TMS320x281x, 280x DSP Serial Peripheral Interface (SPI) Reference Guide

Literature Number: SPRU059C
June 2002 − Revised August 2006

TEXAS INSTRUMENTS

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

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

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

Preface

Read This First

About This Manual

This guide describes how the serial peripheral interface works on the TMS320x281x and TMS320x280x DSPs.

Related Documentation From Texas Instruments

The following books describe the TMS320x281x and related support tools that are available on the TI website.

TMS320F2801, TMS320F2806, TMS320F2808 Digital Signal Processors (literature number SPRS230) data sheet contains the pinout, signal descriptions, as well as electrical and timing specifications for the F280x devices.

TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, and TMS320C2812 Digital Signal Processors (literature number SPRS174) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320R2811 and TMS320R2812 Digital Signal Processors (literature number SPRS257) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320C28x DSP CPU and Instruction Set Reference Guide (literature number SPRU430) describes the central processing unit (CPU) and the assembly language instructions of the TMS320C28x™ fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

TMS320x280x Analog-to-Digital Converter (ADC) Reference Guide (literature number SPRU716) describes the ADC module. The module is a 12−bit pipelined ADC. The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers (MUXs), sample−and−hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits. Digital circuits, referred to as the wrapper in this document, include programmable conversion sequencer, result registers, interface to analog circuits, interface to device peripheral bus, and interface to other on-chip modules.

TMS320x280x Boot ROM Reference Guide (literature number SPRU722) describes the purpose and features of the bootloader (factory-programmed boot-loading software). It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

TMS320x280x Enhanced Capture (eCAP) Module Reference Guide (literature number SPRU807) describes the enhanced Capture Module. It includes the module description and registers.

TMS320x280x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide (literature number SPRU791). The PWM peripheral is an essential part of controlling many of the power related systems found in both commercial and industrial equipments. This guide describes the main areas that include digital motor control, switch mode power supply control, UPS (uninterruptible power supplies), and other forms of power conversion. The PWM peripheral can be considered as performing a DAC function, where the duty cycle is equivalent to a DAC analog value, it is sometimes referred to as a Power DAC.

TMS320x280x Enhanced Quadrature Encoder Pulse (eQEP) Reference Guide (literature number SPRU790) describes the eQEP module, which is used for interfacing with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine in high performance motion and position control systems. It includes the module description and registers.

TMS320x280x System Control and Interrupts Reference Guide (literature number SPRU712) describes the various interrupts and system control features of the 280x digital signal processors (DSPs).

TMS320x281x, 280x Enhanced Controller Area Network (eCAN) Reference Guide (literature number SPRU074) describes the eCAN that uses established protocol to communicate serially with other controllers in electrically noisy environments. With 32 fully configurable mailboxes and time-stamping feature, the eCAN module provides a versatile and robust serial communication interface. The eCAN module implemented in the C28x DSP is compatible with the CAN 2.0B standard (active).

TMS320x281x, 280x Peripheral Reference Guide (literature number SPRU566) describes the peripheral reference guides of the 28x digital signal processors (DSPs).

TMS320x281x, 280x Serial Communication Interface (SCI) Reference Guide (literature number SPRU051) describes the SCI that is a two-wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format.

TMS320x281x Analog-to-Digital Converter (ADC) Reference Guide (literature number SPRU060) describes the ADC module. The module is a 12−bit pipelined ADC. The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers (MUXs), sample−and−hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits. Digital circuits, referred to as the wrapper in this document, include programmable conversion sequencer, result registers, interface to analog circuits, interface to device peripheral bus, and interface to other on-chip modules.

TMS320x281x Boot ROM Reference Guide (literature number SPRU095) describes the purpose and features of the bootloader (factory-programmed boot-loading software). It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

TMS320x281x Event Manager (EV) Reference Guide (literature number SPRU065) describes the EV modules that provide a broad range of functions and features that are particularly useful in motion control and motor control applications. The EV modules include general-purpose (GP) timers, full-compare/PWM units, capture units, and quadrature-encoder pulse (QEP) circuits.

TMS320x281x External Interface (XINTF) Reference Guide (literature number SPRU067) describes the external interface (XINTF) of the 28x digital signal processors (DSPs).

TMS320x281x Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU061) describes the McBSP) available on the C28x devices. The McBSPs allow direct interface between a DSP and other devices in a system.

TMS320x281x System Control and Interrupts Reference Guide (literature number SPRU078) describes the various interrupts and system control features of the 281x digital signal processors (DSPs).

The TMS320C28x Instruction Set Simulator Technical Overview (literature number SPRU608) describes the simulator, available within the

Code Composer Studio for TMS320C2000 IDE, that simulates the instruction set of the C28x core.

TMS320C28x DSP/BIOS Application Programming Interface (API) Reference Guide (literature number SPRU625) describes development using DSP/BIOS.

3.3 V DSP for Digital Motor Control Application Report (literature number SPRA550). New generations of motor control digital signal processors (DSPs) lower their supply voltages from 5 V to 3.3 V to offer higher performance at lower cost. Replacing traditional 5-V digital control circuitry by 3.3-V designs introduce no additional system cost and no significant complication in interfacing with TTL and CMOS compatible components, as well as with mixed voltage ICs such as power transistor gate drivers. Just like 5-V based designs, good engineering practice should be exercised to minimize noise and EMI effects by proper component layout and PCB design when 3.3-V DSP, ADC, and digital circuitry are used in a mixed signal environment, with high and low voltage analog and switching signals, such as a motor control system. In addition, software techniques such as Random PWM method can be used by special features of the Texas Instruments (TI) TMS320x24xx DSP controllers to significantly reduce noise effects caused by EMI radiation.

This application report reviews designs of 3.3-V DSP versus 5-V DSP for low HP motor control applications.The application report first describes a scenario of a 3.3-V-only motor controller indicating that for most applications, no significant issue of interfacing between 3.3 V and 5 V exists. Cost-effective 3.3-V − 5-V interfacing techniques are then discussed for the situations where such interfacing is needed. On-chip 3.3-V ADC versus 5-V ADC is also discussed. Sensitivity and noise effects in 3.3-V and 5-V ADC conversions are addressed. Guidelines for component layout and printed circuit board (PCB) design that can reduce system’s noise and EMI effects are summarized in the last section.

Thermo-Electric Cooler Control Using a TMS320F2812 DSP & DRV592 Power Amplifier Application Note (literature number SPRA873). This application report presents a thermoelectric cooler system consisting of a Texas Instruments TMS320F2812 digital signal processor (DSP) and DRV592 power amplifier. The DSP implements a digital proportional-integral-derivative feedback controller using an integrated 12-bit analog-to-digital converter to read the thermistor, and direct output of pulse-width-modulated waveforms to the H-bridge DRV592 power amplifier. The system presented provides up to 6.1 watts of heating or cooling to the laser mount, although the DRV592 amplifier is actually capable of delivering up to 15 watts when configured appropriately. The closed-loop TEC system is seen to achieve ±0.0006°C temperature accuracy, depending on the needed operating temperature range, with a step response settling time of 14 to 16 seconds. A complete description of the experimental system, along with software and software operating instructions, are provided.

Running an Application from Internal Flash Memory on the TMS320F281x DSP Application Report (literature number SPRA958). Several special requirements exist for running an application from on-chip flash memory on the TMS320F28x DSP. These requirements generally do not manifest themselves during development in RAM since the Code Composer Studio™ debugger can mask problems associated with initialized sections and how they are linked to memory. This application report covers the requirements needed to properly configure application software for execution from on-chip flash memory. Requirements for both DSP/BIOS™ and non-DSP/BIOS projects are presented. Some performance considerations and techniques are also discussed. Example code projects are included that run from on-chip flash on the eZdsp™ F2812 development board (or alternately any F2812, F2811, or F2810 DSP board). Code examples that run from internal RAM are also provided for completeness. These code examples provide a starting point for code development, if desired.

Trademarks

Code Composer Studio and C28x are trademarks of Texas Instruments.

Contents

Figures

Tables

Chapter 1

Serial Peripheral Interface (SPI)

The serial peripheral interface (SPI) is a high-speed synchronous serial input/output (I/O) port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communications between the DSP controller and external peripherals or another controller. Typical applications include external I/O or peripheral expansion via devices such as shift registers, display drivers, and analog-to-digital converters (ADCs). Multidevice communications are supported by the master/slave operation of the SPI. On the C28x™, the port supports a 16-level, receive and transmit FIFO for reducing CPU servicing overhead.

This reference guide is applicable for the SPI found on both the TMS320x280x and TMS3201x281x families of processors. This includes all Flash-based, ROM-based and RAM-based devices within these families.

Topic **Page**

Note: 28x Enhanced Features

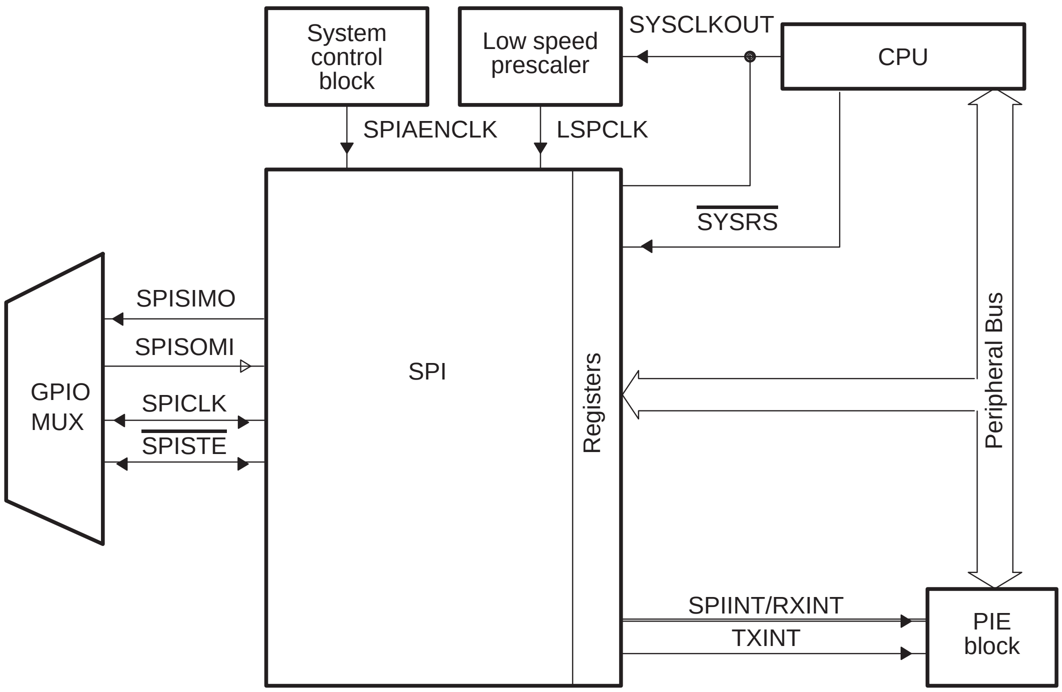
The 28x SPI features several enhancements compared to the 240xA SPI. See section 1.5 for a description of these features.

C28x is a trademark of Texas Instruments.

1.1 Enhanced SPI Module Overview

Figure 1−1 shows the SPI CPU interfaces.

Figure 1−1. SPI CPU Interface



The SPI module features include:

- ❑ Four external pins:
  - SPISOMI: SPI slave-output/master-input pin
  - SPISIMO: SPI slave-input/master-output pin
  - $\overline{\text{SPISTE}}$: SPI slave transmit-enable pin
  - SPICLK: SPI serial-clock pin

Note: All four pins can be used as GPIO, if the SPI module is not used.

- ❑ Two operational modes: master and slave
- ❑ Baud rate: 125 different programmable rates. The maximum baud rate that can be employed is limited by the maximum speed of the I/O buffers used on the SPI pins. See the device-specific data sheet for more details.
- ❑ Data word length: one to sixteen data bits

- Four clocking schemes (controlled by clock polarity and clock phase bits) include:
  - Falling edge without phase delay: SPICLK active-high. SPI transmits data on the falling edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
  - Falling edge with phase delay: SPICLK active-high. SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge without phase delay: SPICLK inactive-low. SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge with phase delay: SPICLK inactive-low. SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
- Simultaneous receive and transmit operation (transmit function can be disabled in software)
- Transmitter and receiver operations are accomplished through either interrupt-driven or polled algorithms.
- 12 SPI module control registers: Located in control register frame beginning at address 7040h.

Note: All registers in this module are 16-bit registers that are connected to Peripheral Frame 2. When a register is accessed, the register data is in the lower byte (7−0), and the upper byte (15−8) is read as zeros. Writing to the upper byte has no effect.

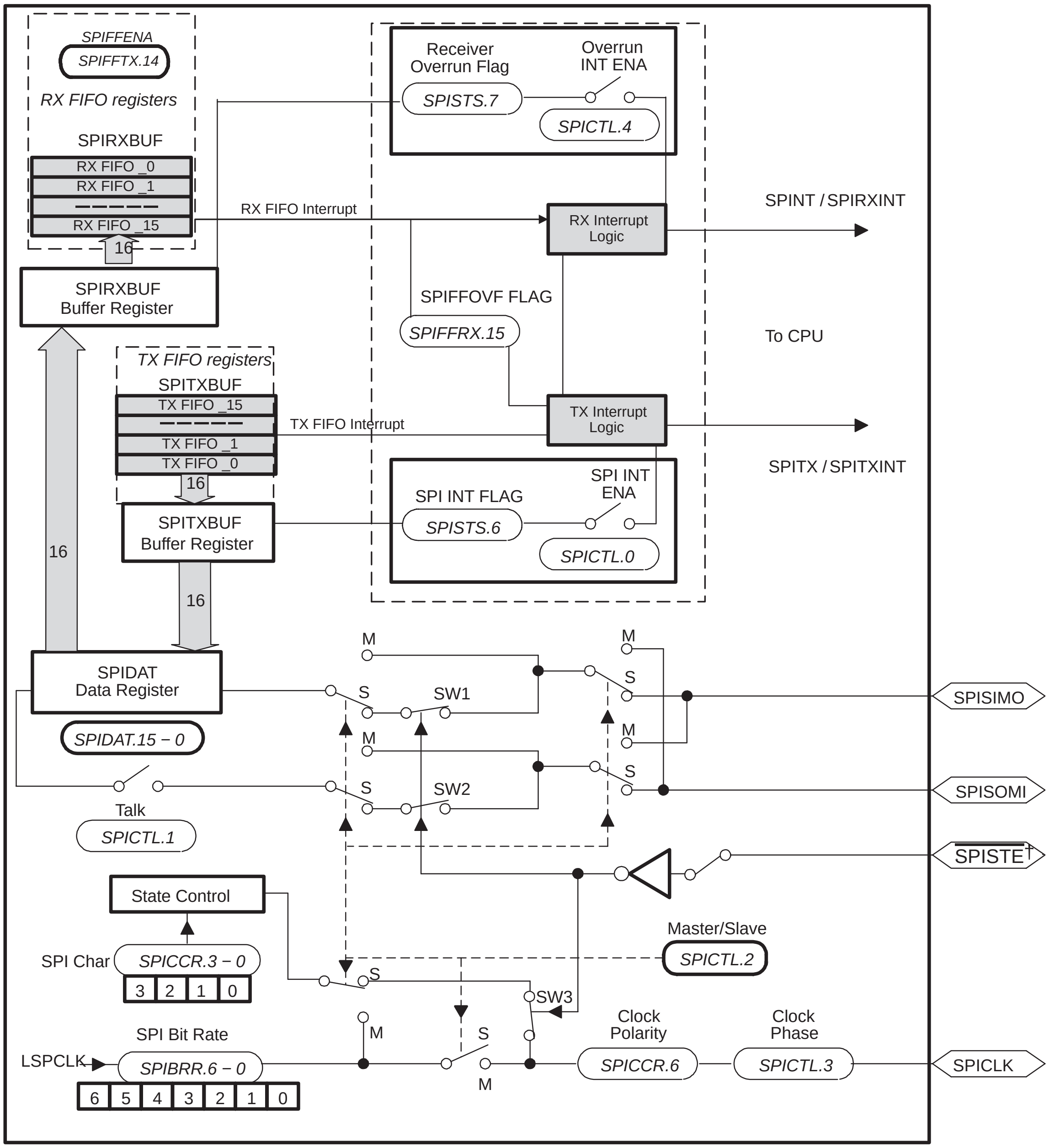
Enhanced feature:

- 16-level transmit/receive FIFO
- Delayed transmit control

1.1.1 SPI Block Diagram

Figure 1−2 is a block diagram of the SPI in slave mode, showing the basic control blocks available on the 28x SPI module.

Figure 1−2. Serial Peripheral Interface Module Block Diagram



† SPISTE of a slave device is driven low by the master.

1.1.2 SPI Module Signal Summary

Signal Name	Description
External Signals	
SPICLK	SPI clock
SPISIMO	SPI slave in, master out
SPISOMI	SPI slave out, master in
$\overline{\text{SPISTE}}$	SPI slave transmit enable
Control	
SPI CLock Rate	LSPCLK
Interrupt signals	
SPIRXINT	Transmit interrupt/ Receive Interrupt in non FIFO mode (referred to as SPI INT)
	Receive in interrupt in FIFO mode
SPITXINT	Transmit interrupt – FIFO

1.2 Overview of SPI Module Registers

The SPI port operation is configured and controlled by the registers listed in Table 1−1.

Table 1−1. SPI Registers

Name	Address Range	Size (x16)	Description
SPICCR	0x0000−7040	1	SPI Configuration Control Register
SPICTL	0x0000−7041	1	SPI Operation Control Register
SPIST	0x0000−7042	1	SPI Status Register
SPIBRR	0x0000−7044	1	SPI Baud Rate Register
SPIEMU	0x0000−7046	1	SPI Emulation Buffer Register
SPIRXBUF	0x0000−7047	1	SPI Serial Input Buffer Register
SPITXBUF	0x0000−7048	1	SPI Serial Output Buffer Register
SPIDAT	0x0000−7049	1	SPI Serial Data Register
SPIFFTX	0x0000−704A	1	SPI FIFO Transmit Register
SPIFFRX	0x0000−704B	1	SPI FIFO Receive Register
SPIFFCT	0x0000−704C	1	SPI FIFO Control Register
SPIPRI	0x0000−704F	1	SPI Priority Control Register

Note: The registers are mapped to Peripheral Frame 2. This space only allows 16-bit accesses. Using 32-bit accesses produces undefined results.

This SPI has 16-bit transmit and receive capability, with double-buffered transmit and double-buffered receive. All data registers are 16-bits wide.

The SPI is no longer limited to a maximum transmission rate of LSPCLK/8 in slave mode. The maximum transmission rate in *both* slave mode *and* master mode is now LSPCLK/4.

Writes of transmit data to the serial data register, SPIDAT (and the new transmit buffer, SPITXBUF), must be left-justified within a 16-bit register.

The control and data bits for general-purpose bit I/O multiplexing have been removed from this peripheral, along with the associated registers, SPIPC1 (704Dh) and SPIPC2 (704Eh). These bits are now in the General-Purpose I/O registers.

Twelve registers inside the SPI module control the SPI operations:

- ❑ SPICCR (SPI configuration control register). Contains control bits used for SPI configuration

  - SPI module software reset
  - SPICLK polarity selection
  - Four SPI character-length control bits

- SPICTL (SPI operation control register). Contains control bits for data transmission
  - Two SPI interrupt enable bits
  - SPICLK phase selection
  - Operational mode (master/slave)
  - Data transmission enable

- SPISTS (SPI status register). Contains two receive buffer status bits and one transmit buffer status bit
  - RECEIVER OVERRUN
  - SPI INT FLAG
  - TX BUF FULL FLAG

- SPIBRR (SPI baud rate register). Contains seven bits that determine the bit transfer rate

- SPIRXEMU (SPI receive emulation buffer register). Contains the received data. This register is used for emulation purposes only. The SPIRXBUF should be used for normal operation

- SPIRXBUF (SPI receive buffer — the serial receive buffer register). Contains the received data

- SPITXBUF (SPI transmit buffer — the serial transmit buffer register). Contains the next character to be transmitted

- SPIDAT (SPI data register). Contains data to be transmitted by the SPI, acting as the transmit/receive shift register. Data written to SPIDAT is shifted out on subsequent SPICLK cycles. For every bit shifted out of the SPI, a bit from the receive bit stream is shifted into the other end of the shift register

- SPIPRI (SPI priority register). Contains bits that specify interrupt priority and determine SPI operation on the XDS™ emulator during program suspensions

1.3 SPI Operation

This section describes the operation of the SPI. Included are explanations of the operation modes, interrupts, data format, clock sources, and initialization. Typical timing diagrams for data transfers are given.

1.3.1 Introduction to Operation

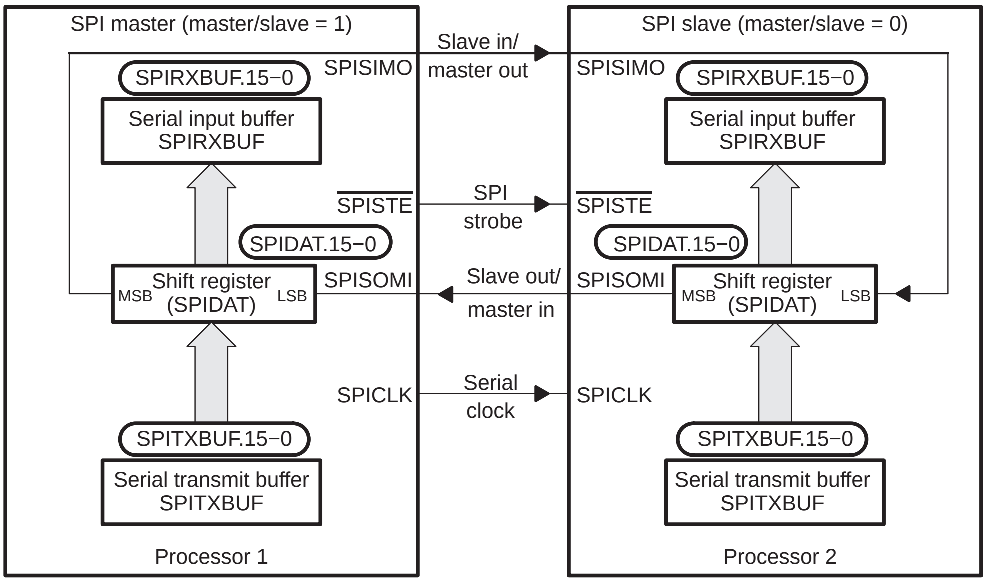
Figure 1−3 shows typical connections of the SPI for communications between two controllers: a master and a slave.

The master initiates data transfer by sending the SPICLK signal. For both the slave and the master, data is shifted out of the shift registers on one edge of the SPICLK and latched into the shift register on the opposite SPICLK clock edge. If the CLOCK PHASE bit (SPICTL.3) is high, data is transmitted and received a half-cycle before the SPICLK transition (see section 1.3.2, *SPI Module Slave and Master Operation Modes*, on page 1-9). As a result, both controllers send and receive data simultaneously. The application software determines whether the data is meaningful or dummy data. There are three possible methods for data transmission:

- Master sends data; slave sends dummy data.
- Master sends data; slave sends data.
- Master sends dummy data; slave sends data.

The master can initiate data transfer at any time because it controls the SPICLK signal. The software, however, determines how the master detects when the slave is ready to broadcast data.

Figure 1−3. SPI Master/Slave Connection



1.3.2 SPI Module Slave and Master Operation Modes

The SPI can operate in master or slave mode. The MASTER/SLAVE bit (SPICTL.2) selects the operating mode and the source of the SPICLK signal.

1.3.2.1 Master Mode

In the master mode (MASTER/SLAVE = 1), the SPI provides the serial clock on the SPICLK pin for the entire serial communications network. Data is output on the SPISIMO pin and latched from the SPISOMI pin.

The SPIBRR register determines both the transmit and receive bit transfer rate for the network. SPIBRR can select 126 different data transfer rates.

Data written to SPIDAT or SPITXBUF initiates data transmission on the SPISIMO pin, MSB (most significant bit) first. Simultaneously, received data is shifted through the SPISOMI pin into the LSB (least significant bit) of SPIDAT. When the selected number of bits has been transmitted, the received data is transferred to the SPIRXBUF (buffered receiver) for the CPU to read. Data is stored right-justified in SPIRXBUF.

When the specified number of data bits has been shifted through SPIDAT, the following events occur:

- ❑ SPIDAT contents are transferred to SPIRXBUF.
- ❑ SPI INT FLAG bit (SPISTS.6) is set to 1.

- ❑ If there is valid data in the transmit buffer SPITXBUF, as indicated by the TXBUF FULL bit in SPISTS, this data is transferred to SPIDAT and is transmitted; otherwise, SPICLK stops after all bits have been shifted out of SPIDAT.
- ❑ If the SPI INT ENA bit (SPICTL.0) is set to 1, an interrupt is asserted.

In a typical application, the $\overline{\text{SPISTE}}$ pin serves as a chip-enable pin for a slave SPI device. This pin is driven low by the master before transmitting data to the slave and is taken high after the transmission is complete.

1.3.2.2 Slave Mode

In the slave mode (MASTER/SLAVE = 0), data shifts out on the SPISOMI pin and in on the SPISIMO pin. The SPICLK 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. The SPICLK input frequency should be no greater than the LSPCLK frequency divided by 4.

Data written to SPIDAT or SPITXBUF is transmitted to the network when appropriate edges of the SPICLK signal are received from the network master. Data written to the SPITXBUF register will be transferred to the SPIDAT register when all bits of the character to be transmitted have been shifted out of SPIDAT. If no character is currently being transmitted when SPITXBUF is written to, the data will be transferred immediately to SPIDAT. To receive data, the SPI waits for the network master to send the SPICLK signal and then shifts the data on the SPISIMO pin into SPIDAT. If data is to be transmitted by the slave simultaneously, and SPITXBUF has not been previously loaded, the data must be written to SPITXBUF or SPIDAT before the beginning of the SPICLK signal.

When the TALK bit (SPICTL.1) is cleared, data transmission is disabled, and the output line (SPISOMI) is put into the high-impedance state. If this occurs while a transmission is active, the current character is completely transmitted even though SPISOMI is forced into the high-impedance state. This ensures that the SPI is still able to receive incoming data correctly. This TALK bit allows many slave devices to be tied together on the network, but only one slave at a time is allowed to drive the SPISOMI line.

The $\overline{\text{SPISTE}}$ pin operates as the slave-select pin. An active-low signal on the $\overline{\text{SPISTE}}$ pin allows the slave SPI to transfer data to the serial data line; an inactive-high signal causes the slave SPI serial shift register to stop and its serial output pin to be put into the high-impedance state. This allows many slave devices to be tied together on the network, although only one slave device is selected at a time.

1.4 SPI Interrupts

This section includes information on the control bits that initialize interrupts, data format, clocking, initialization, and data transfer.

1.4.1 SPI Interrupt Control Bits

Five control bits are used to initialize the SPI interrupts:

- ❑ SPI INT ENA bit (SPICTL.0)
- ❑ SPI INT FLAG bit (SPISTS.6)
- ❑ OVERRUN INT ENA bit (SPICTL.4)
- ❑ RECEIVER OVERRUN FLAG bit (SPISTS.7)

1.4.1.1 SPI INT ENA Bit (SPICTL.0)

When the SPI interrupt-enable bit is set and an interrupt condition occurs, the corresponding interrupt is asserted.

0	Disable SPI interrupts
1	Enable SPI interrupts

1.4.1.2 SPI INT FLAG Bit (SPISTS.6)

This status flag indicates that a character has been placed in the SPI receiver buffer and is ready to be read.

When a complete character has been shifted into or out of SPIDAT, the SPI INT FLAG bit (SPISTS.6) is set, and an interrupt is generated if enabled by the SPI INT ENA bit (SPICTL.0). The interrupt flag remains set until it is cleared by one of the following events:

- ❑ The interrupt is acknowledged (this is different from the C240).
- ❑ The CPU reads the SPIRXBUF (reading the SPIRXEMU does not clear the SPI INT FLAG bit).
- ❑ The device enters IDLE2 or HALT mode with an IDLE instruction.
- ❑ Software clears the SPI SW RESET bit (SPICCR.7).
- ❑ A system reset occurs.

When the SPI INT FLAG bit is set, a character has been placed into the SPIRX-BUF and is ready to be read. If the CPU does not read the character by the time the next complete character has been received, the new character is written into SPIRXBUF, and the RECEIVER OVERRUN Flag bit (SPISTS.7) is set.

1.4.1.3 OVERRUN INT ENA Bit (SPICTL.4)

Setting the overrun interrupt enable bit allows the assertion of an interrupt whenever the RECEIVER OVERRUN Flag bit (SPISTS.7) is set by hardware. Interrupts generated by SPISTS.7 and by the SPI INT FLAG bit (SPISTS.6) share the same interrupt vector.

0 Disable RECEIVER OVERRUN Flag bit interrupts

1 Enable RECEIVER OVERRUN Flag bit interrupts

1.4.1.4 RECEIVER OVERRUN FLAG Bit (SPISTS.7)

The RECEIVER OVERRUN Flag bit is set whenever a new character is received and loaded into the SPIRXBUF before the previously received character has been read from the SPIRXBUF. The RECEIVER OVERRUN Flag bit must be cleared by software.

1.4.2 Data Format

Four bits (SPICCR.3–0) specify the number of bits (1 to 16) in the data character. This information directs the state control logic to count the number of bits received or transmitted to determine when a complete character has been processed. The following statements apply to characters with fewer than 16 bits:

- Data must be left-justified when written to SPIDAT and SPITXBUF.
- Data read back from SPIRXBUF is right-justified.
- SPIRXBUF contains the most recently received character, right-justified, plus any bits that remain from previous transmission(s) that have been shifted to the left (shown in Example 1−1).

Example 1−1. Transmission of Bit From SPIRXBUF

Conditions:

1) Transmission character length = 1 bit (specified in bits SPICCR.3−0)
2) The current value of SPIDAT = 737Bh

SPIDAT (before transmission)

0	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1

SPIDAT (after transmission)

(TXed) 0 ←	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	x	← (RXed)

SPIRXBUF (after transmission)

1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	x

Note: x = 1 if SPISOMI data is high; x = 0 if SPISOMI data is low; master mode is assumed.

1.4.3 Baud Rate and Clocking Schemes

The SPI module supports 125 different baud rates and four different clock schemes. Depending on whether the SPI clock is in slave or master mode, the SPICLK pin can receive an external SPI clock signal or provide the SPI clock signal, respectively.

- In the slave mode, the SPI clock is received on the SPICLK pin from the external source, and can be no greater than the LSPCLK frequency divided by 4.
- In the master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin, and can be no greater than the LSPCLK frequency divided by 4.

Baud Rate Determination

Equation 1−1 shows how to determine the SPI baud rates.

Equation 1−1. SPI Baud-Rate Calculations

- For SPIBRR = 3 to 127:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)}$$

- For SPIBRR = 0, 1, or 2:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{4}$$

where:

LSPCLK = Low-speed peripheral clock frequency of the device

SPIBRR = Contents of the SPIBRR in the master SPI device

To determine what value to load into SPIBRR, you must know the device system clock (LSPCLK) frequency (which is device-specific) and the baud rate at which you will be operating.

Example 1−2 shows how to determine the maximum baud rate at which a 240xA can communicate. Assume that LSPCLK = 40 MHz.

Example 1−2. Maximum Baud-Rate Calculation

$$\begin{aligned}\text{Maximum SPI Baud Rate} &= \frac{\text{LSPCLK}}{4} \\ &= \frac{40 \times 10^6}{4} \\ &= 10 \times 10^6 \text{ bps}\end{aligned}$$

SPI Clocking Schemes

The CLOCK POLARITY bit (SPICCR.6) and the CLOCK PHASE bit (SPICTL.3) control four different clocking schemes on the SPICLK pin. The CLOCK POLARITY bit selects the active edge, either rising or falling, of the clock. The CLOCK PHASE bit selects a half-cycle delay of the clock. The four different clocking schemes are as follows:

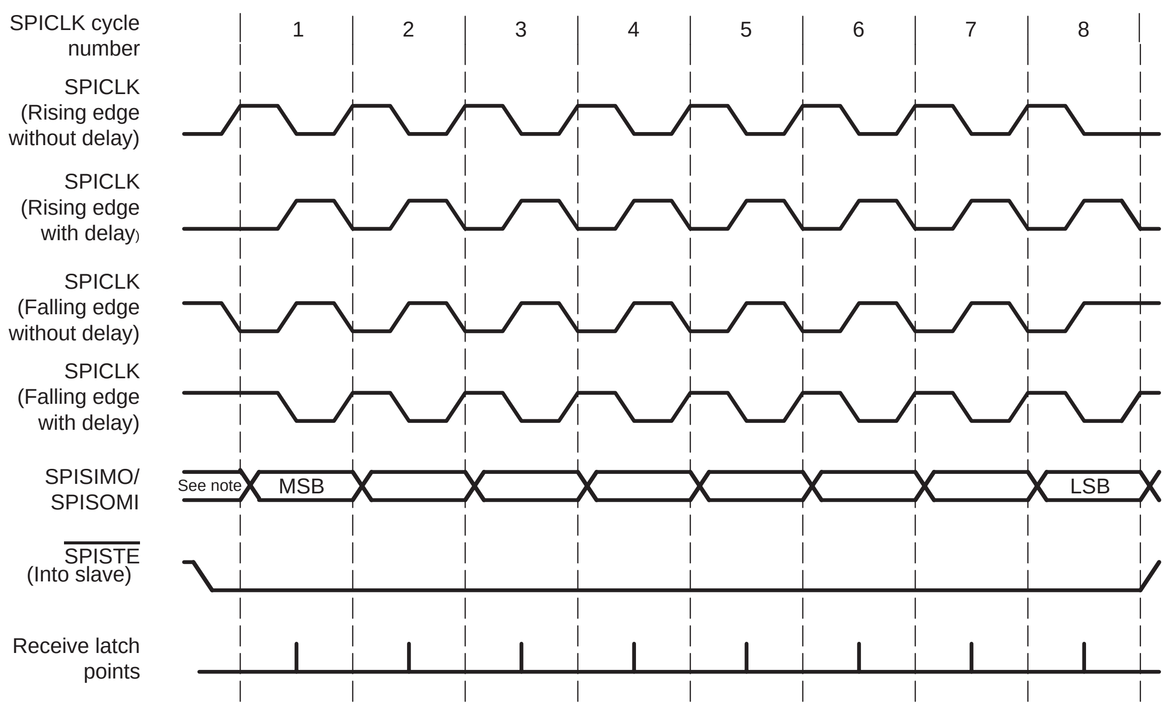
- Falling Edge Without Delay. The SPI transmits data on the falling edge of the SPICLK and receives data on the rising edge of the SPICLK.
- Falling Edge With Delay. The SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
- Rising Edge Without Delay. The SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
- Rising Edge With Delay. The SPI transmits data one half-cycle ahead of the rising edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.

The selection procedure for the SPI clocking scheme is shown in Table 1−2. Examples of these four clocking schemes relative to transmitted and received data are shown in Figure 1−4.

Table 1−2. SPI Clocking Scheme Selection Guide

SPICLK Scheme	CLOCK POLARITY (SPICCR.6)	CLOCK PHASE (SPICTL.3)
Rising edge without delay	0	0
Rising edge with delay	0	1
Falling edge without delay	1	0
Falling edge with delay	1	1

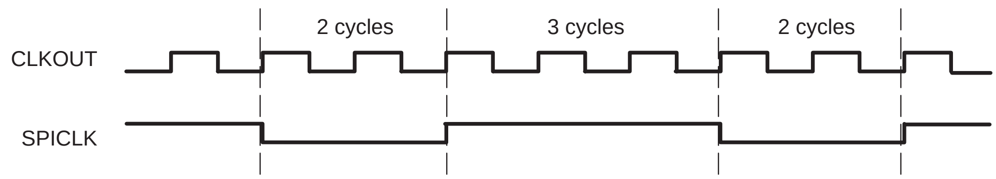
Figure 1−4. SPICLK Signal Options



Note: Previous data bit

For the SPI, SPICLK symmetry is retained only when the result of (SPIBRR+1) is an even value. When (SPIBRR + 1) is an odd value and SPIBRR is greater than 3, SPICLK becomes asymmetrical. The low pulse of SPICLK is one CLKOUT longer than the high pulse when the CLOCK POLARITY bit is clear (0). When the CLOCK POLARITY bit is set to 1, the high pulse of the SPICLK is one CLKOUT longer than the low pulse, as shown in Figure 1−5.

Figure 1−5. SPI: SPICLK-CLKOUT Characteristic When (BRR + 1) is Odd, BRR > 3, and CLOCK POLARITY = 1



1.4.4 Initialization Upon Reset

A system reset forces the SPI peripheral module into the following default configuration:

- ❑ Unit is configured as a slave module (MASTER/SLAVE = 0)
- ❑ Transmit capability is disabled (TALK = 0)
- ❑ Data is latched at the input on the falling edge of the SPICLK signal
- ❑ Character length is assumed to be one bit
- ❑ SPI interrupts are disabled
- ❑ Data in SPIDAT is reset to 0000h
- ❑ SPI module pin functions are selected as general-purpose inputs (this is done in I/O MUX control register B [MCRB])

To change this SPI configuration:

1) Clear the SPI SW RESET bit (SPICCR.7) to 0 to force the SPI to the reset state.

2) Initialize the SPI configuration, format, baud rate, and pin functions as desired.

3) Set the SPI SW RESET bit to 1 to release the SPI from the reset state.

4) Write to SPIDAT or SPITXBUF (this initiates the communication process in the master).

5) Read SPIRXBUF after the data transmission has completed (SPISTS.6 = 1) to determine what data was received.

To prevent unwanted and unforeseen events from occurring during or as a result of initialization changes, clear the SPI SW RESET bit (SPICCR.7) before making initialization changes, and then set this bit after initialization is complete.

Note:

Do not change SPI configuration when communication is in progress.

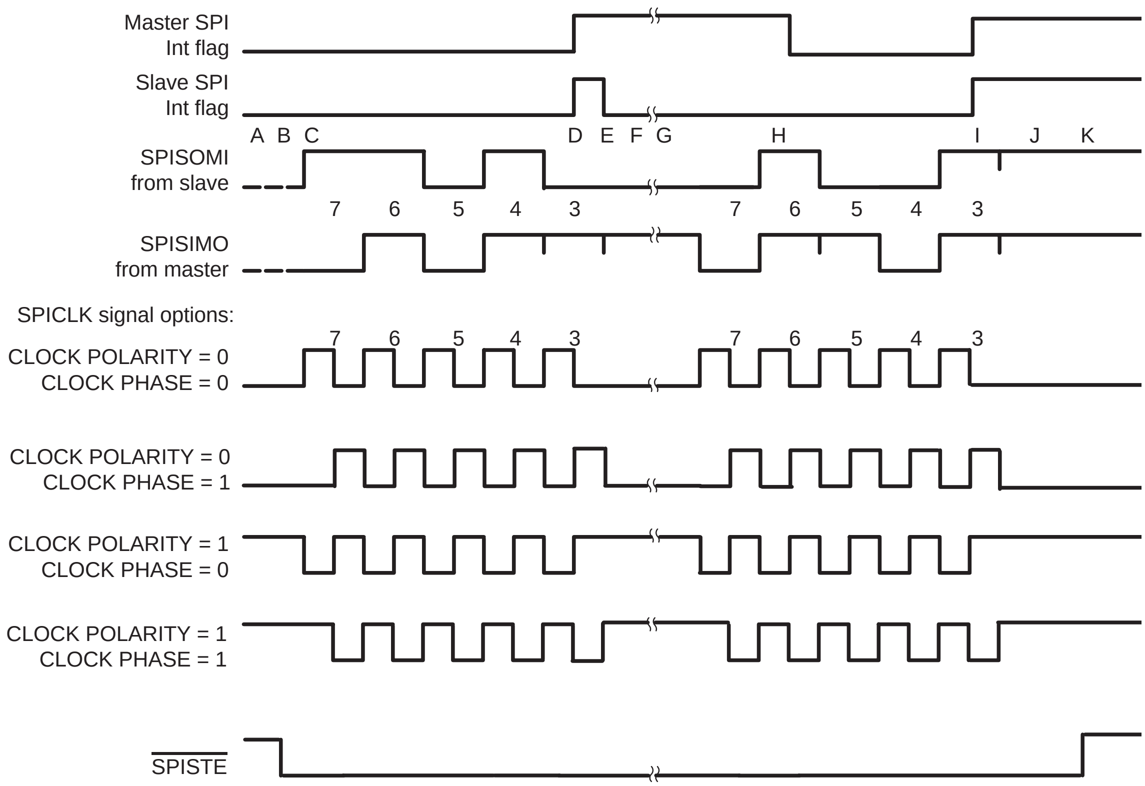
1.4.5 Data Transfer Example

The timing diagram shown in Figure 1−6 illustrates an SPI data transfer between two devices using a character length of five bits with the SPICLK being symmetrical.

The timing diagram with SPICLK unsymmetrical (Figure 1−5) shares similar characterizations with Figure 1−6 except that the data transfer is one CLKOUT cycle longer per bit during the low pulse (CLOCK POLARITY = 0) or during the high pulse (CLOCK POLARITY = 1) of the SPICLK.

Figure 1−6 is applicable for 8-bit SPI only and is not for 24x devices that are capable of working with 16-bit data. The figure is shown for illustrative purposes only.

Figure 1−6. Five Bits per Character



A. Slave writes 0D0h to SPIDAT and waits for the master to shift out the data.
B. Master sets the slave $\overline{\text{SPISTE}}$ signal low (active).
C. Master writes 058h to SPIDAT, which starts the transmission procedure.
D. First byte is finished and sets the interrupt flags.
E. Slave reads 0Bh from its SPIRXBUF (right-justified).
F Slave writes 04Ch to SPIDAT and waits for the master to shift out the data.
G. Master writes 06Ch to SPIDAT, which starts the transmission procedure.
H. Master reads 01Ah from the SPIRXBUF (right−justified).
I. Second byte is finished and sets the interrupt flags.
J. Master reads 89h and the slave reads 8Dh from their respective SPIRXBUF. After the user's software masks off the unused bits, the master receives 09h and the slave receives 0Dh.
K. Master clears the slave $\overline{\text{SPISTE}}$ signal high (inactive).

1.5 SPI FIFO Description

The following steps explain the the FIFO features and help with programming the SPI FIFOs:

1) Reset. At reset the SPI powers up in standard SPI mode, the FIFO function is disabled. The FIFO registers SPIFFTX, SPIFFRX and SPIFFCT remain inactive.

2) Standard SPI. The standard 240x SPI mode will work with SPIINT/SPIRXINT as the interrupt source.

3) Mode change. FIFO mode is enabled by setting the SPIFFEN bit to 1 in the SPIFFTX register. SPIRST can reset the FIFO mode at any stage of its operation.

4) Active registers. All the SPI registers and SPI FIFO registers SPIFFTX, SPIFFRX, and SPIFFCT will be active.

5) Interrupts. FIFO mode has two interrupts one for transmit FIFO, SPITXINT and one for receive FIFO, SPIINT/SPIRXINT. SPIINT/SPIRXINT is the common interrupt for SPI FIFO receive, receive error and receive FIFO overflow conditions. The single SPIINT for both transmit and receive sections of the standard SPI will be disabled and this interrupt will service as SPI receive FIFO interrupt.

6) Buffers. Transmit and receive buffers are supplemented with two 16x16 FIFOs. The one-word transmit buffer (TXBUF) of the standard SPI functions as a transition buffer between the transmit FIFO and shift register. The one-word transmit buffer will be loaded from transmit FIFO only after the last bit of the shift register is shifted out.

7) Delayed transfer. The rate at which transmit words in the FIFO are transferred to transmit shift register is programmable. The SPIFFCT register bits (7−0) FFTXDLY7−FFTXDLY0 define the delay between the word transfer. The delay is defined in number SPI serial clock cycles. The 8-bit register could define a minimum delay of 0 serial clock cycles and a maximum of 256 serial clock cycles. With zero delay, the SPI module can transmit data in continuous mode with the FIFO words shifting out back to back. With the 256 clock delay, the SPI module can transmit data in a maximum delayed mode with the FIFO words shifting out with a delay of 256 SPI clocks between each words. The programmable delay facilitates glueless interface to various slow SPI peripherals, such as EEPROMs, ADC, DAC etc.

8) FIFO status bits. Both transmit and receive FIFOs have status bits TXFFST or RXFFST (bits 12− 0) that define the number of words available

in the FIFOs at any time. The transmit FIFO reset bit TXFIFO and receive reset bit RXFIFO will reset the FIFO pointers to zero when these bits are set to 1. The FIFOs will resume operation from start once these bits are cleared to zero.

9) Programmable interrupt levels. Both transmit and receive FIFO can generate CPU interrupts. The interrupt trigger is generated whenever the transmit FIFO status bits TXFFST (bits 12−8) match (less than or equal to) the interrupt trigger level bits TXFFIL (bits 4−0). This provides a programmable interrupt trigger for transmit and receive sections of the SPI. The default value for these trigger level bits will be 0x11111 for receive FIFO and 0x00000 for transmit FIFO respectively.

1.5.1 SPI Interrupts

Figure 1−7. SPI FIFO Interrupt Flags and Enable Logic Generation

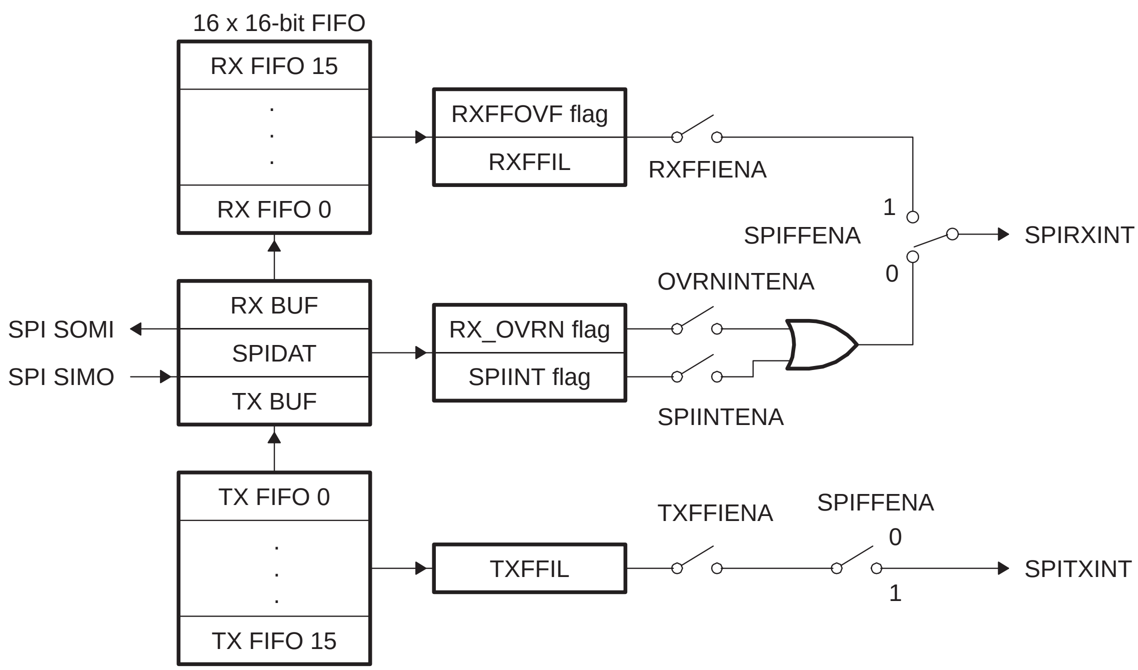


Table 1−3. SPI Interrupt Flag Modes

FIFO Options	SPI Interrupt Source	Interrupt Flags	Interrupt Enables	FIFO Enable SPIFFENA	Interrupt line
SPI without FIFO					
	Receive overrun	RXOVRN	OVRNINTENA	0	SPIRXINT†
	Data receive	SPIINT	SPIINTENA	0	SPIRXINT†
	Transmit empty	SPIINT	SPIINTENA	0	SPIRXINT†
SPI FIFO mode					
	FIFO receive	RXFFIL	RXFFIENA	1	SPIRXINT†
	Transmit empty	TXFFIL	TXFFIENA	1	SPITXINT†

† In nonFIFO mode, SPIRXINT is the same as the SPIINT interrupt in 240x devices.

Chapter 2

SPI Registers and Waveforms

This section contains the registers, bit descriptions, and waveforms.

Topic **Page**

2.1 SPI Control Registers

The SPI is controlled and accessed through registers in the control register file.

2.1.1 SPI Configuration Control Register (SPICCR)

SPICCR controls the setup of the SPI for operation.

Figure 2−1. SPI Configuration Control Register (SPICCR) — Address 7040h

7	6	5	4	3	2	1	0
SPI SW Reset	CLOCK POLARITY	Reserved	SPILBK	SPI CHAR3	SPI CHAR2	SPI CHAR1	SPI CHAR0
R/W-0	R/W−0	R−0	R−0	R/W−0	R/W−0	R/W−0	R/W−0

Legend: R = Read access; W = Write access; −x = value after reset

Table 2−1. SPI Configuration Control Register (SPICCR) Bit Descriptions

Bit(s)	Name	Description
7	SPI SW RESET	SPI software reset. When changing configuration, you should clear this bit before the changes and set this bit before resuming operation. 1 — SPI is ready to transmit or receive the next character. When the SPI SW RESET bit is a 0, a character written to the transmitter will not be shifted out when this bit is set. A new character must be written to the serial data register. 0 — Initializes the SPI operating flags to the reset condition. Specifically, the RECEIVER OVERRUN Flag bit (SPISTS.7), the SPI INT FLAG bit (SPISTS.6), and the TXBUF FULL Flag bit (SPISTS.5) are cleared. The SPI configuration remains unchanged. If the module is operating as a master, the SPICLK signal output returns to its inactive level.
6	CLOCK POLARITY	Shift Clock Polarity. This bit controls the polarity of the SPICLK signal. CLOCK POLARITY and CLOCK PHASE (SPICTL.3) control four clocking schemes on the SPICLK pin. See Section 1.4.3, *SPI Clocking Schemes*, on page 1-13. 1 — Data is output on falling edge and input on rising edge. When no SPI data is sent, SPICLK is at high level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows: ❑ CLOCK PHASE = 0: Data is output on the falling edge of the SPICLK signal; input data is latched on the rising edge of the SPICLK signal. ❑ CLOCK PHASE = 1: Data is output one half-cycle before the first falling edge of the SPICLK signal and on subsequent rising edges of the SPICLK signal; input data is latched on the falling edge of the SPICLK signal.

Table 2−1. SPI Configuration Control Register (SPICCR) Bit Descriptions (Continued)

Bit(s)	Name	Description	
6	CLOCK POLARITY (continued)	0	Data is output on rising edge and input on falling edge. When no SPI data is sent, SPICLK is at low level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows: ❑ CLOCK PHASE = 0: Data is output on the rising edge of the SPICLK signal; input data is latched on the falling edge of the SPICLK signal. ❑ CLOCK PHASE = 1: Data is output one half-cycle before the first rising edge of the SPICLK signal and on subsequent falling edges of the SPICLK signal; input data is latched on the rising edge of the SPICLK signal.
5	Reserved	Reads return zero; writes have no effect.	
4	SPILBK	SPI loopback. Loop back mode allows module validation during device testing. This mode is valid only in master mode of the SPI.	
		1	SPI loop back mode enabled, SIMO/SOMI lines are connected internally. Used for module self tests.
		0	SPI loop back mode disabled – default value after reset
3−0	SPI CHAR3 − SPI CHAR0	Character Length Control Bits 3−0. These four bits determine the number of bits to be shifted in or out as a single character during one shift sequence. Table 2−2 lists the character length selected by the bit values.	

Table 2−2. Character Length Control Bit Values

SPI CHAR3	SPI CHAR2	SPI CHAR1	SPI CHAR0	Character Length
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
0	0	1	1	4
0	1	0	0	5
0	1	0	1	6
0	1	1	0	7
0	1	1	1	8
1	0	0	0	9
1	0	0	1	10
1	0	1	0	11
1	0	1	1	12
1	1	0	0	13
1	1	0	1	14

Table 2−2. Character Length Control Bit Values (Continued)

SPI CHAR3	SPI CHAR2	SPI CHAR1	SPI CHAR0	Character Length
1	1	1	0	15
1	1	1	1	16

2.1.2 SPI Operation Control Register (SPICTL)

SPICTL controls data transmission, the SPI's ability to generate interrupts, the SPICLK phase, and the operational mode (slave or master).

Figure 2−2. SPI Operation Control Register (SPICTL) — Address 7041h

7–5	4	3	2	1	0
Reserved	OVERRUN INT ENA	CLOCK PHASE	MASTER/ SLAVE	TALK	SPI INT ENA
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access; W = Write access; −x = value after reset

Table 2−3. SPI Operation Control Register (SPICTL) Bit Descriptions

Bit(s)	Name	Description
7−5	Reserved	Reads return zero; writes have no effect.
4	Overrun INT ENA	Overrun Interrupt Enable. Setting this bit causes an interrupt to be generated when the RECEIVER OVERRUN Flag bit (SPISTS.7) is set by hardware. Interrupts generated by the RECEIVER OVERRUN Flag bit and the SPI INT FLAG bit (SPISTS.6) share the same interrupt vector. 1 Enable RECEIVER OVERRUN Flag bit (SPISTS.7) interrupts 0 Disable RECEIVER OVERRUN Flag bit (SPISTS.7) interrupts
3	CLOCK PHASE	SPI Clock Phase Select. This bit controls the phase of the SPICLK signal. CLOCK PHASE and CLOCK POLARITY (SPICCR.6) make four different clocking schemes possible (see Figure 1−4). When operating with CLOCK PHASE high, the SPI (master or slave) makes the first bit of data available after SPIDAT is written and before the first edge of the SPICLK signal, regardless of which SPI mode is being used. 1 SPICLK signal delayed by one half-cycle; polarity determined by the CLOCK POLARITY bit 0 Normal SPI clocking scheme, depending on the CLOCK POLARITY bit (SPICCR.6)

Table 2−3. SPI Operation Control Register (SPICTL) Bit Descriptions (Continued)

Bit(s)	Name	Description
2	MASTER / SLAVE	SPI Network Mode Control. This bit determines whether the SPI is a network master or slave. During reset initialization, the SPI is automatically configured as a network slave. 1 SPI configured as a master. 0 SPI configured as a slave.
1	TALK	Master/Slave Transmit Enable. The TALK bit can disable data transmission (master or slave) by placing the serial data output in the high-impedance state. If this bit is disabled during a transmission, the transmit shift register continues to operate until the previous character is shifted out. When the TALK bit is disabled, the SPI is still able to receive characters and update the status flags. TALK is cleared (disabled) by a system reset. 1 Enables transmission For the 4-pin option, ensure to enable the receiver's $\overline{\text{SPISTE}}$ input pin. 0 Disables transmission: ❑ Slave mode operation: If not previously configured as a general-purpose I/O pin, the SPISOMI pin will be put in the high-impedance state. ❑ Master mode operation: If not previously configured as a general-purpose I/O pin, the SPISIMO pin will be put in the high-impedance state.
0	SPI INT ENA	SPI Interrupt Enable. This bit controls the SPI's ability to generate a transmit/receive interrupt. The SPI INT FLAG bit (SPISTS.6) is unaffected by this bit. 1 Enables interrupt 0 Disables interrupt

2.1.3 SPI Status Register (SPIST)

Figure 2−3. SPI Status Register (SPIST) — Address 7042h

7	6	5	4–0
RECEIVER OVERRUN FLAG†‡	SPI INT FLAG†‡	TX BUF FULL FLAG‡	Reserved
R/C-0	R/C−0	R/C−0	R−0

Legend: R = Read access; C = Clear; −x = value after reset

† The RECEIVER OVERRUN FLAG bit and the SPI INT FLAG bit share the same interrupt vector.

‡ Writing a 0 to bits 5, 6, and 7 has no effect.

Table 2−4. SPI Status Register (SPIST) Bit Descriptions

Bit(s)	Name	Description
7	RECEIVER OVERRUN FLAG	SPI Receiver Overrun Flag. This bit is a read/clear-only flag. The SPI hardware sets this bit when a receive or transmit operation completes before the previous character has been read from the buffer. The bit indicates that the last received character has been overwritten and therefore lost (when the SPIRXBUF was overwritten by the SPI module before the previous character was read by the user application). The SPI requests one interrupt sequence each time this bit is set if the OVERRUN INT ENA bit (SPICTL.4) is set high. The bit is cleared in one of three ways: ❑ Writing a 1 to this bit ❑ Writing a 0 to SPI SW RESET (SPICCR.7) ❑ Resetting the system If the OVERRUN INT ENA bit (SPICTL.4) is set, the SPI requests only one interrupt upon the first occurrence of setting the RECEIVER OVERRUN Flag bit. Subsequent overruns will not request additional interrupts if this flag bit is already set. This means that in order to allow *new* overrun interrupt requests the user must clear this flag bit by writing a 1 to SPISTS.7 each time an overrun condition occurs. In other words, if the RECEIVER OVERRUN Flag bit is left set (not cleared) by the interrupt service routine, another overrun interrupt will not be immediately re-entered when the interrupt service routine is exited. However, the RECEIVER OVERRUN Flag bit should be cleared during the interrupt service routine because the RECEIVER OVERRUN Flag bit and SPI INT FLAG bit (SPISTS.6) share the same interrupt vector. This will alleviate any possible doubt as to the source of the interrupt when the next byte is received.
6	SPI INT FLAG	SPI Interrupt Flag. SPI INT FLAG is a read-only flag. The SPI hardware sets this bit to indicate that it has completed sending or receiving the last bit and is ready to be serviced. The received character is placed in the receiver buffer at the same time this bit is set. This flag causes an interrupt to be requested if the SPI INT ENA bit (SPICTL.0) is set. This bit is cleared in one of three ways: ❑ Reading SPIRXBUF ❑ Writing a 0 to SPI SW RESET (SPICCR.7) ❑ Resetting the system
5	TX BUF FULL FLAG	SPI Transmit Buffer Full Flag. This read-only bit gets set to 1 when a character is written to the SPI Transmit buffer SPITXBUF. It is cleared when the character is automatically loaded into SPIDAT when the shifting out of a previous character is complete. It is cleared at reset.
4−0	Reserved	Reads return zero; writes have no effect.

2.1.4 SPI Baud Rate Register (SPIBRR)

SPIBRR contains the bits used for baud-rate selection.

Figure 2−4. SPI Baud Rate Register (SPIBRR) — Address 7044h

7	6	5	4	3	2	1	0
Reserved	SPI BIT RATE 6	SPI BIT RATE 5	SPI BIT RATE 4	SPI BIT RATE 3	SPI BIT RATE 2	SPI BIT RATE 1	SPI BIT RATE 0
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−5. SPI Baud Rate Register (SPIBRR) Bit Descriptions

Bit(s)	Name	Description
7	Reserved	Reads return zero; writes have no effect.
6−0	SPI BIT RATE 6− SPI BIT RATE 0	SPI Bit Rate (Baud) Control. These bits determine the bit transfer rate if the SPI is the network master. There are 125 data-transfer rates (each a function of the CPU clock, LSPCLK) that can be selected. One data bit is shifted per SPICLK cycle. (SPICLK is the baud rate clock output on the SPICLK pin.) If the SPI is a network slave, the module receives a clock on the SPICLK pin from the network master; therefore, these bits have no effect on the SPICLK signal. The frequency of the input clock from the master should not exceed the slave SPI's SPICLK signal divided by 4. In master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin. The SPI baud rates are determined by the following formula: ❑ For SPIBRR = 3 to 127: $\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)}$ ❑ For SPIBRR = 0, 1, or 2: $\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{4}$ where: LSPCLK = Function of CPU clock frequency X low-speed peripheral clock of the device SPIBRR = Contents of the SPIBRR in the master SPI device

2.1.5 SPI Emulation Buffer Register (SPIRXEMU)

SPIRXEMU contains the received data. Reading SPIRXEMU does not clear the SPI INT FLAG bit (SPISTS.6). This is not a real register but a dummy address from which the contents of SPIRXBUF can be read by the emulator without clearing the SPI INT FLAG.

Figure 2−5. SPI Emulation Buffer Register (SPIRXEMU) — Address 7046h

15	14	13	12	11	10	9	8
ERXB15	ERXB14	ERXB13	ERXB12	ERXB11	ERXB10	ERXB9	ERXB8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
ERXB7	ERXB6	ERXB5	ERXB4	ERXB3	ERXB2	ERXB1	ERXB0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Legend: R = Read access, -0 = value after reset

Table 2−6. SPI Emulation Buffer Register (SPIRXEMU) Bit Descriptions

Bit(s)	Name	Description
15−0	ERXB15−ERXB0	Emulation Buffer Received Data. SPIRXEMU functions almost identically to SPIRXBUF, except that reading SPIRXEMU does not clear the SPI INT FLAG bit (SPISTS.6). Once the SPIDAT has received the complete character, the character is transferred to SPIRXEMU and SPIRXBUF, where it can be read. At the same time, SPI INT FLAG is set. This mirror register was created to support emulation. Reading SPIRXBUF clears the SPI INT FLAG bit (SPISTS.6). In the normal operation of the emulator, the control registers are read to continually update the contents of these registers on the display screen. SPIRXEMU was created so that the emulator can read this register and properly update the contents on the display screen. Reading SPIRXEMU does not clear the SPI INT FLAG bit, but reading SPIRXBUF clears this flag. In other words, SPIRXEMU enables the emulator to emulate the true operation of the SPI more accurately. It is recommended that you view SPIRXEMU in the normal emulator run mode.

2.1.6 SPI Serial Receive Buffer Register (SPIRXBUF)

SPIRXBUF contains the received data. Reading SPIRXBUF clears the SPI INT FLAG bit (SPISTS.6).

Figure 2−6. SPI Serial Receive Buffer Register (SPIRXBUF) — Address 7047h

15	14	13	12	11	10	9	8
RXB15	RXB14	RXB13	RXB12	RXB11	RXB10	RXB9	RXB8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Legend: R = Read access, -0 = value after reset

Table 2−7. SPI Serial Receive Buffer Register (SPIRXBUF Bit Descriptions

Bit(s)	Name	Description
15−0	RXV15 − RXB0	Received Data. Once SPIDAT has received the complete character, the character is transferred to SPIRXBUF, where it can be read. At the same time, the SPI INT FLAG bit (SPISTS.6) is set. Since data is shifted into the SPI's most significant bit first, it is stored right-justified in this register.

2.1.7 SPI Serial Transmit Buffer Register (SPITXBUF)

SPITXBUF stores the next character to be transmitted. Writing to this register sets the TX BUF FULL Flag bit (SPISTS.5). When transmission of the current character is complete, the contents of this register are automatically loaded in SPIDAT and the TX BUF FULL Flag is cleared. If no transmission is currently active, data written to this register falls through into the SPIDAT register and the TX BUF FULL Flag is not set.

In master mode, if no transmission is currently active, writing to this register initiates a transmission in the same manner that writing to SPIDAT does.

Figure 2−7. SPI Serial Transmit Buffer Register (SPITXBUF) — Address 7048h

15	14	13	12	11	10	9	8
TXB15	TXB14	TXB13	TXB12	TXB11	TXB10	TXB9	TXB8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−8. SPI Serial Transmit Buffer Register (SPITXBUF) Bit Descriptions

Bit(s)	Name	Description
15−0	TXV15−TXV0	Transmit Data Buffer. This is where the next character to be transmitted is stored. When the transmission of the current character has completed, if the TX BUF FULL Flag bit is set, the contents of this register is automatically transferred to SPIDAT, and the TX BUF FULL Flag is cleared. **Note:** Writes to SPITXBUF must be left-justified.

2.1.8 SPI Serial Data Register (SPIDAT)

SPIDAT is the transmit/receive shift register. Data written to SPIDAT is shifted out (MSB) on subsequent SPICLK cycles. For every bit (MSB) shifted out of the SPI, a bit is shifted into the LSB end of the shift register.

Figure 2−8. SPI Serial Data Register (SPIDAT) — Address 7049h

15	14	13	12	11	10	9	8
SDAT15	SDAT14	SDAT13	SDAT12	SDAT11	SDAT10	SDAT9	SDAT8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−9. SPI Serial Data Register (SPIDAT)Bit Descriptions

Bit(s)	Name	Description
15−0	SDAT15−SDAT0	Serial data. Writing to the SPIDAT performs two functions: ❑ It provides data to be output on the serial output pin if the TALK bit (SPICTL.1) is set. ❑ When the SPI is operating as a master, a data transfer is initiated. When initiating a transfer, see the CLOCK POLARITY bit (SPICCR.6) described in section 2.1.1, *SPI Configuration Control Register* on page 2-2, and the CLOCK PHASE bit (SPICTL.3) described in section 2.1.2, *SPI Operation Control Register* on page 2-4, for the requirements. In master mode, writing dummy data to SPIDAT initiates a receiver sequence. Since the data is not hardware-justified for characters shorter than sixteen bits, transmit data must be written in left-justified form, and received data read in right-justified form.

2.1.9 SPI FIFO Transmit, Receive, and Control Registers

Figure 2−9. SPI FIFO Transmit (SPIFFTX) Register − Address 704Ah

15	14	13	12	11	10	9	8
SPIRST	SPIFFENA	TXFIFO	TXFFST4	TXFFST3	TXFFST2	TXFFST1	TXFFST0
R/W-1	R/W−0	R/W−1	R−0	R−0	R−0	R−0	R−0

7	6	5	4	3	2	1	0
TXFFINT Flag	TXFFINT CLR	TXFFIENA	TXFFIL4	TXFFIL3	TXFFIL2	TXFFIL1	TXFFIL0
R/W-0	W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−10. SPI FIFO Transmit (SPIFFTX) Register Bit Descriptions

Bit(s)	Name	Description	
15	SPIRST	SPI reset	
		0	Write 0 to reset the SPI transmit and receive channels. The SPI FIFO register configuration bits will be left as is.
		1	SPI FIFO can resume transmit or receive. No effect to the SPI registers bits.
14	SPIFFENA	SPI FIFO enhancements enable	
		0	SPI FIFO enhancements are disabled
		1	SPI FIFO enhancements are enabled
13	TXFIFO Reset	Transmit FIFO reset	
		0	Write 0 to reset the FIFO pointer to zero, and hold in reset.
		1	Re-enable Transmit FIFO operation
8−12	TXFFST4−0	Transmit FIFO status	
		00000	Transmit FIFO is empty.
		00001	Transmit FIFO has 1 word.
		00010	Transmit FIFO has 2 words.
		00011	Transmit FIFO has 3 words.
		0xxxx	Transmit FIFO has x words.
		10000	Transmit FIFO has 16 words.
7	TXFFINT	TXFIFO interrupt	
		0	TXFIFO interrupt has not occurred, This is a read-only bit.
		1	TXFIFO interrupt has occurred, This is a read-only bit.
6	TXFFINT CLR	TXFIFO clear	
		0	Write 0 has no effect on TXFIFINT flag bit, Bit reads back a zero.
		1	Write 1 to clear TXFFINT flag in bit 7.
5	TXFFIENA	TX FIFO interrupt enable	
		0	TX FIFO interrupt based on TXFFIVL match (less than or equal to) will be disabled .
		1	TX FIFO interrupt based on TXFFIVL match (less than or equal to) will be enabled.
0−4	TXFFIL4−0	00000	TXFFIL4−0 transmit FIFO interrupt level bits. Transmit FIFO will generate interrupt when the FIFO status bits (TXFFST4−0) and FIFO level bits (TXFFIL4−0) match (less than or equal to). Default value is 0x00000.

Figure 2−10. SPI FIFO Receive (SPIFFRX) Register − Address 704Bh

15	14	13	12	11	10	9	8
RXFFOVF Flag	RXFFOVF CLR	RXFIFO Reset	RXFFST4	RXFFST3	RXFFST2	RXFFST1	RXFFST0
R-0	W−0	R/W−1	R−0	R−0	R−0	R−0	R−0

7	6	5	4	3	2	1	0
RXFFINT Flag	RXFFINT CLR	RXFFIENA	RXFFIL4	RXFFIL3	RXFFIL2	RXFFIL1	RXFFIL0
R-0	W−0	R/W−0	R/W−1	R/W−1	R/W−1	R/W−1	R/W−1

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−11. SPI FIFO Receive (SPIFFRX) Register Bit Descriptions

Bit(s)	Name	Description	
15	RXFFOVF	Receive FIFO overflow flag	
		0	Receive FIFO has not overflowed. This is a read-only bit.
		1	Receive FIFO has overflowed, read-only bit. More than 16 words have been received in to the FIFO, and the first received word is lost.
14	RXFFOVF CLR	Receive FIFO overflow clear	
		0	Write 0 does not affect RXFFOVF flag bit, Bit reads back a zero
		1	Write 1 to clear RXFFOVF flag in bit 15
13	RXFIFO Reset	Receive FIFO reset	
		0	Write 0 to reset the FIFO pointer to zero, and hold in reset.
		1	Re-enable transmit FIFO operation
8−12	RXFFST4−0	00000	Receive FIFO is empty.
		00001	Receive FIFO has 1 word.
		00010	Receive FIFO has 2 words.
		00011	Receive FIFO has 3 words.
		0xxxx	Receive FIFO has x word.
			Note: 10000: Receive FIFO has 16 words.
7	RXFFINT	Receive FIFO interrupt	
		0	RXFIFO interrupt has not occurred. This is a read-only bit.
		1	RXFIFO interrupt has occurred. This is a read-only bit.
6	RXFFINT CLR	Receive FIFO interrupt clear	
		0	Write 0 has no effect on RXFIFINT flag bit, Bit reads back a zero.
		1	Write 1 to clear RXFFINT flag in bit 7.

Table 2−11. SPI FIFO Receive (SPIFFRX) Register Bit Descriptions (Continued)

Bit(s)	Name	Description	
5	RXFFIENA	RX FIFO interrupt enable	
		0	RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be disabled.
		1	RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be enabled.
4−0	RXFFIL4−0	Receive FIFO interrupt level bits	
		11111	Receive FIFO will generate interrupt when the FIFO status bits (RXFFST4−0) and FIFO level bits (RXFFIL4−0) match (greater than or equal to) the default value of these bits after reset − 11111. This will avoid frequent interrupts after reset, as the receive FIFO will be empty most of the time.

Figure 2−11. SPI FIFO Control (SPIFFCT) Register − Address 704Ch

15–8
Reserved
R-0

7	6	5	4	3	2	1	0
FFTXDLY7	FFTXDLY6	FFTXDLY5	FFTXDLY4	FFTXDLY3	FFTXDLY2	FFTXDLY1	FFTXDLY0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−12. SPI FIFO Control (SPIFFCT) Register Bit Descriptions

Bit(s)	Name	Description
15−8	Reserved	Reserved
7−0	FFTXDLY7−0	FIFO transmit delay bits These bits define the delay between every transfer from FIFO transmit buffer to transmit shift register. The delay is defined in number SPI serial clock cycles. The 8 bit register could define a minimum delay of 0 serial clock cycles and a maximum of 25 serial clock cycles. In the FIFO mode the buffer (TXBUF) between the shift register and the FIFO should be filled only after the shift register has completed shifting of the last bit. This is required to pass on the delay between transfers to the data stream. In the FIFO mode TXBUF should not be treated as one additional level of buffer.

2.1.10 SPI Priority Control Register (SPIPRI)

Figure 2−12. SPI Priority Control Register (SPIPRI) — Address 704Fh

7	6	5	4	3–0
Reserved	Reserved	SPI SUSP SOFT	SPI SUSP FREE	Reserved
R-0	R/W-0	R/W	R/W−0	R−0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−13. SPI Priority Control Register (SPIPRI) Bit Descriptions

Bit(s)	Name	Description
7:6	Reserved	Reads return zero; writes have no effect.
5−4	SPI SUSP SOFT SPI SUSP FREE	These bits determine what occurs when an emulation suspend occurs (for example, when the debugger hits a breakpoint). The peripheral can continue whatever it is doing (free-run mode) or, if in stop mode, it can either stop immediately or stop when the current operation (the current receive/transmit sequence) is complete. Bit 5 SOFT / Bit 4 FREE 0 0 — Transmission will stop after midway in the bit stream while TSPEND is asserted. Once TSUSPEND is deasserted without a system reset, the remainder of the bits pending in the DATBUF will be shifted. Example: If SPIDAT has shifted 3 out of 8 bits, the communication will freeze right there. However, if TSUSPEND is later deasserted without resetting the SPI, SPI will start transmitting from where it had stopped (fourth bit in this case) and will transmit 8 bits from that point. The SCI module operates differently. 1 0 — If the emulation suspend occurs before the start of a transmission, (i.e., before the first SPICLK pulse) then the transmission will not occur. If the emulation suspend occurs after the start of a transmission, then the data will be shifted out to completion. When the start of transmission occurs is dependent on the baud rate used. *Standard SPI mode:* Stop after transmitting the words in the shift register and buffer. That is after TXBUF and SPIDAT is empty. *In the FIFO mode:* Stop after transmitting the words in the shift register and buffer. That is after TX FIFO and SPIDAT is empty. x 1 — Free run, continue SPI operation regardless of suspend or when the suspend occurred.
3−0	Reserved	Reads return zero; writes have no effect.

2.2 SPI Example Waveforms

Figure 2−13. CLOCK POLARITY = 0, CLOCK PHASE = 0 (All data transitions are during the rising edge, non-delayed clock. Inactive level is low.)

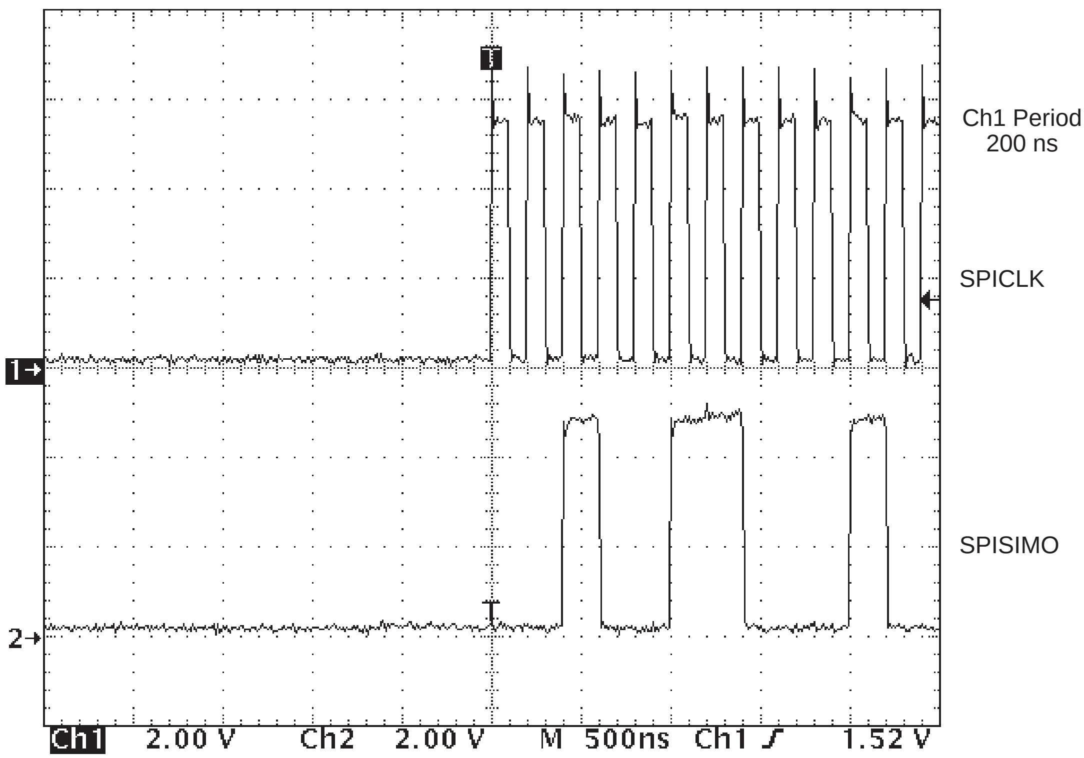


Figure 2−14. CLOCK POLARITY = 0, CLOCK PHASE = 1 (All data transitions are during the rising edge, but delayed by half clock cycle. Inactive level is low.)

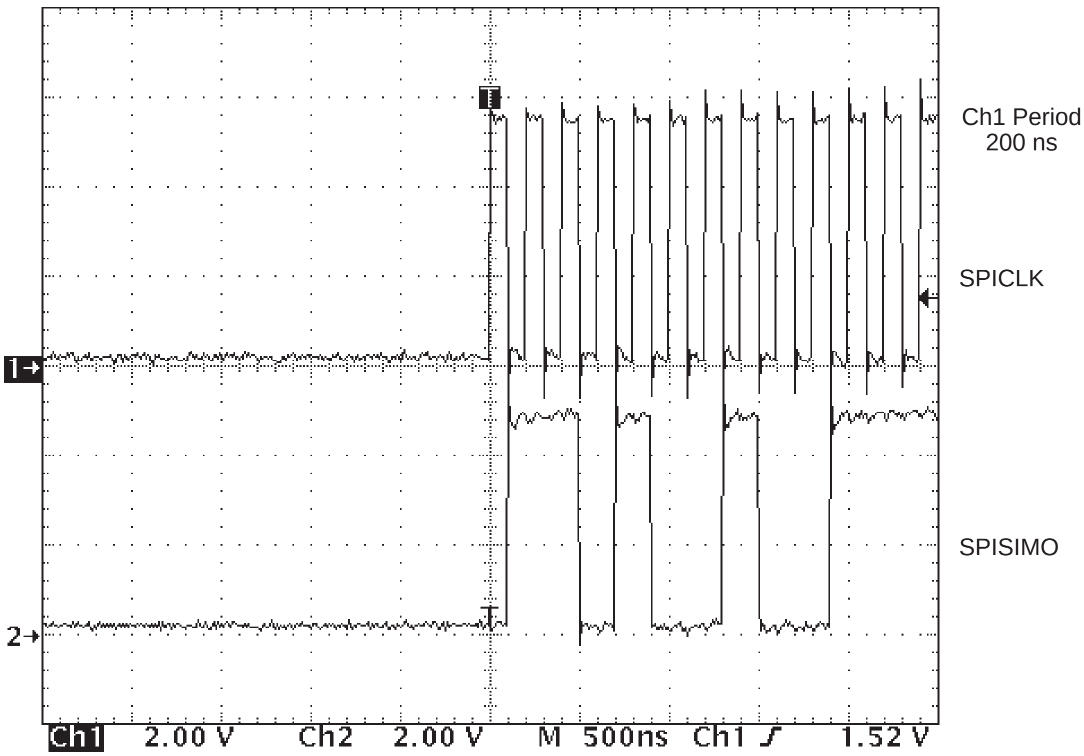


Figure 2−15. CLOCK POLARITY = 1, CLOCK PHASE = 0 (All data transitions are during the falling edge. Inactive level is high.)

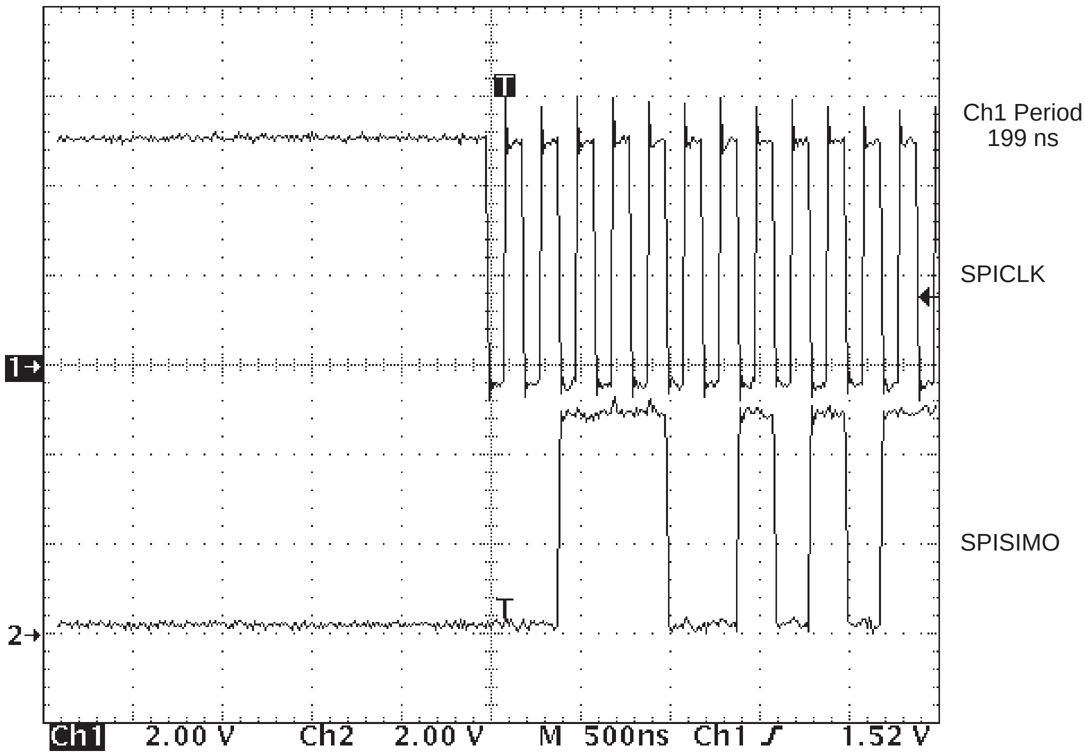


Figure 2−16. CLOCK POLARITY = 1, CLOCK PHASE = 1 (All data transitions are during the falling edge, but delayed by half clock cycle. Inactive level is high.)

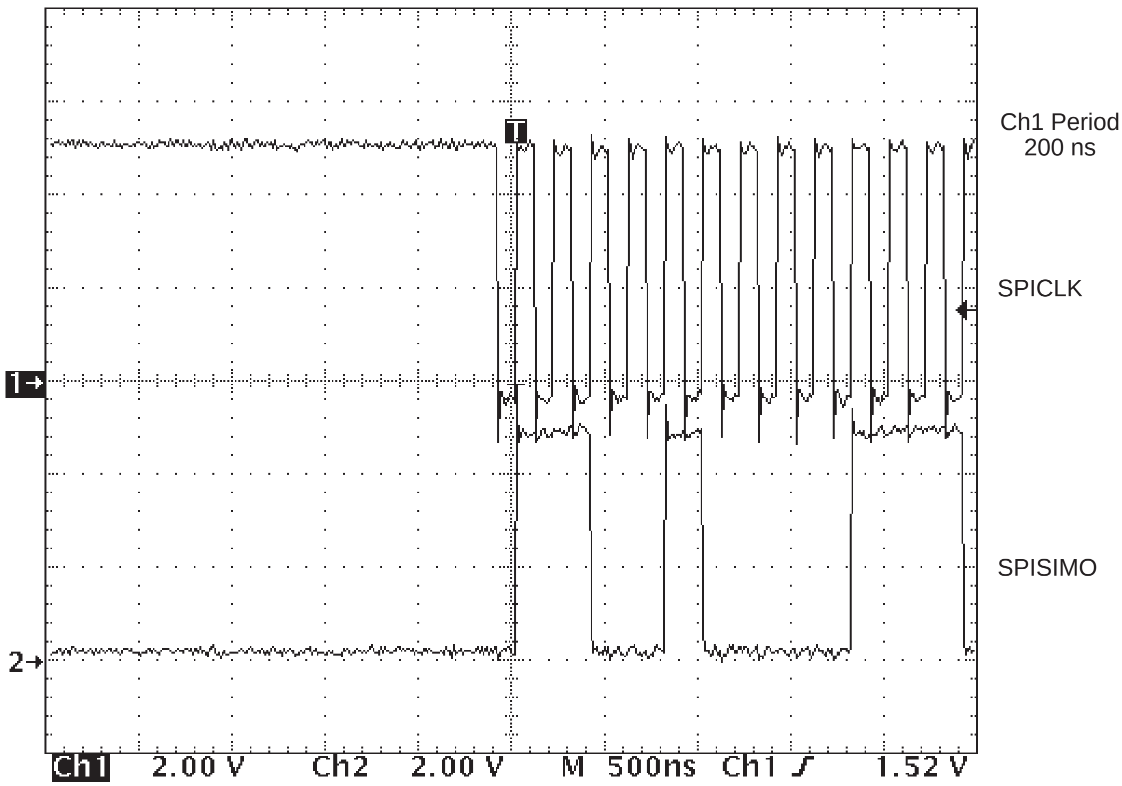


Figure 2−17. $\overline{\text{SPISTE}}$ Behavior in Master Mode (Master lowers $\overline{\text{SPISTE}}$ during the entire 16 bits of transmission.)

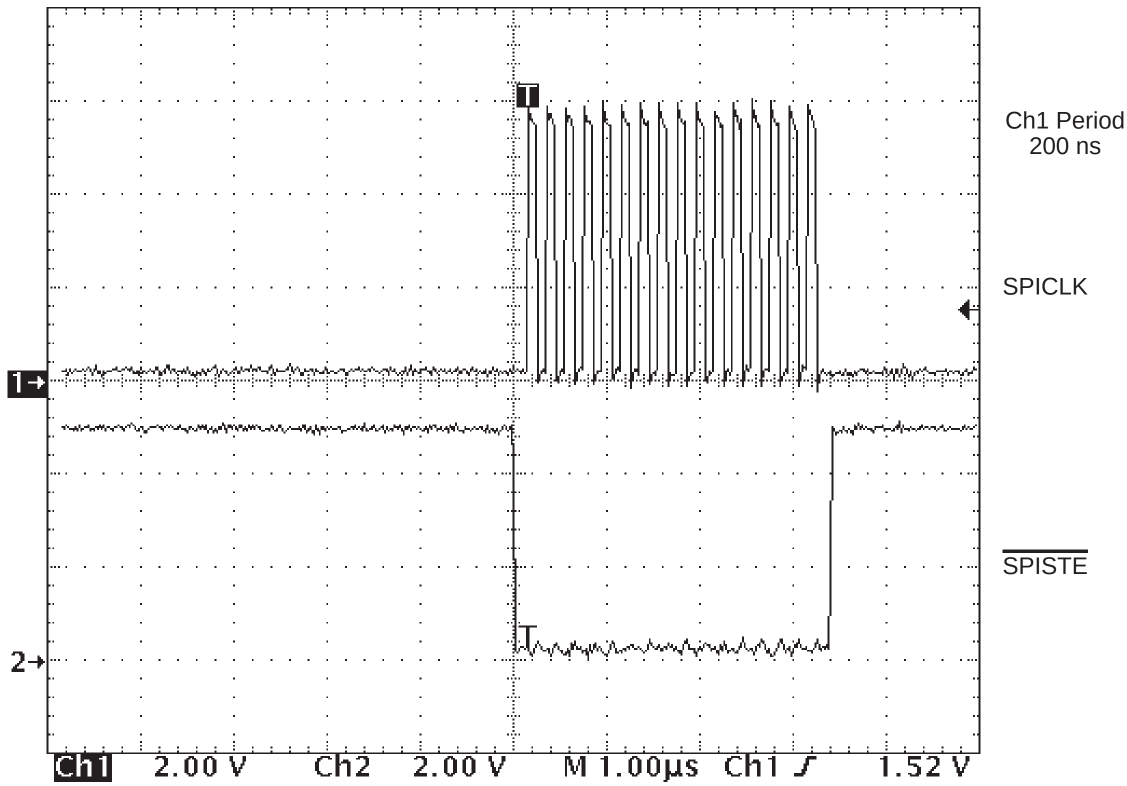
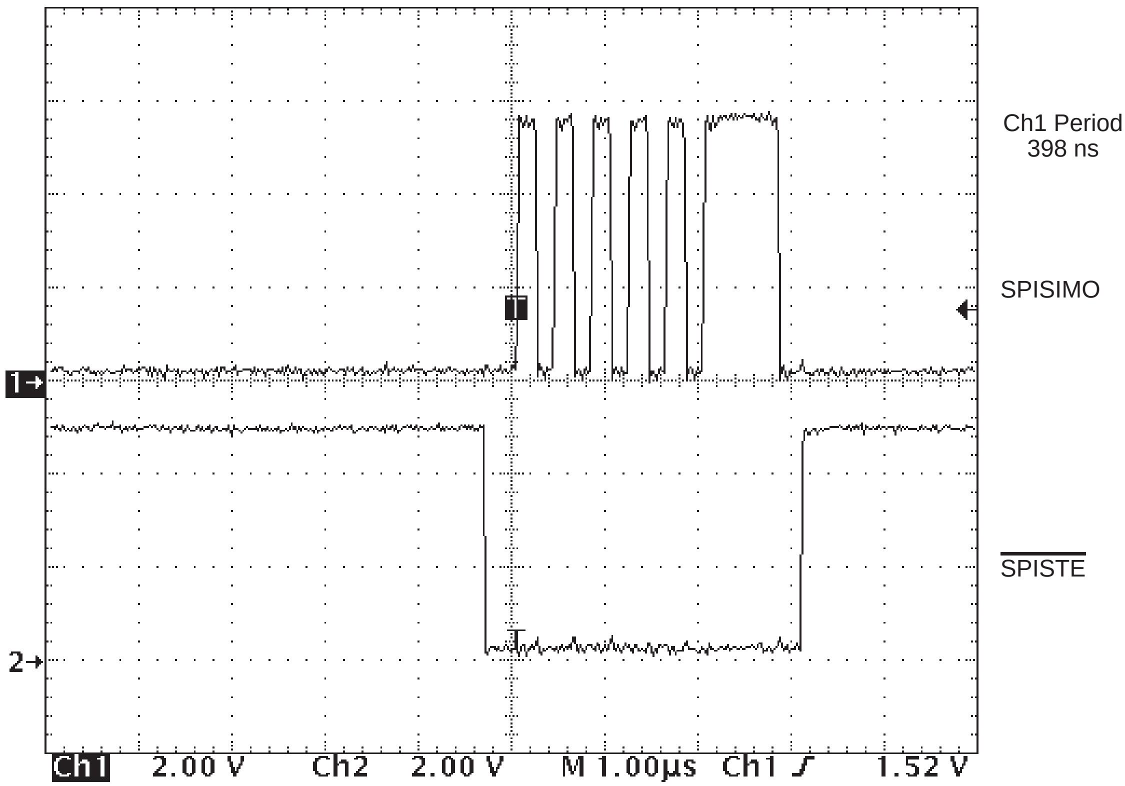


Figure 2−18. $\overline{\text{SPISTE}}$ Behavior in Slave Mode (Slave's $\overline{\text{SPISTE}}$ is lowered during the entire 16 bits of transmission.)



Appendix A

Revision History

This document was revised to SPRU059C from SPRU059B.

The scope of the revisions was limited to adding technical changes as described on the next page.

A.1 Changes Made in This Revision

The following changes were made in this revision:

Page	Additions/Modifications/Deletions
1-11	Deleted the bullet from Section 1.4.1 that listed bit 6 as initializing interrupts
2-15	Separated bits 7 and 6 to make bit 6 R/W instead of R only

Index

W