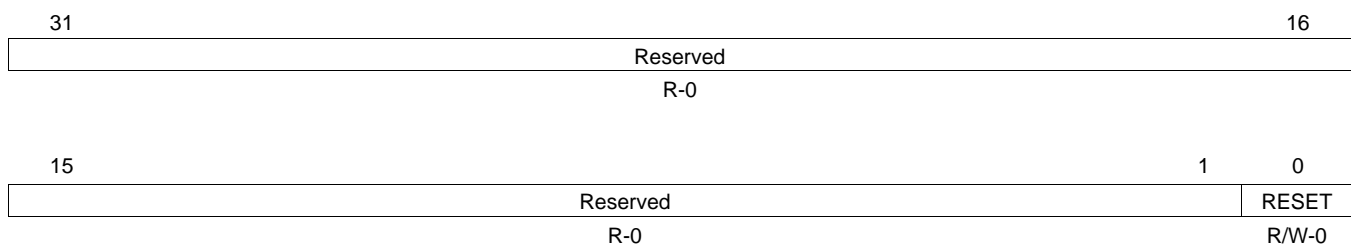
**Table 3. SPI Registers**

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

#### 3.1 SPI Global Control Register 0 (SPIGCR0)

The SPI global control register 0 (SPIGCR0) is shown in Figure 11 and described in Table 4.

**Figure 11. SPI Global Control Register 0 (SPIGCR0)**



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

**Table 4. SPI Global Control Register 0 (SPIGCR0) Field Descriptions**

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

### 3.2 SPI Global Control Register 1 (SPIGCR1)

The SPI global control register 1 (SPIGCR1) is shown in [Figure 12](#) and described in [Table 5](#).

**Figure 12. SPI Global Control Register 1 (SPIGCR1)**

|          |    |        |          |        |          |   |
|----------|----|--------|----------|--------|----------|---|
| 31       | 25 | 24     | 23       | 17     | 16       |   |
| Reserved |    | SPIENA | Reserved |        | LOOPBACK |   |
| R-0      |    | R/W-0  | R-0      |        | R/W-0    |   |
|          |    |        |          |        |          |   |
| 15       |    |        |          | 2      | 1        | 0 |
| Reserved |    |        |          | CLKMOD | MASTER   |   |
| R-0      |    |        |          | R/W-0  | R/W-0    |   |

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

**Table 5. SPI Global Control Register 1 (SPIGCR1) Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31-25 | Reserved | 0     | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.   |
| 24    | SPIENA   | 0     | SPI enable. Holds the SPI in a reset state after a chip reset. The SPI is enabled only after a 1 is written to this bit. This bit must be set to 1 after all other SPI configuration bits have been written. This prevents an invalid operation of the SPI while the clock polarity is being changed.   |
|       |          | 1     | Activates SPI.  |
| 23-17 | Reserved | 0     | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.   |
| 16    | LOOPBACK | 0     | Internal loop-back test mode. The internal self-test option is enabled by setting this bit to 1. If the SPI_DO and SPI_DI pins are configured with SPI functionality, then the SPI_DO pin is internally connected to the SPI_DI pin. The transmit data is looped back as receive data and is stored in the receive field of the concerned buffer. |
|       |          | 1     | Externally, during loop-back operation, the SPI_CLK pin outputs an inactive value and SPI_DI remains in a high-impedance state. The SPI has to be initialized in master mode before the loop-back is selected. If a data transfer is ongoing, errors may result.  |
| 15-2  | Reserved | 0     | Reserved. The reserved bit location is always read as 0. This field must be written as zeroes.  |
| 1     | CLKMOD   | 0     | Clock mode. This bit must be set for the SPI module to operate.   |
|       |          | 1     | Reserved  |
|       |          | 1     | SPI module clock mode is enabled.   |
| 0     | MASTER   | 0     | Master mode. This bit must be set for the SPI module to operate.  |
|       |          | 1     | Reserved  |
|       |          | 1     | SPI module master mode is enabled.  |

### 3.3 SPI Interrupt Register (SPIINT)

The SPI interrupt register (SPIINT) is shown in [Figure 13](#) and described in [Table 6](#).

**Figure 13. SPI Interrupt Register (SPIINT)**

|          |          |  |  |         |      |           |      |           |          |   |          |    |
|----------|----------|--|--|---------|------|-----------|------|-----------|----------|---|----------|----|
| 31       | Reserved |  |  |         |      |           |      |           |          |   | 17       | 16 |
|          |          |  |  |         |      |           |      |           |          |   | DMAREQEN |    |
| R-0      |          |  |  |         |      |           |      |           |          |   | R/W-0    |    |
| 15       |          |  |  | 9       | 8    | 7         | 6    | 5         | 4        | 3 | 0        |    |
| Reserved |          |  |  | RXINTEN | Rsvd | OVRNINTEN | Rsvd | BITERRENA | Reserved |   |          |    |
| R-0      |          |  |  | R/W-0   | R-0  | R/W-0     | R-0  | R/W-0     | R-0      |   |          |    |

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

**Table 6. SPI Interrupt Register (SPIINT) Field Descriptions**

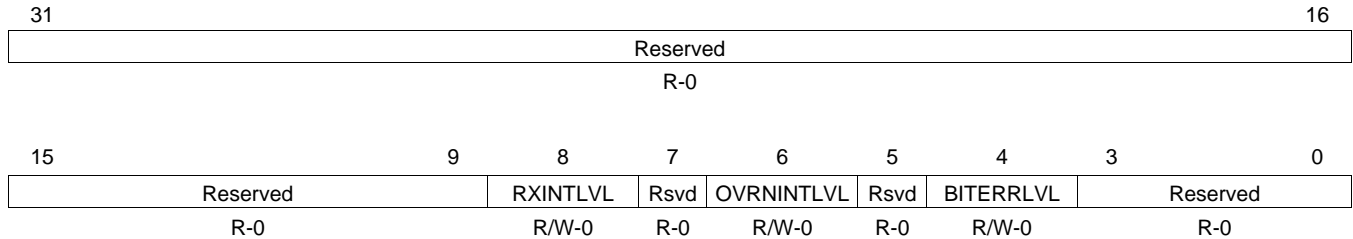
| Bit   | Field     | Value  | Description   |
|-------|-----------|--------|---|
| 31-17 | Reserved  | 0      | Reserved. The reserved bit location is always read as 0. This field must be written with zeroes.  |
| 16    | DMAREQEN  | 0<br>1 | DMA request enable. Enables the DMA request signal to be generated for both receive and transmit channels. When using the SPI module with the EDMA, it is important that the related EDMA channels are configured and enabled before enabling this bit.<br>DMA is not used.<br>DMA is used. |
| 15-9  | Reserved  | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.   |
| 8     | RXINTEN   | 0<br>1 | Receive interrupt enable. An interrupt is generated when the RXINTFLAG bit in the SPI flag register (SPIFLG) is set by hardware; otherwise, no interrupt is generated.<br>Interrupt is not generated.<br>Interrupt is generated.  |
| 7     | Reserved  | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.   |
| 6     | OVRNINTEN | 0<br>1 | Overrun interrupt enable. An interrupt is generated when the OVRNINTFLAG bit in the SPI flag register (SPIFLG) is set by hardware; otherwise, no interrupt is generated.<br>Overrun interrupt is not generated.<br>Overrun interrupt is generated.  |
| 5     | Reserved  | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.   |
| 4     | BITERRENA | 0<br>1 | Enables interrupt on bit error.<br>No interrupt asserted upon bit error.<br>Enables an interrupt on a bit error (BITERR = 1).   |
| 3-0   | Reserved  | 0      | Reserved. The reserved bit location is always read as 0. This field must be written with zeroes.  |



### 3.4 SPI Interrupt Level Register (SPILVL)

The SPI interrupt level register (SPILVL) is shown in [Figure 14](#) and described in [Table 7](#).

**Figure 14. SPI Interrupt Level Register (SPILVL)**



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

**Table 7. SPI Interrupt Level Register (SPILVL) Field Descriptions**

| Bit  | Field      | Value  | Description  |
|------|------------|--------|--|
| 31-9 | Reserved   | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 8    | RXINTLVL   | 0<br>1 | Receive interrupt level.<br>0 Receive interrupt is mapped to interrupt line INT0.<br>1 Receive interrupt is mapped to interrupt line INT1.                         |
| 7    | Reserved   | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 6    | OVRNINTLVL | 0<br>1 | Receive overrun interrupt level.<br>0 Receive overrun interrupt is mapped to interrupt line INT0.<br>1 Receive overrun interrupt is mapped to interrupt line INT1. |
| 5    | Reserved   | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 4    | BITERRLVL  | 0<br>1 | Bit error interrupt level.<br>0 Bit error interrupt is mapped to interrupt line INT0.<br>1 Bit error interrupt is mapped to interrupt line INT1.                   |
| 3-0  | Reserved   | 0      | Reserved. The reserved bit location is always read as 0. This field must be written with zeroes.   |

### 3.5 SPI Flag Register (SPIFLG)

The SPI flag register (SPIFLG) is shown in [Figure 15](#) and described in [Table 8](#).

**Figure 15. SPI Flag Register (SPIFLG)**

|          |          |           |      |            |      |           |          |    |
|----------|----------|-----------|------|------------|------|-----------|----------|----|
| 31       | Reserved |           |      |            |      |           |          | 16 |
| R-0      |          |           |      |            |      |           |          |    |
| 15       | 9        | 8         | 7    | 6          | 5    | 4         | 3        | 0  |
| Reserved |          | RXINTFLAG | Rsvd | OVRNINTFLG | Rsvd | BITERRFLG | Reserved |    |
| R-0      |          | R/W1C-0   | R-0  | R/W1C-0    | R-0  | RC-0      | R-0      |    |

LEGEND: R/W = Read/Write; R = Read only; RC = Read bit to clear; W1C = Write 1 to clear (writing 0 has no effect); -n = value after reset

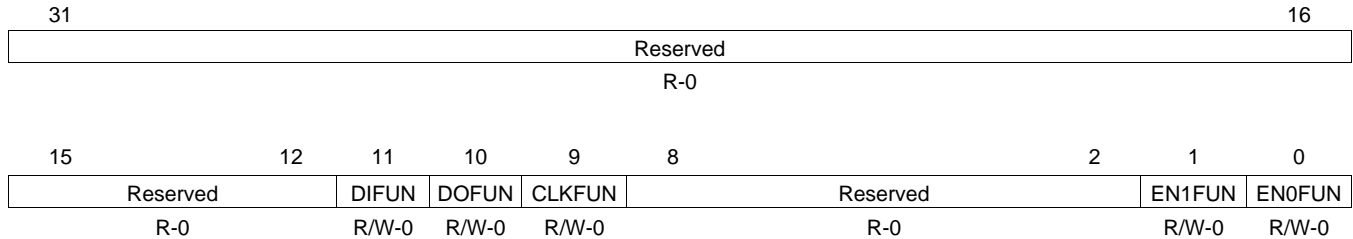
**Table 8. SPI Flag Register (SPIFLG) Field Descriptions**

| Bit  | Field      | Value  | Description  |
|------|------------|--------|--|
| 31-9 | Reserved   | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 8    | RXINTFLAG  | 0<br>1 | Receive interrupt flag. RXINTFLAG is set when a word is received and copied into the SPI buffer register (SPIBUF). If the RXINTEN bit in the SPI interrupt register (SPIINT) is set to 1 (enabled), an interrupt is also generated. During emulation mode, a read of the SPI emulation register (SPIEMU) does not clear RXINTFLAG. This bit is cleared by: <ul style="list-style-type: none"> <li>• Reading SPIBUF</li> <li>• Writing a 1 to this bit</li> <li>• Writing a 0 to the SPIENA bit in the SPI global control register 1 (SPIGCR1)</li> <li>• System reset</li> </ul> 0 Interrupt condition did not occur.<br>1 Interrupt condition did occur.  |
| 7    | Reserved   | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 6    | OVRNINTFLG | 0<br>1 | Receiver overrun flag. OVRNINTFLG is set by the SPI hardware when an operation completes before the previous character has been read from the buffer. OVRNINTFLG bit indicates that the last received character has been overwritten and therefore lost. If the OVRNINTEN bit in the SPI interrupt register (SPIINT) is set to 1 (enabled), an interrupt is also generated. This bit is cleared by: <ul style="list-style-type: none"> <li>• Reading INTVECT0 or INTVECT1</li> <li>• Writing a 1 to this bit</li> <li>• Writing a 0 to the SPIENA bit in the SPI global control register 1 (SPIGCR1)</li> <li>• System reset</li> </ul> Reading the SPIBUF register does not clear this bit.<br>0 Overrun condition did not occur.<br>1 Overrun condition did occur. |
| 5    | Reserved   | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 4    | BITERRFLG  | 0<br>1 | Bit error flag. The SPI samples the signal of the transmit pin (master: SPI_DO) at the receive point (half clock cycle after transmit point). If the sampled value differs from the transmitted value, a bit error is detected and the BITERRFLG bit is set and the BITERR bit in the SPI buffer register (SPIBUF) is set. Possible reasons for a bit error are a high bit rate/capacitive load or another master/slave trying to transmit at the same time. If the BITERRENA bit in the SPI interrupt register (SPIINT) is set to 1 (enabled) and the BITERR bit (in SPIBUF) is set to 1, an interrupt is also generated. This bit is cleared by reading the bit.<br>0 Bit error did not occur.<br>1 Bit error did occur.   |
| 3-0  | Reserved   | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |

### 3.6 SPI Pin Control Register (SPIPC0)

The SPI pin control register (SPIPC0) is shown in [Figure 16](#) and described in [Table 9](#).

**Figure 16. SPI Pin Control Register (SPIPC0)**



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

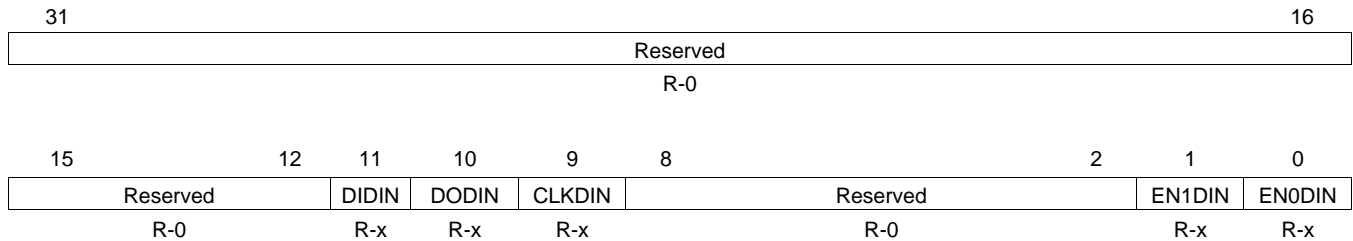
**Table 9. SPI Pin Control Register (SPIPC0) Field Descriptions**

| Bit   | Field    | Value | Description   |
|-------|----------|-------|---|
| 31-12 | Reserved | 0     | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect. |
| 11    | DIFUN    | 0     | Reserved  |
|       |          | 1     | SPI_DI is a functional pin.   |
| 10    | DOFUN    | 0     | Reserved  |
|       |          | 1     | SPI_DO is a functional pin.   |
| 9     | CLKFUN   | 0     | Reserved  |
|       |          | 1     | SPI_CLK is a functional pin.  |
| 8-2   | Reserved | 0     | Reserved. The reserved bit location is always read as 0. This field must be written with zeroes.      |
| 1     | EN1FUN   | 0     | Reserved  |
|       |          | 1     | SPI_EN1 is a functional pin.  |
| 0     | EN0FUN   | 0     | Reserved  |
|       |          | 1     | SPI_EN0 is a functional pin.  |

### 3.7 SPI Pin Control Register 2 (SPIPC2)

The SPI pin control register 2 (SPIPC2) is shown in [Figure 17](#) and described in [Table 10](#).

**Figure 17. SPI Pin Control Register 2 (SPIPC2)**



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

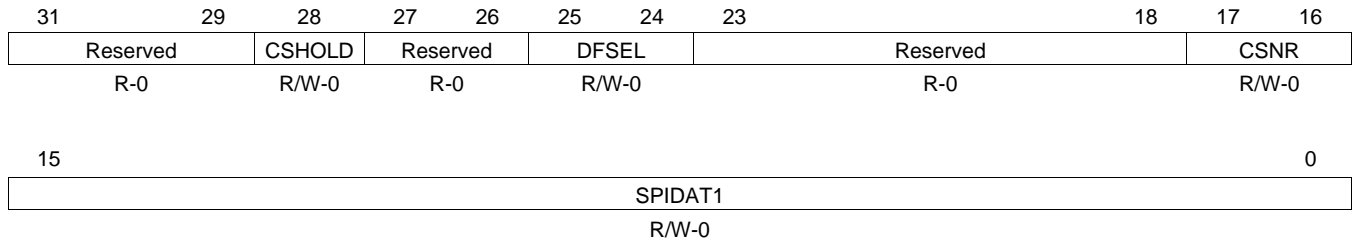
**Table 10. SPI Pin Control Register 2 (SPIPC2) Field Descriptions**

| Bit   | Field    | Value  | Description  |
|-------|----------|--------|--|
| 31-12 | Reserved | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 11    | DIDIN    | 0<br>1 | SPI data input (SPI_DI) pin value. This bit reflects the value on the SPI_DI pin.<br>Current value on SPI_DI pin is logic 0.<br>Current value on SPI_DI pin is logic 1.  |
| 10    | DODIN    | 0<br>1 | SPI data output (SPI_DO) pin value. This bit reflects the value on the SPI_DO pin.<br>Current value on SPI_DO pin is logic 0.<br>Current value on SPI_DO pin is logic 1. |
| 9     | CLKDIN   | 0<br>1 | SPI clock (SPI_CLK) pin value. This bit reflects the value on the SPI_CLK pin.<br>Current value on SPI_CLK pin is logic 0.<br>Current value on SPI_CLK pin is logic 1.   |
| 8-2   | Reserved | 0      | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 1     | EN1DIN   | 0<br>1 | SPI slave 1 (SPI_EN1) pin value. This bit reflects the value on the SPI_EN1 pin.<br>Current value on SPI_EN1 pin is logic 0.<br>Current value on SPI_EN1 pin is logic 1. |
| 0     | EN0DIN   | 0<br>1 | SPI slave 0 (SPI_EN0) pin value. This bit reflects the value on the SPI_EN0 pin.<br>Current value on SPI_EN0 pin is logic 0.<br>Current value on SPI_EN0 pin is logic 1. |

### 3.8 SPI Shift Register (SPIDAT1)

The SPI shift register (SPIDAT1) is shown in [Figure 18](#) and described in [Table 11](#).

**Figure 18. SPI Shift Register (SPIDAT1)**



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

**Table 11. SPI Shift Register (SPIDAT1) Field Descriptions**

| Bit   | Field    | Value   | Description  |
|-------|----------|---------|--|
| 31-29 | Reserved | 0       | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 28    | CSHOLD   | 0       | Chip select hold mode. CSHOLD is considered in 4-pin mode only. CSHOLD defines the behavior of the chip select line at the end of a transfer.  |
|       |          | 1       | Chip select signal is deactivated at the end of a transfer after the T2CDELAY time has passed. If two consecutive transfers are dedicated to the same chip select, this chip select signal is shortly deactivated before it is activated again.  |
| 27-26 | Reserved | 0       | Reserved. The reserved bit location is always read as 0. This field must be written with zeroes.   |
| 25-24 | DFSEL    | 0-3h    | Data word format select.   |
|       |          | 0       | Data format 0 is selected.   |
|       |          | 1h      | Data format 1 is selected.   |
|       |          | 2h      | Data format 2 is selected.   |
|       |          | 3h      | Data format 3 is selected.   |
| 23-18 | Reserved | 0       | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 17-16 | CSNR     | 0-3h    | Chip select number. CSNR defines the chip select that shall be activated during the data transfer. Note that the $\overline{\text{SPI\_ENn}}$ signals are active low.  |
|       |          | 0       | Both chip select $\overline{\text{SPI\_EN0}}$ and $\overline{\text{SPI\_EN1}}$ are selected.   |
|       |          | 1h      | Chip select $\overline{\text{SPI\_EN1}}$ is selected only.   |
|       |          | 2h      | Chip select $\overline{\text{SPI\_EN0}}$ is selected only.   |
|       |          | 3h      | No chip select is used.  |
| 15-0  | SPIDAT1  | 0-FFFFh | SPI shift data 1. This field makes up the SPI shift register. Data is shifted out of the MSB (bit 15) and into the LSB (bit 0). The SPIENA bit in the SPI global control register 1 (SPIGCR1) must be set to 1 before this register can be written to. Writing a 0 to the SPIENA bit forces the SPIDAT1 bit to 0.<br><br>A write to this field drives the $\overline{\text{SPI\_EN[1:0]}}$ signal low if the $\overline{\text{SPI\_EN[1:0]}}$ signals are enabled in the SPI pin control register (SPIPC0).<br><br>When data is read from this field, the value is indeterminate because of the shift operation. The value in the SPI buffer register (SPIBUF) should be read after the shift operation is complete to determine what data was shifted into SPIDAT1. |

### 3.9 SPI Buffer Register (SPIBUF)

The SPI buffer register (SPIBUF) is shown in [Figure 19](#) and described in [Table 12](#). Reading SPIBUF clears the OVRNINTFLG and RXINTFLAG bits in the SPI flag register (SPIFLG).

**Figure 19. SPI Buffer Register (SPIBUF)**

|         |        |        |        |          |    |    |       |
|---------|--------|--------|--------|----------|----|----|-------|
| 31      | 30     | 29     | 28     | 27       | 18 | 17 | 16    |
| RXEMPTY | RXOVR  | TXFULL | BITERR | Reserved |    |    | LCSNR |
| R-1     | RC-0   | R-0    | RC-0   | R-0      |    |    | R-0   |
|         |        |        |        |          |    |    | 0     |
| 15      | SPIBUF |        |        |          |    |    | 0     |
| R-x     |        |        |        |          |    |    |       |

LEGEND: R/W = Read/Write; R = Read only; RC = Read bit to clear; -n = value after reset; -x = value is indeterminate after reset

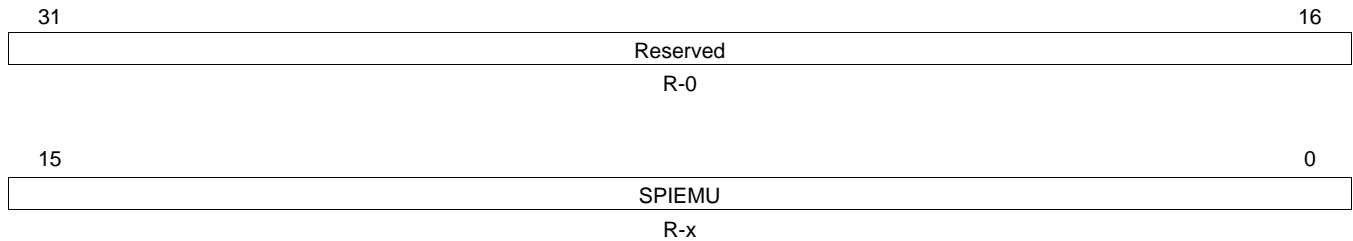
**Table 12. SPI Buffer Register (SPIBUF) Field Descriptions**

| Bit   | Field    | Value   | Description  |
|-------|----------|---------|--|
| 31    | RXEMPTY  | 0<br>1  | Receive data buffer empty. This is a read-only flag. When the host reads an SPIBUF field or the entire SPIBUF, this automatically sets the RXEMPTY bit. When a data transfer has been finished and the received data is copied into SPIBUF, the RXEMPTY bit is cleared.<br>Data is received and copied into an SPIBUF field.<br>No data received since last reading SPIBUF.  |
| 30    | RXOVR    | 0<br>1  | Receive data buffer overrun. This is a read-to-clear only flag, that is, reading this bit automatically clears the bit. When a data transfer has been finished and the received data is copied into SPIBUF with the RXEMPTY bit already cleared, the RXOVR bit is set.<br>No receive data overrun condition occurred since last time reading the status field.<br>A receive data overrun condition occurred since last time reading the status field.  |
| 29    | TXFULL   | 0<br>1  | Transmit data buffer full. This is a read-only flag. Writing into SPIDAT1 bits in the SPI shift register (SPIDAT1) automatically sets the TXFULL bit. After transfer of the transmit data the TXFULL flag is cleared.<br>No new transmit data from the host since previous transfer of transmit data.<br>Host provided new transmit data to SPIDAT1.   |
| 28    | BITERR   | 0<br>1  | Bit error. This is a read-to-clear only flag, that is, reading this bit automatically clears the bit. This bit represents a copy of the BITERRFLG bit in the SPI flag register (SPIFLG). The SPI samples the signal of the transmit pin (master: SIMO, slave: 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 BITERRFLG bit is set and the BITERR bit is set. Possible reasons for a bit error are a high bit rate/capacitive load or another master/slave trying to transmit at the same time. If the BITERRENA bit in the SPI interrupt register (SPIINT) is set to 1 (enabled) and the BITERR bit is set to 1, an interrupt is also generated.<br>Bit error did not occur.<br>Bit error did occur. |
| 27-18 | Reserved | 0       | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 17-16 | LCSNR    | 0-3h    | Last chip select number. LCSNR in the status field is a copy of the CSNR bit in the corresponding control field. It defines the chip select that has been activated during the last data transfer from the corresponding buffer. LCSNR is copied after transmission during write back of received data.  |
| 15-0  | SPIBUF   | 0-FFFFh | SPI receive buffer. This field makes up the SPI receive buffer. Data in this field is the data received via the SPI_DI pin and transferred from the SPIDAT1 bits in the SPI shift register (SPIDAT1). Since the data is shifted into the SPI with the most-significant bit first, for word lengths less than 16, the data is stored right-justified in the register.   |

### 3.10 SPI Emulation Register (SPIEMU)

The SPI emulation register (SPIEMU) is shown in [Figure 20](#) and described in [Table 13](#).

**Figure 20. SPI Emulation Register (SPIEMU)**



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

**Table 13. SPI Emulation Register (SPIEMU) Field Descriptions**

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

### 3.11 SPI Delay Register (SPIDELAY)

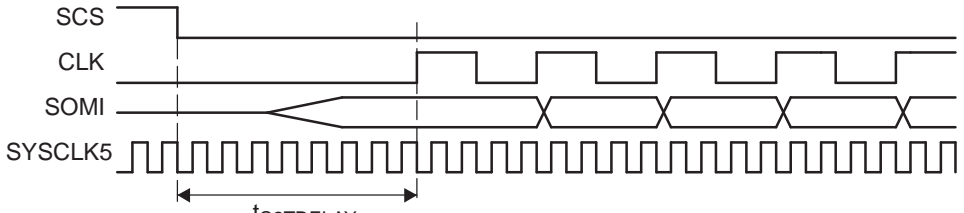
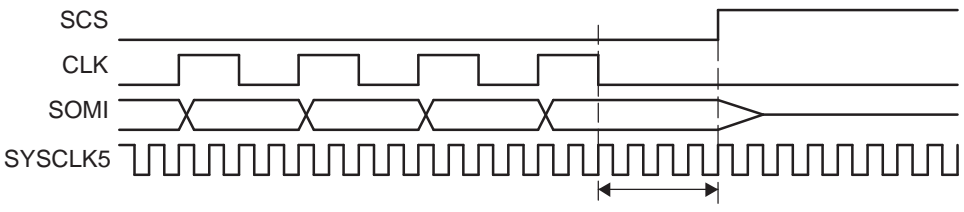
The SPI delay register (SPIDELAY) is shown in [Figure 21](#) and described in [Table 14](#).

**Figure 21. SPI Delay Register (SPIDELAY)**

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

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

**Table 14. SPI Delay Register (SPIDELAY) Field Descriptions**

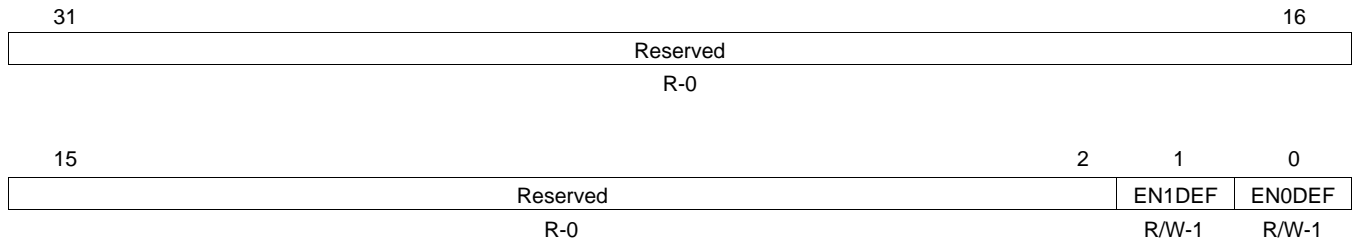
| Bit   | Field    | Value | Description  |
|-------|----------|-------|--|
| 31-29 | Reserved | 0     | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 28-24 | C2TDELAY | 0-16h | <p>Chip-select-active-to-transmit-start-delay. C2TDELAY defines a setup time for the slave device that delays the data transmission from the chip select active edge by a multiple of SYSCLK5 clock cycles. C2TDELAY is configured between 2 and 22 SYSCLK5 clock cycles.</p>  <p>The setup time value is calculated as:</p> $t_{C2TDELAY} = \frac{C2TDELAY}{SYSCLK5} + 2$ <p>Example: SYSCLK5 = 25 MHz and C2TDELAY = 06h; therefore, <math>t_{C2TDELAY} = 320</math> ns. When the chip select signal becomes active, the slave has to prepare data transfer within 320 ns.</p>      |
| 23-21 | Reserved | 0     | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 20-16 | T2CDELAY | 0-16h | <p>Transmit-end-to-chip-select-inactive-delay. T2CDELAY defines a hold time for the slave device that delays the chip select deactivation by a multiple of SYSCLK5 clock cycles after the last bit is transferred. T2CDELAY is configured between 1 and 22 SYSCLK5 clock cycles.</p>  <p>The hold time value is calculated as:</p> $t_{T2CDELAY} = \frac{T2CDELAY}{SYSCLK5} + 1$ <p>Example: SYSCLK5 = 25 MHz and T2CDELAY = 02h; therefore, <math>t_{T2CDELAY} = 120</math> ns. After the last data bit is being transferred, the chip select signal is held active for 120 ns.</p> |
| 15-0  | Reserved | 0     | Reserved. The reserved bit location is always read as 0. This field must be written with zeroes.   |



### 3.12 SPI Default Chip Select Register (SPIDEF)

The SPI default chip select register (SPIDEF) is shown in [Figure 22](#) and described in [Table 15](#).

**Figure 22. SPI Default Chip Select Register (SPIDEF)**



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

**Table 15. 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    | EN1DEF   | 0<br>1 | Chip select default pattern. Selects the output to the slave 1 chip select pin ( $\overline{\text{SPI\_EN1}}$ ) when no transmission is currently in progress. EN1DEF allows you to set a chip select pattern that deselects all the SPI slaves.<br><br>0 $\overline{\text{SPI\_EN1}}$ is driven to logic 0 when no transaction is in progress.<br>1 $\overline{\text{SPI\_EN1}}$ is driven to logic 1 when no transaction is in progress. |
| 0    | EN0DEF   | 0<br>1 | Chip select default pattern. Selects the output to the slave 0 chip select pin ( $\overline{\text{SPI\_EN0}}$ ) when no transmission is currently in progress. EN0DEF allows you to set a chip select pattern that deselects all the SPI slaves.<br><br>0 $\overline{\text{SPI\_EN0}}$ is driven to logic 0 when no transaction is in progress.<br>1 $\overline{\text{SPI\_EN0}}$ is driven to logic 1 when no transaction is in progress. |

### 3.13 SPI Data Format Registers (SPIFMT<sub>n</sub>)

The SPI data format register (SPIFMT<sub>0</sub>, SPIFMT<sub>1</sub>, SPIFMT<sub>2</sub>, and SPIFMT<sub>3</sub>) is shown in [Figure 23](#) and described in [Table 16](#).

**Figure 23. SPI Data Format Register (SPIFMT<sub>n</sub>)**

|                       |    |          |                       |                      |                       |                    |
|-----------------------|----|----------|-----------------------|----------------------|-----------------------|--------------------|
| 31                    | 21 | 20       | 19                    | 18                   | 17                    | 16                 |
| Reserved              |    |          | SHIFTDIR <sub>n</sub> | Reserved             | POLARITY <sub>n</sub> | PHASE <sub>n</sub> |
| R/W-0                 |    |          | R/W-0                 | R-0                  | R/W-0                 | R/W-0              |
| 15                    | 8  | 7        | 5                     | 4                    |                       |                    |
| PRESCALE <sub>n</sub> |    | Reserved |                       | CHARLEN <sub>n</sub> |                       |                    |
| R/W-0                 |    | R-0      |                       | R/W-0                |                       |                    |

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

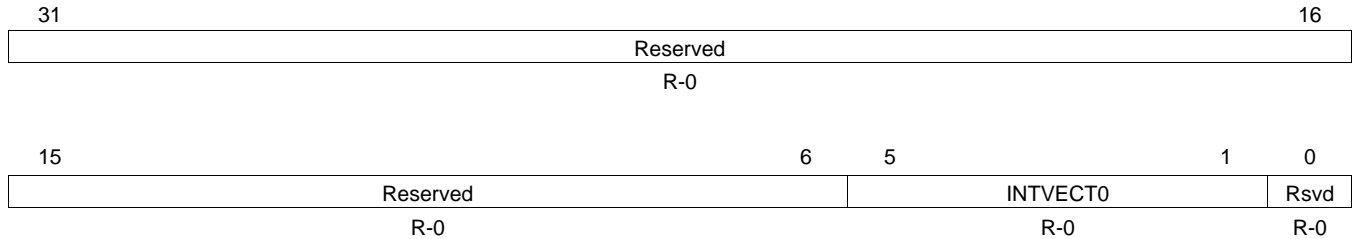
**Table 16. SPI Data Format Register (SPIFMT<sub>n</sub>) Field Descriptions**

| Bit   | Field                 | Value                              | Description   |
|-------|-----------------------|------------------------------------|---|
| 31-21 | Reserved              | 0                                  | Reserved. The reserved bit location is always read as 0. This field must be written with zeroes.  |
| 20    | SHIFTDIR <sub>n</sub> | 0<br>1                             | Shift direction for data format <i>n</i> . SHIFTDIR <sub>n</sub> selects the shift direction for data format <i>n</i> ( <i>n</i> = 1,2,3).<br>0 Most-significant bit is shifted out first.<br>1 Least-significant bit is shifted out first.   |
| 19-18 | Reserved              | 0                                  | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.   |
| 17    | POLARITY <sub>n</sub> | 0<br>1                             | Clock polarity for data format <i>n</i> . POLARITY <sub>n</sub> defines the clock polarity for data format <i>n</i> ( <i>n</i> = 1,2,3).<br>0 SPI clock signal is low-inactive, that is, before and after data transfer the clock signal is low.<br>1 SPI clock signal is high-inactive, that is, before and after data transfer the clock signal is high.  |
| 16    | PHASE <sub>n</sub>    | 0<br>1                             | Clock delay for data format <i>n</i> . PHASE <sub>n</sub> defines the clock delay for data format <i>n</i> ( <i>n</i> = 1,2,3).<br>0 SPI clock signal is not delayed versus the transmit/receive data stream. The first data bit is transmitted with the first clock edge and the first bit is received with the second (inverse) clock edge.<br>1 SPI clock signal is delayed by a half SPI clock cycle versus the transmit/receive data stream. The first transmit bit has to output prior to the first clock edge. Master and slave receive the first bit with the first edge. |
| 15-8  | PRESCALE <sub>n</sub> | 2-FFh                              | Prescaler for data format <i>n</i> . PRESCALE <sub>n</sub> defines the bit transfer rate for data format <i>n</i> ( <i>n</i> = 1,2,3). PRESCALE <sub>n</sub> is directly derived from SYSCLK5 CLK. The clock rate is calculated as:<br>$SPI\_CLK = \frac{SYSCLK5}{(PRESCALE_n + 1)}$<br>PRESCALE <sub>n</sub> is only supported for values >1, where the maximum SPI clock rate is SYSCLK5/3.   |
| 7-5   | Reserved              | 0                                  | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.   |
| 4-0   | CHARLEN <sub>n</sub>  | 0-1Fh<br>0-1h<br>2h-10h<br>11h-1Fh | Data word length for data format <i>n</i> . CHARLEN <sub>n</sub> defines the word length for data format <i>n</i> ( <i>n</i> = 1,2,3). Legal values are 2h (data word length = 2 bits) to 10h (data word length = 16 bits). Other values, such as 0 or 1Fh, are not detected and their effect is indeterminate.<br>0-1h Not detected and their effect is indeterminate.<br>2h-10h Data word length is 2 bits to 16 bits.<br>11h-1Fh Not detected and their effect is indeterminate.   |

### 3.14 SPI Interrupt Vector Register 0 (INTVECT0)

The SPI interrupt vector register 0 (INTVECT0) is shown in [Figure 24](#) and described in [Table 17](#).

**Figure 24. SPI Interrupt Vector Register 0 (INTVECT0)**



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

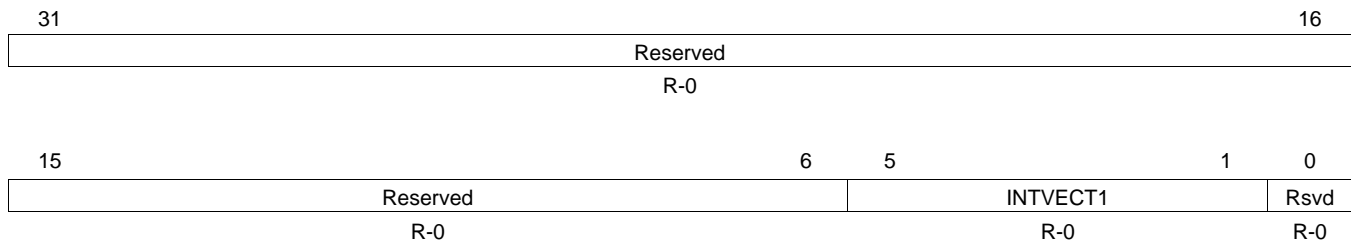
**Table 17. SPI Interrupt Vector Register 0 (INTVECT0) Field Descriptions**

| Bit  | Field    | Value | Description  |
|------|----------|-------|--|
| 31-6 | Reserved | 0     | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 5-1  | INTVECT0 | 0-1Fh | <p>Interrupt vector for interrupt line INT0. INTVECT0 returns the vector of the pending interrupt at interrupt line INT0. If more than one interrupt is pending, INTVECT0 always references the highest prior interrupt source first.</p> <p>The interrupts available for SPI, in the descending order of their priorities:</p> <ul style="list-style-type: none"> <li>• Receive overrun interrupt</li> <li>• Receive interrupt</li> <li>• Transmission error interrupt</li> </ul> <p>Vectors for each of these interrupts is reflected in the INTVECT0 bit, when they occur. Reading the vectors for the receive overrun interrupt and receive interrupt automatically clears the respective flags in the SPI flag register (SPIFLG). On reading the INTVECT0 bit, the vector of the next highest priority interrupt (if any) is then reflected in the INTVECT0 bit. If two or more interrupts occur simultaneously, the vector for the highest priority interrupt is reflected in the INTVECT0 bits.</p> <p style="margin-left: 20px;">0            No interrupt is pending.</p> <p style="margin-left: 20px;">1h-10h    Reserved</p> <p style="margin-left: 20px;">11h        Error interrupt is pending. Refer to lower halfword of the SPI interrupt register (SPIINT) to determine more details about the type of error and the concerned buffer.</p> <p style="margin-left: 20px;">12h        Receive interrupt is pending.</p> <p style="margin-left: 20px;">13h        Receive overrun interrupt is pending.</p> <p style="margin-left: 20px;">14h-1Fh   Reserved</p> |
| 0    | Reserved | 0     | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |

### 3.15 SPI Interrupt Vector Register 1 (INTVECT1)

The SPI interrupt vector register 1 (INTVECT1) is shown in [Figure 25](#) and described in [Table 18](#).

**Figure 25. SPI Interrupt Vector Register 1 (INTVECT1)**



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

**Table 18. SPI Interrupt Vector Register 1 (INTVECT1) Field Descriptions**

| Bit  | Field    | Value   | Description  |
|------|----------|---------|--|
| 31-6 | Reserved | 0       | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |
| 5-1  | INTVECT1 | 0-1Fh   | Interrupt vector for interrupt line INT1. INTVECT1 returns the vector of the pending interrupt at interrupt line INT1. If more than one interrupt is pending, INTVECT1 always references the highest prior interrupt source first.<br><br>The interrupts available for SPI, in the descending order of their priorities: <ul style="list-style-type: none"> <li>• Receive overrun interrupt</li> <li>• Receive interrupt</li> <li>• Transmission error interrupt</li> </ul> Vectors for each of these interrupts is reflected in the INTVECT1 bit, when they occur. Reading the vectors for the receive overrun interrupt and receive interrupt automatically clears the respective flags in the SPI flag register (SPIFLG). On reading the INTVECT1 bit, the vector of the next highest priority interrupt (if any) is then reflected in the INTVECT1 bit. If two or more interrupts occur simultaneously, the vector for the highest priority interrupt is reflected in the INTVECT1 bits. |
|      |          | 0       | No interrupt is pending.   |
|      |          | 1h-10h  | Reserved   |
|      |          | 11h     | Error interrupt is pending. Refer to lower halfword of the SPI interrupt register (SPIINT) to determine more details about the type of error and the concerned buffer.   |
|      |          | 12h     | Receive interrupt is pending.  |
|      |          | 13h     | Receive overrun interrupt is pending.  |
|      |          | 14h-1Fh | Reserved   |
| 0    | Reserved | 0       | Reserved. The reserved bit location is always read as 0. A value written to this field has no effect.  |

## 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

|                             |  |
|-----------------------------|--|
| Amplifiers                  | <a href="http://amplifier.ti.com">amplifier.ti.com</a>             |
| Data Converters             | <a href="http://dataconverter.ti.com">dataconverter.ti.com</a>     |
| DSP                         | <a href="http://dsp.ti.com">dsp.ti.com</a>                         |
| Clocks and Timers           | <a href="http://www.ti.com/clocks">www.ti.com/clocks</a>           |
| Interface                   | <a href="http://interface.ti.com">interface.ti.com</a>             |
| Logic                       | <a href="http://logic.ti.com">logic.ti.com</a>                     |
| Power Mgmt                  | <a href="http://power.ti.com">power.ti.com</a>                     |
| Microcontrollers            | <a href="http://microcontroller.ti.com">microcontroller.ti.com</a> |
| RFID                        | <a href="http://www.ti-rfid.com">www.ti-rfid.com</a>               |
| RF/IF and ZigBee® Solutions | <a href="http://www.ti.com/lprf">www.ti.com/lprf</a>               |

### Applications

|                    |  |
|--------------------|--|
| Audio              | <a href="http://www.ti.com/audio">www.ti.com/audio</a>                   |
| Automotive         | <a href="http://www.ti.com/automotive">www.ti.com/automotive</a>         |
| Broadband          | <a href="http://www.ti.com/broadband">www.ti.com/broadband</a>           |
| Digital Control    | <a href="http://www.ti.com/digitalcontrol">www.ti.com/digitalcontrol</a> |
| Medical            | <a href="http://www.ti.com/medical">www.ti.com/medical</a>               |
| Military           | <a href="http://www.ti.com/military">www.ti.com/military</a>             |
| Optical Networking | <a href="http://www.ti.com/opticalnetwork">www.ti.com/opticalnetwork</a> |
| Security           | <a href="http://www.ti.com/security">www.ti.com/security</a>             |
| Telephony          | <a href="http://www.ti.com/telephony">www.ti.com/telephony</a>           |
| Video & Imaging    | <a href="http://www.ti.com/video">www.ti.com/video</a>                   |
| Wireless           | <a href="http://www.ti.com/wireless">www.ti.com/wireless</a>             |

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2008, Texas Instruments Incorporated