TMS470R1x Multi-Buffer Serial Peripheral Interface (MibSPI) Reference Guide

Literature Number: SPNU217B
October 2003

Texas Instruments

REVISION HISTORY

REVISION	DATE	NOTES
B	10/03	Register formats updated, Pages 30-101
A	9/02	Converted to stand-alone book
*	9/02	*Initial version*

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 application s 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 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.

Resale of TI products or services with statements different form 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.

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
		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

Contents

1 Overview 2
2 MibSPI Operation Modes 3
2.1 MibSPI Internal Registers 5
2.2 MibSPI Operation; Three-Pin Option 7
2.3 MibSPI Operation; Four-Pin Option 8
2.3.1 Four-Pin Option with SPISCS [7:0] 8
2.3.2 Four-Pin Option with SPIENA 9
2.4 MibSPI Operation; Five-Pin Option (Hardware Handshaking) 10
2.5 Data Format 12
2.5.1 Compatibility Mode with TMS470 SPI 12
2.5.2 Multi-buffer Mode (MibSPI) 12
2.6 Clocking Modes 14
2.7 Data Transfer Example 17
2.8 Multi-buffer RAM 18
2.8.1 Buffer Initialization 19
2.9 Multiple Chip Select (Master only) 19
2.10 Slave Mode in Multi-buffer Configuration 19
2.11 Internal Loop-Back Test Mode (Master only) 21
2.12 Transmission Continuous Self-test (Master only) 21
2.13 Variable Chip Select Setup and Hold Timing (Master only) 21
2.14 Lock Transmission 23
2.15 Hold Chip-Select Active (Master only) 23
2.16 Detection of Slave De-synchronization (Master only) 23
2.17 ENA Signal Time-out (Master only) 24
3 General Purpose I/O 25
4 Low Power Mode 26
5 Interrupts 27
5.1 Compatibility Mode 27
5.2 Multi-buffer Mode 27
6 DMA Interface 29
6.1 Compatibility Mode 29
6.2 Multi-buffer Mode 29
7 Control Registers and RAM 30
7.1 Control Registers 30
7.2 MibSPI RAM 37

7.3 MibSPI Control Register 1 (SPICTRL1) 39
7.4 MibSPI Control Register 2 (SPICTRL2) 41
7.5 MibSPI Control Register 3 (SPICTRL3) 44
7.6 MibSPI Shift Register 0 (SPIDAT0) 46
7.7 MibSPI Shift Register 1 (SPIDAT1) 47
7.8 MibSPI Buffer Register (SPIBUF) 49
7.9 MibSPI Emulation Register (SPIEMU) 53
7.10 MibSPI Pin Control Register 1 (SPIPC1) 54
7.11 MibSPI Pin Control Register 2 (SPIPC2) 56
7.12 MibSPI Pin Control Register 3 (SPIPC3) 57
7.13 MibSPI Pin Control Register 4 (SPIPC4) 59
7.14 MibSPI Pin Control Register 5 (SPIPC5) 61
7.15 MibSPI Pin Control Register 6 (SPIPC6) 63
7.16 SPI Control Register 4 (SPICTRL4) 65
7.17 SPI Control Register 5 (SPICTRL5) 66
7.18 SPI Control Register 6 (SPICTRL6) 70
7.19 SPI Control Register 7 (SPICTRL7) (SPICTRL8, SPICTRL9) 71
7.20 SPI Status Register (SPISTAT) 74
7.21 Transfer Group Interrupt Enable Register (TGINTENA) 77
7.22 Transfer Group Interrupt Level Register (TGINTLVL) 78
7.23 Transfer Group Interrupt Flag Register (TGINTFLAG) 79
7.24 Transfer Group Interrupt Vector Register 0 (TGINTVECT0) 81
7.25 Transfer Group Interrupt Vector Register 1 (TGINTVECT1) 83
7.26 Tick Count Register (TICKCNT) 85
7.27 Last Transfer Group End Pointer (LTGPEND) 87
7.28 MibSPI Transfer Group Control Register (TGxCTRL) 88
7.29 MibSPI DMA Channel Control Register (DMAxCTRL) 92
7.30 Multi-buffer RAM 95
7.30.1 Control Field 96
7.30.2 Transmit Field 98
7.30.3 Status Field 99
7.30.4 Receive Field 101

Figures

1 MibSPI Module Block Diagram. 4
2 MibSPI Three-Pin Option 7
3 MibSPI Four-Pin Option with SPISCS 9
4 MibSPI Four-Pin Option with SPIENA 10
5 MibSPI Five-Pin Option with SPIENA and SPISCS 11
6 Clock Mode with POLARITY = 0 and PHASE = 0 15
7 Clock Mode with POLARITY = 0 and PHASE = 1 15
8 Clock Mode with POLARITY = 1 and PHASE = 0 16
9 Clock Mode with POLARITY = 1 and PHASE = 1 16
10 Five Bits per Character (Five-Pin Option) 17
11 Example: Different Modes for a Multi-buffer RAM of 64 Buffers. 18
12 Multi-buffer in Slave Mode 20
13 Transfer Group Interrupt Structure 28
14 SPISTAT Interrupt Structure 28
15 Example: tC2TDELAY = 8 ICLK cycles 66
16 Example: tT2CDELAY = 4 ICLK cycles 67
17 Transmit-data-finished-to-ENA-inactive-time-out 68
18 Chip-select-active-to-ENA-signal-active-time-out. 69
19 Multi-buffer RAM Configuration 95

Tables

1 MibSPI Internal Registers Mode 5
2 Clocking Modes 14
3 MibSPI Registers 30
4 MibSPI RAM 37
5 Interrupt Vector for Interrupt Line INT0 82
6 Interrupt Vector for Interrupt Line INT1 84
7 Trigger Event Types 90
8 Trigger Sources 91
9 Buffer Mode Selection Bits BUFMODE[2:0] 96

Multi-Buffer Serial Peripheral Interface (MibSPI)

This reference guide provides the specifications for a 16-bit configurable synchronous multi-buffer serial peripheral interface (MibSPI). The MibSPI is, in effect, a programmable-length shift register used for high speed communication between external peripherals or other microcontrollers. Its multi-buffer allows multiple transmissions with different peripherals without any CPU action.

Topic **Page**

1 **Overview** 2

2 **MibSPI Operation Modes** 3

3 **General Purpose I/O** 25

4 **Low Power Mode** 26

5 **Interrupts** 27

6 **DMA Interface** 29

7 **Control Registers and RAM** 30

1 Overview

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

The MibSPI is available with three, four, or five pins to ensure full compatibility with the TMS470 SPI. The MibSPI also allows multiple programmable chip-selects, as well as a programmable multi-buffer array that enables programed transmission to be completed without CPU intervention. The buffers are combined in different *transfer group*s that can be triggered by external events (Timers, I/O, etc.) or by the internal tick counter. The internal tick counter can support periodic trigger events.

Each buffer can be associated with different DMA channels in different transfer groups, allowing the user to move data from/to internal memory to/ from external slave with a minimal CPU interaction.

The pins SPICLK, SPISIMO, and SPISOMI are used in all MibSPI pin modes. The pins SPIENA and SPISCS[7:0] are optional and may be used if the pins are present on a given device.

The MibSPI has the following attributes:

- ❏ 16-bit shift register
- ❏ Receive buffer register
- ❏ 8-bit baud clock generator
- ❏ Serial clock (SPICLK) I/O pin
- ❏ Slave in, master out (SPISIMO) I/O pin
- ❏ Slave out, master in (SPISOMI) I/O pin
- ❏ SPI enable (SPIENA) I/O pin (4-or 5-pin mode only)
- ❏ Slave chip select (SPISCS[7:0]) I/O pin (4- or 5-pin mode only)

The MibSPI allows software to program the following options:

- ❏ SPISOMI/SPISIMO pin direction configuration
- ❏ SPICLK pin source (external/internal)
- ❏ SPICLK frequency (interface clock [ICLK] /2 through /256)
- ❏ MibSPI pins as functional or digital I/O pins
- ❏ Character length (2 to 16 bits)
- ❏ Phase (delay/no delay), Polarity (high or low).

2 MibSPI Operation Modes

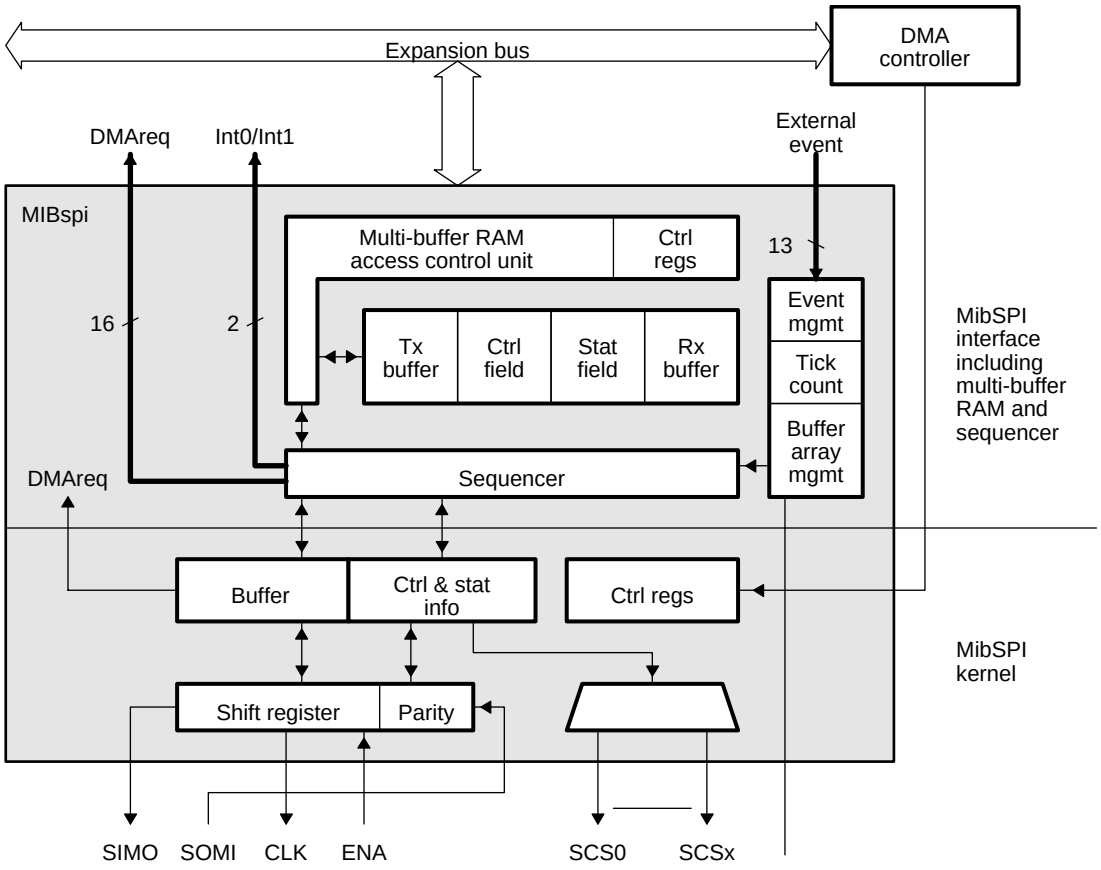
The MibSPI operates in master or slave mode. The MASTER bit (SPICTRL2.3) selects the configuration of the SPISIMO and SPISOMI pins and the CLKMOD bit (SPICTRL2.5) determines whether an internal or external clock source will be used.

The slave chip select ($\overline{\text{SPISCS}}$[7:0]) pins are used when communicating with multiple slave devices. When the master (MibSPI sending out the clock stream) writes to SPIDAT1, the $\overline{\text{SPISCS}}$ pins are automatically driven to select the slave connected to that signal. Writing to SPIDAT0 will not drive any $\overline{\text{SPISCS}}$ pins, thus allowing the master to communicate with all slave devices connected to the same SPI bus.

In addition, a handshaking mechanism, provided by the $\overline{\text{SPIENA}}$ pin, enables the slave to delay the generation of the clock signal supplied by the master as long as it is not prepared for the next exchange of data.

Figure 1 on page 4 shows the MibSPI module block diagram.

Figure 1. MibSPI Module Block Diagram



2.1 MibSPI Internal Registers

A general representation of the MibSPI internal registers is shown in Table 1. The page column provides a cross reference to additional information on the individual registers. For more information regarding individual bytes, see Table 3, on page 30.

Table 1. MibSPI Internal Registers Mode†‡

Offset Address†	Mnemonic	Name	Description	Page
0x00	SPICTRL1	SPI Control Register 1	Sets transfer rate and character length	39
0x04	SPICTRL2	SPI Control Register 2	Controls SPI clock	41
0x08	SPICTRL3	SPI Control Register 3	Controls system interface	44
0x0C	SPIDAT0	SPI Shift Register 0	Main shift register	46
0x10	SPIDAT1	SPI Shift Register 1	Shift register used in automatic slave chip select mode only	47
0x14	SPIBUF	SPI Buffer Register	Holds received word	49
0x18	SPIEMU	SPI Emulation Register	Mirror of SPIBUF. Read does not clear flags	53
0x1C	SPIPC1	SPI Pin Control Register 1	Controls the direction of data on the I/O pins	54
0x20	SPIPC2	SPI Pin Control Register 2	Reflects the values on the I/O pins	56
0x24	SPIPC3	SPI Pin Control Register 3	Controls the values sent to the I/O pins	57
0x28	SPIPC4	SPI Pin Control Register 4	Sets data values in the SPIPC3 register	59
0x2C	SPIPC5	SPI Pin Control Register 5	Clears values in the SPIPC3 register	61
0x30	SPIPC6	SPI Pin Control Register 6	Determines if pins will operate as general I/O or SPI functional pin.	63
040h	SPICTRL4	Control Register 4	Enables MibSPI mode	65
044h	SPICTRL5	Control Register 5	Sets CS mode, CS pre-/post-transfer delay time and ENA time-out	66

† The actual address of these registers is device specific and CPU specific. See the specific device data sheet to verify the MibSPI register addresses.

‡ The shaded registers are only accessible in MibSPI mode, registers from 0x00 to 0x30 are compatible with the TMS470 SPI module (SPNU195)

Table 1. MibSPI Internal Registers Mode†‡ (Continued)

Offset Address†	Mnemonic	Name	Description	Page
048h	SPICTRL6	Control Register 6	In CS decoded mode only: sets high low/ active CS signals	70
04Ch	SPICTRL7	Control Register 7	Configuration of 2nd data word format	71
050h	SPICTRL8	Control Register 8	Configuration of 3rd data word format	71
054h	SPICTRL9	Control Register 9	Configuration of 4th data word format	71
058h	SPISTAT	Status Register	Enables error interrupts and indicates the actual error status	74
05Ch	TGIN-TENA	TG Interrupt Enable	Transfer group interrupt enable bits for “transfer finished” event and “transfer suspended” event	77
060h	TGINTLVL	TG Interrupt Level	Transfer group interrupt line select (INT0 or INT1)	78
064h	TGINT-FLAG	TG Interrupt Flag	Transfer group interrupt flags for “transfer finished” event and “transfer suspended” event	79
068h	TGINTVE CT0	TG Interrupt Vector	Transfer group interrupt vector for line INT0	81
06Ch	TGINTVE CT1	TG Interrupt Vector	Transfer group interrupt vector for line INT1	83
070h	TICKICNT	Initial Tick Count Value	Initial tick count value defines periodic trigger to start group transfers	85
074h	LTGPEND	MibSPI Last Transfer Group End Address	Defines the end address for the last transfer group	87
078h 07Ch ... 0B4h	TGxCTRL\ (x=0...15)	MibSPI Transfer Group Control	Up to 16 identical transfer group control register. Each comprises TGENA, ONE-SHOT, PRST, TRIGEVT, TRIGSRC, PCURRENT, PSTART	88
0B8h 0BCh ... 0D4h	DMAxC-TRL (x=0...7)	DMA Channel Control Register	DMA channel control register comprises buffer ID, ONESHOT, RX/TXDMACH, RX/ TXDMAENA, NOBRK, ICOUNT and COUNT	92

† The actual address of these registers is device specific and CPU specific. See the specific device data sheet to verify the MibSPI register addresses.

‡ The shaded registers are only accessible in MibSPI mode, registers from 0x00 to 0x30 are compatible with the TMS470 SPI module (SPNU195).

Table 1. MibSPI Internal Registers Mode†‡ (Continued)

Offset Address†	Mnemonic	Name	Description	Page
0D8h... 0FFh	Reserved		Reserved	
base1 + 000h... 1FFh	CTRL+TX buffers	Multi-buffer RAM Read/ Write Addresses	Transmit and control RAM	96
200h 3FFh	STAT+RX buffers	Multi-buffer Read-only Addresses	Receive and status RAM	99

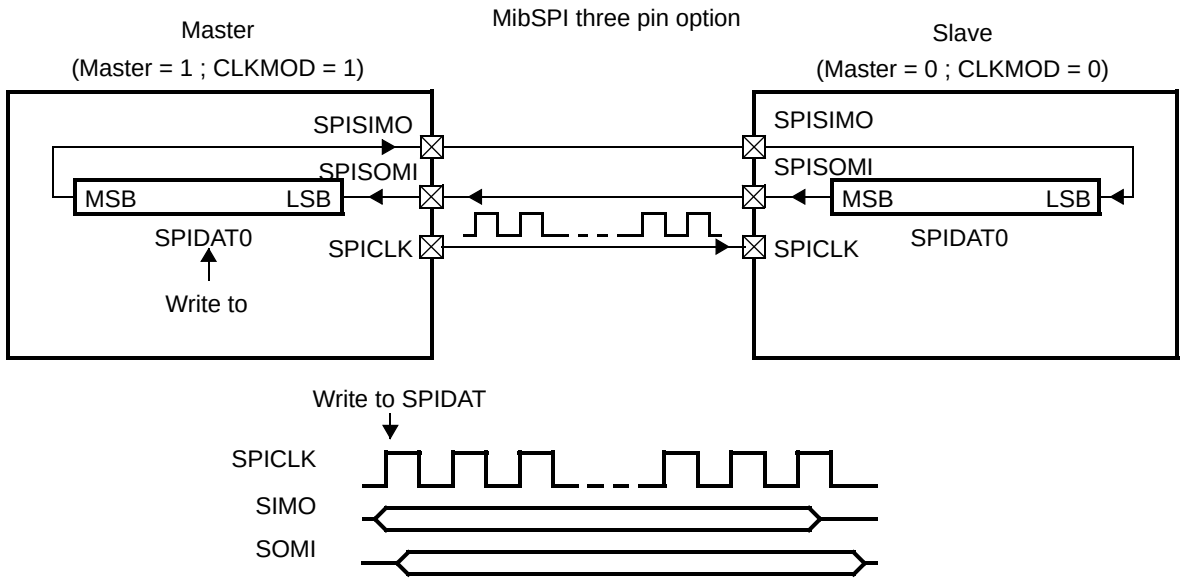
† The actual address of these registers is device specific and CPU specific. See the specific device data sheet to verify the MibSPI register addresses.

‡ The shaded registers are only accessible in MibSPI mode, registers from 0x00 to 0x30 are compatible with the TMS470 SPI module (SPNU195)

2.2 MibSPI Operation; Three-Pin Option

In master mode configuration (MASTER=1 (SPICTRL2.3) and CLKMOD=1 (SPICTRL2.5)), the MibSPI provides the serial clock on the SPICLK pin for the entire serial communications network. Data is output on the SPISIMO pin and latched in from the SPISOMI pin (see Figure 2).

Figure 2. MibSPI Three-Pin Option



Data written to the shift register (SPIDAT0) initiates data transmission on the SPISIMO pin, most significant bit (MSB) first. Simultaneously, received data is shifted through the SPISOMI pin into the least significant bit (LSB) of the

SPIDAT0 register. When the selected number of bits has been transmitted, the data is transferred to the SPIBUF register for the CPU to read. Data is stored right-justified in SPIBUF.

When the specified number of bits has been shifted through the SPIDAT0 register, the following events occur:

- ❏ The RXINTFLAG bit (SPICTRL3.0) is set to 1
- ❏ The SPIDAT0 register contents transfer to the SPIBUF register
- ❏ An interrupt is asserted if the RXINTEN bit (SPICTRL3.1) is set to 1

In slave mode configuration (MASTER = 0 and CLKMOD = 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.

Data written to the SPIDAT0 register is transmitted to the network when the SPICLK signal is received from the network master. To receive data, the MibSPI waits for the network master to send the SPICLK signal and then shifts data on the SPISIMO pin into the SPIDAT0 register. If data is to be transmitted by the slave simultaneously, it must be written to the SPIDAT0 register before the beginning of the SPICLK signal.

When the MIPSPI mode is enabled, the three-pin option works by setting all the SPISCS pins in GPIO mode. The chip-select field in the buffers becomes meaningless.

2.3 MibSPI Operation; Four-Pin Option

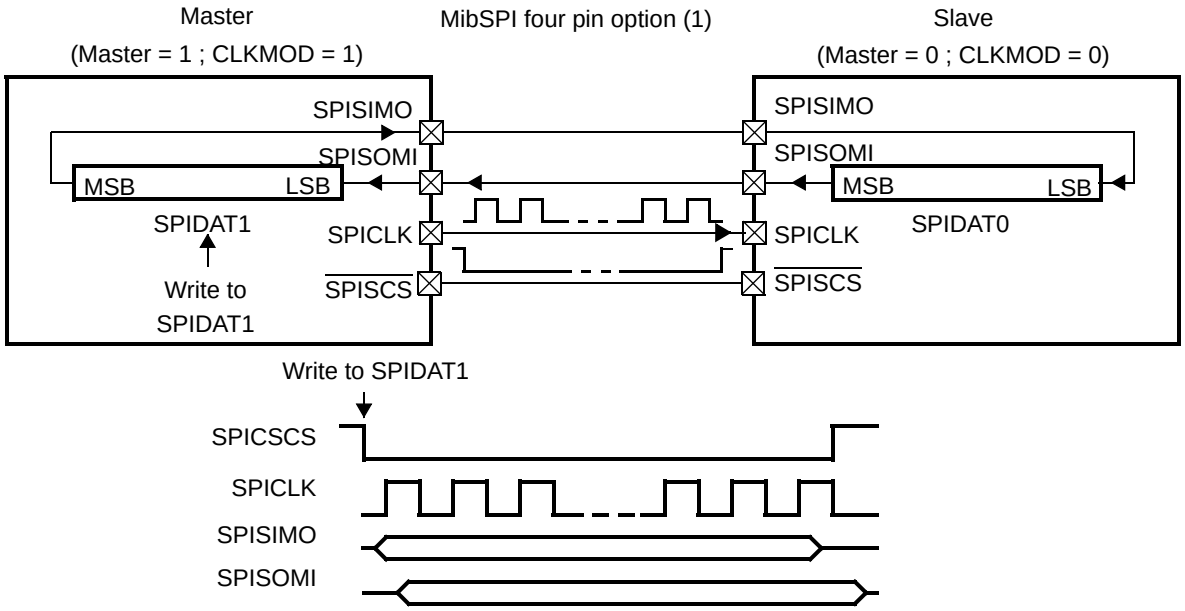
The three-pin option and the four-pin options of the MibSPI are identical in the master mode (CLKMOD = 1), except that the four-pin option uses either SPIENA or SPISCS [7:0] pins. The I/O direction of these pins is determined by the CLKMOD control bit as MibSPI not general purpose I/O.

2.3.1 Four-Pin Option with SPISCS [7:0]

Compatibility mode

To use the SPISCS [0] as an automatic chip select pin, the SPISCS [0] pin must be configured to be functional (SPIPC6.4 = 1). In this mode, the master will drive this signal low when data has been written to SPIDAT1 and then drive the pin high again after a character transmission has completed. If data is written to SPIDAT0, SPISCS [0] remains high (see Figure 3).

Figure 3. MibSPI Four-Pin Option with $\overline{SPISCS}$



To use the $\overline{SPISCS}$ [0] as a chip select, the slave $\overline{SPISCS}$ [0] pin must be configured as MibSPI functional (SPIPC6.4 = 1). In this mode, an active low signal on the $\overline{SPISCS}$ [0] pin will allow the slave SPI to transfer data to the serial data line. An inactive high signal will put the slave SPI's serial output pin in a high-impedance state. Therefore many slave devices can be tied together on the network, but only one slave at a time is allowed to talk. While the slave is not selected, no shifting or interrupts will occur.

MibSPI mode

In MibSPI mode, the $\overline{SPISCS}$ [7:0] pins that are going to be used, must be configured as functional (SPIPC6.[11:4]). The default pattern to be put on the SPISCS [7:0] when all the slave are deactivate is set in the SPICRTL6 register. This pattern allows a different slave with different chip-select polarity to be activated by the MibSPI.

During transmission, the CSNR field of the SPIDAT1 register or of the current buffer is applied on the pins, this pattern will select the slave to which the transmission is dedicated. (see section 2.9)

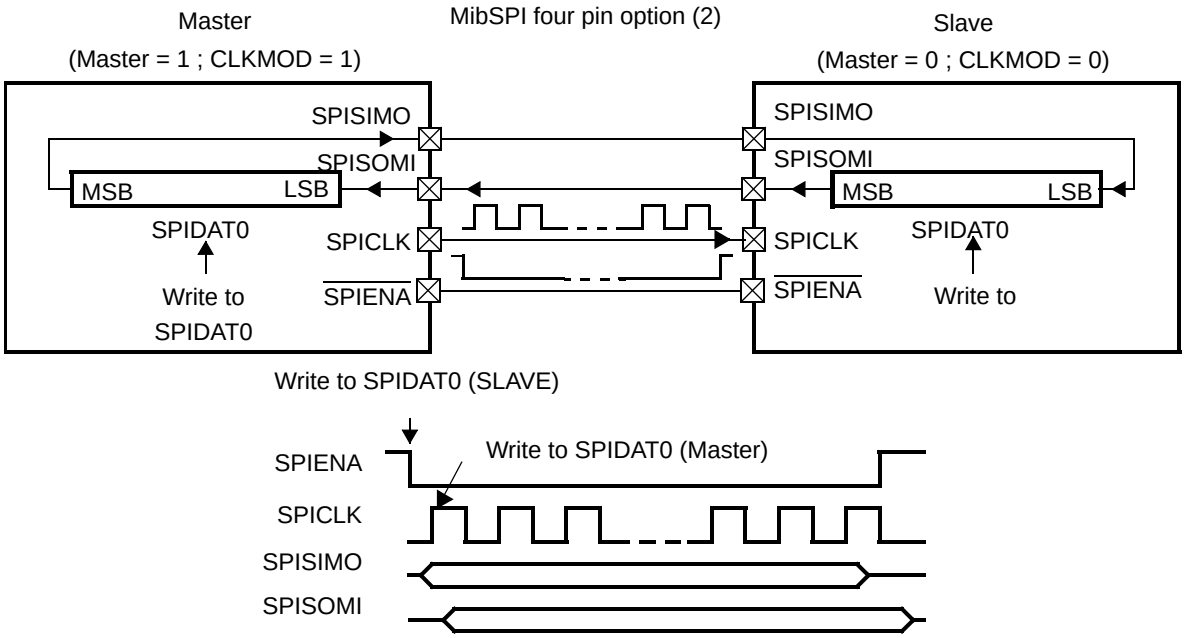
2.3.2 Four-Pin Option with $\overline{SPIENA}$

To use the $\overline{SPIENA}$ as a WAIT signal pin, the $\overline{SPIENA}$ pin must be configured to be functional (SPIPC6.0 = 1). In this mode, an active low signal on the

SPIENA pin will allow the master SPI to drive the clock pulse stream; otherwise, the master will hold the clock signal.

To use the SPIENA as a WAIT signal pin, the slave SPIENA pin must be configured as functional (SPIPC6.0 = 1). If the SPIENA pin is in high-z mode (ENABLE_HIGHZ = 1), the slave will put SPIENA into the high-impedance once it receives a new character. If the SPIENA pin is in push-pull mode (ENABLE_HIGHZ = 0), the slave will drive SPIENA high once it receives a new character. The slave will drive SPIENA low again after new data is written to the slave shift register (SPIDAT0).

Figure 4. MibSPI Four-Pin Option with SPIENA



2.4 MibSPI Operation; Five-Pin Option (Hardware Handshaking)

To use the hardware handshaking mechanism, both the SPIENA pin and SPISCS [7:0] pin must be configured as functional pins.

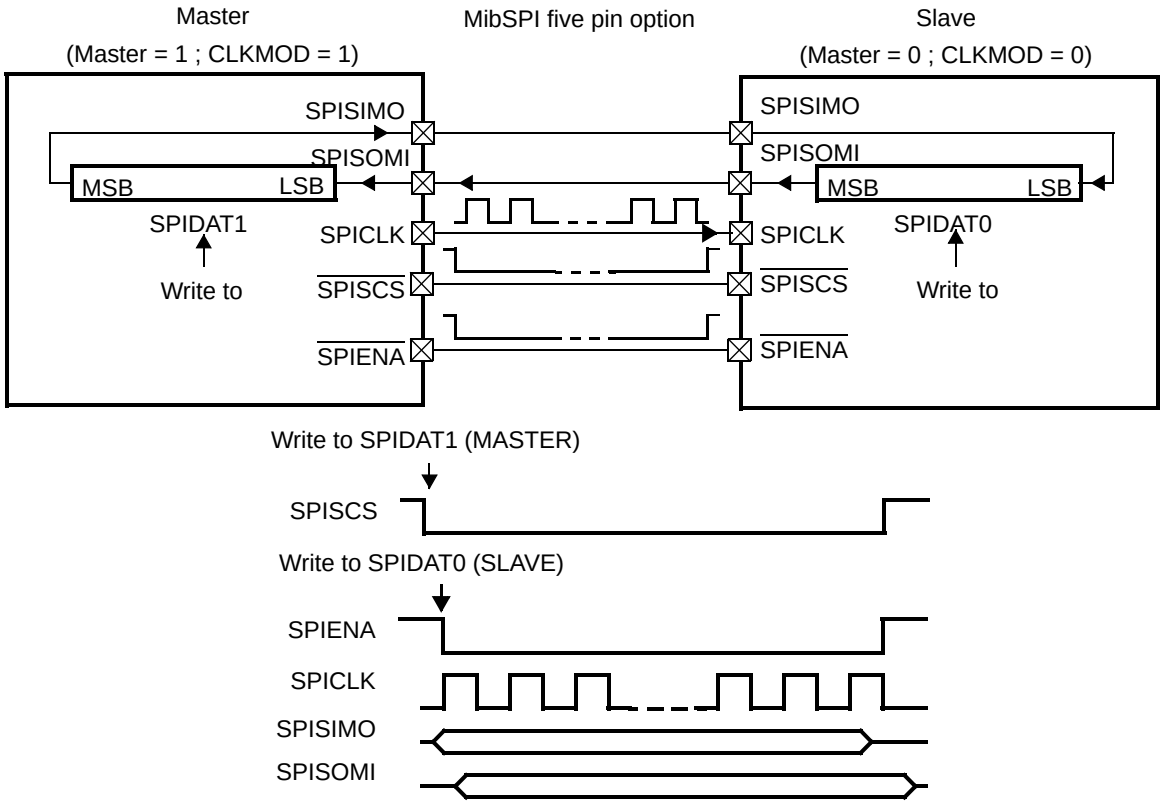
Compatibility mode

In the master SPI (CLKMOD = 1), the SPIENA pin is configured as a functional input. If configured as a slave SPI, the SPIENA pin is configured as a functional output. If the SPIENA pin is in high-z mode (ENABLE_HIGHZ = 1), the slave SPI will put this signal into the high-impedance state if it receives a new character from the master or if the slave becomes de-selected by the master (SPISCS goes high). The slave will drive the signal SPIENA low when

new data is written to the slave shift register (SPIDAT0) and the slave has been selected by the master (SPISCS is low).

If the SPIENA pin is in push-pull mode (ENABLE_HIGHZ = 0), the slave will drive SPIENA high only if there is new data in the buffer register and the slave is selected by the master (SPISCS is low). The slave SPI will drive the SPIENA signal low when new data is written to the slave shift register (SPIDAT0) and the slave is selected by the master (SPISCS is low). If the slave is de-selected by the master (SPISCS goes high), the slave SPIENA signal is driven low, allowing the master SPI to communicate with other slave SPIs.

Figure 5. MibSPI Five-Pin Option with SPIENA and SPISCS



In the master SPI (CLKMOD = 1), the SPISCS pin is configured as a functional output. If configured as a slave SPI (CLKMOD = 0), the SPISCS pin is configured as a functional input. A write to the master's SPIDAT1 shift register will automatically drive the SPISCS signal low. The master will drive the SPISCS signal high again after transmitting the new character. If the new data is written to the master's SPIDAT0 shift register, the SPISCS signal will NOT be driven low.

2.5 Data Format

2.5.1 Compatibility Mode with TMS470 SPI

In compatibility mode, the data formats for the three-, four- and five-pin options are identical.

CHARLEN[4:0] (SPICTRL1.4-0) specifies the number of bits (one to 16) in the data word. The CHARLEN[4:0] value directs the state control logic to count the number of bits received or transmitted to determine when a complete word is processed.

Data word length **must** be programmed in **master mode** and in **slave mode**.

The following conditions apply for words with fewer than 16 bits:

- ❏ Data must be left-justified when it is written to the MibSPI for transmission
- ❏ Data is right-justified when read back from the receive register

The buffer contains the most recently received word, right-justified, plus any bits that are left over from previous transmissions that have been shifted to the left. The diagram below shows how a 14-bit word is stored in the buffer once it is received.

Bits	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

In transmit mode, the SPIBUF register contains the most recently transmitted word, left-justified. The diagram below shows how a 14-bit word needs to be written to the buffer in order to be transmitted correctly.

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

To allow for the efficient transmission of byte-sized words, if a character length is programmed for 8 bits or less, the SDPDAT[7] bit instead of SDPDAT[15] is the source of the serial out data. This prevents the need to further add eight justification bits.

2.5.2 Multi-buffer Mode (MibSPI)

To support different types of slaves in one MibSPI network, four independent data word formats are implemented that allow configuration of individual data word length, polarity, phase and bit rate. For each buffer, the user can select which data format to use via the bits DFSEL[1:0] in the control field for one of the four data word formats.

Data word format 0 is identical to the data word format available in the standard SPI.

Data formats 1, 2 and 3 can be configured through new control registers described in Section 7.19.

Each MibSPI data format includes the standard SPI data format with enhanced features:

- For each of the four data word formats, the shift direction can be configured individually. It can be specified as most significant bit first or least significant bit first, where the position of the most significant bit depends on the configured data word length.
- MibSPI supports automatic right-alignment of receive data independent from shift direction and data word length. In addition, the transmit data does not need to be pre-aligned by the host. If it is written right-aligned into the MibSPI, the internal shift register will sort out the correct transfer according to selected shift direction and data word length.
- To increase fault detection of data transmission and reception, an odd or even parity bit can be enabled to be transmitted at the end of a data word. The parity generator can be enabled or disabled individually for each buffer. If a received parity bit does not match with the locally calculated parity bit, the parity error flag (PARITYERR) is set and an interrupt is asserted if enabled.
- Since the MibSPI is able to have two consecutive accesses to the same slave, a delay counter has been implemented to satisfy the delay time for data to be refreshed in the accessed slave. This delay counter is a programmable 6-bit counter and is loaded with the data format. It is enabled individually within each buffer.

Note:

These enhanced features are not accessible with the data format zero when the MibSPI is operating in compatibility mode.

2.6 Clocking Modes

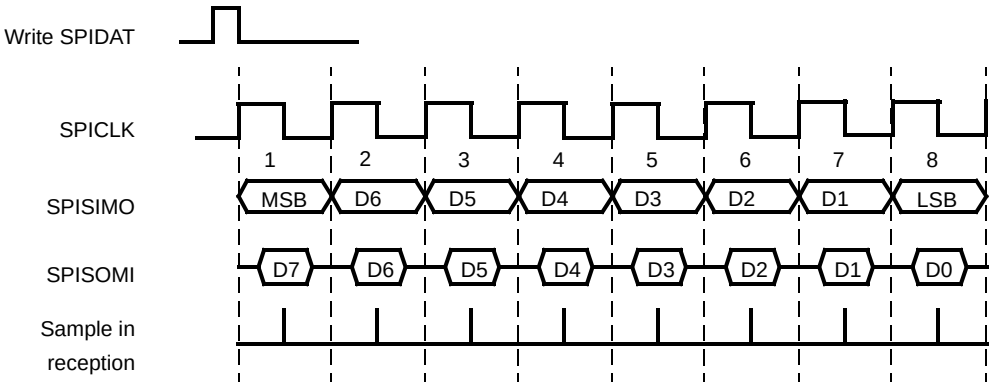
There are four clock modes in which SPICLK may operate, depending on the choice of the phase (delay/no delay) and the polarity (rising edge/falling edge) of the clock. When operating with PHASE active, the MibSPI makes the first bit of data available after the SPIDAT0 register is written and before the first edge of SPICLK. The data input and output edges depend on the values of both POLARITY and PHASE as shown in Table 2.

Table 2. Clocking Modes

POLARITY	PHASE	ACTION
0	0	Data is output on the rising edge of SPICLK. Input data is latched on the falling edge.
0	1	Data is output one half-cycle before the first rising edge of SPICLK and on subsequent falling edges. Input data is latched on the rising edge of SPICLK.
1	0	Data is output on the falling edge of SPICLK. Input data is latched on the rising edge.
1	1	Data is output one half-cycle before the first falling edge of SPICLK and on subsequent rising edges. Input data is latched on the falling edge of SPICLK.

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

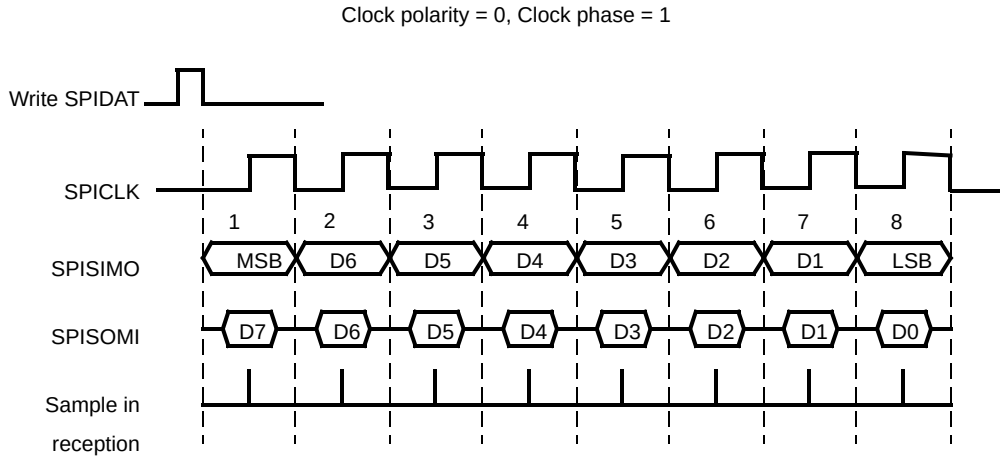
Figure 6. Clock Mode with POLARITY = 0 and PHASE = 0



Clock phase = 0 (SPICLK without delay)

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

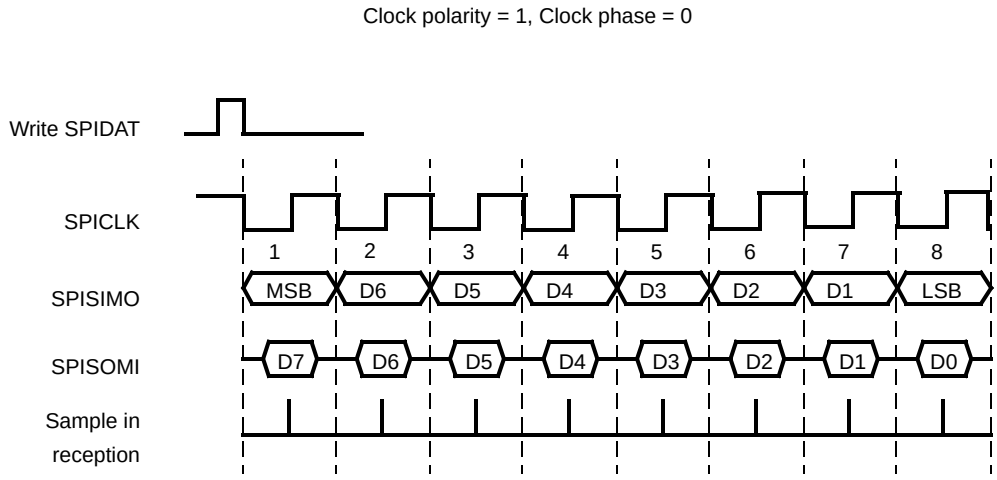
Figure 7. Clock Mode with POLARITY = 0 and PHASE = 1



Clock phase = 1 (SPICLK with delay)

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

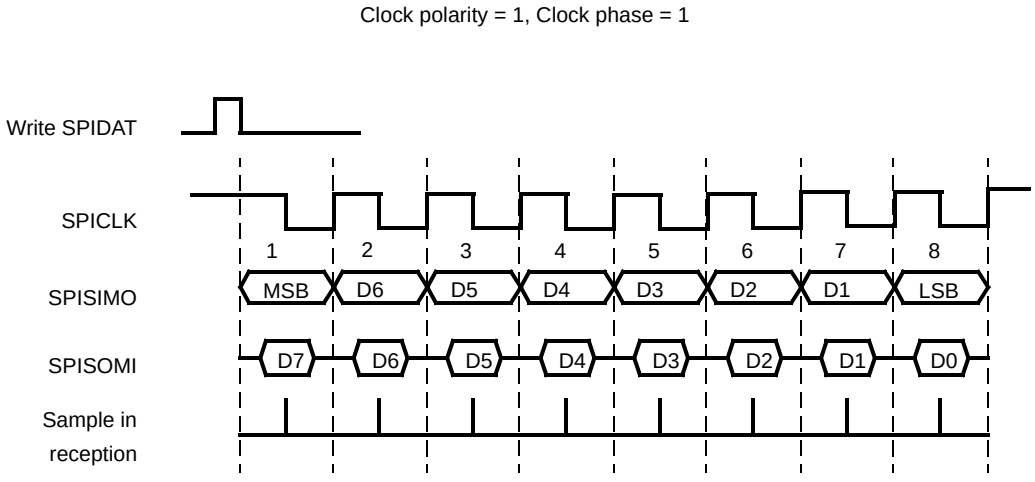
Figure 8. Clock Mode with POLARITY = 1 and PHASE = 0



Clock phase = 0 (SPICLK without delay)

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

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



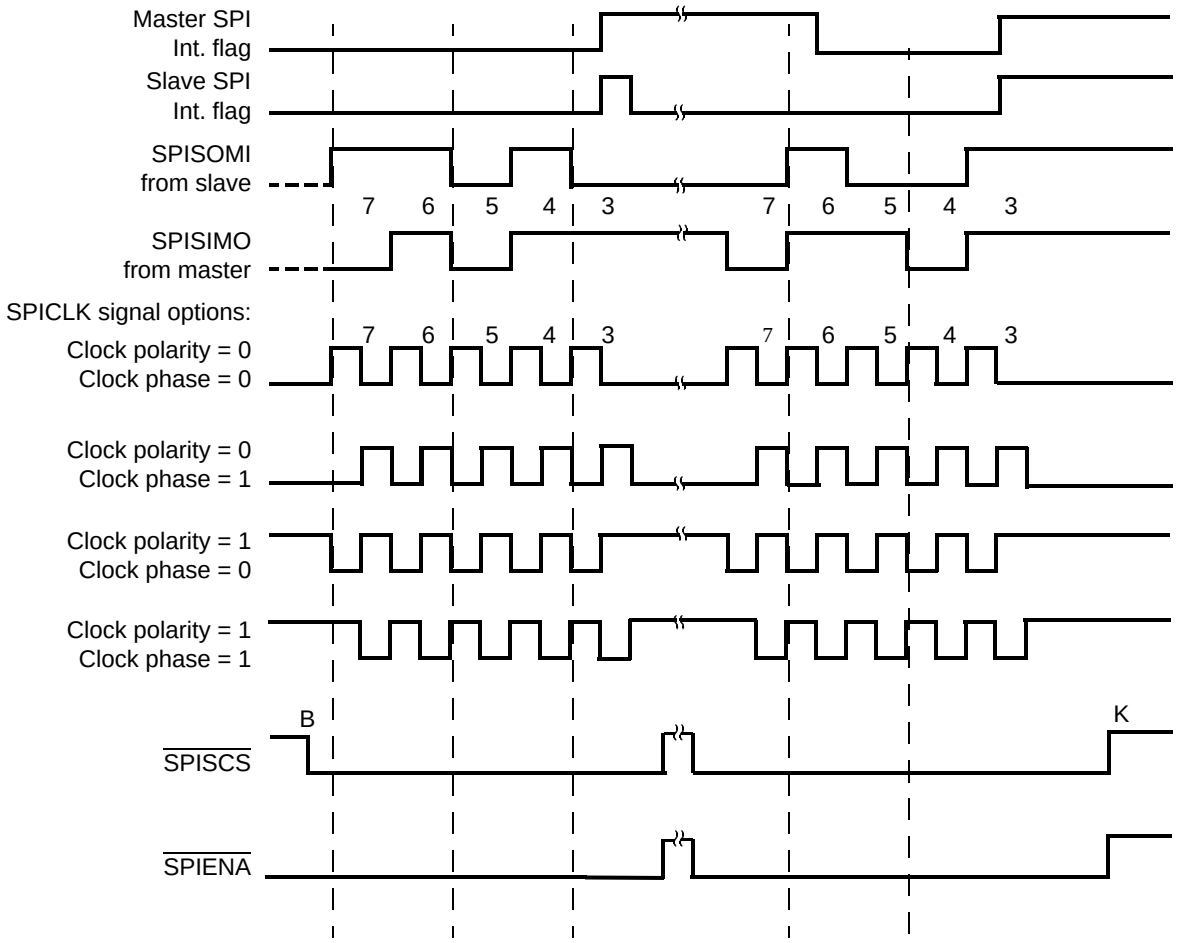
Clock phase 1 (SPICLK with delay)

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

2.7 Data Transfer Example

The following timing diagram illustrates an MibSPI data transfer between two devices using a character length of five bits.

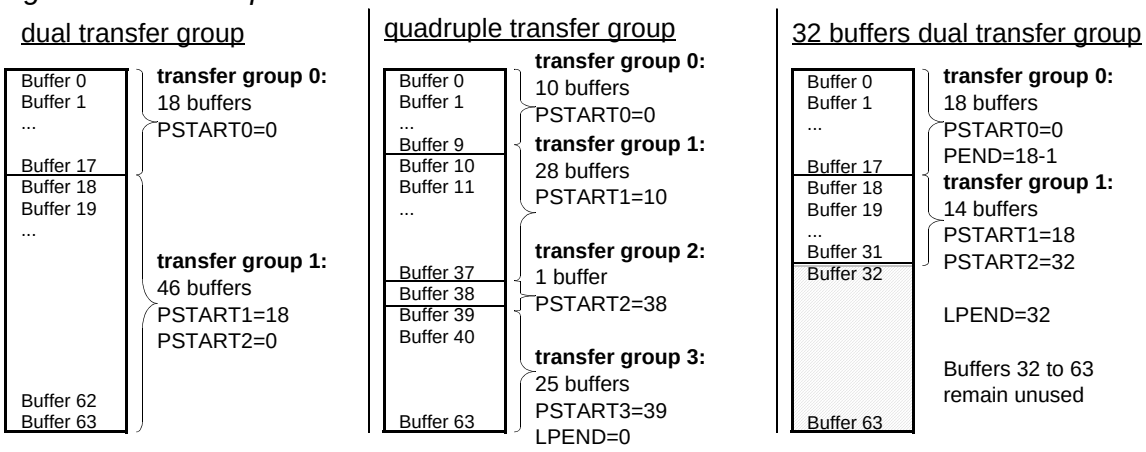
Figure 10. Five Bits per Character (Five-Pin Option)



2.8 Multi-buffer RAM

The size of the multi-buffer RAM depends on the implementation. It is comprised of 0 to 128 buffers, where 0 buffers represents the special case of no multi-buffer RAM. Each entry in the multi-buffer RAM consists of four parts: a 16-bit transmit field, a 16-bit receive field, a 16-bit control field and a 16-bit status field.The multi-buffer RAM can be partitioned into multiple transfer groups with a variable number of buffers each. In Figure 11 three examples are shown for a multi-buffer RAM with 64 buffers and four transfer groups. The first example shows two transfer groups used to partition the 64 buffers. In the second example all buffers are partitioned four times and in the third example only the first 32 buffers are utilized.

Figure 11. Example: Different Modes for a Multi-buffer RAM of 64 Buffers



Each of the transfer groups can be configured individually. For each of the transfer groups, a trigger event and a trigger source can be chosen. A trigger event can be, for example, a rising edge or a permanent low level at a selectable trigger source. A last trigger type is “always”, which means either continuous mode or single mode depending on the ONESHOT control bit. Up to 15 trigger sources are available, which can be utilized by each transfer group. One of these trigger sources is called tick counter. This tick counter is implemented in the MibSPI and generates periodic trigger events. Other trigger sources can be MibSPI external signals coming from another peripheral module like the High-End Timer (HET) or a general-purpose input pin (GPIO).

An interrupt can be generated upon finishing a group transfer or when an ongoing group transfer is suspended due to “suspend to wait” state of one of the buffers inside the transfer group. This suspend interrupt enables the host to quickly provide new transmit data or to quickly consume received data from the suspended buffer.

The trigger sources have to be defined individually for each implementation of MibSPI into a TMS470 derivative. External trigger sources might by a HET I/O channel or a GIO pin re-used as trigger input.

The number of transfer groups must be defined individually for each MibSPI macro cell. Up to 16 transfer groups are supported.

2.8.1 Buffer Initialization

After reset, the buffer RAM is initialized. This process takes as many cycles as the number of buffers contained in the RAM, that is, a 128 buffer RAM will take 128 cycles to initialize. During this process, the CPU is held in reset by the MibSPI.

2.9 Multiple Chip Select (Master only)

The MibSPI supports multi chip-select (multiCS) modes. The MibSPI is able to support encoded chip-select and decoded chip-select.

The MibSPI could connect up to eight individual slave devices that use decoded chip-select by routing one wire to each slave. For the decoded “mode”, the eight chip-selects in the control field are directly connected to the eight pins. The default value of each chip-select can be configured via the register CSDEF. During the transmission, the value of the chip-select control field (CSNR[7:0]) of the SPIDAT1 register (SPIDAT1[23:16]) is applied to the pins. When the transmission finishes, the default register value (CSDEF) is applied to the chip-select pins.

To connect the MibSPI with encoded slave devices, the CSNR field allows multiple active bits at a time. The user could apply any value from 0 to 255 to provide a binary-encoded chip-select signal via the eight chip-select lines. All eight chip-select lines must be connected to each slave device and each slave needs to have unique address decoding. The CSDEF register provides the address at which slaves devices are all de-selected.

Using only on part of the chip select line as encoded, users could use the remaining chip select as decoded lines. Since the control field is 8 bit wide, the encoding of some chip select could be done easily, the remaining decoded lines just have to be set once at a time for only one active slave.

2.10 Slave Mode in Multi-buffer Configuration

When operating in slave mode, the MibSPI uses the chip-select pins 0 to 3 to generate a trigger to the corresponding Transfer Group Setting “0000” on the chip-select pins triggers Transfer Group 0, “0001” triggers TG1. When the value “1111” is set to the chip-select, the MibSPI is deselected - that is Transfer Group 15 is not available in slave mode. Chip-select pins 4 to 7 should be kept in GPIO mode. In slave mode, the fields TRIGSRC and

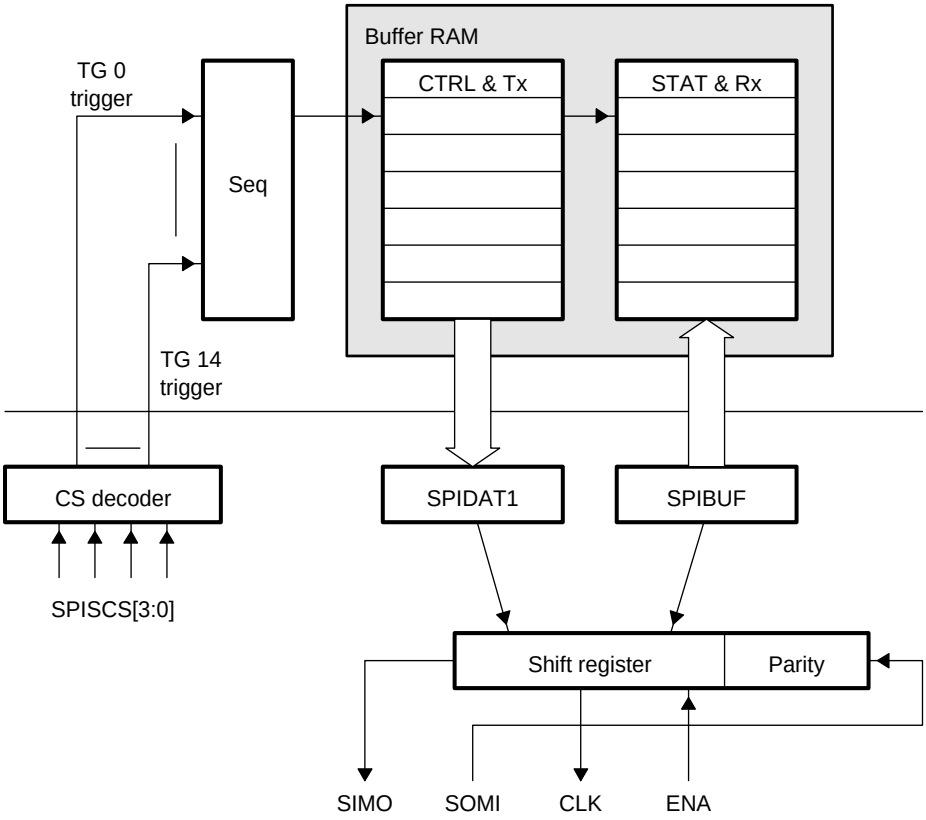
TRIGEVT are not taken into account by the sequencer, so only the SPISCS pins could trigger a Transfer Group. The chip-select trigger operates as a level sensitive trigger. The PRST and ONESHOT field should be set to 0.

If the corresponding Transfer Group is enabled, the multi-buffer reads the current buffer of the TG and writes it into SPIDAT1. If the Transfer Group is disabled the multi-buffer does not update the SPIDAT1 register.

Note:

If the Transfer Group is disabled and no update of the SPIDAT1 register has been done, the data to be transferred is meaningless.

Figure 12. Multi-buffer in Slave Mode



When the SPIDAT1 register is updated, the enable signal is released, and the transaction could begin. If the enable signal is not used, the master should wait for six ICLK cycles before sending the clock to begin the transaction. This time allows the MibSPI to update the SPIDAT1 register.

Once the transaction is finished, the MibSPI writes back the content of the shift-register into the Rx buffer and updates the status field.

Note:

If all the Transfer Groups are not needed, the number of SPISCS could be reduced to 3, 2 or 1 using the SPIPC6 register. In these cases, the maximum number of Transfer Groups accessible are 7, 3 and 1, respectively. The pins that are set in GPIO mode are no longer decoded.

When using only one SPISCS pin, only the Transfer Group 0 is triggerable.

Note: Maximum Input Frequency

The maximum input frequency on the SPICLK pin when in slave mode is the ICLK frequency divided by two.

2.11 Internal Loop-Back Test Mode (Master only)

The internal loop-back self-test mode can be utilized to test the MibSPI transmit path and receive path, including the buffers and the parity generator. In this mode, the transmit signal is internally fed back to the receiver, whereas the SIMO, SOMI and CLK pin are disconnected. 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.

Note:

This mode could not be set during transmission.

2.12 Transmission Continuous Self-test (Master only)

During data transfer, MibSPI is comparing at the sample point (half the SPI clock after the transmit point) its own internal transmit data with its transmit data on the bus. If the data on the bus does not match with the expected value, the bit error (BITERR) flag is set and an interrupt is asserted if enabled.

Note:

The compare is made from the output pin by the way of the input buffer.

2.13 Variable Chip Select Setup and Hold Timing (Master only)

In order to support slow slave devices, a 5-bit delay counter can be configured to delay the data transmission after the chip select is activated. A second 5-bit delay counter can be configured to delay the chip select deactivation after the last data bit transfer. Both delay counters are clocked with ICLK (see section 7.17).

Note:

If the CSHOLD bit is set within the control field, the current hold time and the following set-up time will not be applied in between transactions.

2.14 Lock Transmission

In order to be accessed, some slave device require, a command followed by data. In this case, the SPI transaction should not be interrupted by another group transfer. The LOCK bit within each buffer allows consecutive transfer to happen without interruption by another group transfer.

2.15 Hold Chip-Select Active (Master only)

There are slave devices available that require the chip-select signal to be held continuously active during several consecutive data word transfers. Other slave devices require the chip-select signal to be deactivated between consecutive data word transfers. Each MibSPI buffer can be individually initialized for either of the two modes via the CSHOLD bit in its control field. If the CSHOLD bit is set in the control field of a buffer, the chip-select signal will not be deactivated until the next control information is loaded with new chip-select information. Since the chip-select is maintained active between two transfers, the chips-select hold delay is not applied at the end of the current transaction, nor is the chip-select set-up time delay applied at the beginning of the following transaction. However, the wait delay could be still applied between the two transactions, if the bit WDEL is set within the control field.

Note:

When CSHOLD is active, no transmission interruptions are allowed. The LOCK bit does not keep the CS active.

2.16 Detection of Slave De-synchronization (Master only)

When a slave supports generation of an enable signal (ENA) a de-synchronization can be detected. With the enable signal, a slave indicates to the master that it is ready to exchange data. A de-synchronization can occur if one or more clock edges are missed by the slave. In this case, the slave may block the SOMI line until it detects clock edges corresponding to the next data word. This would corrupt the data word of the de-synchronized slave and the consecutive data word. A configurable 8-bit time-out counter, which is clocked with SPI clock, is implemented to detect this slave malfunction. After the transmission has finished (end of last bit transferred: either last data bit or parity bit) the counter is started. If the ENAble signal generated by the slave isn't becoming inactive before the counter expires the DESYNC flag is set and a interrupt is asserted if enabled. The DESYNC flag is set as well if the slave deactivates the ENAble signal before the last bit is transferred.

2.17 ENA Signal Time-out (Master only)

The MibSPI in master mode is able to wait for the hardware handshake signal (ENA) coming from the addressed slave before performing the data transfer. To avoid stalling the MibSPI by a non-responsive slave device, a time-out value can be configured. If the time-out counter expires before an active ENA signal is sampled, the TIMEOUT flag in the status register SPISTAT is set and the TIMEOUT flag in the status field of the corresponding buffer is set.

Note:

When the CS becomes active, no transmission interruption are allowed. The next arbitration is done while waiting for the time-out to occur.

3 General Purpose I/O

Each of the SPI pins may be programmed via the SPI Pin Control Registers (SPIPC1, SPIPC2, SPIPC3, SPIPC4, SPIPC5, SPIPC6) to be a general-purpose I/O pin.

When the MibSPI module is not used, the MibSPI pins may be programmed to be either general input or general output pins. The direction is controlled in the SPIPC1 register. Note that each pin can be programmed to be either a SPI pin or a GPIO pin through register SPIPC6.

If the MibSPI function is used, application software must ensure that each pin is configured as a MibSPI pin and not a GPIO pin, or else unexpected behavior may result.

4 Low Power Mode

The MibSPI module can enter low-power mode two ways: a global low-power mode from the system and a local low-power mode via the POWERDOWN bit (SPICTRL2.2). The net effect on the MibSPI is the same, regardless of the source.

In effect, low-power mode shuts down all the clocks to the module. During a global low-power mode, no registers are visible to the software; nothing can be written to or read from any register. A local low-power mode has the same effect when both the local POWERDOWN bit and the system level PPWNOVR bit are set. If only the local POWERDOWN bit is set, then the MibSPI logic is not clocked, but the registers continue to be clocked.

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

5 Interrupts

5.1 Compatibility Mode

In compatibility mode, the MiBSPI generates interrupts to the highest priority. Vectorization is not enabled, which implies that the program should poll the register SPICTRL3 flags.

To enable the interrupt in compatibility mode, the program should set the bit RXINTEN for receive interrupt or the bit OVRNINTEN for overrun interrupt.

This bit are available in the SPICTRL3 register.

The transfer group interrupt, as well as the error interrupt, are not available in compatibility mode. Therefore, there is no way to access these registers.

5.2 Multi-buffer Mode

In multi-buffer mode, the MibSPI could generates vectorized interrupt on two levels.

The overrun interrupt and receive interrupt are disabled and therefore the enable bits within the SPICTRL3 are discarded.

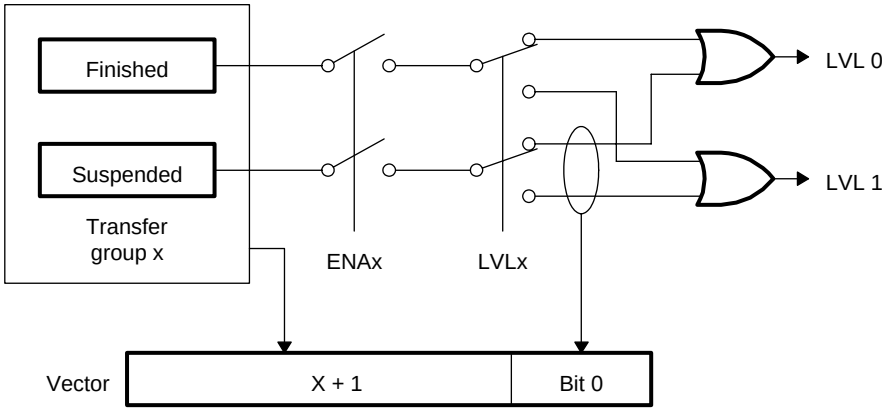
The interrupts available are:

- ❏ Transfer group completed
- ❏ Transfer group suspended
- ❏ Transmission error

When a transfer group has finished and the corresponding enable bit in the TGINTENA register is set, an interrupt “transfer finished” is generated. The level of priority of the interrupt is determined by the corresponding bit in the TGINTLVL register.

When a transfer group is suspended by a buffer that has been set as “suspend to wait” until TXFULL flag or/and RXEMPTY flag are set, and if the corresponding bit in the TGINTENA register is set, an interrupt “transfer suspended” is generated. The level of priority of the interrupt is determined by the corresponding bit in the TGINTLVL register.

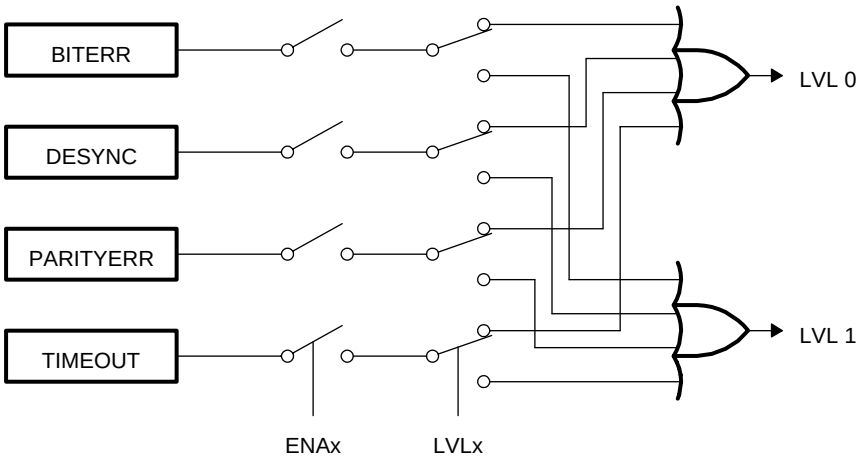
Figure 13. Transfer Group Interrupt Structure



During transmission, if one of the following error occurs: BITERR, DESYNC, PARITYERR, TIMEOUT. The corresponding flag in the SPISTAT register is set. If the enable bit is set then an interrupt is generated. The level of the interrupt could be generated according to the bit field in SPISTAT.

The error interrupt are enabled, and prioritized independently from each other, but the vector generated by the MibSPI will be the same if multiple error are enabled on the same level.

Figure 14. SPISTAT Interrupt Structure



6 DMA Interface

If handling the MibSPI message traffic on a character-by-character basis requires too much CPU overhead and if the particular device is equipped with the DMA controller, the MibSPI may use the DMA controller to receive or transmit data directly to or from memory.

6.1 Compatibility Mode

When use in compatibility mode, the MibSPI module uses one DMA request enable bit (DMA REQ EN).

When a character is being transmitted or received, the MibSPI will signal the DMA via a DMA request signal. The DMA controller will then perform the needed data manipulation.

For DMA-based transmissions, all characters are assembled in RAM, and DMA transfers move the message, word-by-word, from RAM into the SPIDAT0 register. (See the DMA controller specification). Data is then read from SPIBUF, clearing RXINTFLAG (SPICTRL3.0).

For efficient behavior, during DMA operations, the receive interrupt enable flag RXINTEN (SPICTRL3.1) should be cleared to 0. For specific DMA features, refer to the DMA controller specification.

6.2 Multi-buffer Mode

When multi-buffer mode is used, the DMA request bit in the SPICRTL 3 register is discarded. Only the DMA to buffers are allowed.

The MibSPI offers up to eight DMA channels (SEND/RECEIVE). All the DMA channels are programmable individually and could be hooked to every buffer.

The DMA transfer could be trigger on Transmit, on receive or on both events.

Each DMA channel has the possibility to transfer a block of up to 32 data without interruption using only one buffer of the array. This enables the transfer of memory block from or into an external SPI memory. For example, a group transfer could be set with three buffers of commands and a buffer hooked to a DMA channel with a no break condition.

7 Control Registers and RAM

7.1 Control Registers

This section describes the MibSPI control, data and pin registers The registers support 16-bit and 32-bit writes. The offset is relative to the associated peripheral select.

Table 3. MibSPI Registers

Offset Address†	Register	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x00	SPICTRL1	Reserved															
		WAIT-ENAx	PARI-TYx	Reserved	PRESCALE.7:0								CHARLEN.4:0				
0x04	SPICTRL2	Reserved															LOOP BACK
		Reserved	WDELAYx							SHIFT DIR	PAR POL	CLK MOD	SPI EN	MASTER	POWER DOWN	POLAR-ITY	PHASE
0x08	SPICRTL3	Reserved															
		Reserved										ENABLE HIGHZ	DMA REQ EN	OVRN INTEN	RCVR OVRN	RXINT EN	RXINT-FLAG
0x0C	SPIDAT0	Reserved															
		SPIDAT0.15:0															
0x10	SPIDAT1	Reserved				CS HOLD	Reserved		WDEL	DFSEL		CSNR					
		SPIDAT1.15:0															

Table 3. MibSPI Registers (Continued)

Offset Address†	Register	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x14	SPIBUF	RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIMEOUT	Reserved							RCVR OVR IMG	RXINT-FLAG IMG
		SPIBUF.15:0															
0x18	SPIEMU	Reserved															
		SPIEMU.15:0															
0x1C	SPIPC1	Reserved															
		Reserved				SCS DIR 7	SCS DIR 6	SCS DIR 5	SCS DIR 4	SCS DIR 3	SCS DIR 2	SCS DIR 1	SCS DIR 0	SOMI DIR	SIMO DIR	CLK DIR	ENABLE DIR
0x20	SPIPC2	Reserved															
		Reserved				SCS DIN 7	SCS DIN 6	SCS DIN 5	SCS DIN 4	SCS DIN 3	SCS DIN 2	SCS DIN 1	SCS DIN 0	SOMI DIN	SIMO DIN	CLK DIN	ENABLE DIN
0x24	SPIPC3	Reserved															
		Reserved				SCS DOUT 7	SCS DOUT 6	SCS DOUT 5	SCS DOUT 4	SCS DOUT 3	SCS DOUT 2	SCS DOUT 1	SCS DOUT 0	SOMI DOUT	SIMO DOUT	CLK DOUT	ENABLE DOUT
0x28	SPIPC4	Reserved															
		Reserved				SCS DOUT SET 7	SCS DOUT SET 6	SCS DOUT SET 5	SCS DOUT SET 4	SCS DOUT SET 3	SCS DOUT SET 2	SCS DOUT SET 1	SCS DOUT SET 0	SOMI DOUT SET	SIMO DOUT SET	CLK DOUT SET	ENABLE DOUT SET

Table 3. MibSPI Registers (Continued)

Offset Address†	Register	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x2C	SPIPC5	Reserved															
		Reserved				SCS DOUT CLR 7	SCS DOUT CLR 6	SCS DOUT CLR 5	SCS DOUT CLR 4	SCS DOUT CLR 3	SCS DOUT CLR 2	SCS DOUT CLR 1	SCS DOUT CLR 0	SOMI DOUT CLR	SIMO DOUT CLR	CLK DOUT CLR	ENABLE DOUT CLR
0x30	SPIPC6	Reserved															
		Reserved				SCS FUN 7	SCS FUN 6	SCS FUN 5	SCS FUN 4	SCS FUN 3	SCS FUN 2	SCS FUN 1	SCS FUN 0	SOMI FUN	SIMO FUN	CLK FUN	ENABLE FUN
0x34		Reserved															
		Reserved															
0x38		Reserved															
		Reserved															
0x3C		Reserved															
		Reserved															
0x40	SPICTRL4	Reserved															
		Reserved															MIBSPI ENA

Table 3. MibSPI Registers (Continued)

Offset Address†	Register	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x44	SPICTRL 5	Reserved			C2TDELAY					Reserved			T2CDELAY				
		T2EDELAY								C2EDELAY							
0x48	SPICTRL 6	Reserved															
		Reserved								CSD EF7	CSD EF6	CSD EF5	CSD EF4	CSD EF3	CSD EF2	CSD EF1	CSD EF0
0x4C	SPICTRL 7	Reserved			WDELAYx					SHIFT DIRx	PAR POL	Reserved				POLARI-TYx	PHASEx
		WAIT-ENAx	PARI-TYx	Reserved	PRESCALEx					CHARLENx							
0x50	SPICTRL 8	Reserved			WDELAYx					SHIFT DIRx	PAR POL	Reserved				POLARI-TYx	PHASEx
		WAIT-ENAx	PARI-TYx	Reserved	PRESCALEx					CHARLENx							
0x54	SPICTRL 9	Reserved			WDELAYx					SHIFT DIRx	PAR POL	Reserved				POLARI-TYx	PHASEx
		WAIT-ENAx	PARI-TYx	Reserved	PRESCALEx					CHARLENx							
0x58	SPISTAT	Reserved					BIT ERR LVL	DESYNC LVL	PAR ERR LVL	TIME OUT LVL	Reserved			BIT ERR ENA	DESYNC ENA	PAR ERR ENA	TIME OUT ENA
		Reserved					BIT ERR	DE-SYNC	PARITY ERR	TIME OUT	LCSNR						

Table 3. MibSPI Registers (Continued)

Offset Address†	Register	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x5C	TGINTENA	INTEN RDY15	INTEN RDY14	INTEN RDY15	INTEN RDY12	INTEN RDY11	INTEN RDY10	INTEN RDY9	INTEN RDY8	INTEN RDY7	INTEN RDY6	INTEN RDY5	INTEN RDY4	INTEN RDY3	INTEN RDY2	INTEN RDY1	INTEN RDY0
		INTEN SUS15	INTEN SUS14	INTEN SUS13	INTEN SUS12	INTEN SUS11	INTEN SUS10	INTEN SUS9	INTEN SUS8	INTEN SUS7	INTEN SUS6	INTEN SUS5	INTEN SUS4	INTEN SUS3	INTEN SUS2	INTEN SUS1	INTEN SUS0
0x60	TGINTLVL	INTLVL RDY15	INTLVL RDY14	INTLVL RDY13	INTLVL RDY12	INTLVL RDY11	INTLVL RDY10	INTLVL RDY9	INTLVL RDY8	INTLVL RDY7	INTLVL RDY6	INTLVL RDY5	INTLVL RDY4	INTLVL RDY3	INTLVL RDY2	INTLVL RDY1	INTLVL RDY0
		INTLVL SUS15	INTLVL SUS14	INTLVL SUS13	INTLVL SUS12	INTLVL SUS11	INTLVL SUS10	INTLVL SUS9	INTLVL SUS8	INTLVL SUS7	INTLVL SUS6	INTLVL SUS5	INTLVL SUS4	INTLVL SUS3	INTLVL SUS2	INTLVL SUS1	INTLVL SUS0
0x64	TGINTFLG	INTFLG RDY15	INTFLG RDY14	INTFLG RDY13	INTFLG RDY12	INTFLG RDY11	INTFLG RDY10	INTFLG RDY9	INTFLG RDY8	INTFLG RDY7	INTFLG RDY6	INTFLG RDY5	INTFLG RDY4	INTFLG RDY3	INTFLG RDY2	INTFLG RDY1	INTFLG RDY0
		INTFLG SUS15	INTFLG SUS14	INTFLG SUS13	INTFLG SUS12	INTFLG SUS11	INTFLG SUS10	INTFLG SUS9	INTFLG SUS8	INTFLG SUS7	INTFLG SUS6	INTFLG SUS5	INTFLG SUS4	INTFLG SUS3	INTFLG SUS2	INTFLG SUS1	INTFLG SUS0
0x68	TGINTVECT0	Reserved															
		Reserved										INTVECT0					SUS-PEND
0x6C	TGINTVECT1	Reserved															
		Reserved										INTVECT1					SUS-PEND
0x70	TICKCNT	TICK ENA	RE LOAD	CLKCTRL	Reserved												
		TICKVALUE															

Table 3. MibSPI Registers (Continued)

Offset Address†	Register	31 / 15	30 / 14	29 / 13	28 / 12	27 / 11	26 / 10	25 / 9	24 / 8	23 / 7	22 / 6	21 / 5	20 / 4	19 / 3	18 / 2	17 / 1	16 / 0
0x74	LTGPEND	Reserved															
		Reserved	LPEND							Reserved							
0x78	TG0CTRL	TG ENA	ONE SHOT	PRST	TGTD	Reserved				TRGEVT				TRIGSRC			
		Reserved	PSTART							Reserved	PCURRENT						
0x7C	TG1CTRL	TG ENA	ONE SHOT	PRST	TGTD	Reserved				TRGEVT				TRIGSRC			
		Reserved	PSTART							Reserved	PCURRENT						
		...															
0xB0	TG14CTRL	TG ENA	ONE SHOT	PRST	TGTD	Reserved				TRGEVT				TRIGSRC			
		Reserved	PSTART							Reserved	PCURRENT						
0xB4	TG15CTRL	TG ENA	ONE SHOT	PRST	TGTD	Reserved				TRGEVT				TRIGSRC			
		Reserved	PSTART							Reserved	PCURRENT						
		RX DMA ENA	TX DMA ENA	NO BRK	ICOUNT					Reserved				COUNT			

Table 3. MibSPI Registers (Continued)

Offset Address†	Register	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0xB8	DMA0CTRL	ONE SHOT	BUFID							RXDMACH				TXDMACH			
0xBC	DMA1CTRL	ONE SHOT	BUFID							RXDMACH				TXDMACH			
		RX DMA ENA	TX DMA ENA	NO BRK	ICOUNT					Reserved				COUNT			
		...															
0xD0	DMA6CTRL	ONE SHOT	BUFID							RXDMACH				TXDMACH			
		RX DMA ENA	TX DMA ENA	NO BRK	ICOUNT					Reserved				COUNT			
0xD4	DMA7CTRL	ONE SHOT	BUFID							RXDMACH				TXDMACH			
		RX DMA ENA	TX DMA ENA	NO BRK	ICOUNT					Reserved				COUNT			

† The actual addresses of these registers are device specific. See the specific device data sheet to verify the MibSPI register addresses.
‡ The SPIBUF is a 32 bit register. Two bits in the upper 16 bits are used for control, all 16 lower bits are data buffers.

7.2 MibSPI RAM

This section describes the MibSPI control and data RAM. The RAM support 16-bit and 32-bit writes. The offset is relative to the Memory chip select affected to the MibSPI RAM. In the SCMRx register (MMC) associated with the Chip select used by the MibSPI RAM, the number of wait state (WS) should be set to:

WS = 2 + (ICLK ratio).

Example: For ICLK = 3 SYSCLK, then WS = 2 + 3 = 5

Table 4. MibSPI RAM

Offset Address†	Register	31/15	30/14	29/13	28/12	27/11	26/10	25/9	24/8	23/7	22/6	21/5	20/4	19/3	18/2	17/1	16/0
0x00	Buffer 0	BUFMODE			CS HOLD	LOCK	WDEL	DFSEL		CSNR							
		TXDATA															
0x04	Buffer 1	BUFMODE			CS HOLD	LOCK	WDEL	DFSEL		CSNR							
		TXDATA															
...																	
0x1F8	Buffer 126	BUFMODE			CS HOLD	LOCK	WDEL	DFSEL		CSNR							
		TXDATA															
0x1FC	Buffer 127	BUFMODE			CS HOLD	LOCK	WDEL	DFSEL		CSNR							
		TXDATA															

Table 4. MibSPI RAM (Continued)

Offset Address†	Register	31 15	30 14	29 13	28 12	27 11	26 10	25 9	24 8	23 7	22 6	21 5	20 4	19 3	18 2	17 1	16 0
0x200	Buffer 0	RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIME OUT	Reserved	LCSNR							
		RXDATA															
0x204	Buffer 1	RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIME OUT	Reserved	LCSNR							
		RXDATA															
		...															
0x3F8	Buffer 126	RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIME OUT	Reserved	LCSNR							
		RXDATA															
0x3FC	Buffer 127	RXEMPTY	RX OVR	TX FULL	BIT ERR	DE SYNC	PARITYERR	TIME OUT	Reserved	LCSNR							
		RXDATA															

7.3 MibSPI Control Register 1 (SPICTRL1)

Bits	31															16
0x00	Reserved															
	U															
Bits	15		14		13	12						5	4			0
	WAITENA0		PARITY0		Reserved	PRESCALE							CHARLEN			
	RWP-0		RWP-0		U	RW-0							RW-0			

Legend: R = Read, W = write, P = Privilege mode U = Undefined; -*n* = Value after reset

Bits 31:16 **Reserved.**

Reads are undefined and writes have no effect.

Bit 15 **WAITENA0.** Master waits for ENA signal from slave for data format 0

WAITENA is considered in master mode only. In slave mode this bit has no meaning. WAITENA enables a flexible SPI network where slaves with ENA signal and slaves without ENA signal can be mixed. WAITENA defines for each buffer whether the addressed slave generates the ENA signal or does not.

1 = Before the MibSPI starts the data transfer it waits for the ENA signal to become low. If the ENA signal is not pulled down by the addressed slave before the internal time-out counter (CE2DELAY) expires.

0 = The MibSPI does not wait for the ENA signal from the slaves and directly starts the transfer.

Note:

This bit is only accessible in MibSPI mode

Bit 14 **PARITY0.** Parity enable for data format 0.

1 = A parity is added after transfer of the data bit. At the end of a transfer the parity generator compares the received parity bit with the locally calculated parity flag. If the parity bits do not match the RXERR flag is set in the corresponding control field. The parity type (even or odd) can be selected via the PARPOL bit (SPICTRL2.6).

0 = No parity generation/ verification is performed for this data format.

Note:

This bit is only accessible in MibSPI mode

Bits 13 **Reserved.**

Reads are undefined and writes have no effect.

Bits 12:5 **PRESCALE.**

Determines the bit transfer rate if the MibSPI is the network master. There are 255 data transfer rates (each a function of the interface clock) that can be selected. One data bit is shifted per SPICLK cycle.

If the MibSPI is a network slave, the module receives a clock signal on the SPICLK pin from the network master.

MibSPI Baud Rate for PRESCALE = 1 to 255

$$SPIBaudRate = \frac{ICLK}{(PRESCALE + 1)}$$

MibSPI Baud Rate for PRESCALE = 0

$$SPIBaudRate = \frac{ICLK}{2}$$

Bits 4:0 **CHARLEN.**

Controls how many times the MibSPI shifts per character transmitted or the number of bits per character. The binary value of the bit length must be programmed into this register. Legal values are 0x02 to 0x10. Illegal values, such as 0x00 or 0x1F are not detected and their effect is indeterminate.

Note: CHARLEN Bits Must Be Initialized

CHARLEN.4:0 must be initialized to the desired character length before the SPIEN bit is set. Otherwise, the first character may be shifted with an incorrect length.

7.4 MibSPI Control Register 2 (SPICTRL2)

Bits	31															17	16
004h	Reserved																LOOPBACK
	U																RWP-0

Bits	15	14	13					8
	Reserved		WDELAY0					
	U		RWP-0					

Bits	7	6	5	4	3	2	1	0
	SHIFTDIR0	PAR POL0	CLKMOD	SPIEN	MASTER	PWRDWN	POLARITY	PHASE
	RWP-0	RWP-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend: RW = Read/Write in all modes, WP = Write in privilege mode only, U = Undefined, -*n* = Value after reset, x = indeterminate

Bits 31:17 Reserved

Bit 16 LOOP BACK. Internal loop-back test mode.

The internal self-test option can be enabled by setting this bit. If the SPISIMO and SPISOMI pins are configured with SPI functionality, then the SPISIMO pin is internally connected to the SPISOMI pin. The transmit data is looped back as receive data and is stored in the receive field of the concerned buffer.

Externally, during loop-back operation, the SPICLK pin outputs an inactive value and SPISOMI remains in high-impedance state. The MibSPI has to be initialized in master mode before the loop-back can be selected. If the MibSPI is initialized in slave mode or a data transfer is ongoing, errors may result.

1 = Internal loop-back test mode enabled.

0 = Internal loop-back test mode disabled.

Note:

This bit is only accessible in MibSPI mode

Bits 15:14 Reserved

Reads are undefined and writes have no effect.

Bits 13:8 **WDELAY0.** Delay in between transmissions for data format 0.

Idle time that will be applied at the end of the current transmission if the bit WDEL is set in the current buffer. (See section 2.5.2)

The delay to be applied is equal to: WDELAY * P_{ICLK} + 2 * P_{ICLK}.

Note:

This bit field is only accessible in MibSPI mode

Bit 7 **SHIFTDIR0.** Shift direction for data format 0.

With bit SHIFTDIR0 the shift direction for data format 0 can be selected.

1 = Data format 0 shift direction: Least significant bit is shifted out first.

0 = Data format 0 shift direction: Most significant bit is shifted out first.

Note:

This bit is only accessible in MibSPI mode

Bit 6 **PARPOL0.** Parity polarity: even or odd.

PARPOL0 can be modified in privilege mode only. It can be used for data format 0.

1 = If PARPOL0 is set to “1” and SPICTRL 1.14 is enabled, a odd parity flag is added at the end of the transmit data stream.

0 = If PARPOL0 is set to “0” and SPICTRL 1.14 is enabled, a even parity flag is added at the end of the transmit data stream.

Note:

This bit is only accessible in MibSPI mode

Bit 5 **CLKMOD.** Clock mode

Selects either an internal or external clock source. This bit also determines the I/O direction of the $\overline{\text{SPIENA}}$ and $\overline{\text{SPISCS}}$[3:0] pins in functional mode.

0 =Clock is external

1 =Clock is internal

Bit 4 **SPIEN.** SPI enable

Holds the MibSPI in a reset state after a chip reset. The MibSPI is enabled only after a 1 is written to this bit. This bit must be set to 1 after all other MibSPI configuration bits have been written. This prevents an invalid operation of the MibSPI while the clock polarity is being changed. When this bit is 0, the MibSPI shift registers (SPIDAT0 and SPIDAT1) is held in reset mode and forced to 0x0000.

The RXINTFLAG (SPICTRL3.0) and RCVROVRN (SPITRL3.2) bits are also held in reset mode and forced to 0 when this bit is 0. SPICLK is disabled when this bit is 0.

0 = MibSPI is in reset

1 = Activates MibSPI

Bit 3 **MASTER.** SPISIMO/SPISOMI pin direction determination.

Determines the direction of the SPISIMO and SPISOMI pins.

0 = SPISIMO pin an input, SPISOMI pin an output

1 = SPISOMI pin an input, SPISIMO pin an output

Bit 2 **POWERDOWN.**

When active, the MibSPI state machines enter a powerdown state.

0 = MibSPI in active mode

1 = MibSPI in powerdown mode

Bit 1 **POLARITY.**

Controls the polarity of the SPICLK. Clock polarity and clock phase (SPICTRL2.0) controls four clocking schemes on the SPICLK pin. See Figure 6 to Figure 9, page -15 for wave form diagrams of the MibSPI clocking schemes.

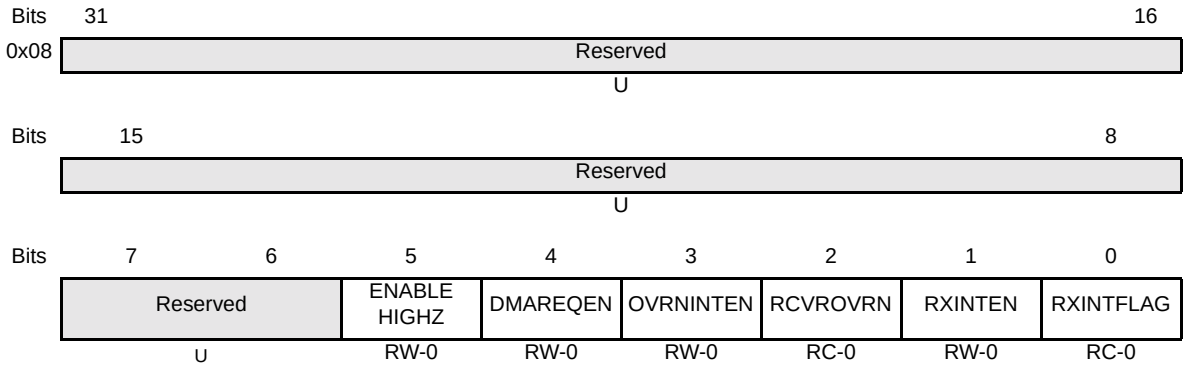
Bit 0 **PHASE.**

Data is sent or latched in-phase with the clock signal. When PHASE = 1, SPICLK is delayed by one-half cycle from when data is output. Polarity is determined by the POLARITY bit (SPICTRL2.1). POLARITY and PHASE make four different clocking schemes possible. For information on the use of the Polarity and Phase bits, see section 2.5, *Data Format,* on page 12

Note: Register Configuration Bits

Since there are configuration bits in this register, two write operations must occur when setting these bits. One write to set the configuration bits and one to set the SPIEN bit.

7.5 MibSPI Control Register 3 (SPICTRL3)



Legend: R = Read, W = write, C = Clear, U = Undefined; *-n* = Value after reset

Bits 31:6 **Reserved.**

Reads are undefined and writes have no effect.

Bit 5 **ENABLE HIGHZ.** $\overline{\text{SPIENA}}$ pin high-z enable.

When active, the $\overline{\text{SPIENA}}$ pin (when it is configured as a WAIT functional output signal in a slave SPI) is forced to place it's output in high-z when not driving a low signal. If inactive, then the pin will output both a high and a low signal.

0 = $\overline{\text{SPIENA}}$ pin is a value
1 = $\overline{\text{SPIENA}}$ pin is in high-z

Bit 4 **DMA REQ EN.** DMA request enable.

Enables the DMA request signal to be generated for both receive and transmit channels.

0 = DMA is not used
1 = DMA is used

Bit 3 **OVRNINTEN.** Overrun interrupt enable.

An interrupt is to be generated when the RCVR OVRN flag bit (SPICTRL3.2) is set by hardware. Otherwise, no interrupt will be generated.

0 = Overrun interrupt will not be generated

1 = Overrun interrupt will be generated

Bit 2 **RCVROVRN.** Receiver overrun flag.

This bit is a read/clear only flag. The MibSPI hardware sets this bit when an operation completes before the previous character has been read from the buffer. The bit indicates that the last received character has been overwritten and therefore lost. The MibSPI will generate an interrupt request if this bit is set and the OVRN INTEN bit (SPICTRL3.3) is set high.

This bit is cleared in one of four ways:

- ❏ Reading the SPIBUF register
- ❏ Writing a 1 to this bit
- ❏ Writing a 0 to SPIEN (SPICTRL2.4)
- ❏ System reset

0 = Overrun condition did not occur

1 = Overrun condition has occurred

Bit 1 **RXINTEN.**

An interrupt is to be generated when the RXINTFLAG bit (SPICTRL3.0) is set by hardware. Otherwise, no interrupt will be generated.

0 = Interrupt will not be generated

1 = Interrupt will be generated

Bit 0 **RXINTFLAG.** Serves as the MibSPI interrupt flag.

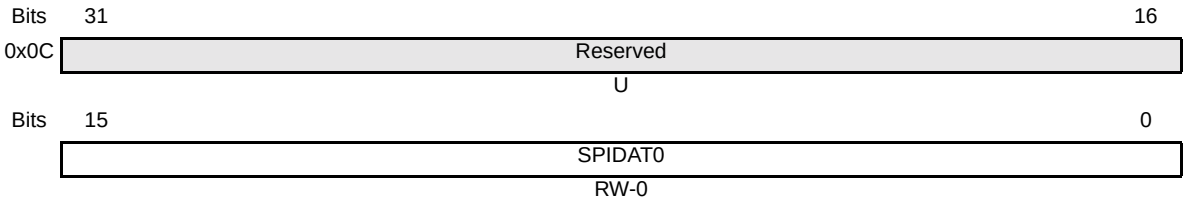
This flag is set when a word is received and copied into the buffer register (SPIBUF). If RXINTEN is enabled, an interrupt is also generated. During emulation mode, however, a read to the emulation register (SPIEMU) does not clear this flag bit. This bit is cleared in one of four ways:

- ❏ Reading the SPIBUF register
- ❏ Writing a 1 to this bit
- ❏ Writing a 0 to SPIEN (SPICTRL2.4)
- ❏ System reset

0 = Interrupt condition did not occur

1 = Interrupt condition did occur

7.6 MibSPI Shift Register 0 (SPIDAT0)



Legend: R = Read, W = write, U = Undefined; -*n* = Value after reset

Note: Accessibility of SPIDAT0 in MibSPI mode

The SPIDAT0 register is not accessible in MibSPI mode. It is only accessible in compatibility mode.

Bits 31:16 **Reserved**

Reads are undefined and writes have no effect.

Bits 15:0 **SPIDAT0.** MibSPI shift data 0.

These bits make up the MibSPI shift register 0. Data is shifted out of the MSB (bit 15) and into the LSB (bit 0).

SPIEN (SPICTRL2.4) must be set to 1 before this register can be written to. Writing a 0 to the SPIEN register forces the lower 16 bits of the SPIDAT0 register to 0x00.

When data is read from this register, the value is indeterminate because of the shift operation. The value in the buffer register (SPIBUF) should be read after the shift operation is complete to determine what data was shifted into the SPIDAT0 register.

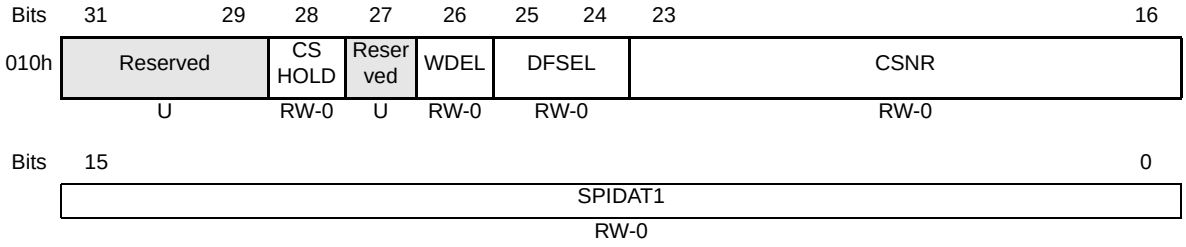
When transmitting data, input data is automatically clocked in at the receive side. As the data is shifted from the MSB, the LSB of the received data is shifted in. Similarly, when the shift register is used as a receiver, the shift register continues to send data out as it receives new data on each input clock cycle. This allows the concurrent transmission and reception of data. The application software must determine whether the data transferred is valid.

For word sizes of 8 bits or less (as determined by CHARLEN) (SPICTRL1.4:0), the shift register is tapped at SPIDAT.7. As a result, data of 8 bits does not need to be justified at all. For data of less than 8 bits, the data should be justified as if it is an 8-bit register.

7.7 MibSPI Shift Register 1 (SPIDAT1)

Note: Accessibility of SPIDAT1 in MibSPI mode

The MibSPI kernel allows access to all bits in SPIDAT1 in MibSPI mode, whereas in compatibility mode only the least significant 16 bits can be writable.



Legend: R = Read, W = Write, C = Clear, U = Undefined, -*n* = Value after reset, x = indeterminate

Bit 31:29 **Reserved.**

Bit 28 **CSHOLD.** Chip select hold mode

CSHOLD is considered in master mode only. In slave mode this bit has no meaning. CSHOLD defines the behavior of the chip select line at the end of a data transfer.

1 = The chip select signal is held active at the end of a transfer until a control field with new data and control information is loaded into SPIDAT1. If the new chip select information equals the previous one the active chip select signal is extended until the end of transfer with CSHOLD cleared or until the chip select information changes.

0 = The chip select signal is deactivated at the end of a transfer after the T2CDELAY time has passed. If two consecutive transfers are dedicated to the same chip select this chip select signal will be shortly deactivated before it is activated again.

Bit 27 **Reserved**

Bit 26 **WDEL.** Enable the delay counter at the end of the current transaction.

1 = After the transaction WDELAY of the corresponding data format is loaded into the delay counter. No transaction is performed until the counter is reset.

0 = No delay is inserted.

Bit 25:24 **DFSEL.** Data word format select.

DFSEL1	DFSEL0	Description
0	0	Data word format 0 is selected (see section 7.19) for this buffer
0	1	Data word format 1 is selected (see specification of existing TMS470SPI) for this buffer
1	0	Data word format 2 is selected
1	1	Data word format 3 is selected

Bits 23:16 **CSNR.** Chip select number.

CSNR defines the chip select that shall be activated during the data transfer.

Bits 15:0 **SPIDAT1.** MibSPI shift data 1.

These bits make up the MibSPI shift register 1. Data is shifted out of the MSB (bit 15) and into the LSB (bit 0).

SPIEN must be set to 1 before this register can be written to. Writing a 0 to the SPIEN register forces the lower 16 bits of the SPIDAT1 register to 0x00.

Write to this register ONLY when using the automatic Slave Chip Select feature. See section 2, *MibSPI Operation Modes,* on page 3. A write to this register will drive the $\overline{\text{SPISCS}}$ signal low.

When data is read from this register, the value is indeterminate because of the shift operation. The value in the buffer register (SPIBUF) should be read after the shift operation is complete to determine what data was shifted into the SPIDAT1 register.

7.8 MibSPI Buffer Register (SPIBUF)

Note: Accessibility of SPIBUF, SPIEMU in MibSPI mode

The SPIBUF and SPIEMU register are not accessible in normal MibSPI mode. They are only accessible in compatibility mode.

Bits	31	30	29	28	27	26	25	24
014h	RXEMPTY	RXOVR	TXFULL	BITERR	DESYNC	PARITYERR	TIMEOUT	Reserved
	R-1	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	U

Bits	23	18	17	16
	Reserved		RCVROVRNIMG	INTFLAGIMG
	U		RC-0	RC-0

Bits	15	0
	SPIBUF	
	R-x	

Legend: R = Read, W = Write, C = Clear, U = Undefined, *-n* = Value after reset, x = indeterminate

Bit 31 **RXEMPTY.** Receive data buffer empty.

This flag is a read-only flag. When host reads the SPIBUF field or the whole SPIBUF register this will automatically set the RXEMPTY flag. When a data transfer has been finished and the received data is copied into SPIBUF the RXEMPTY flag is cleared.

1 = No data received since last reading of SPIBUF register.

0 = Data is received and copied into SPIBUF field.

Bit 30 **RXOVR.** Receive data buffer overrun.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it.

It has the same meaning as the RCVROVRNIMG flag at bit position 17, but is re-mapped to bit 30 due to compatibility reason with the buffer status fields. When a data transfer has been finished and the received data is copied into the SPIBUF while RXEMPTY flag is already cleared RXOVR is set.

1 = A receive data overrun condition occurred since last time reading the status field.

0 = No receive data overrun condition occurred since last time reading the status field

Bit 29 **TXFULL.** Transmit data buffer full.

This flag is a read-only flag. Writing into SPIDAT0 or SPIDAT1 field will automatically set the TXFULL flag. After transfer of the transmit data the TXFULL flag is cleared.

1 = Host provided new transmit data to SPIDAT0 or SPIDAT1.

0 = No new transmit data from host since previous transfer of transmit data.

Bit 28 **BITERR.** Mismatch of internal transmit data and transmitted data.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it. It represents a copy of the BITERR flag in SPISTAT.

1 = A bit error occurred. The MibSPI 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 flag BITERR is set. A possible reason for a bit error can be noise, a to high bit rate / capacitive load or another master/slave trying to transmit at the same time.

0 = No bit error occurred.

Bit 27 **DESYNC.** De-synchronization of slave device.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it. De-synchronization monitor is active in master mode only. DESYNC represents a copy of the DESYNC flag in SPISTAT.

1 = A slave device is de-synchronized. The master monitors the ENA signal coming from the slave device and sets the DESYNC flag if ENA is deactivated before the last reception point or after the last bit is transmitted plus $t_{T2EDELAY}$ (see Section 7.17). If DESYNCENA is set an interrupt is asserted. De-synchronization can occur if a slave device misses a clock edge coming from the master or is detecting an additional clock edge due to perturbation.

0 = No slave de-synchronization detected.

Bit 26 **PARITYERR.** Calculated parity differs from received parity bit.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it. It represents a copy of the PARITYERR flag in SPISTAT.

1 = A parity error occurred. If the parity generator is enabled (can be selected individually for each buffer) an even or odd parity bit is added at the end of a data word (see Section 7.19). During reception of the data word the parity generator calculates the reference parity and compares it to the received parity bit. In the event of a mismatch the PARITYERR flag is set.

0 = No parity error detected.

Bit 25 **TIMEOUT.** Time-out due to non-activation of ENA signal.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it.

1 = An ENA signal time-out occurred. The MibSPI generates a time-out because the slave hasn't responded in time by activating the ENA signal after the chip select signal has been activated. If a time-out condition is detected the corresponding chip select is deactivated immediately and the TIMEOUT flag is set. In addition the TIMOUT flag in the status field of the corresponding buffer and in the SPISTAT register is set.

0 = No ENA-signal time-out occurred.

Bits 24:18 **Reserved.** Reads are undefined and writes have no effect

Bit 17 **RCVR OVRN IMG.** MibSPI receiver overrun flag image.

This is a mirror bit of the RCVROVRN flag bit (SPICTRL3.2) and is used to reduce the interrupt latency and execution time.

This bit is cleared in one of four ways.

- ❏ Reading the SPIBUF register
- ❏ Writing a 1 to this bit
- ❏ Writing a 0 to SPIEN (SPICTRL2.4)
- ❏ System reset

0 = Overrun condition did not occur

1 = Overrun condition has occurred

Bit 16 **RXINTFLAG IMG.** MibSPI interrupt flag image.

This is a mirror bit of the RXINTFLAG bit (SPICTRL3.0).

This bit is cleared in one of four ways.

- ❏ Reading the SPIBUF register
- ❏ Writing a 1 to this bit
- ❏ Writing a 0 to SPIEN (SPICTRL2.4)
- ❏ System reset

0 = Interrupt condition did not occur

1 = Interrupt condition did occur

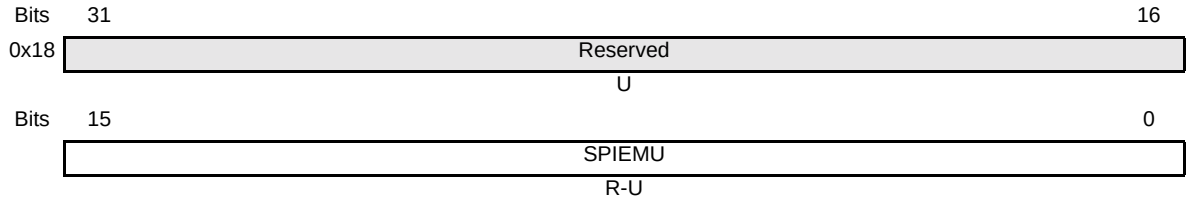
Bits 15:0 **SPIBUF.** MibSPI buffer.

The data in this register is the data transferred from the shift-register (SPIDAT). Since the data is shifted into the MibSPI most significant bit first, for word lengths less than 16, the data is stored right-justified in the register.

Note: MibSPI Buffer

Reading the SPIBUF register clears the RCVROVRN (SPICTRL3.2), RXINTFLAG (SPICTRL3.0), RCVR OVRN IMG (SPIBUF.17), and the RXINTFLAG IMG (SPIBUF.16) bits.

7.9 MibSPI Emulation Register (SPIEMU)



Legend: R = Read, U = Undefined; -*n* = Value after reset

Bits 31:16 **Reserved.**

Reads are undefined and writes have no effect

Bits 15:0 **SPIEMU.** MibSPI emulation.

MibSPI emulation is a mirror of the SPIBUF register. The only difference between SPIEMU and SPIBUF is that a read from SPIEMU does not clear the RCVR OVRN (SPICTRL3.2) or RXINTFLAG (SPICTRL3.0) bits.

7.10 MibSPI Pin Control Register 1 (SPIPC1)

Bits	31							16
0x1C	Reserved							
	U							

Bits	15			12	11	10	9	8
	Reserved				SCSDIR7	SCSDIR6	SCSDIR5	SCSDIR4
	U				RW-0	RW-0	RW-0	RW-0

Bits	7	6	5	4	3	2	1	0
	SCSDIR3	SCSDIR2	SCSDIR1	SCSDIR0	SOMIDIR	SIMOIDIR	CLKDIR	ENADIR
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend: R = Read, C = Clear, U = Undefined; *-n* = Value after reset

Bits 31:12 **Reserved.**

Reads are undefined and writes have no effect

Bit 11:4 **SCSDIR x.** $\overline{\text{SPISCSx}}$ direction.

Controls the direction of the $\overline{\text{SPISCSx}}$ pins when they are used as a general-purpose I/O pin. Each pins could be configured independently from the others If the $\overline{\text{SPISCSx}}$ is used as a SPI functional pin, the I/O direction is determined by the CLKMOD bit (SPICTRL2.5).

0 = $\overline{\text{SPISCSx}}$ pin is an input

1 = $\overline{\text{SPISCSx}}$ pin is an output

Bit 3 **SOMIDIR.** SPISOMI direction.

Controls the direction of the SPISOMI pin when it is used as a general-purpose I/O pin. If the SPISOMI pin is used as a MibSPI functional pin, the I/O direction is determined by the MASTER bit (SPICTRL2.3).

0 = SPISOMI pin is an input

1 = SPISOMI pin is an output

Bit 2 **SIMODIR.** SPISIMO direction.

Controls the direction of the SPISIMO pin when it is used as a general-purpose I/O pin. If the SPISIMO pin is used as a MibSPI functional pin, the I/O direction is determined by the MASTER bit (SPICTRL2.3).

0 = SPISIMO pin is an input

1 = SPISIMO pin is an output

Bit 1 **CLKDIR.** SPICLK direction.

Controls the direction of the SPICLK pin when it is used as a general-purpose I/O pin. In functional mode, the I/O direction is determined by the CLKMOD bit (SPICTRL2.5).

0 = SPICLK pin is an input

1 = SPICLK pin is an output

Bit 0 **ENADIR.** $\overline{\text{SPIENA}}$ direction.

Controls the direction of the $\overline{\text{SPIENA}}$ pin when it is used as a general-purpose I/O. If the $\overline{\text{SPIENA}}$ pin is used as a functional pin, then the I/O direction is determined by the CLKMOD bit (SPICTRL2.5).

0 = SPIENA pin is an input

1 = SPIENA pin is an output

7.11 MibSPI Pin Control Register 2 (SPIPC2)

Bits	31							16
0x20	Reserved							
	U							

Bits	15			12	11	10	9	8
	Reserved				SCSDIN7	SCSDIN6	SCSDIN5	SCSDIN4
	U				R-U	R-U	R-U	R-U

Bits	7	6	5	4	3	2	1	0
	SCSDIN3	SCSDIN2	SCSDIN1	SCSDIN0	SOMIDIN	SIMODIN	CLKDIN	ENADIN
	R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U

Legend: R = Read, C = Clear, U = Undefined; *-n* = Value after reset

Bits 31:12 **Reserved.**
Reads are undefined and writes have no effect

Bit 11:4 **SCSDINx.** $\overline{\text{SPISCSx}}$ data in.
Reflects the value of the $\overline{\text{SPISCSx}}$ pins.
0 = Current value on $\overline{\text{SPISCSx}}$ pin is logic 0.
1 = Current value on $\overline{\text{SPISCSx}}$ pin is logic 1

Bit 3 **SOMIDIN.** SPISOMI data in.
Reflects the value of the SPISOMI pin.
0 = Current value on SPISOMI pin is logic 0.
1 = Current value on SPISOMI pin is logic 1

Bit 2 **SIMODIN.** SPISIMO data in.
Reflects the value of the SPISIMO pin.
0 = Current value on SPISIMO pin is logic 0.
1 = Current value on SPISIMO pin is logic 1

Bit 1 **CLKDIN.** Clock data in.
Reflects the value of the SPICLK pin.
0 = Current value on SPICLK pin is logic 0.
1 = Current value on SPICLK pin is logic 1

Bit 0 **ENADIN.** $\overline{\text{SPIENA}}$ data in.
Reflects the value of the $\overline{\text{SPIENA}}$ pin.
0 = Current value on $\overline{\text{SPIENA}}$ pin is logic 0.
1 = Current value on $\overline{\text{SPIENA}}$ pin is logic 1

7.12 MibSPI Pin Control Register 3 (SPIPC3)

Bits	31							16
0x24	Reserved							
	U							

Bits	15			12	11	10	9	8
	Reserved				SCSDOUT7	SCSDOUT6	SCSDOUT5	SCSDOUT4
	U				RW-0	RW-0	RW-0	RW-0

Bits	7	6	5	4	3	2	1	0
	SCSDOUT3	SCSDOUT2	SCSDOUT1	SCSDOUT0	SOMIDOUT	SIMODOUT	CLKDOUT	ENADOUT
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend: R = Read, W = Write, U = Undefined; -*n* = Value after reset

Bits 31:12 **Reserved.**

Reads are undefined and writes have no effect.

Bit 11:4 **SCSDOUTx.** $\overline{\text{SPISCSx}}$ dataout write.

Only active when the $\overline{\text{SPISCSx}}$ pins are configured as a general-purpose I/O pins and configured as an output pins. The value of these bit indicates the value sent to the pins.

0 = Current value on $\overline{\text{SPISCSx}}$ pin is logic 0.

1 = Current value on $\overline{\text{SPISCSx}}$ pin is logic 1

Bit 3 **SOMIDOUT.** SPISOMI dataout write.

Only active when the SPISOMI pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.

0 = Current value on SPISOMI pin is logic 0.

1 = Current value on SPISOMI pin is logic 1

Bit 2 **SIMODOUT.** SPISIMO dataout write.

Only active when the SPISIMO pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.

0 = Current value on SPISIMO pin is logic 0.

1 = Current value on SPISIMO pin is logic 1

Bit 1 **CLKDOUT.** SPICLK dataout write.

Only active when the SPICLK pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.

0 = Current value on SPICLK pin is logic 0.

1 = Current value on SPICLK pin is logic 1

Bit 0 **ENADOUT.** $\overline{\text{SPIENA}}$ dataout write.

Only active when the $\overline{\text{SPIENA}}$ pin is configured as a general-purpose I/O pin and configured as an output pin. The value of this bit indicates the value sent to the pin.

0 = Current value on $\overline{\text{SPIENA}}$ pin is logic 0.

1 = Current value on $\overline{\text{SPIENA}}$ pin is logic 1

7.13 MibSPI Pin Control Register 4 (SPIPC4)

Bits	31							16
0x28	Reserved							
	U							

Bits	15			12	11	10	9	8
	Reserved				SCSDSET7	SCSDSET6	SCSDSET5	SCSDSET4
	U				RW-0	RW-0	RW-0	RW-0

Bits	7	6	5	4	3	2	1	0
	SCSDSET3	SCSDSET2	SCSDSET1	SCSDSET0	SOMIDSET	SIMODSET	CLKDSET	ENADSET
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend: R = Read, W = Write, U = Undefined; -*n* = Value after reset

Bits 31:12 **Reserved.**

Reads are undefined and writes have no effect

Bit 11:4 **SCSDOUTSETx.** $\overline{\text{SPISCSx}}$ dataout set.

Only active when the $\overline{\text{SPISCSx}}$ pins are configured as a general-purpose output pins. A value of one written to these bits set the corresponding $\overline{\text{SCS}}$DOUT bit (SPIPC3.4) to one.

Write:

0 = Has no effect

1 = Logic 1 placed on $\overline{\text{SPISCSx}}$ pin

Read:

0 = Current value on $\overline{\text{SPISCSx}}$ pin is logic 0.

1 = Current value on $\overline{\text{SPISCSx}}$ pin is logic 1

Bit 3 **SOMIDSET.** SPISOMI dataout set.

Only active when the SPISOMI pin is configured as a general-purpose output pin. A value of one written to this bit sets the corresponding SPISOMIDOUT bit (SPIPC3.3) to one.

Write:

0 = Has no effect

1 = Logic 1 placed on SPISOMI pin

Read:

0 = Current value on SPISOMI pin is logic 0.

1 = Current value on SPISOMI pin is logic 1

Bit 2 **SIMODSET.** SPISIMO dataout set.

Only active when the SPISIMO pin is configured as a general-purpose output pin. A value of one written to this bit sets the corresponding SPISIMODOUT bit (SPIPC3.2) to one.

Write:

0 = Has no effect

1 = Logic 1 placed on SPISIMO pin

Read:

0 = Current value on SPISIMO pin is logic 0.

1 = Current value on SPISIMO pin is logic 1

Bit 1 **CLKDSET.** SPICLK dataout set.

Only active when the SPICLK pin is configured as a general-purpose output pin. A value of one written to this bit sets the corresponding CLKDOUT bit (SPIPC3.1) to one.

Write:

0 = Has no effect

1 = Logic 1 placed on SPICLK pin

Read:

0 = Current value on SPICLK pin is logic 0.

1 = Current value on SPICLK pin is logic 1

Bit 0 **ENADSET.** $\overline{\text{SPIENA}}$ dataout set.

Only active when the $\overline{\text{SPIENA}}$ pin is configured as a general-purpose output pin. A value of one written to this bit sets the corresponding ENABLEDOUT bit (SPIPC3.0) to one.

Write:

0 = Has no effect

1 = Logic 1 placed on $\overline{\text{SPIENA}}$ pin

Read:

0 = Current value on $\overline{\text{SPIENA}}$ pin is logic 0.

1 = Current value on $\overline{\text{SPIENA}}$ pin is logic 1

Note: Register Read

A read to this register gives the corresponding value of the SPIPC3 register.

7.14 MibSPI Pin Control Register 5 (SPIPC5)

Bits	31							16
0x2C	Reserved							
	U							

Bits	15			12	11	10	9	8
	Reserved				SCSDCLR7	SCSDCLR6	SCSDCLR5	SCSDCLR4
	U				RW-0	RW-0	RW-0	RW-0

Bits	7	6	5	4	3	2	1	0
	SCSDCLR3	SCSDCLR2	SCSDCLR1	SCSDCLR0	SOMIDCLR	SIMODCLR	CLKDCLR	ENADCLR
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend :R = Read, W = Write, U = Undefined; -*n* = Value after reset

Bits 31:12 **Reserved.**

Reads are undefined and writes have no effect

Bit 11:4 **SCSDCLRx.** $\overline{\text{SPISCSx}}$ dataout clear.

Only active when the $\overline{\text{SPISCSx}}$ pins are configured as a general-purpose output pins. A value of one written to this bit clears the corresponding SCSDOUT bit (SPIPC3.4) to zero.

Write:

0 = Has no effect

1 = Logic 0 placed on $\overline{\text{SPISCSx}}$ pin

Read:

0 = Current value on $\overline{\text{SPISCSx}}$ pin is logic 0.

1 = Current value on $\overline{\text{SPISCSx}}$ pin is logic 1

Bit 3 **SOMIDCLR.** SPISOMI dataout clear.

Only active when the SPISOMI pin is configured as a general-purpose output pin. A value of one written to this bit clears the corresponding SPISOMIDOUT bit (SPIPC3.3) to zero.

Write:

0 = Has no effect

1 = Logic 0 placed on SPISOMI pin

Read:

0 = Current value on SPISOMI pin is logic 0.

1 = Current value on SPISOMI pin is logic 1

Bit 2 **SIMODCLR.** SPISIMO dataout clear.

Only active when the SPISIMO pin is configured as a general-purpose output pin. A value of one written to this bit clears the corresponding SPISIMODOUT bit (SPIPC3.2) to zero.

Write:

0 = Has no effect

1 = Logic 0 placed on SPISIMO pin

Read:

0 = Current value on SPISIMO pin is logic 0.

1 = Current value on SPISIMO pin is logic 1

Bit 1 **CLKDCLR.** SPICLK dataout clear.

Only active when the SPICLK pin is configured as a general-purpose output pin. A value of one written to this bit clears the corresponding CLKDOUT bit (SPIPC3.1) to zero.

Write:

0 = Has no effect

1 = Logic 0 placed on SPICLK pin

Read:

0 = Current value on SPICLK pin is logic 0.

1 = Current value on SPICLK pin is logic 1

Bit 0 **ENADCLR.** $\overline{\text{SPIENA}}$ dataout clear.

Only active when the $\overline{\text{SPIENA}}$ pin is configured as a general-purpose output pin. A value of one written to this bit clears the corresponding ENABLEDOUT bit (SPIPC3.0) to zero.

Write:

0 = Has no effect

1 = Logic 0 placed on $\overline{\text{SPIENA}}$ pin

Read:

0 = Current value on $\overline{\text{SPIENA}}$ pin is logic 0.

1 = Current value on $\overline{\text{SPIENA}}$ pin is logic 1

Note: Register Read

A read to this register gives the corresponding value of the SPIPC3 register.

7.15 MibSPI Pin Control Register 6 (SPIPC6)

Bits	31							16
0x30	Reserved							
	U							

Bits	15			12	11	10	9	8
	Reserved				SCSFUN7	SCSFUN6	SCSFUN5	SCSFUN4
	U				RW-0	RW-0	RW-0	RW-0

Bits	7	6	5	4	3	2	1	0
	SCSFUN3	SCSFUN2	SCSFUN1	SCSFUN0	SOMIFUN	SIMOFUN	CLKFUN	ENAFUN
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend: R = Read, W = Write, U = Undefined; -*n* = Value after reset

Bits 31:12 **Reserved.**

Reads are undefined and writes have no effect

Bit 11:4 **SCSFUNx.** $\overline{\text{SPISCSx}}$ function.

Determines whether the $\overline{\text{SPISCSx}}$ pins are to be used as a general-purpose I/O pins or as SPI functional pins. If the slave $\overline{\text{SPISCSx}}$ pins are in functional mode and receive an inactive high signal, the slave SPI will place it's output in high-z and disable shifting.

0 = $\overline{\text{SPISCSx}}$ pin is a GPIO

1 = $\overline{\text{SPISCSx}}$ pin is a SPI functional pin

Bit 3 **SOMIFUN.** Slave out, master in function.

Determines whether the SPISOMI pin is to be used as a general-purpose I/O pin or as a MibSPI functional pin.

0 = SPISOMI pin is a GPIO

1 = SPISOMI pin is a MibSPI functional pin

Bit 2 **SIMOFUN.** Slave in, master out function.

Determines whether the SPISIMO pin is to be used as a general-purpose I/O pin, or as a MibSPI functional pin.

0 = SPISIMO pin is a GPIO

1 = SPISIMO pin is a MibSPI functional pin

Bit 1 **CLKFUN.** MibSPI clock function.

Determines whether the SPICLK pin is to be used as a general-purpose I/O pin, or as a MibSPI functional pin.

0 = SPICLK pin is a GPIO

1 = SPICLK pin is a MibSPI functional pin

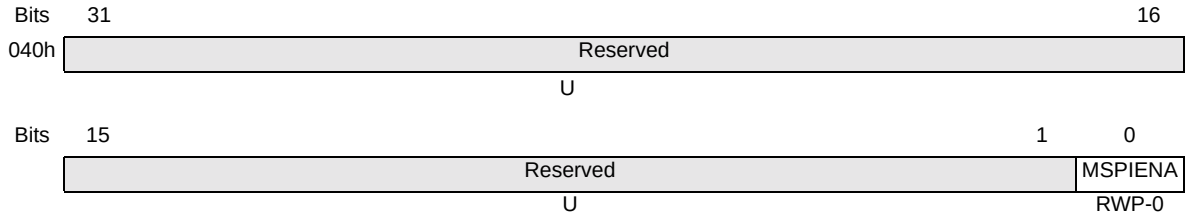
Bit 0 **ENAFUN.** $\overline{\text{SPIENA}}$ function.

Determines whether the $\overline{\text{SPIENA}}$ pin is to be used as a general-purpose I/O pin, or as a MibSPI functional pin.

0 = $\overline{\text{SPIENA}}$ pin is a GPIO

1 = $\overline{\text{SPIENA}}$ pin is a MibSPI functional pin

7.16 SPI Control Register 4 (SPICTRL4)



Legend: R = Read in all modes, WP = Write in privilege mode only, U =Undefined, -*n* = Value after reset, x = indeterminate

Bits 31:1 **Reserved**

Bit 0 **MSPIENA.** MibSPI enable.

After power-up or reset MSPIENA is cleared, which means that the MibSPI runs in compatibility mode. MSPIENA can be modified in privilege mode only.

1 = The MibSPI is configured to run in MibSPI mode. In this mode the additional features are available. If no multi-buffer RAM is attached to the MibSPI kernel only the features implemented in the kernel are available.

0 = The MibSPI runs in compatibility mode, i.e. in this mode the MibSPI is fully code compliant to the standard TMS470SPI. All enhanced features are deactivated.

7.17 SPI Control Register 5 (SPICTRL5)

Bits	31	29	28	24	23	21	20	16
044h	Reserved		C2TDELAY		Reserved		T2CDELAY	
	U		RW-0		U		RW-0	

Bits	15	8	7	0
	T2EDELAY		C2EDELAY	
	RW-0		RW-0	

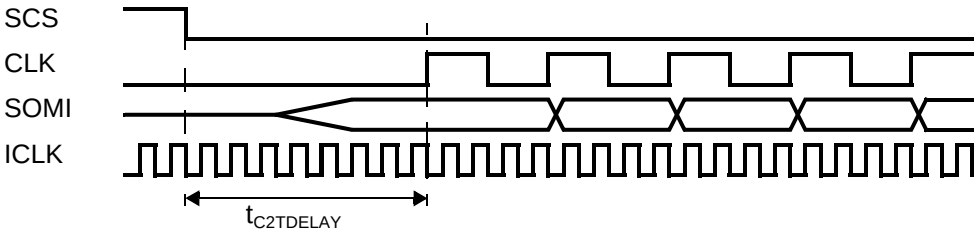
Legend: RW = Read/Write in all modes, WP = Write in privilege mode only, U = Undefined, *-n* = Value after reset

Bit 31:29 **Reserved**

Bits 28:24 **C2TDELAY.** Chip-select-active-to-transmit-start-delay.

C2TDELAY is used in master mode only. If the slave device indicates the ready-to-transfer status via the ENA signal, C2TDELAY time is not delayed after the ENA signal is activated. It defines a setup time for the slave device that delays the data transmission from the chip select active edge by a multiple of ICLK cycles. C2TDELAY can be configured between 2 and 33 ICLK cycles.

Figure 15. Example: $t_{C2TDELAY}$ = 8 ICLK cycles



The setup time value is calculated as shown in Equation 1..

Equation 1. Chip-Select-Active-to-Transmit-Start-Delay-Time

$$t_{C2TDELAY} = \frac{C2TDELAY}{ICLK} + 2$$

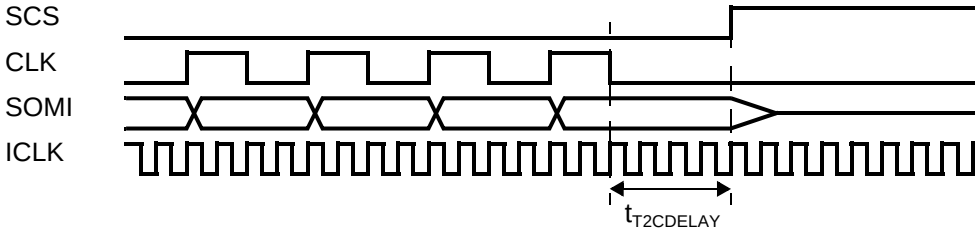
Example: ICLK = 25 MHz; C2TDELAY = 06h;
> $t_{C2TDELAY}$ = 320 ns;
When the chip select signal becomes active the slave has to prepare data transfer within 320 ns.

Bits 23:21 **Reserved**

Bits 20:16 **T2CDELAY.** Transmit-end-to-chip-select-inactive-delay.

T2CDELAY is used in master mode only. It defines a hold time for the slave device that delays the chip select deactivation by a multiple of ICLK cycles after the last bit is transferred. T2CDELAY can be configured between 1 and 32 ICLK cycles.

Figure 16. Example: $t_{T2CDELAY}$ = 4 ICLK cycles



The hold time value is calculated as shown in Equation 2..

Equation 2. Transmit-End-to-Chip-Select-Inactive-Delay-*Time*

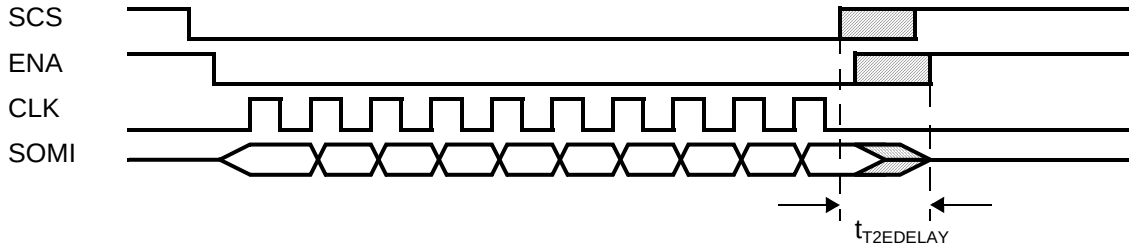
$$t_{T2CDELAY} = \frac{T2CDELAY}{ICLK} + 1$$

Example: ICLK = 25 MHz; T2CDELAY = 02h;
> $t_{T2CDELAY}$ = 120 ns;
After the last data bit (or parity bit) is being transferred the chip select signal is held active for 120 ns.

Bits 15:8 **T2EDELAY.** Transmit-data-finished-to-ENA-pin-inactive-time-out.

T2EDELAY is used in master mode only. It defines a time-out value as a multiple of SPI clock before the ENAble signal has to become inactive and after the CS becomes inactive. The SPI clock depends on which data format is selected. If the slave device is missing one or more clock edges, it's becoming de-synchronized (see Section 7.20). Although the master has finished the data transfer the slave is still waiting for the missed clock pulses and the ENA signal isn't disabled. The T2EDELAY defines a time-out value that triggers the DESYNC flag, if the ENA signal isn't deactivated in time.

Figure 17. Transmit-data-finished-to-ENA-inactive-time-out



The time-out value is calculated as following:

Equation 3. Transmit-data-finished-to-ENA-inactive-time-out Value

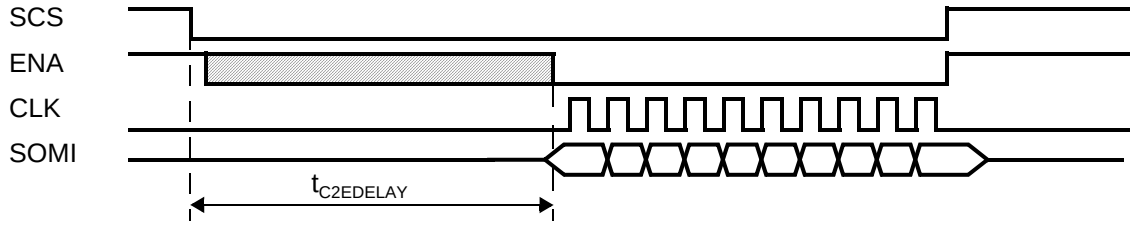
$$t_{T2EDELAY} = \frac{T2EDELAY}{SPIclock}$$

Example: SPIclock = 8 Mbit/s; T2EDELAY = 10h;
> $t_{T2EDELAY}$=2μs;
The slave device has to disable the ENA signal within 2μs, otherwise the SDESYNC flag is set and an interrupt is asserted if enabled.

Bits 7:0 **C2EDELAY.** Chip-select-active-to-ENA-signal-active-time-out

C2EDELAY is utilized only in master mode and it applies only if the addressed slave generates an ENA signal as a hardware handshake response. C2EDELAY defines the maximum time between the MibSPI activates the chip select signal and the addressed slave has to respond by activating the ENA signal. C2EDELAY defines a time-out value as a multiple of SPI clocks. The SPI clock depends on whether data format 0 or data format 1 is selected. If the slave device is not responding with the ENA signal before the time-out value is reached, the TIMEOUT flag in SPISTAT register is set and a interrupt is asserted if enabled. If a time-out occurs the MibSPI clears the transmit request of the timed-out buffer and continuous the transfer with the next buffer in the sequence that is enabled.

Figure 18. Chip-select-active-to-ENA-signal-active-time-out



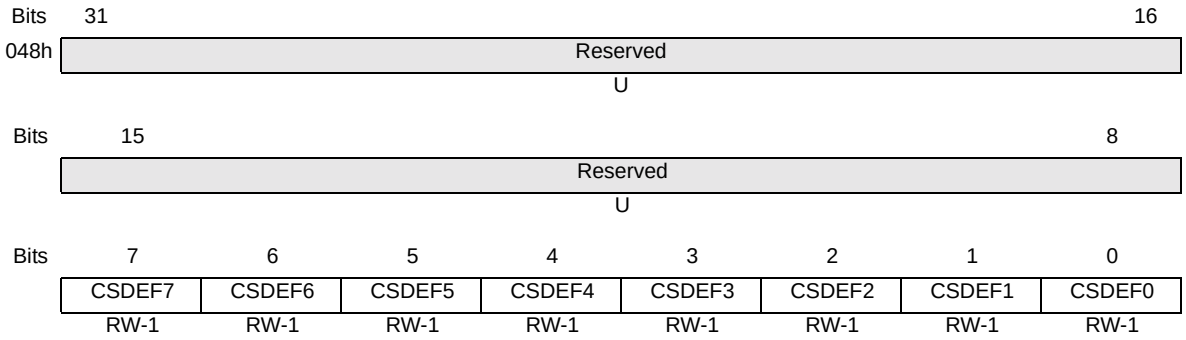
The time-out value is calculated as following:

Equation 4. Chip-select-active-to-ENA-signal-active-time-out Value

$$t_{C2EDELAY} = \frac{C2EDELAY}{SPIclock}$$

Example: SPIclock = 8 Mbit/s; C2EDELAY = 30h;
> $t_{C2EDELAY}$=6ms;
The slave device has to active the ENA signal within 6 ms after the MibSPI has activated the chip select signal (SCS), otherwise the TIMEOUT flag is set and a interrupt is asserted if enabled.

7.18 SPI Control Register 6 (SPICTRL6)



Legend: RW = Read/Write, U = Undefined, *-n* = Value after reset, x = indeterminate

Bits 31:8 **Reserved**

Bits 7:0 **CSDEFx.** Chip select default pattern.

Master mode behavior.

The CSDEFx register is output to the chip select pin when no transmission are currently performed. It allows the user to set a chip select pattern which deselect all the SPI slaves.

1 = If CSDEFx is set to “1” the corresponding chip select is set to “1” when no transfer occurs.

0 = If CSPOLx is set to “0” the corresponding chip select is set to “0” when no transfer occurs.

7.19 SPI Control Register 7 (SPICTRL7) (SPICTRL8, SPICTRL9)

The MibSPI supports 4 data format transmission. The data format specifies the number of bit, the clock speed, the phase, the polarity, and the direction of a transaction. This allows the MibSPI to communicate with different type of SPI Slave without reprogramming each time the MibSPI.

SPICTRL7 comprises the configuration bits for data format 1.

SPICTRL8 comprises the configuration bits for data format 2.

SPICTRL9 comprises the configuration bits for data format 3.

Bits	31	30	29	24	23	22	21	18	17	16
04Ch	Reserved		WDELAYx		SHIFT DIRx	PAR-POLx	Reserved		POLA RITYx	PHASEx
	U		RWP-0				U		RWP-0	RWP-0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	0
	WAIT ENAx	PARI-TYx	Reserved	PRESCALEx								CHARLENx	
	RWP-0	RWP-0	RWP-0	RWP-0								RWP-0	

Legend: RW = Read/Write in all modes, WP = Write in privilege mode only, U = Undefined, -*n* = Value after reset, x = indeterminate

Bits 31:30 **Reserved**

Bits 29:24 **WDELAYx.** Delay in between transmissions for data format x.

Idle time that will be applied at the end of the current transmission if the bit WDEL is set in the current buffer. (see section 2.5.2)

The delay to be applied is equal to: WDELAY * P_{ICLK} + 2 * P_{ICLK}.

Bit 23 **SHIFTDIRx.** Shift direction for data format x.

With bit SHIFTDIRx the shift direction for data format x (x=1,2,3) can be selected.

1 = Data format x shift direction: Least significant bit is shifted out first.

0 = Data format x shift direction: Most significant bit is shifted out first.

Bit 22 **PARPOLx.** Parity polarity: even or odd.

PARPOLx can be modified in privilege mode only. It can be used for data format x (x= 1,2,3).

1 = If PARPOLx is set to “1” and PARITYx is enabled, a odd parity flag is added at the end of the transmit data stream.

0 = If PARPOLx is set to “0” and PARITYx is enabled, a even parity flag is added at the end of the transmit data stream.

Bits 21:18 **Reserved**

Bit 15 **WAITENAx.** Master waits for ENA signal from slave for data format x.

WAITENA is considered in master mode only. In slave mode this bit has no meaning. WAITENA enables a flexible SPI network where slaves with ENA signal and slaves without ENA signal can be mixed. WAITENA defines for each buffer whether the addressed slave generates the ENA signal or does not.

1 = Before the MibSPI starts the data transfer it waits for the ENA signal to become low. If the ENA signal is not pulled down by the addressed slave before the internal time-out counter (CE2DELAY) expires.

0 = The MibSPI does not wait for the ENA signal from the slaves and directly starts the transfer.

Bit 16 **PARITYx.** Parity enable for data format x.

1 = A parity is added after transfer of the data bit. At the end of a transfer the parity generator compares the received parity bit with the locally calculated parity flag. If the parity bits do not match the RXERR flag is set in the corresponding control field. The parity type (even or odd) can be selected via the PARPOL bit.

0 = No parity generation/ verification is performed for this data format.

Bit 14 **PHASEx.** SPI Data format x clock delay.

PHASEx defines the clock delay of data format x. PHASEx can be modified in privilege mode only.

1 = If PHASEx is set to “1” the SPI clock signal is delayed by a half SPI clock cycle versus the transmit / receive data stream. The first transmit bit has to output prior to the first clock edge. Master and slave receive the first bit with the first edge

0 = If PHASEx is set to “0” the SPI clock signal is not delayed versus the transmit / receive data stream. The first data bit is transmitted with the first clock edge and the first bit is received with the second (inverse) clock edge.

Bit 13 **POLARITYx.** SPI data format x clock polarity.

POLARITYx defines the clock polarity of data format x. POLARITYx can be modified in privilege mode only.

1 = If POLARITYx is set to “1” the SPI clock signal is high-inactive, i.e. before and after data transfer the clock signal is high.

0 = If POLARITYx is set to “0” the SPI clock signal is low-inactive, i.e. before and after data transfer the clock signal is low.

Bits 12:5 **PRESCALEx.** SPI data format x prescaler.

PRESCALEx can be modified in privilege mode only.

PRESCALEx determines the bit transfer rate of data format x if the SPI is the network master. PRESCALEx is directly derived from ICLK. If the MibSPI is configured as slave PRESCALEx DOES NOT NEED to be configured.

The clock rate for data format x can be calculated as shown in Equation 5..

Equation 5. SPI clock rate for data format x

$$BR_{Formatx} = \frac{ICLCK}{(PRESCALEx + 1)}$$

When PRESCALEx is set to zero (0), the SPI clock rate is ICLK/2.

Bits 4:0 **CHARLENx.** SPI data format x data word length.

CHARLENx defines the word length of data format x. Legal values are 0x02 (data word length = 2 bit) to 0x10 (data word length = 16). Illegal values, such as 0x00 or 0x1F are not detected and their effect is indeterminate.

7.20 SPI Status Register (SPISTAT)

Bits	31			28	27	26	25	24
058h	Reserved				BITERRLVL	DESYNCLVL	PARERRLVL	TIMEOUTLVL
	U				RW-0	RW-0	RW-0	RW-0
Bits	23			20	19	18	17	16
	Reserved				BITERRENA	DESYNCENA	PARERRENA	TIMEOUTENA
	U				RW-0	RW-0	RW-0	RW-0
Bits	15			12	11	10	9	8
	Reserved				BITERR	DESYNC	PARITYERR	TIMEOUT
	U				RC-0	RC-0	RC-0	RC-0
Bits	7							0
	LCSNR							
	R-0							

Legend: R = Read, W = Write, C = Clear, U = Undefined, -*n* = Value after reset, x = indeterminate

Bits 31:28 Reserved

Bit 27 **BITERRLVL.** Bit error interrupt level.

1 = A bit error interrupt is mapped to interrupt line INT1.

0 = A bit error interrupt is mapped to interrupt line INT0.

Bit 26 **DESYNCLVL.** De-synchronized slave interrupt level.

DESYNCLVL is used in master mode only.

1 = An interrupt due to de-synchronization of the slave (DESYNC = 1) is mapped to interrupt line INT1.

0 = An interrupt due to de-synchronization of the slave (DESYNC = 1) is mapped to interrupt line INT0.

Bit 25 **PARERRLVL.** Parity error interrupt level.

1 = A parity error interrupt (PARITYERR = 1) is mapped to interrupt line INT1.

0 = A parity error interrupt (PARITYERR = 1) is mapped to interrupt line INT0.

Bit 24 **TIMEOUTLVL.** ENA signal time-out interrupt level.

1 = An interrupt on a time-out of the ENA signal (TIMEOUT = 1) is mapped to interrupt line INT1.

0 = An interrupt on a time-out of the ENA signal (TIMEOUT = 1) is mapped to interrupt line INT0.

Bits 23:20 **Reserved**

Bit 19 **BITERRENA.** Enables interrupt on bit error.

1 = Enables an interrupt on a bit error (BITERR = 1).

0 = No interrupt asserted upon bit error.

Bit 18 **DESYNCENA.** Enables interrupt on de-synchronized slave.

DESYNCENA is used in master mode only.

1 = Enables an interrupt on de-synchronization of the slave (DESYNC = 1).

0 = No interrupt asserted upon de-synchronization error.

Bit 17 **PARERRENA.** Enables interrupt on parity error.

1 = Enables an interrupt on a parity error (PARITYERR = 1).

0 = No interrupt asserted upon parity error.

Bit 16 **TIMEOUTENA.** Enables interrupt on ENA signal time-out.

1 = Enables an interrupt on a time-out of the ENA signal (TIMEOUT = 1).

0 = No interrupt asserted upon ENA signal time-out.

Bits 15:12 **Reserved**

Bit 11 **BITERR.** Mismatch of internal transmit data and transmitted data.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it.

1 = A bit error occurred. The MibSPI 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 Flag BITERR is set. If BITERRENA is set an interrupt is asserted. A possible reason for a bit error can be a to high bit rate / capacitive load or another master/slave trying to transmit at the same time.

0 = No bit error occurred.

Bit 10 **DESYNC.** De-synchronization of slave device.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it. De-synchronization monitor is active in master mode only.

1 = A slave device is de-synchronized. The master monitors the ENAble signal coming from the slave device and sets the DESYNC flag if ENA is deactivated before the last reception point or after the last bit is transmitted plus $t_{T2EDELAY}$ (see Section 7.17). If DESYNCENA is set an interrupt is asserted. De-synchronization can occur if a slave device misses a clock edge coming from the master or is detecting an additional clock edge due to perturbation.

0 = No slave de-synchronization detected.

Bit 9 **PARITYERR.** Calculated parity differs from received parity bit.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it.

1 = A parity error occurred. If the parity generator is enabled (can be selected individually for each buffer) an even or odd parity bit is added at the end of a data word (see Section 7.19, bit 10). During reception of the data word the parity generator calculates the reference parity and compares it to the received parity bit. In the event of a mismatch the PARITYERR flag is set and an interrupt is asserted if PARERRENA is set.

0 = No parity error detected.

Bit 8 **TIMEOUT.** Time-out due to non-activation of ENA signal.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it.

1 = An ENA signal time-out occurred. The MibSPI generates a time-out because the slave hasn't responded in time by activating the ENA signal after the chip select signal has been activated. If a time-out condition is detected the corresponding chip select is deactivated immediately and the TIMEOUT flag is set. In addition the TIMOUT flag in the status field of the corresponding buffer is set. The transmit request of the concerned buffer is cleared, i.e. the MibSPI doesn't restart a data transfer from this buffer.

0 = No ENA-signal time-out occurred.

Bits 7:0 **LCSNR.** Chip select with erroneous transfer (BITERR, DESYNC, PARITYERR or TIMEOUT).

These bits are read only and comprise valid data in master mode only. If a BITERR, a DESYNC, a PARITYERR or a TIMEOUT occurs the current chip select number is copied into LCSNR[7:0].

7.21 Transfer Group Interrupt Enable Register (TGINTENA)

The register TGINTENA comprises the transfer group interrupt enable flags for “transfer finished” and for “transfer suspended”. The number of implemented transfer groups is dependent on the MibSPI macro-cell. Each of the enable bits in the higher half-word and the lower half-word of TGINTENA belongs to one transfer group. In below register map a super-set MibSPI with 16 transfer groups is assumed. The real number of interrupt enable bits depends on the implemented number of transfer groups.

Bits	31	30	29	28	27	26	25	24
05Ch	INTENRDY15	INTENRDY14	INTENRDY15	INTENRDY12	INTENRDY11	INTENRDY10	INTENRDY9	INTENRDY8
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Bits	23	22	21	20	19	18	17	16
	INTENRDY7	INTENRDY6	INTENRDY5	INTENRDY4	INTENRDY3	INTENRDY2	INTENRDY1	INTENRDY0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Bits	15	14	13	12	11	10	9	8
	INTENSUS15	INTENSUS14	INTENSUS13	INTENSUS12	INTENSUS11	INTENSUS10	INTENSUS9	INTENSUS8
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Bits	7	6	5	4	3	2	1	0
	INTENSUS7	INTENSUS6	INTENSUS5	INTENSUS4	INTENSUS3	INTENSUS2	INTENSUS1	INTENSUS0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend: R = Read, W = Write, C = Clear, U = Undefined, -*n* = Value after reset, x = indeterminate

Bits 31:16 **INTENRDYx.** Transfer group interrupt enable when transfer finished

1 = A interrupt is asserted when the transfer from transfer group x has been finished.

0 = No interrupt at the end of a transfer group transfer.

Bits 15:0 **INTENSUSx.** Transfer group interrupt enable when transfer suspended

1 = A interrupt is asserted when a transfer from transfer group x is suspended.

0 = No interrupt due to suspending of a transfer group transfer.

7.22 Transfer Group Interrupt Level Register (TGINTLVL)

The register TGINTLVL comprises the transfer group interrupt level flags for “transfer finished” event and for “transfer suspended” event. With each of the flags the corresponding interrupt can be mapped to either INT0 interrupt line or INT1 interrupt line.

The number of implemented transfer groups is dependent on the MibSPI macro-cell. Each of the level bits in the higher half-word and the lower half-word of TGINTLVL belongs to one transfer group. In below register map a super-set MibSPI with 16 transfer groups is assumed. The real number of interrupt level bits depends on the implemented number of transfer groups.

Bits	31	30	29	28	27	26	25	24
060h	INTLVL RDY15	INTLVL RDY14	INTLVL RDY13	INTLVL RDY12	INTLVL RDY11	INTLVL RDY10	INTLVL RDY9	INTLVL RDY8
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
Bits	23	22	21	20	19	18	17	16
	INTLVLRDY7	INTLVLRDY6	INTLVLRDY5	INTLVLRDY4	INTLVLRDY3	INTLVLRDY2	INTLVLRDY1	INTLVLRDY0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
Bits	15	14	13	12	11	10	9	8
	INTLVL SUS15	INTLVL SUS14	INTLVL SUS13	INTLVL SUS12	INTLVL SUS11	INTLVL SUS10	INTLVL SUS9	INTLVL SUS8
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
Bits	7	6	5	4	3	2	1	0
	INTLVLSUS7	INTLVLSUS6	INTLVLSUS5	INTLVLSUS4	INTLVLSUS3	INTLVLSUS2	INTLVLSUS1	INTLVLSUS0
	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend: R = Read, W = Write, C = Clear, U = Undefined, -*n* = Value after reset, x = indeterminate

Bits 31:16 **INTLVLRDYx.** Transfer group interrupt level for “transfer finished” event.

1 = A transfer finished interrupt of the transfer group x is mapped to interrupt line INT1.

0 = A transfer finished interrupt of the transfer group x is mapped to interrupt line INT0.

Bits 15:0 **INTLVLSUSx.** Transfer group interrupt level for “transfer suspended” event.

1 = A transfer suspended interrupt of the transfer group x is mapped to interrupt line INT1.

0 = A transfer suspended interrupt of the transfer group x is mapped to interrupt line INT0.

7.23 Transfer Group Interrupt Flag Register (TGINTFLAG)

The register TGINTFLAG comprises the transfer group interrupt flags for “transfer finished” interrupts (INTFLGRDYx) and for “transfer suspended” interrupts (INTFLGSUSx). The number of implemented transfer groups is dependent on the MibSPI macro-cell. Each of the interrupt flags in the higher half-word and the lower half-word of TGINTFLAG belongs to one transfer group. In below register map a super-set MibSPI with 16 transfer groups is assumed. The real number of interrupt flags depends on the implemented number of transfer groups.

Bits	31	30	29	28	27	26	25	24
064h	INTFLG RDY15	INTFLG RDY14	INTFLG RDY13	INTFLG RDY12	INTFLG RDY11	INTFLG RDY10	INTFLG RDY9	INTFLG RDY8
	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0
Bits	23	22	21	20	19	18	17	16
	INTFLGRDY7	INTFLGRDY6	INTFLGRDY5	INTFLGRDY4	INTFLGRDY3	INTFLGRDY2	INTFLGRDY1	INTFLGRDY0
	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0
Bits	15	14	13	12	11	10	9	8
	INTFLG SUS15	INTFLG SUS14	INTFLG SUS13	INTFLG SUS12	INTFLG SUS11	INTFLG SUS10	INTFLG SUS9	INTFLG SUS8
	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0
Bits	7	6	5	4	3	2	1	0
	INTFLGSUS7	INTFLGSUS6	INTFLGSUS5	INTFLGSUS4	INTFLGSUS3	INTFLGSUS2	INTFLGSUS1	INTFLGSUS0
	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

Legend: R = Read, W = Write, C = Clear, U = Undefined, -*n* = Value after reset, x = indeterminate

Bits 31:16 **INTFLGRDYx.** Transfer group interrupt flag for “transfer finished” interrupt.

These flag are read/clear register.Reading the interrupt vector registers TGINTVECT0 or TGINTVECT1 is clearing automatically the referenced interrupt flag INTFLGRDYx.

1 = A “transfer finished” interrupt from transfer group x occurred. No matter whether the interrupt is enabled or disabled (INTENRDYx = don’t care) or whether the interrupt is mapped to line INT0 or INT1, INTFLGRDYx is set right after the transfer from transfer group x is finished.

Writing a one (1) in bit field will clear the corresponding bit flag.

0 = No “transfer finished” interrupt occurred since last clearing of the flag INTFLGRDYx.

Writing a zero (0) has no effect.

Bits 15:0 **INTFLGSUSx.** Transfer group interrupt flag for “transfer suspend” interrupt.

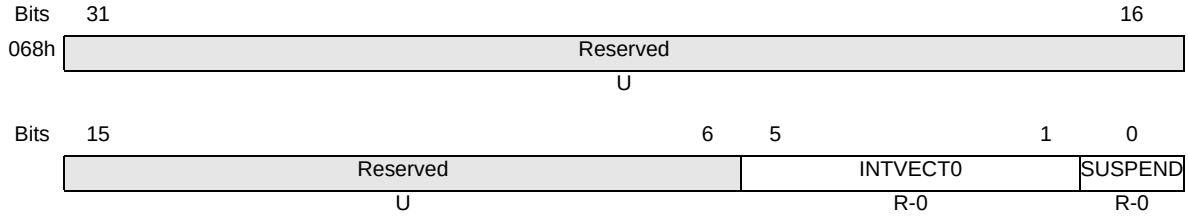
These flag are read/clear register.Reading the interrupt vector registers TGINTVECT0 or TGINTVECT1 is clearing automatically the referenced interrupt flag INTFLGSUSx.

1 = A “transfer suspended” interrupt from transfer group x occurred. No matter whether the interrupt is enabled or disabled (INTENSUSx = don’t care) or whether the interrupt is mapped to line INT0 or INT1, INTFLGSUSx is set right after the transfer from transfer group x is suspended.
Writing a one (1) in bit field will clear the corresponding bit flag.

0 = No ”transfer suspended” interrupt occurred since last clearing of the flag INTFLGSUSx.
Writing a zero (0) has no effect.

7.24 Transfer Group Interrupt Vector Register 0 (TGINTVECT0)

The register TGINTVECT0 returns a vector of the highest prior interrupt that is enabled for interrupt line INT0, no matter whether it is a “transfer finished” interrupt or a “transfer suspended” interrupt. The transfer group with the lowest number (transfer group 0) has the highest priority.



Legend: R = Read, W = Write, C = Clear, U = Undefined, -*n* = Value after reset, x = indeterminate

Bits 31:6 Reserved

Bits 5:1 INTVECT0. Interrupt vector for interrupt line INT0.

INTVECT0 returns the vector of the pending interrupt at interrupt line INT0. If more than one interrupt is pending, INTVECT0 always references the highest prior interrupt source first. The transfer group with the smallest number has the highest priority. Reading INTVECT0 automatically updates the TGINTVECT0 register with the interrupt vector coming next in the interrupt priority chain and the corresponding type (SUSPEND flag). Error interrupts have lowest priority.

Table 5. Interrupt Vector for Interrupt Line INT0

INTVECT0[5:1]	Description
00000b	no interrupt pending
00001b	Pending interrupt of transfer group 0. Refer to the flag SUSPEND to determine the interrupt type (transfer suspended, transfer finished).
00010b	Pending interrupt of transfer group 1. Refer to the flag SUSPEND to determine the interrupt type (transfer suspended, transfer finished).
00011b	Pending interrupt of transfer group 2. Refer to the flag SUSPEND to determine the interrupt type (transfer suspended, transfer finished).
00001b + x	Pending interrupt of transfer group x (x=0,..., 15). Refer to the flag SUSPEND to determine the interrupt type (transfer suspended, transfer finished).
10001b	Error interrupt pending. Refer to lower halfword of SPISTAT to determine more details about the type of error and the concerned buffer.
10010b ... 11111b	Reserved

Bits 0 **SUSPEND.** “Transfer suspended” or “transfer finished” interrupt.

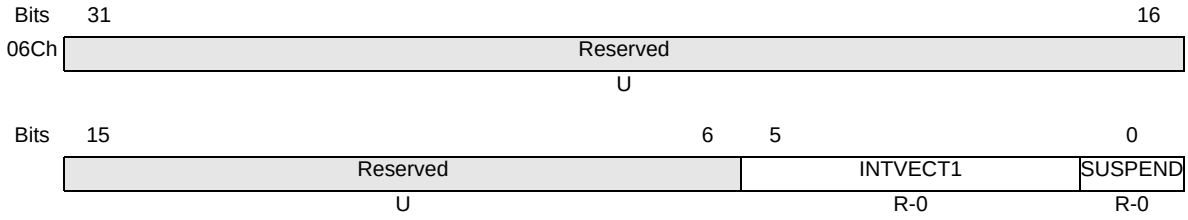
Every time TGINTVECT0 is read by the host the corresponding interrupt flag of the referenced transfer group is cleared and TGINTVECT0 is updated with the vector coming next in the priority chain. In parallel the SUSPEND flag is updated depending on the type of interrupt.

1 = The interrupt type is a “transfer suspended” interrupt. I.e. the transfer group referenced by INTVECT0 has asserted an interrupt, because the buffer to be transferred next is in “suspend to wait” mode.

0 = The interrupt type is a “transfer finished” interrupt. I.e. the buffer array referenced by INTVECT0 has asserted an interrupt, because all data from the whole transfer group has been transferred.

7.25 Transfer Group Interrupt Vector Register 1 (TGINTVECT1)

The register TGINTVECT1 returns a vector of the highest prior interrupt that is enabled for interrupt line INT1, no matter whether it is a “transfer finished” interrupt or a “transfer suspended” interrupt. The transfer group with the lowest number (transfer group 0) has the highest priority.



Legend: R = Read, W = Write, C = Clear, U = Undefined, -*n* = Value after reset, x = indeterminate

Bits 31:6 **Reserved**

Bits 5:1 **INTVECT1.** Interrupt vector for interrupt line INT1.

INTVECT1 returns the vector of the pending interrupt at interrupt line INT1. If more than one interrupt is pending, INTVECT1 always references the highest prior interrupt source first. The transfer group with the smallest number has the highest priority. Reading INTVECT1 automatically updates the TGINTVECT1 register with the interrupt vector coming next in the interrupt priority chain and the corresponding type (SUSPEND flag). Error interrupts have lowest priority.

Bits 0 **SUSPEND.** “Transfer suspended” or “transfer finished” interrupt.

Every time TGINTVECT1 is read by the host the corresponding interrupt flag of the referenced transfer group is cleared and TGINTVECT1 is updated with the vector coming next in the priority chain. In parallel the SUSPEND flag is updated depending on the type of interrupt.

1 = The interrupt type is a “transfer suspended” interrupt. I.e. the transfer group referenced by INTVECT1 has asserted an interrupt, because the buffer to be transferred next is in “suspend to wait” mode.

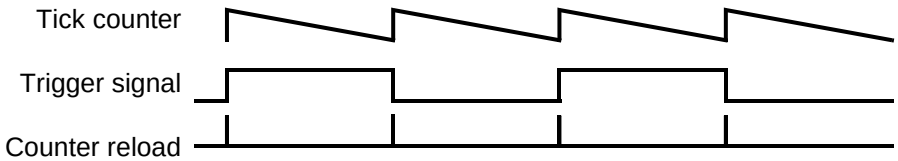
0 = The interrupt type is a “transfer finished” interrupt. I.e. the transfer group referenced by INTVECT1 has asserted an interrupt, because all data from the whole transfer group has been transferred.

Table 6. Interrupt Vector for Interrupt Line INT1

INTVECT1[5:1]	Description
00000b	no interrupt pending
00001b	Pending interrupt of transfer group 0. Refer to the flag SUSPEND to determine the interrupt type (transfer suspended, transfer finished).
00010b	Pending interrupt of transfer group 1. Refer to the flag SUSPEND to determine the interrupt type (transfer suspended, transfer finished).
00011b	Pending interrupt of transfer group 2. Refer to the flag SUSPEND to determine the interrupt type (transfer suspended, transfer finished).
00001b + x	Pending interrupt of transfer group x (x=0,..., 15). Refer to the flag SUSPEND to determine the interrupt type (transfer suspended, transfer finished).
10001b	Error interrupt pending. Refer to lower halfword of SPISTAT to determine more details about the type of error and the concerned buffer.
10010b ... 11111b	Reserved

7.26 Tick Count Register (TICKCNT)

One of the trigger sources for transfer groups is a MibSPI internal periodic time trigger. This time trigger is called tick counter and is basically a down-counter with a pre-load/reload value. Every time the tick counter detects an under-flow it reloads the initial value and toggles the trigger signal provided to the transfer groups.



The above shown trigger signal as a square wave signal support very well the different trigger event types for the transfer groups (e.g. falling edge, rising edge and both edges).

Bits	31	30	29	28	27	16
070h	TICKENA	RELOAD	CLKCTRL		Reserved	
	RW-0	R0S-0	RW-0		U	

Bits	15	0
	TICKVALUE	
	RW-0	

Legend: R = Read, R0 = always read 0, W = Write, S = Set, C = Clear, U = Undefined, *-n* = Value after reset, x = indeterminate

Bit 31 **TICKENA.** Tick counter enable.

1 = The MibSPI internal tick counter is enabled and is clocked by the clock source selected by CLKCTRL[1:0]. When the tick counter is enabled it starts down-counting from its current value. When TICKENA goes from “0” to “1” the tick counter is automatically loaded with the TICKVALUE.

0 = The MibSPI internal tick counter is disabled. The counter value remains unchanged.

Note:

When the tick counter is disabled the trigger signal is forced low.

Bit 30 **RELOAD.** Pre-load tick counter.

RELOAD is a set-only bit, i.e. writing a “1” to it automatically reloads the tick counter with the value stored in TICKICNT. Reading RELOAD always returns a “0”.

Note:

When the tick counter is reloaded by the RELOAD bit, the trigger signal is not toggled.

Bit 29:28 **CLKCTRL.** Tick counter clock source control.

CLKCTRL[1:0] defines the clock source that is used to clock the MibSPI internal tick counter.

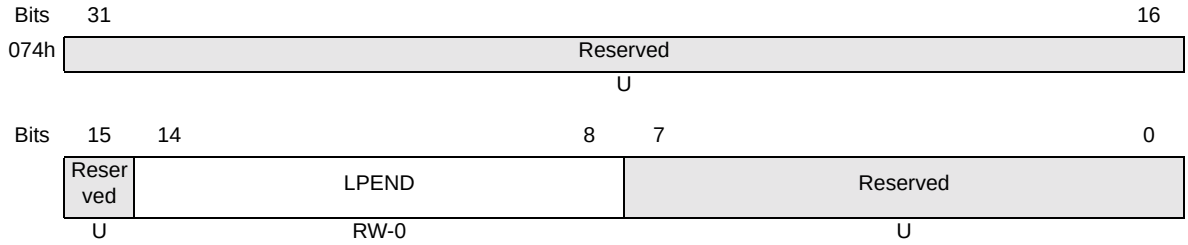
CLKCTRL[1:0]	Description
00b	SPICLK of Data word format 0 is selected as clock source of tick counter
01b	SPICLK of Data word format 1 is selected as clock source of tick counter
10b	SPICLK of Data word format 2 is selected as clock source of tick counter
11b	SPICLK of Data word format 3 is selected as clock source of tick counter

Bits 27:16 **Reserved**

Bits 15:0 **TICKVALUE.** Initial value for tick counter.

TICKVALUE stores the initial value for the tick counter. The tick counter is loaded with TICKVALUE every time an under-flow condition occurs and every time the PRELOAD flag is set by the host.

7.27 Last Transfer Group End Pointer (LTGPEND)



Legend: R = Read, W = Write, C = Clear, U = Undefined, *-n* = Value after reset, x = indeterminate

Bits 31:15 **Reserved**

Bits 14:8 **LPEND.** Last transfer group end pointer.

Usually the transfer groups end address (PEND) is inherently defined by the start value of the starting pointer of the subsequent transfer group (PSTART). The transfer group ends at the buffer one before the next transfer group starts (PEND[x]=PSTART[x+1] - 1). The 15th transfer group has no subsequent transfer group, i.e. no end address is inherently defined. Therefore LPEND has to be programmed to specify explicitly the end address of the 15th transfer group.

Note: LPEND allows SW compatibility for MibSPI variants

Due to software compatibility reasons the last transfer group end address has to be configurable, otherwise a super-set MibSPI can't emulate a MibSPI with less transfer groups without software changes.

Bits 7:0 **Reserved**

7.28 MibSPI Transfer Group Control Register (TGxCTRL)

The number of transfer groups is scalable by design. Depending on the implementation the number of transfer groups and hence the number of transfer group control register may vary. Each transfer group can be configured via one dedicated control register. The register description below shows one exemplary control register which is identical for all transfer groups. E.g. the control register for transfer group 2 is named “TG2CTRL” and is located at address *base0+78h+4*2*.

Bits	31	30	29	28	27	24	23	20	19	16
078h+ 4*X	TG ENA	ONE SHOT	PRST	TGTD	Reserved		TRGEVT		TRIGSRC	
X=0...15	RW-0	RW-0	RW-0	R-0	U		RW-0		RW-0	

Bits	15	14	8	7	6	0
	Reserved	PSTART		Reserved	PCURRENT	
X=0...15	U	RW-0		U	R-0	

Legend: R = Read, W = Write, C = Clear, U = Undefined, *-n* = Value after reset, x = indeterminate

Bits 31 **TGENA.** Transfer group enable.

1 = The corresponding transfer group is enabled. If the correct event (TRIGEVT) occurs at the selected source (TRIGSRC) a group transfer is initiated if no higher prior transfer group is in active transfer mode or if one or more higher prior transfer groups are in transfer suspend mode. If higher prior transfer groups are in transfer mode

0 = The corresponding transfer group is disabled. Disabling a transfer group while a transfer is ongoing, will finish the ongoing buffer transfer but not the whole group transfer.

Bits 30 **ONESHOT.** Single transfer group transfer.

1 = A transfer from the corresponding transfer group will be performed only once (= one shot) after a valid trigger event at the selected trigger source. After the transfer is finished, the TGENA control bit will be cleared by the MibSPI and therefore no additional transfer can be triggered before the host enables the transfer group again. This one shot mode ensures that after one group transfer the host has enough time to read the received data and to provide new transmit data.

0 = The corresponding transfer group initiates a transfer every time a trigger event occurs and TGENA is set.

Bits 29 **PRST.** Transfer group pointer reset mode.

With PRST the way of resolving trigger events during an ongoing transfer from the concerned transfer group can be configured.

1 = The corresponding transfer group pointer (PCURRENT) will be reset to the start address (PSTART) when a valid trigger event occurs at the selected trigger source while a transfer from the same transfer group is ongoing. I.e. every trigger event resets PCURRENT no matter whether the concerned transfer group is in transfer mode or not. The trigger events have priority over the ongoing transfer.

0 = If a trigger event occurs during a transfer from the concerned transfer group, the event is ignored and is not stored internally. The transfer group transfer has priority over additional trigger events.

Bits 28 **TGTD.** Transfer group triggered.

This bit is read-only.

1 = The transfer group has been triggered and is either currently service or waiting for servicing.

0 = The corresponding transfer group has not been triggered or is no more waiting for service.

Bits 27:24 **Reserved**

Bits 23:20 **TRIGEVT.** Type of trigger event.

After reset the trigger event types of all transfer groups are set to inactive.

Table 7. Trigger Event Types

TRIGEVT[3:0]	Type	Description
0000b	never	
0001b	rising edge	A rising edge (0>1) at the selected trigger source (TRIGSRC) initiates a transfer from the corresponding transfer group
0010b	falling edge	A falling edge (1>0) at the selected trigger source (TRIGSRC) initiates a transfer from the corresponding transfer group
0011b	both edges	Rising and falling edges at the selected trigger source (TRIGSRC) initiates a transfer from the corresponding transfer group
0100b	reserved	
0101b	high-active	Repetitive group transfer while trigger is high: While the selected trigger source (TRIGSRC) is at a logic high level (1) the group transfer is continued and at the end of one group transfer restarted at the beginning. If ONESHOT is set the transfer is performed only once. If the logic level changes to low (0) during an ongoing group transfer, the whole group transfer will be suspended.
0110b	low-active	Repetitive group transfer while trigger is low: While the selected trigger source (TRIGSRC) is at a logic low level (0) the group transfer is continued and at the end of one restarted at the beginning. If ONESHOT is set the transfer is performed only once. If the logic level changes to high (1) during an ongoing group transfer, the whole group transfer will be suspended.
0111b	always	A repetitive group transfer will be performed. Setting ONESHOT allows a software controlled single transfer mode*. If ONESHOT is cleared a continuous mode for this buffer is selected.
1xxxb	reserved	

*. By setting the TRIGSRC to 0000b, the TRIGEVT to ALWAYS (0111b) the ONESHOT bit to 1. This allows a software control trigger on this Transfer Group. Then by setting the TGENA bit, the Transfer Group is immediately triggered.

Bits 19:16 **TRIGSRC.** Trigger source.

After reset the trigger sources of all transfer groups are disabled.

Table 8. Trigger Sources

TRIGSRC[3:0]	Type	Description
0000b	disabled	
0001b	EXT0	MibSPI external trigger source 0. Source has to be defined individually for each Microcontroller derivative (e.g. HET I/O channel, event pin, etc.).
0010b	EXT1	MibSPI external trigger source 1. Source has to be defined individually for each Microcontroller derivative (e.g. HET I/O channel, event pin, etc.).
0011b	EXT2	MibSPI external trigger source 2. Source has to be defined individually for each Microcontroller derivative (e.g. HET I/O channel, event pin, etc.).
0100b	EXT3	MibSPI external trigger source 3. Source has to be defined individually for each Microcontroller derivative (e.g. HET I/O channel, event pin, etc.).
...	...	...
1110b	EXT13	MibSPI external trigger source 13. Source has to be defined individually for each Microcontroller derivative (e.g. HET I/O channel, event pin, etc.).
1111b	TICK	MibSPI internal periodic event trigger. The tick counter can initiate periodic group transfers.

Bits 15 **Reserved**

Bits 14:8 **PSTART.** Transfer group start address.

PSTART stores the start address of the corresponding transfer group. The corresponding end address is inherently defined by the subsequent transfer groups start address minus one (PEND[TGx] = PSTART[TGx+1]-1). PSTART is copied into PCURRENT when one of the following occurs:

- ❏ the transfer group is enabled
- ❏ the end of the transfer group is reached during a transfer
- ❏ a trigger event occurs and PRST is set to 1

Bits 7 **Reserved**

Bits 6:0 **PCURRENT.** Transfer group pointer to current buffer.

PCURRENT is read-only. PCURRENT stores the address (0...127) of the buffer that is currently transferred or that will be transferred after a trigger event occurs (if PRST=0) or after the transfer group resumes from suspend to wait mode.

If the transfer group switches mode from active transfer mode to suspend to wait, PCURRENT keeps the address of the next buffer. After the transfer group resumes from suspend to wait mode the next buffer will be transferred, i.e., no buffer data is multiply transferred or not at all transferred due to suspend to wait mode.

7.29 MibSPI DMA Channel Control Register (DMAxCTRL)

The number of bidirectional DMA channels (requires two physical DMA channels at the DMA controller: one for the transmit path and one for the receive path) is scalable by design. Depending on the implementation the number of DMA channels and hence the number of DMA channel control registers may vary. Each DMA channel can be configured via one dedicated control register. The register description below shows one exemplary control register which is identical for all DMA channels. For example, the control register for DMA channel 0 is named "DMA0CTRL". The MibSPI supports up to 8 bi-directional DMA channels.

Bits	31	30 – 24	23 – 20	19 – 16
0B8h+ 4*X	ONE SHOT	BUFID	RXDMACH	TXDMACH
X=0...7	RW-0	RW-0	RW-0	RW-0

Bits	15	14	13	12 – 8
	RXDMAENA	TXDMAENA	NOBRK	ICOUNT
X=0...7	RW-0	RW-0	RW-0	RW-0

Bits	7 – 6	5 – 0
	Reserved	COUNT
X=0...7	U	R-0

Legend: R = Read, W = Write, C = Clear, U = Undefined, -*n* = Value after reset, x = indeterminate

Bits 31 **ONESHOT.** Auto-disable of DMA channel after ICOUNT+1 transfers.

1 = ONESHOT allows a block transfer of defined length (ICOUNT+1) mainly controlled by the MibSPI and not by the DMA controller. After ICOUNT+1 transfers the enable bits RXDMAENA and TXDMAENA are automatically cleared by the MibSPI, hence no more DMA requests are generated. In conjunction with NOBRK a burst transfer can be initiated without any other transfer through another buffer.

0 = The length of the block transfer is fully controlled by the DMA controller. The enable bits RXDMAENA and TXDMAENA are not modified by the MibSPI.

Bits 30:24 **BUFID.** Buffer utilized for DMA transfer.

BUFID defines the buffer that is utilized for the DMA transfer. In order to synchronize the transfer with the DMA controller with the NOBRK condition the “suspend to wait until...” modes must be used (for more details refer to Section 7.30.1).

Bits 23:20 **RXDMACH.** Receive data DMA channel

Each MibSPI DMA channel can be linked to two physical DMA channels of the DMA controller. One channel for receive data and one channel for transmit data.

RXDMACH[3:0] defines the number of the physical DMA channel that is connected to the receive path of the MibSPI DMA channel.

If RXDMAENA and TXDMAENA are both set then RXDMACH[3:0] shall differ from TXDMACH[3:0] and shall differ from any other used physical DMA channel. Otherwise unexpected interference may occur.

Bits 19:16 **TXDMACH.** Transmit data DMA channel

Each MibSPI DMA channel can be linked to two physical DMA channels of the DMA controller. One channel for receive data and one channel for transmit data.

TXDMACH[3:0] defines the number of the physical DMA channel that is connected to the transmit path of the MibSPI DMA channel.

If RXDMAENA and TXDMAENA are both set then TXDMACH[3:0] shall differ from RXDMACH[3:0] and shall differ from any other used physical DMA channel. Otherwise unexpected interference may occur.

Bit 15 **RXDMAENA.** Receive data DMA channel enable.

1 = The physical DMA channel for the receive path is enabled. The first DMA request pulse is generated after the first transfer from the referenced buffer (BUFID) is finished. The concerned buffer should be configured in the mode “skip until RXEMPTY is set” or “suspend to wait until RXEMPTY is set” in order to ensure synchronization between DMA controller and MibSPI sequencer.

0 = No DMA request upon new receive data.

Bit 14 **TXDMAENA.** Transmit data DMA channel enable.

1 = The physical DMA channel for the transmit path is enabled. The first DMA request pulse is generated right after setting TXDMAENA to load the first transmit data. The concerned buffer should be configured in the mode “skip until TXFULL is set” or “suspend to wait until TXFULL is set” in order to ensure synchronization between DMA controller and MibSPI sequencer.

0 = No DMA request upon new transmit data.

Bit 13 **NOBRK.** non-interleaved DMA block transfer.

1 = NOBRK ensures that ICOUNT+1 data transfers are performed from the buffer referenced by BUFID without a data transfer from any other buffer. The sequencer remains at the DMA buffer until ICOUNT+1 transfers have been processed.
This can be used to generate a burst transfer to one device without disabling the chip select signal in-between (the concerned buffer has to be configured with CSHOLD=1). Another example would be to have a defined block data transfer in slave mode, synchronous to the master SPI.
Triggering of higher prior transfer groups or enabling of higher prior DMA channels is not able to interrupt NOBRK block transfer.

0 = The DMA transfers through the buffer referenced by BUFID are interleaved by data transfers from other active buffers or transfer groups. Every time the sequencer checks the DMA buffer it performs one transfer and then steps to the next buffer.

Bits 12:8 **ICOUNT.** Initial number of DMA transfers

ICOUNT[4:0] is used to preset the transfer counter COUNT[4:0]. Every time COUNT[4:0] hits zero it is reloaded with ICOUNT[4:0]. The real number of transfer equals ICOUNT[4:0] plus one.

If ONESHOT is set, ICOUNT[4:0] defines the number of DMA transfers that are performed before the MibSPI automatically disables the DMA channels. If NOBRK is set, ICOUNT[4:0] defines the number of DMA transfers that are performed in one sequence without a transfer from any other buffer.

Bits 7:5 **Reserved.**

Bits 4:0 **COUNT.** Actual number of remaining DMA transfer

COUNT[4:0] is a read-only bit field. It comprises the actual number of DMA transfers that remain, until the DMA channel is disabled if ONESHOT is set.

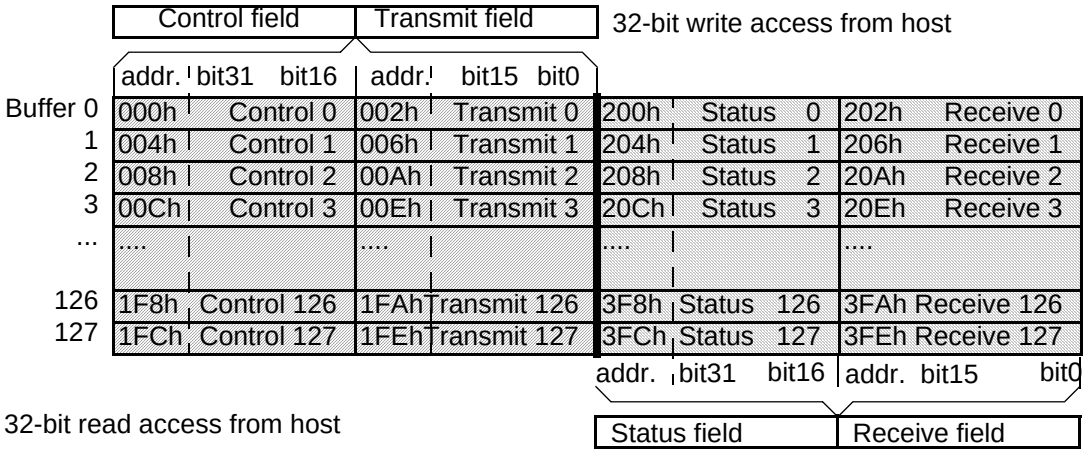
Note:

In the case of TX and RX DMA request are enabled, the COUNT register will be decremented when RX has been serviced.

7.30 Multi-buffer RAM

The multi-buffer RAM comprises all buffers, which can be configured identically. The buffers can be partitioned into multiple transfer groups, each containing a variable number of buffers. Each of the buffers can be sub-divided into a 16-bit transmit field, a 16-bit receive field, a 16-bit control field and a 16-bit status field, as displayed in Figure 19.

Figure 19. Multi-buffer RAM Configuration



The MibSPI RAM location is set by one of the memory chip select within the decoder. The corresponding SMCR register should be programmed to match the number of wait-state required (see section 7.2). The different fields can be read and written 8-bit, 16-bit or 32-bit wide.

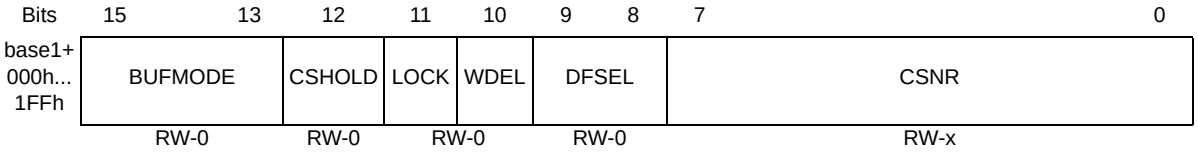
The transmit fields can be written and read in the address range 000h to 1FDh. The control field array can be written and read in the address range 002h to 1FFh, i.e. the transmit fields are interleaved with the control fields. When performing a 32-bit access the transmit field and the control field can be accessed in parallel. The low half-word (xxxx xx00b) is allocated by the transmit field and the high half-word (xxxx xx10b) is allocated by the corresponding control field.

The receive fields are read-only and can be accessed through the address range 202h to 3FFh. The corresponding status fields are mapped into the same address range (200h to 3FDh), i.e. the receive fields are interleaved with the status fields. When performing a 32-bit read from a receive buffer, the high half-word contains the corresponding status information and the low half-word comprises the receive data.

The chip select number bit field CSNR[5:0] of the control field is mirrored into the status field after transmission.

The actual size of the multi-buffer RAM is implementation dependent. E.g. an implementation of 16 buffers would consume 4*16*2 bytes = 128 bytes.

7.30.1 Control Field



Legend: R = Read, W = Write, U = Undefined, *-n* = Value after reset, x = indeterminate

Bits 15:13 **BUFMODE** Buffer sequencing mode.

The three BUFMODE bits define the sequencing mode of the corresponding buffer, i.e. the sequencer knows due to the BUFMODE setting what conditions have to be met that a data transfer is initiated from this buffer. The meaning of these bits is summarized in Table 9.

Table 9. Buffer Mode Selection Bits BUFMODE[2:0]

BUFMOD2	BUFMOD1	BUFMOD0	Description
0	0	0	The corresponding buffer is disabled
0	0	1	The sequencer skips the buffer until the corresponding TXFULL flag is set, i.e. new transmit data available. (single transfer mode)
0	1	0	The sequencer skips the buffer until the corresponding RXEMPTY flag is set, i.e. new receive data can be stored in RXDATA without data loss (overwrite-protect mode)
0	1	1	The sequencer skips the buffer until the corresponding TXFULL flag and RXEMPTY flag are set, i.e. new transmit data available and previous receive data consumed (read) by the host.
1	0	0	Sequencer initiates a transfer every time it checks this buffer (continuous mode) Data transmit are retransmitted if it has not been updated, data receive is overwritten even if it has not been read.
1	0	1	Sequencer suspends to wait until the corresponding TXFULL flag is set, i.e. the sequencer stops at the current buffer until new transmit data is written into the TXDATA field
1	1	0	Sequencer suspends to wait until the corresponding RXEMPTY flag is set, i.e. the sequencer stops at the current buffer until the previously received data is consumed (read) by the host.
1	1	1	Sequencer suspends to wait until the corresponding TXFULL flag and RXEMPTY flags are set, i.e. the sequencer stops at the current buffer until new transmit data is written into the TXDATA field and the previously received data is consumed (read) by the host.

When one of the “skip” modes is selected, the sequencer checks the buffer status every time it comes along. If the current buffer status (TXFULL, RXEMPTY) does not match the buffer is skipped without data transfer.

When one of the “suspend” modes is selected, the sequencer checks the buffer status and if it doesn’t match it waits until it matches. I.e. no data transfer is initiated until the status condition of the concerned buffer matches.

Bit 12 **CSHOLD.** Chip select hold mode

CSHOLD is considered in master mode only. In slave mode this bit has no meaning. CSHOLD defines the behavior of the chip select line at the end of a data transfer.

1 = The chip select signal is held active at the end of a transfer until a control field with new chip select information is loaded into the MibSPI kernel. If the new chip select information equals the previous one the active chip select signals is extended until end of a transfer with CSHOLD cleared or until the chip select information changes.
If CSHOLD is set in the control field of the last buffer to be transferred, then the corresponding chip select signal stays active until a new transfer is initiated.

0 = The chip select signal is deactivated at the end of a transfer after the T2CDELAY time has passed. If two transfers are dedicated to the same chip select the appropriate chip select signal will be shortly deactivated before it is activated again.

Bit 11 **LOCK.** Lock two consecutive buffers to be transmit without being interrupted by a transfer group that has higher priority.

1 = The transfer of this buffer and the buffer located at the next address is not interruptible.

0 = Any higher interruption could occurs after the end of the current transaction.

Bit 10 **WDEL.** Enable the delay counter at the end of the current transaction.

1 = after the transaction WDELAY of the corresponding data format is loaded into the delay counter. No transaction is performed until the counter is reset.

0 = No delay is inserted.

Bit 9:8 **DFSEL.** Data word format select.

DFSEL1	DFSEL0	Description
0	0	Data word format 0 is selected (see section 7.3 and section 7.4) for this buffer
0	1	Data word format 1 is selected (see section 7.19) for this buffer
1	0	Data word format 2 is selected (see section 7.19) for this buffer
1	1	Data word format 3 is selected (see section 7.19) for this buffer

Bits 7:0 **CSNR** Chip select number.

CSNR defines the chip select that shall be activated during a data transfer from the corresponding buffer. The chip value contain in CSNR register is directly output to the external pins.

7.30.2 Transmit Field

Bits	15	0
base1+ 000h... 1FFh	TXDATA	
	RW-x	

Legend: R = Read, W = Write, C = Clear, U = Undefined, -*n* = Value after reset, x = indeterminate

Bits 15:0 **TXDATA.** Transmit data field.

Transmit data has to be written into TXDATA right-aligned. 16-bit and 8-bit read/write access is supported. In the event of a 32-bit access the low half-word addresses the transmit field and the high half-word addresses the corresponding control field.

7.30.3 Status Field

Bits	15	14	13	12	11	10	9	8
base1+ 200h... 3FFh	RXEMPTY	RXOVR	TXFULL	BITERR	DESYNC	PARITYERR	TIMEOUT	Reserved
	R-1	RC-0	R-0	RC-0	RC-0	RC-0	RC-0	U

Bits	7	0
base1+ 200h... 3FFh	LCSNR	
	R-0	

Legend: RC = Read with auto-clear, U = Undefined, -*n* = Value after reset, x = indeterminate

Bit 15 **RXEMPTY.** Receive data buffer empty.

This flag is read only flag. When host reads the corresponding RXDATA field this will automatically set the RXEMPTY flag. When a data transfer has been finished and the received data is copied into the corresponding RXDATA field the RXEMPTY flag is cleared.

1 = No data received since last reading of the corresponding RXDATA field or RX Status field.

0 = Data is received in the corresponding RXDATA field.

Bit 14 **RXOVR.** Receive data buffer overrun.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it. When a data transfer has been finished and the received data is copied into the corresponding RXDATA field while RXEMPTY flag is already cleared RXOVR is set.

1 = A receive data overrun condition occurred since last time reading the status field.

0 = No receive data overrun condition occurred since last time reading the status field

Bit 13 **TXFULL.** Transmit data buffer full.

This flag is read only flag. Writing into the corresponding TXDATA field will automatically set the TXFULL flag. After a SPI transfer of the transmit data the TXFULL flag is cleared

1 = Host provided new transmit data to corresponding TXDATA field.

0 = No new transmit data from host since previous transfer of transmit data.

Bit 12 **BITERR.** Mismatch of internal transmit data and transmitted data.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it. It represents a copy of the BITERR flag in SPISTAT.

1 = A bit error occurred. The MibSPI 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 flag BITERR is set. A possible reason for a bit error can be noise, a to high bit rate / capacitive load or another master/slave trying to transmit at the same time.

0 = No bit error occurred.

Bit 11 **DESYNC.** De-synchronization of slave device.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it. De-synchronization monitor is active in master mode only. DESYNC represents a copy of the DESYNC flag in SPISTAT.

1 = A slave device is de-synchronized. The master monitors the ENA signal coming from the slave device and sets the DESYNC flag if ENA is deactivated before the last reception point or after the last bit is transmitted plus $t_{T2EDELAY}$ (see Section 7.17). If DESYNCENA is set an interrupt is asserted. De-synchronization can occur if a slave device misses a clock edge coming from the master or is detecting an additional clock edge due to perturbation.

0 = No slave de-synchronization detected.

Bit 10 **PARITYERR.** Calculated parity differs from received parity bit.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it. It represents a copy of the PARITYERR flag in SPISTAT.

1 = A parity error occurred. If the parity generator is enabled (can be selected individually for each buffer) an even or odd parity bit is added at the end of a data word (see Section 7.19). During reception of the data word the parity generator calculates the reference parity and compares it to the received parity bit. In the event of a mismatch the PARITYERR flag in the SPISTAT register and in the status field of the involved buffer is set.

0 = No parity error detected.

Bit 9 **TIMEOUT.** Time-out due to non-activation of ENA signal.

This flag is read/clear only flag, i.e. reading the flag will automatically clear it.

1 = An ENA signal time-out occurred. The MibSPI generates a time-out because the slave hasn't responded in time by activating the ENA signal after the chip select signal has been activated. If a time-out condition is detected the corresponding chip select is deactivated immediately and the TIMEOUT flag is set. In addition the TIMOUT flag in the status field of the corresponding buffer is set. The transmit request of the concerned buffer is cleared, i.e. the MibSPI doesn't restart a data transfer from this buffer.

0 = No ENA-signal time-out occurred.

Bit 8 **Reserved.**

Bits 7:0 **LCSNR.** Last chip select number.

LCSNR in the status field is a copy of CSNR in the corresponding control field. It defines the chip select that has been activated during the last data transfer from the corresponding buffer.

LCSNR is copied from the Kernel MibSPI after transmission during write back of received data.

7.30.4 Receive Field

Bits	15	0
base1+ 200h... 3FFh	RXDATA	
	R-x	

R = Read, W = Write, C = Clear, U = Undefined, *-n* = Value after reset, x = indeterminate

Bits 15:0 **RXDATA.** Receive data field.

Receive data can be read through RXDATA. The received data is automatically stored right-aligned into RXDATA, independent from the selected data word length. 16-bit and 8-bit accesses are supported. In the event of a 32-bit read access the low half-word contains the receive data and the high half-word contains the corresponding status field.