41. Serial Peripheral Interface (SPI)

41.1 Description

The Serial Peripheral Interface (SPI) circuit is a synchronous serial data link that provides communication with external devices in Master or Slave mode. It also enables communication between processors if an external processor is connected to the system.

The Serial Peripheral Interface is essentially a Shift register that serially transmits data bits to other SPIs. During a data transfer, one SPI system acts as the “master”' which controls the data flow, while the other devices act as “slaves'' which have data shifted into and out by the master. Different CPUs can take turn being masters (multiple master protocol, contrary to single master protocol where one CPU is always the master while all of the others are always slaves). One master can simultaneously shift data into multiple slaves. However, only one slave can drive its output to write data back to the master at any given time.

A slave device is selected when the master asserts its NSS signal. If multiple slave devices exist, the master generates a separate slave select signal for each slave (NPCS).

The SPI system consists of two data lines and two control lines:

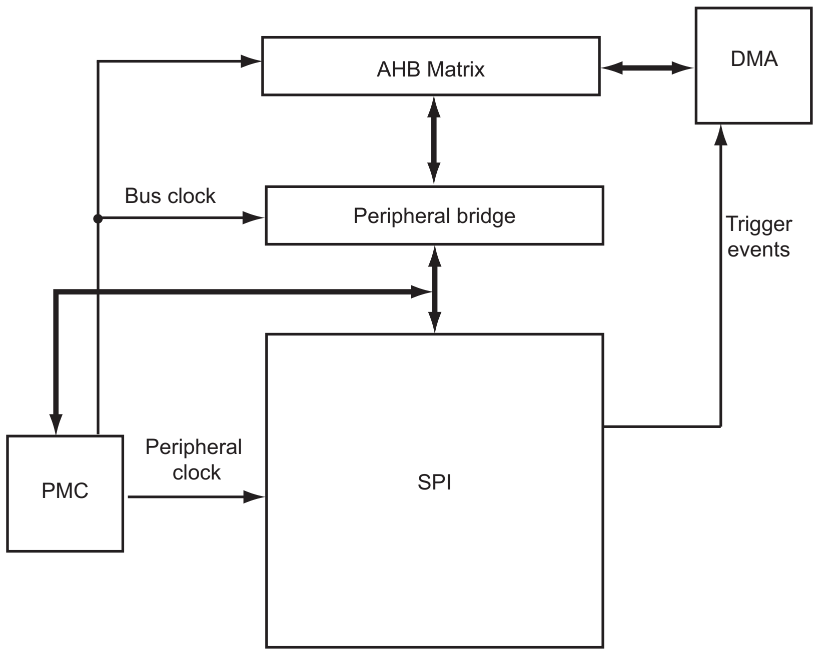
- Master Out Slave In (MOSI)—This data line supplies the output data from the master shifted into the input(s) of the slave(s).
- Master In Slave Out (MISO)—This data line supplies the output data from a slave to the input of the master. There may be no more than one slave transmitting data during any particular transfer.
- Serial Clock (SPCK)—This control line is driven by the master and regulates the flow of the data bits. The master can transmit data at a variety of baud rates; there is one SPCK pulse for each bit that is transmitted.
- Slave Select (NSS)—This control line allows slaves to be turned on and off by hardware.

41.2 Embedded Characteristics

- Master or Slave Serial Peripheral Bus Interface
  - 8-bit to 16-bit programmable data length per chip select
  - Programmable phase and polarity per chip select
  - Programmable transfer delay between consecutive transfers and delay before SPI clock per chip select
  - Programmable delay between chip selects
  - Selectable mode fault detection
- Master Mode can drive SPCK up to Peripheral Clock
- Master Mode Bit Rate can be Independent of the Processor/Peripheral Clock
- Slave mode operates on SPCK, asynchronously with core and bus clock
- Four chip selects with external decoder support allow communication with up to 15 peripherals
- Communication with Serial External Devices Supported
  - Serial memories, such as DataFlash and 3-wire EEPROMs
  - Serial peripherals, such as ADCs, DACs, LCD controllers, CAN controllers and sensors
  - External coprocessors
- Connection to DMA Channel Capabilities, Optimizing Data Transfers
  - One channel for the receiver
  - One channel for the transmitter
- Register Write Protection

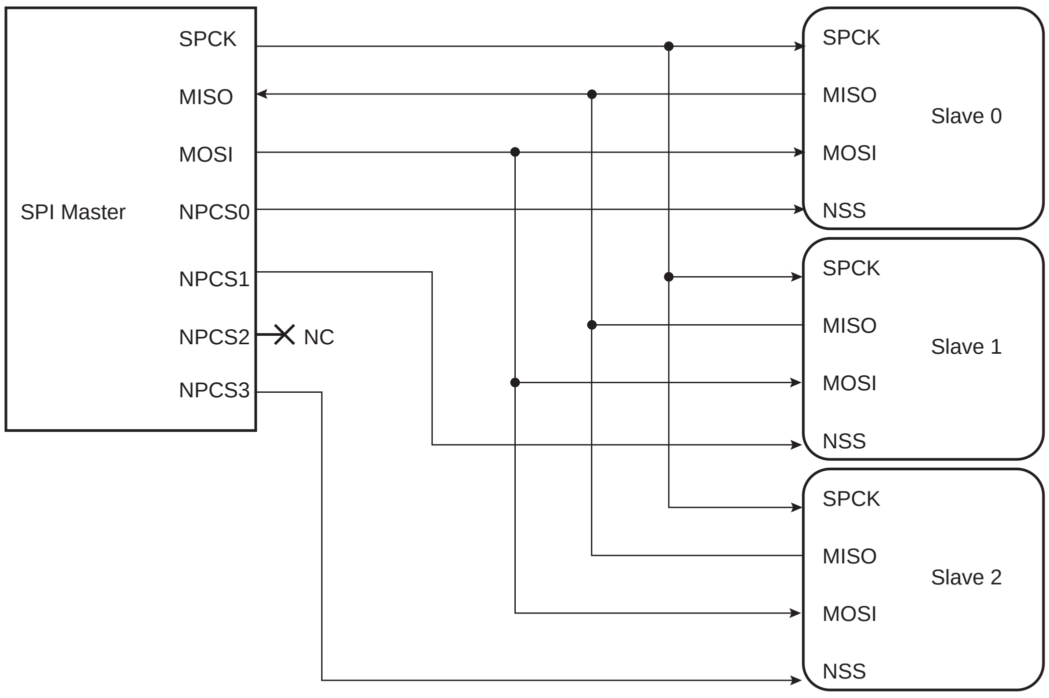
41.3 Block Diagram

Figure 41-1. Block Diagram



41.4 Application Block Diagram

Figure 41-2. Application Block Diagram: Single Master/Multiple Slave Implementation



41.5 Signal Description

Table 41-1. Signal Description

Pin Name	Pin Description	Type	
		Master	Slave
MISO	Master In Slave Out	Input	Output
MOSI	Master Out Slave In	Output	Input
SPCK	Serial Clock	Output	Input
NPCS1–NPCS3	Peripheral Chip Selects	Output	Unused
NPCS0/NSS	Peripheral Chip Select/Slave Select	Output	Input

41.6 Product Dependencies

41.6.1 I/O Lines

The pins used for interfacing the compliant external devices can be multiplexed with PIO lines. The programmer must first program the PIO controllers to assign the SPI pins to their peripheral functions.

Table 41-2. I/O Lines

Instance	Signal	I/O Line	Peripheral
SPI0	SPI0_MISO	PD20	B
SPI0	SPI0_MOSI	PD21	B
SPI0	SPI0_NPCS0	PB2	D
SPI0	SPI0_NPCS1	PA31	A
SPI0	SPI0_NPCS1	PD25	B
SPI0	SPI0_NPCS2	PD12	C
SPI0	SPI0_NPCS3	PD27	B
SPI0	SPI0_SPCK	PD22	B
SPI1	SPI1_MISO	PC26	C
SPI1	SPI1_MOSI	PC27	C
SPI1	SPI1_NPCS0	PC25	C
SPI1	SPI1_NPCS1	PC28	C
SPI1	SPI1_NPCS1	PD0	C
SPI1	SPI1_NPCS2	PC29	C
SPI1	SPI1_NPCS2	PD1	C
SPI1	SPI1_NPCS3	PC30	C
SPI1	SPI1_NPCS3	PD2	C
SPI1	SPI1_SPCK	PC24	C

41.6.2 Power Management

The SPI can be clocked through the Power Management Controller (PMC), thus the programmer must first configure the PMC to enable the SPI clock.

41.6.3 Interrupt

The SPI interface has an interrupt line connected to the interrupt controller. Handling the SPI interrupt requires programming the interrupt controller before configuring the SPI.

Table 41-3. Peripheral IDs

Instance	ID
SPI0	21
SPI1	42

41.6.4 Direct Memory Access Controller (DMAC)

The SPI interface can be used in conjunction with the DMAC in order to reduce processor overhead. For a full description of the DMAC, refer to the relevant section.

41.7 Functional Description

41.7.1 Modes of Operation

The SPI operates in Master mode or in Slave mode.

- The SPI operates in Master mode by setting the MSTR bit in the SPI Mode Register (SPI_MR):
  - Pins NPCS0 to NPCS3 are all configured as outputs
  - The SPCK pin is driven
  - The MISO line is wired on the receiver input
  - The MOSI line is driven as an output by the transmitter.
- The SPI operates in Slave mode if the MSTR bit in the SPI_MR is written to ‘0’:
  - The MISO line is driven by the transmitter output
  - The MOSI line is wired on the receiver input
  - The SPCK pin is driven by the transmitter to synchronize the receiver.
  - The NPCS0 pin becomes an input, and is used as a slave select signal (NSS)
  - NPCS1 to NPCS3 are not driven and can be used for other purposes.

The data transfers are identically programmable for both modes of operation. The baud rate generator is activated only in Master mode.

41.7.2 Data Transfer

Four combinations of polarity and phase are available for data transfers. The clock polarity is programmed with the CPOL bit in the SPI chip select registers (SPI_CSRx). The clock phase is programmed with the NCPHA bit. These two parameters determine the edges of the clock signal on which data is driven and sampled. Each of the two parameters has two possible states, resulting in four possible combinations that are incompatible with one another. Consequently, a master/slave pair must use the same parameter pair values to communicate. If multiple slaves are connected and require different configurations, the master must reconfigure itself each time it needs to communicate with a different slave.

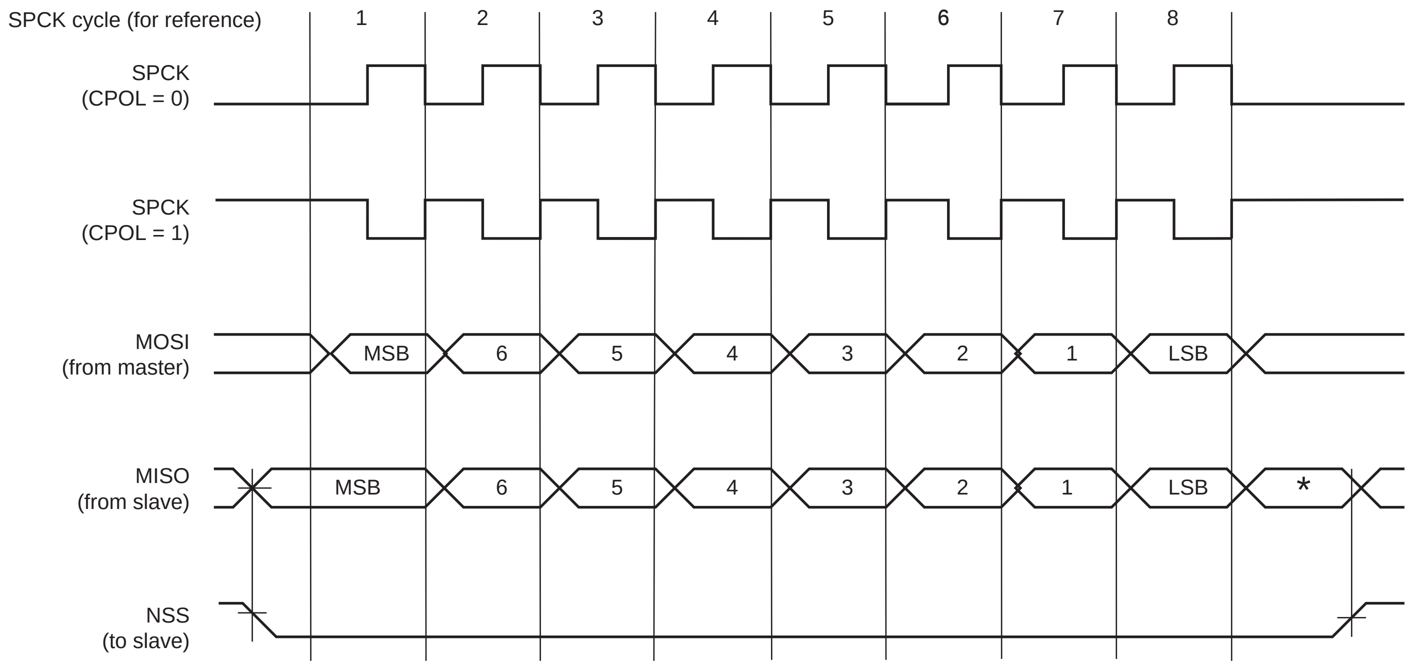
Table 41-4 shows the four modes and corresponding parameter settings.

Table 41-4. SPI Bus Protocol Modes

SPI Mode	CPOL	NCPHA	Shift SPCK Edge	Capture SPCK Edge	SPCK Inactive Level
0	0	1	Falling	Rising	Low
1	0	0	Rising	Falling	Low
2	1	1	Rising	Falling	High
3	1	0	Falling	Rising	High

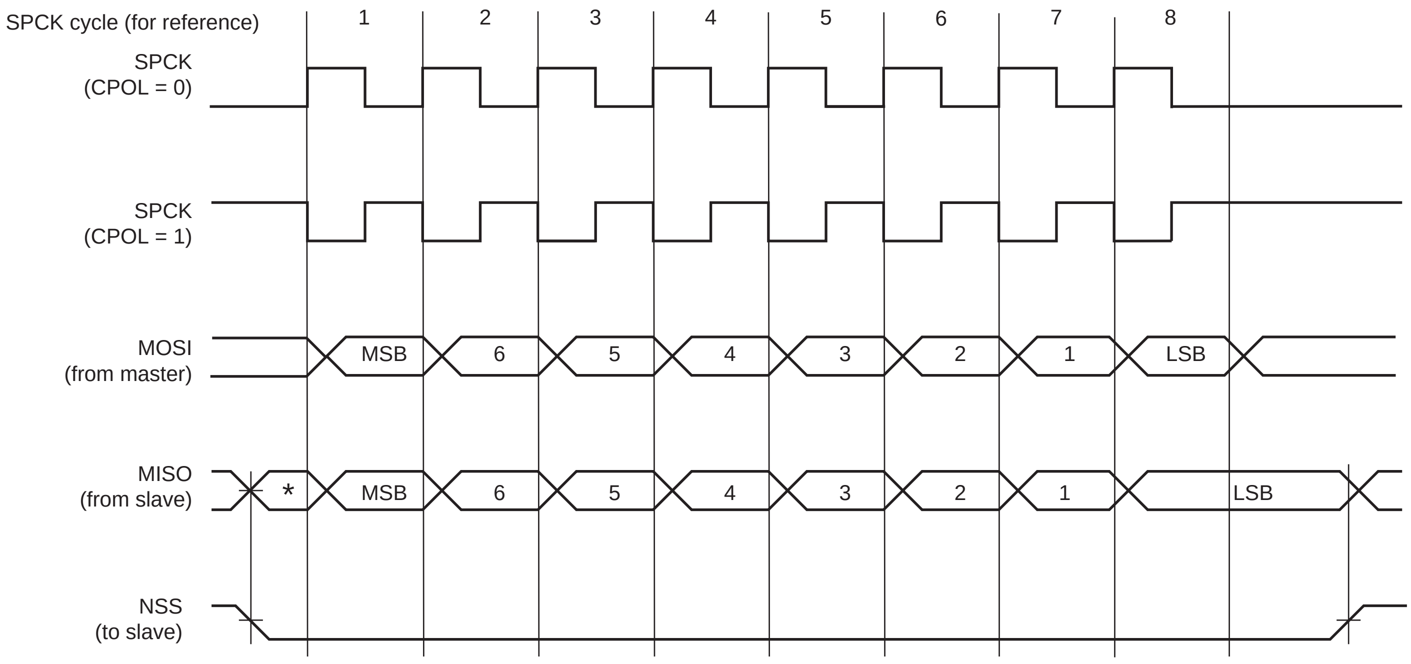
Figure 41-3 and Figure 41-4 show examples of data transfers.

Figure 41-3. SPI Transfer Format (NCPHA = 1, 8 bits per transfer)



* Not defined.

Figure 41-4. SPI Transfer Format (NCPHA = 0, 8 bits per transfer)



* Not defined.

41.7.3 Master Mode Operations

When configured in Master mode, the SPI operates on the clock generated by the internal programmable baud rate generator. It fully controls the data transfers to and from the slave(s) connected to the SPI bus. The SPI drives the chip select line to the slave and the serial clock signal (SPCK).

The SPI features two holding registers, the Transmit Data Register (SPI_TDR) and the Receive Data Register (SPI_RDR), and a single shift register. The holding registers maintain the data flow at a constant rate.

After enabling the SPI, a data transfer starts when the processor writes to the SPI_TDR. The written data is immediately transferred in the Shift register and the transfer on the SPI bus starts. While the data in the Shift register is shifted on the MOSI line, the MISO line is sampled and shifted in the Shift register. Data cannot be loaded in the SPI_RDR without transmitting data. If there is no data to transmit, dummy data can be used (SPI_TDR filled with ones). If the SPI_MR.WDRBT bit is set, transmission can occur only if the SPI_RDR has been read. If Receiving mode is not required, for example when communicating with a slave receiver only (such as an LCD), the receive status flags in the SPI Status register (SPI_SR) can be discarded.

Before writing the SPI_TDR, the PCS field in the SPI_MR must be set in order to select a slave.

If new data is written in the SPI_TDR during the transfer, it is kept in the SPI_TDR until the current transfer is completed. Then, the received data is transferred from the Shift register to the SPI_RDR, the data in the SPI_TDR is loaded in the Shift register and a new transfer starts.

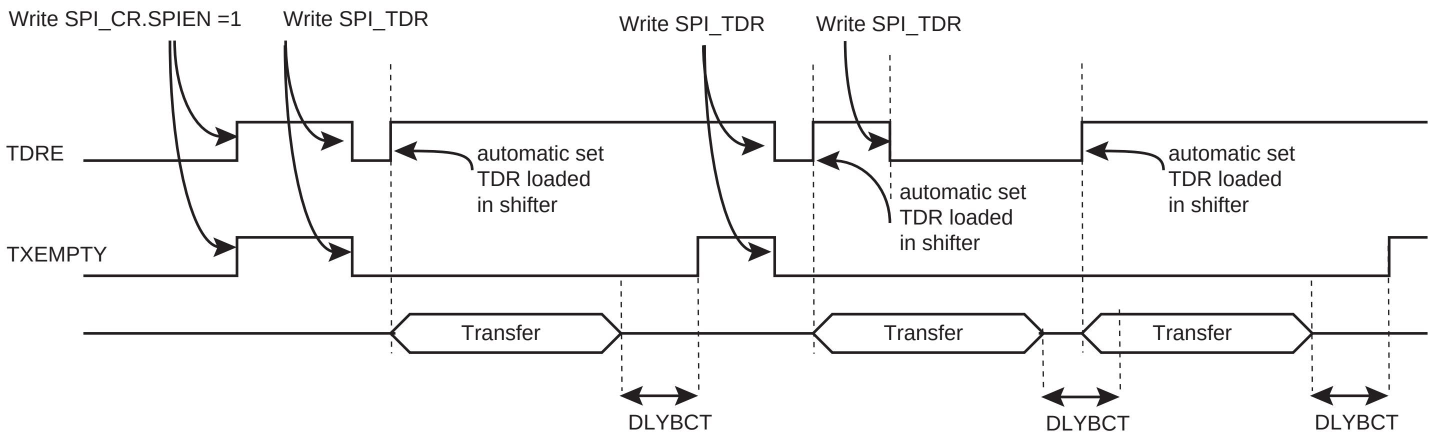
As soon as the SPI_TDR is written, the Transmit Data Register Empty (TDRE) flag in the SPI_SR is cleared. When the data written in the SPI_TDR is loaded into the Shift register, the TDRE flag in the SPI_SR is set. The TDRE bit is used to trigger the Transmit DMA channel.

See Figure 41-5.

The end of transfer is indicated by the TXEMPTY flag in the SPI_SR. If a transfer delay (DLYBCT) is greater than 0 for the last transfer, TXEMPTY is set after the completion of this delay. The peripheral clock can be switched off at this time.

Note: When the SPI is enabled, the TDRE and TXEMPTY flags are set.

Figure 41-5. TDRE and TXEMPTY flag behavior



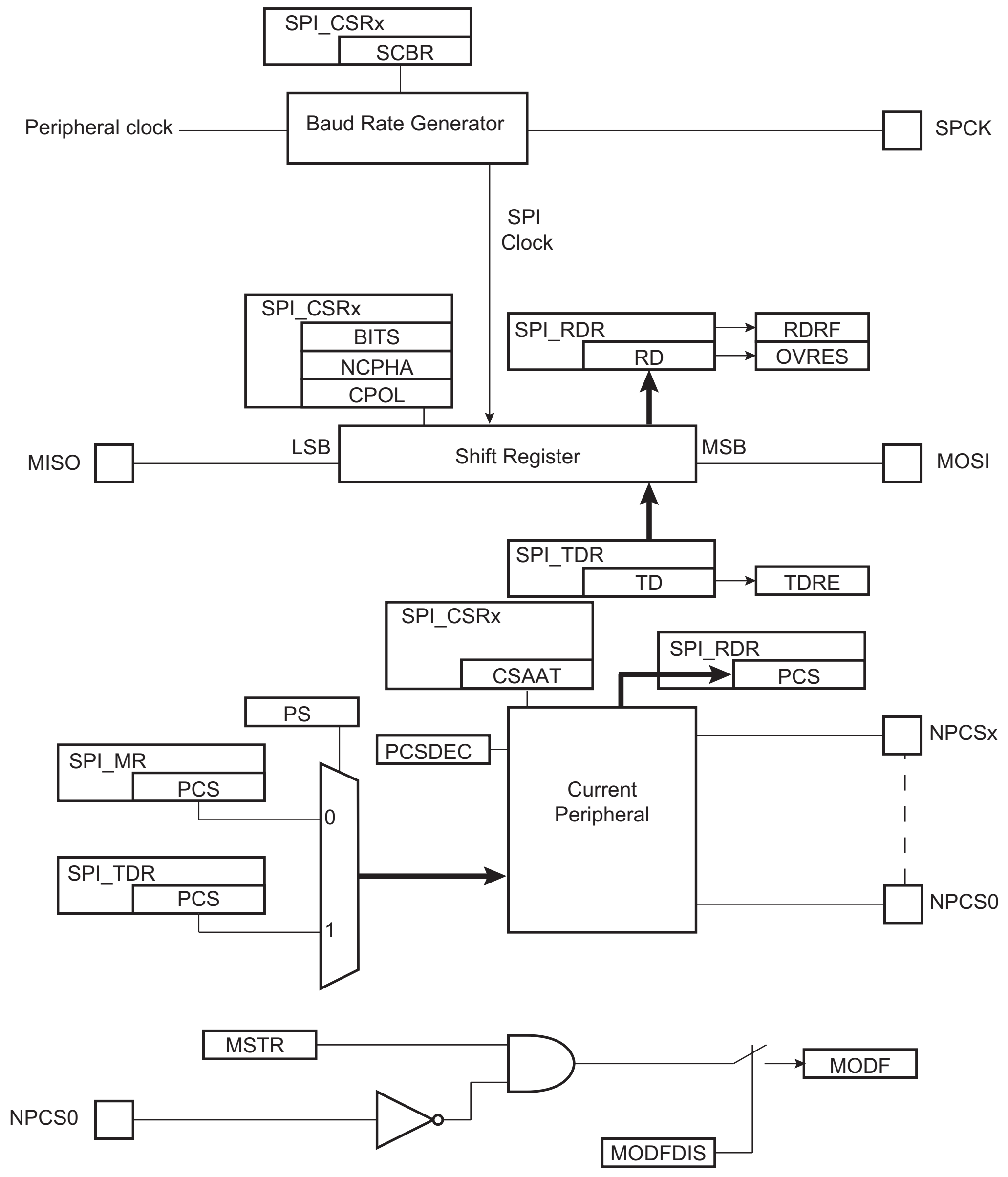
The transfer of received data from the Shift register to the SPI_RDR is indicated by the Receive Data Register Full (RDRF) bit in the SPI_SR. When the received data is read, the RDRF bit is cleared.

If the SPI_RDR has not been read before new data is received, the Overrun Error (OVRES) bit in the SPI_SR is set. As long as this flag is set, data is loaded in the SPI_RDR. The user has to read the SPI_SR to clear the OVRES bit.

Figure 41-6 shows a block diagram of the SPI when operating in Master mode. Figure 41-7 shows a flow chart describing how transfers are handled.

41.7.3.1 Master Mode Block Diagram

Figure 41-6. Master Mode Block Diagram



41.7.3.2 Master Mode Flow Diagram

Figure 41-7. Master Mode Flow Diagram

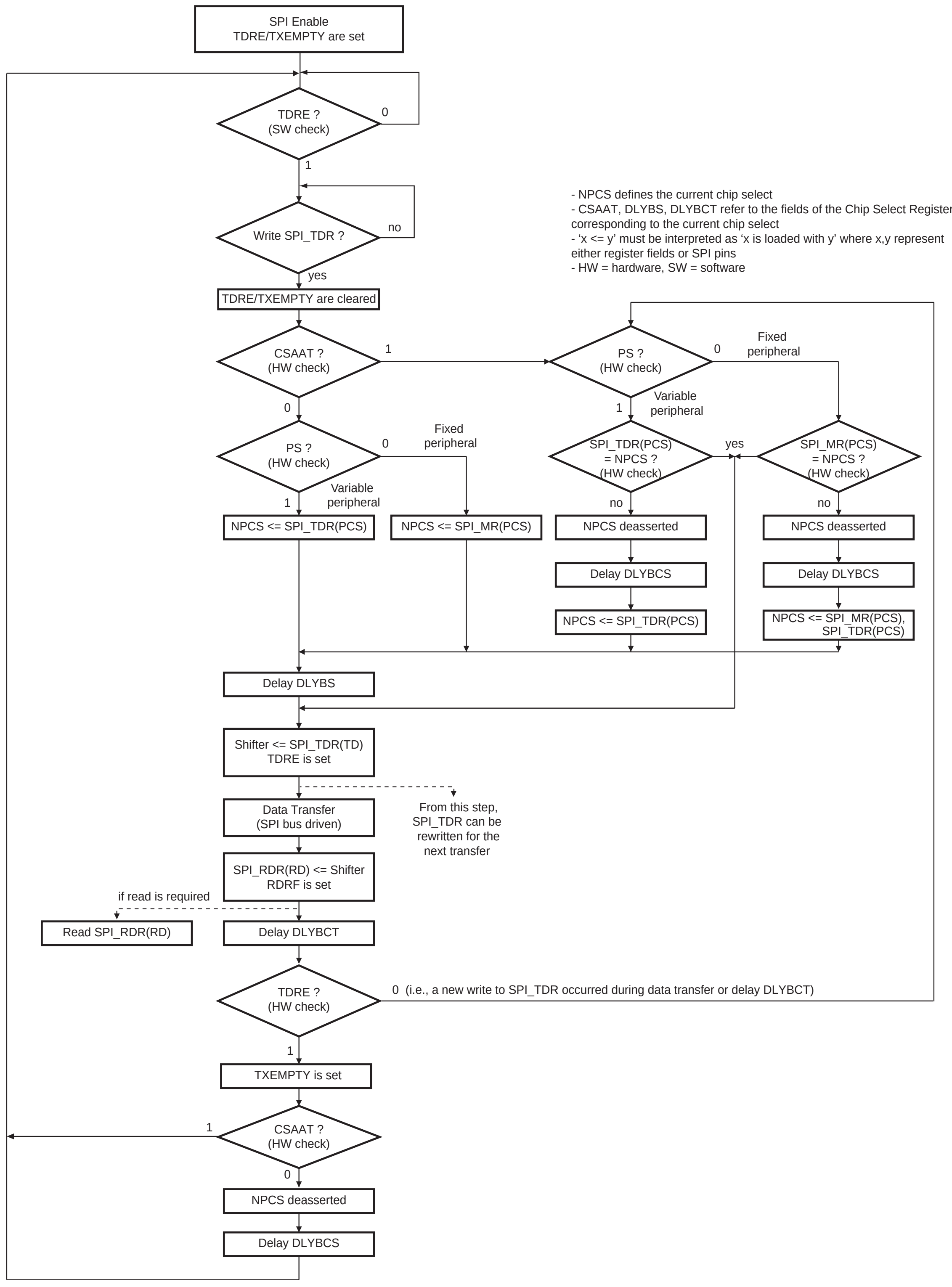
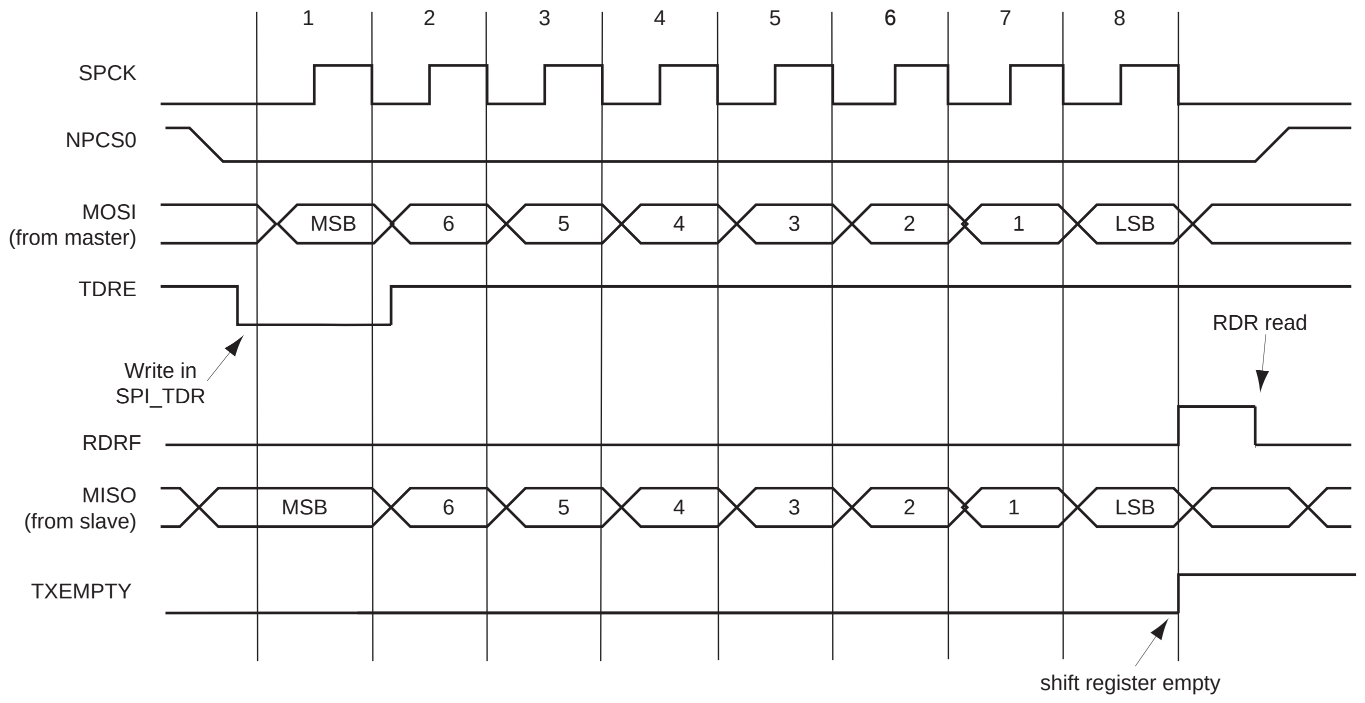


Figure 41-8 shows the behavior of Transmit Data Register Empty (TDRE), Receive Data Register (RDRF) and Transmission Register Empty (TXEMPTY) status flags within the SPI_SR during an 8-bit data transfer in Fixed mode without the DMA involved.

Figure 41-8. Status Register Flags Behavior



41.7.3.3 Clock Generation

The SPI Baud rate clock is generated by dividing the peripheral clock by a value between 1 and 255.

If the SCBR field in the SPI_CSRx is programmed to 1, the operating baud rate is peripheral clock (see the electrical characteristics section for the SPCK maximum frequency). Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

At reset, SCBR is 0 and the user has to program it to a valid value before performing the first transfer.

The divisor can be defined independently for each chip select, as it has to be programmed in the SCBR field. This allows the SPI to automatically adapt the baud rate for each interfaced peripheral without reprogramming.

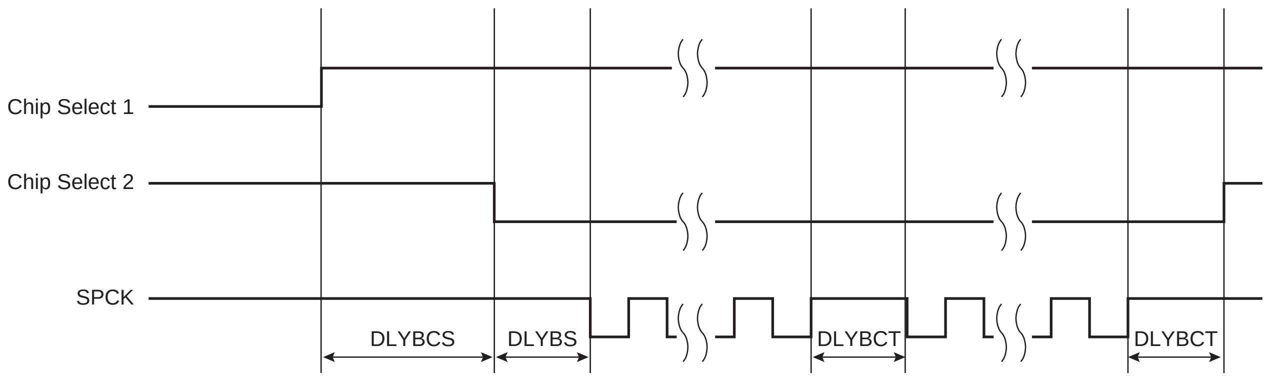
41.7.3.4 Transfer Delays

Figure 41-9 shows a chip select transfer change and consecutive transfers on the same chip select. Three delays can be programmed to modify the transfer waveforms:

- Delay between the chip selects—programmable only once for all chip selects by writing the DLYBCS field in the SPI_MR. The SPI slave device deactivation delay is managed through DLYBCS. If there is only one SPI slave device connected to the master, the DLYBCS field does not need to be configured. If several slave devices are connected to a master, DLYBCS must be configured depending on the highest deactivation delay. Refer to the SPI slave device electrical characteristics.
- Delay before SPCK—independently programmable for each chip select by writing the DLYBS field. The SPI slave device activation delay is managed through DLYBS. Refer to the SPI slave device electrical characteristics to define DLYBS.
- Delay between consecutive transfers—independently programmable for each chip select by writing the DLYBCT field. The time required by the SPI slave device to process received data is managed through DLYBCT. This time depends on the SPI slave system activity.

These delays allow the SPI to be adapted to the interfaced peripherals and their speed and bus release time.

Figure 41-9. Programmable Delays



41.7.3.5 Peripheral Selection

The serial peripherals are selected through the assertion of the NPCS0 to NPCS3 signals. By default, all NPCS signals are high before and after each transfer.

- **Fixed Peripheral Select Mode**: SPI exchanges data with only one peripheral.
  Fixed Peripheral Select mode is enabled by clearing the PS bit in the SPI_MR. In this case, the current peripheral is defined by the PCS field in the SPI_MR and the PCS field in the SPI_TDR has no effect.
- **Variable Peripheral Select Mode:** Data can be exchanged with more than one peripheral without having to reprogram the NPCS field in the SPI_MR.
  Variable Peripheral Select mode is enabled by setting the PS bit in the SPI_MR. The PCS field in the SPI_TDR is used to select the current peripheral. This means that the peripheral selection can be defined for each new data. The value to write in the SPI_TDR has the following format:

  [xxxxxxx(7-bit) + LASTXFER(1-bit)[(1)]+ xxxx(4-bit) + PCS (4-bit) + DATA (8 to 16-bit)] with PCS equals the chip select to assert, as defined in Section 41.8.4 “SPI Transmit Data Register” and LASTXFER bit at 0 or 1 depending on the CSAAT bit.

Note: 1. Optional

CSAAT, LASTXFER and CSNAAT bits are discussed in Section 41.7.3.9 “Peripheral Deselection with DMA”.

If LASTXFER is used, the command must be issued after writing the last character. Instead of LASTXFER, the user can use the SPIDIS command. After the end of the DMA transfer, it is necessary to wait for the TXEMPTY flag and then write SPIDIS into the SPI Control Register (SPI_CR). This does not change the configuration register values). The NPCS is disabled after the last character transfer. Then, another DMA transfer can be started if the SPIEN has previously been written in the SPI_CR.

41.7.3.6 SPI Direct Access Memory Controller (DMAC)

In both Fixed and Variable modes, the Direct Memory Access Controller (DMAC) can be used to reduce processor overhead.

The fixed peripheral selection allows buffer transfers with a single peripheral. Using the DMAC is an optimal means, as the size of the data transfer between the memory and the SPI is either 8 bits or 16 bits. However, if the peripheral selection is modified, the SPI_MR must be reprogrammed.

The variable peripheral selection allows buffer transfers with multiple peripherals without reprogramming the SPI_MR. Data written in the SPI_TDR is 32 bits wide and defines the real data to be transmitted and the destination peripheral. Using the DMAC in this mode requires 32-bit wide buffers, with the data in the LSBs and the

PCS and LASTXFER fields in the MSBs. However, the SPI still controls the number of bits (8 to 16) to be transferred through MISO and MOSI lines with the chip select configuration registers. This is not the optimal means in terms of memory size for the buffers, but it provides a very effective means to exchange data with several peripherals without any intervention of the processor.

41.7.3.7 Peripheral Chip Select Decoding

The user can program the SPI to operate with up to 15 slave peripherals by decoding the four chip select lines, NPCS0 to NPCS3 with an external decoder/demultiplexer (refer to Figure 41-10). This can be enabled by setting the PCSDEC bit in the SPI_MR.

When operating without decoding, the SPI makes sure that in any case only one chip select line is activated, i.e., one NPCS line driven low at a time. If two bits are defined low in a PCS field, only the lowest numbered chip select is driven low.

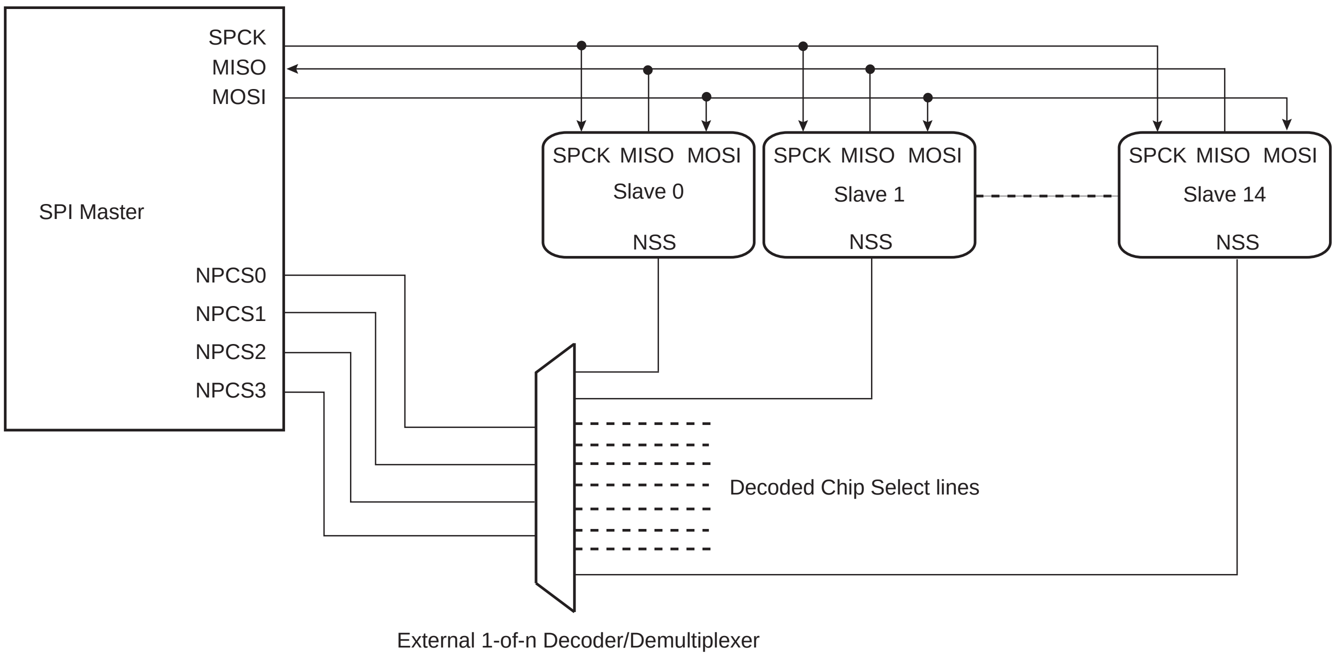
When operating with decoding, the SPI directly outputs the value defined by the PCS field on the NPCS lines of either SPI_MR or SPI_TDR (depending on PS).

As the SPI sets a default value of 0xF on the chip select lines (i.e., all chip select lines at 1) when not processing any transfer, only 15 peripherals can be decoded.

The SPI has four chip select registers (SPI_CSR0...SPI_CSR3). As a result, when external decoding is activated, each NPCS chip select defines the characteristics of up to four peripherals. As an example, SPI_CRS0 defines the characteristics of the externally decoded peripherals 0 to 3, corresponding to the PCS values 0x0 to 0x3. Consequently, the user has to make sure to connect compatible peripherals on the decoded chip select lines 0 to 3, 4 to 7, 8 to 11 and 12 to 14. Figure 41-10 shows this type of implementation.

If the CSAAT bit is used, with or without the DMAC, the Mode Fault detection for NPCS0 line must be disabled. This is not needed for all other chip select lines since Mode Fault detection is only on NPCS0.

Figure 41-10. Chip Select Decoding Application Block Diagram: Single Master/Multiple Slave Implementation



41.7.3.8 Peripheral Deselection without DMA

During a transfer of more than one unit of data on a chip select without the DMA, the SPI_TDR is loaded by the processor, the TDRE flag rises as soon as the content of the SPI_TDR is transferred into the internal Shift register. When this flag is detected high, the SPI_TDR can be reloaded. If this reload by the processor occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the chip select is not deasserted between the two transfers. But depending on the application software handling the SPI status register flags (by interrupt or polling method) or servicing other interrupts or other tasks, the processor may not reload the SPI_TDR in time to keep the chip select active (low). A null DLYBCT value (delay between consecutive transfers) in the SPI_CSR, gives even less time for the processor to reload the SPI_TDR. With some SPI slave peripherals, if the chip select line must remain active (low) during a full set of transfers, communication errors can occur.

To facilitate interfacing with such devices, the chip select registers [SPI_CSR0...SPI_CSR3] can be programmed with the Chip Select Active After Transfer (CSAAT) bit at 1. This allows the chip select lines to remain in their current state (low = active) until a transfer to another chip select is required. Even if the SPI_TDR is not reloaded, the chip select remains active. To deassert the chip select line at the end of the transfer, the Last Transfer (LASTXFER) bit in SPI_CR must be set after writing the last data to transmit into SPI_TDR.

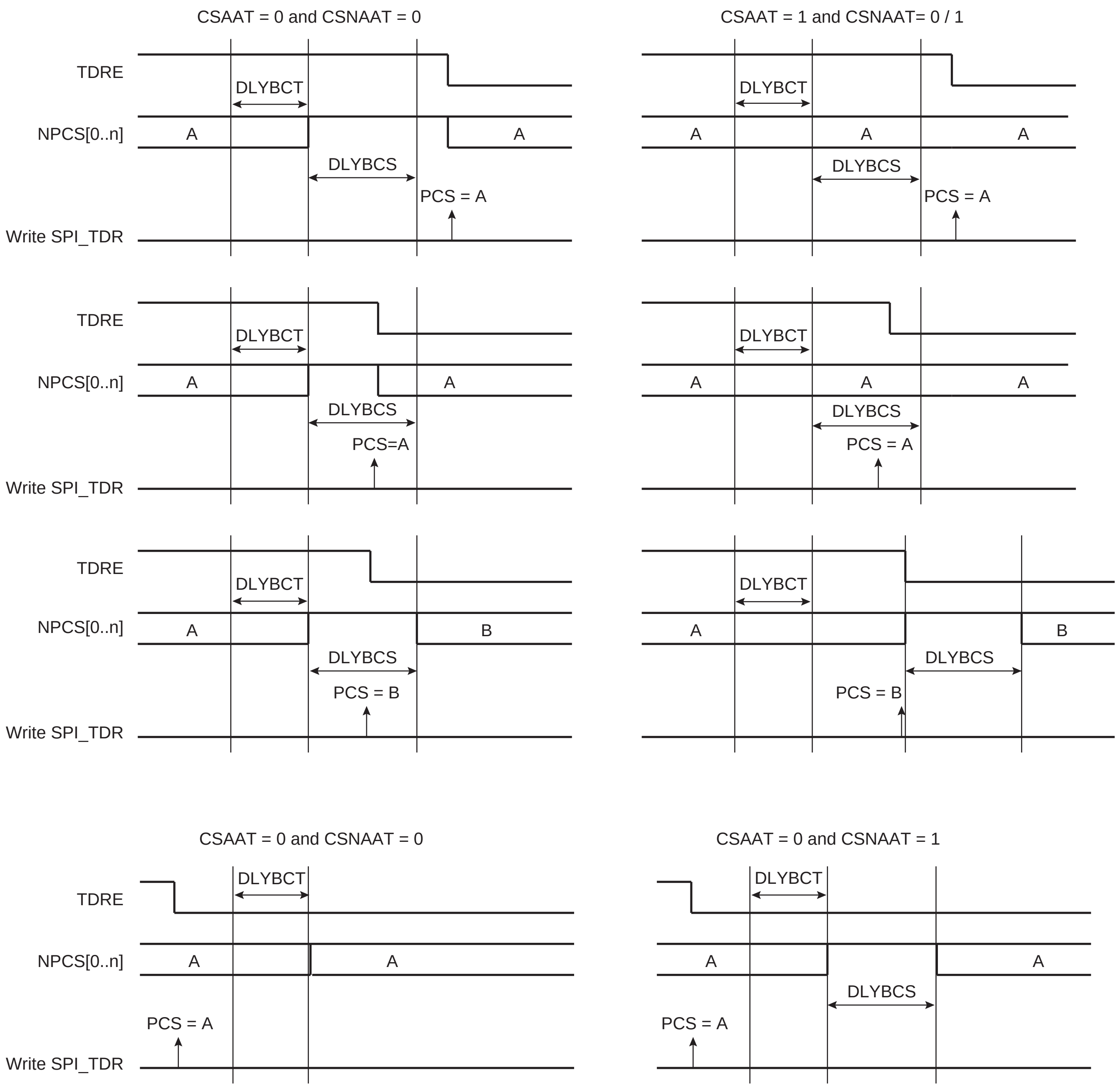
41.7.3.9 Peripheral Deselection with DMA

DMA provides faster reloads of the SPI_TDR compared to software. However, depending on the system activity, it is not guaranteed that the SPI_TDR is written with the next data before the end of the current transfer. Consequently, data can be lost by the deassertion of the NPCS line for SPI slave peripherals requiring the chip select line to remain active between two transfers. The only way to guarantee a safe transfer in this case is the use of the CSAAT and LASTXFER bits.

When the CSAAT bit is configured to 0, the NPCS does not rise in all cases between two transfers on the same peripheral. During a transfer on a chip select, the TDRE flag rises as soon as the content of the SPI_TDR is transferred into the internal shift register. When this flag is detected, the SPI_TDR can be reloaded. If this reload occurs before the end of the current transfer and if the next transfer is performed on the same chip select as the current transfer, the chip select is not deasserted between the two transfers. This can lead to difficulties to interface with some serial peripherals requiring the chip select to be deasserted after each transfer. To facilitate interfacing with such devices, the SPI_CSR can be programmed with the Chip Select Not Active After Transfer (CSNAAT) bit at 1. This allows the chip select lines to be deasserted systematically during a time “DLYBCS” (the value of the CSNAAT bit is processed only if the CSAAT bit is configured to 0 for the same chip select).

Figure 41-11 shows different peripheral deselection cases and the effect of the CSAAT and CSNAAT bits.

Figure 41-11. Peripheral Deselection



41.7.3.10 Mode Fault Detection

The SPI has the capability to operate in multimaster environment. Consequently, the NPCS0/NSS line must be monitored. If one of the masters on the SPI bus is currently transmitting, the NPCS0/NSS line is low and the SPI must not transmit any data. A mode fault is detected when the SPI is programmed in Master mode and a low level is driven by an external master on the NPCS0/NSS signal. In multimaster environment, NPCS0, MOSI, MISO and SPCK pins must be configured in open drain (through the PIO controller). When a mode fault is detected, the SPI_SR.MODF bit is set until SPI_SR is read and the SPI is automatically disabled until it is reenabled by setting the SPI_CR.SPIEN bit.

By default, the mode fault detection is enabled. The user can disable it by setting the SPI_MR.MODFDIS bit.

41.7.4 SPI Slave Mode

When operating in Slave mode, the SPI processes data bits on the clock provided on the SPI clock pin (SPCK).

The SPI waits until NSS goes active before receiving the serial clock from an external master. When NSS falls, the clock is validated and the data is loaded in the SPI_RDR depending on the BITS field configured in SPI_CSR0. These bits are processed following a phase and a polarity defined respectively by the NCPHA and CPOL bits in SPI_CSR0. Note that the fields BITS, CPOL and NCPHA of the other chip select registers (SPI_CSR1...SPI_CSR3) have no effect when the SPI is programmed in Slave mode.

The bits are shifted out on the MISO line and sampled on the MOSI line.

Note: For more information on the BITS field, see also the note below the SPI_CSRx bitmap (Section 41.8.9 "SPI Chip Select Register").

When all bits are processed, the received data is transferred in the SPI_RDR and the RDRF bit rises. If the SPI_RDR has not been read before new data is received, the Overrun Error Status (OVRES) bit in the SPI_SR is set. As long as this flag is set, data is loaded in the SPI_RDR. The user must read SPI_SR to clear the OVRES bit.

When a transfer starts, the data shifted out is the data present in the Shift register. If no data has been written in the SPI_TDR, the last data received is transferred. If no data has been received since the last reset, all bits are transmitted low, as the Shift register resets to 0.

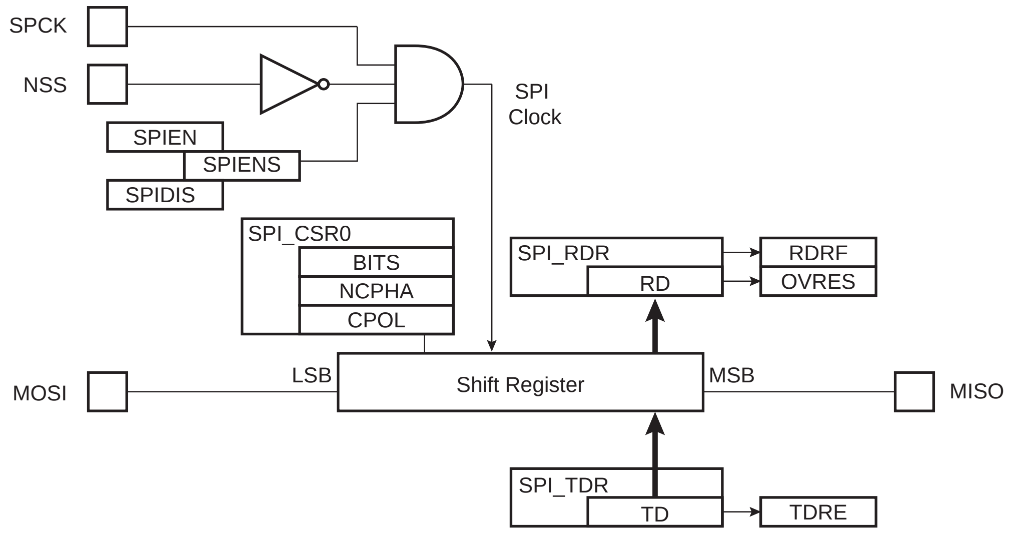
When a first data is written in the SPI_TDR, it is transferred immediately in the Shift register and the TDRE flag rises. If new data is written, it remains in the SPI_TDR until a transfer occurs, i.e., NSS falls and there is a valid clock on the SPCK pin. When the transfer occurs, the last data written in the SPI_TDR is transferred in the Shift register and the TDRE flag rises. This enables frequent updates of critical variables with single transfers.

Then, new data is loaded in the Shift register from the SPI_TDR. If no character is ready to be transmitted, i.e., no character has been written in the SPI_TDR since the last load from the SPI_TDR to the Shift register, the SPI_TDR is retransmitted. In this case the Underrun Error Status Flag (UNDES) is set in the SPI_SR.

In Slave mode, if the NSS line rises and the received character length does not match the configuration defined in SPI_CSR0.BITS the flag SFERR is set in SPI_SR.

Figure 41-12 shows a block diagram of the SPI when operating in Slave mode.

Figure 41-12. Slave Mode Functional Block Diagram



41.7.5 Register Write Protection

To prevent any single software error from corrupting SPI behavior, certain registers in the address space can be write-protected in the SPI Write Protection Mode Register (SPI_WPMR).

If a write access to a write-protected register is detected, the WPVS flag in the SPI Write Protection Status Register (SPI_WPSR) is set and the WPVSRC field indicates the register in which the write access has been attempted.

The WPVS bit is automatically cleared after reading SPI_WPSR.

The following registers are write-protected when WPEN is set in SPI_WPMR:

- SPI Mode Register
- SPI Chip Select Register

The following register is write-protected when WPCREN is set in SPI_WPMR:

- SPI Control Register

The following registers are write-protected when WPITEN is set in SPI_WPMR:

- SPI Interrupt Enable Register
- SPI Interrupt Disable Register

41.8 Serial Peripheral Interface (SPI) User Interface

In the “Offset” column of Table 41-5, ‘CS_number’ denotes the chip select number.

Table 41-5. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Control Register	SPI_CR	Write-only	–
0x04	Mode Register	SPI_MR	Read/Write	0x0
0x08	Receive Data Register	SPI_RDR	Read-only	0x0
0x0C	Transmit Data Register	SPI_TDR	Write-only	–
0x10	Status Register	SPI_SR	Read-only	0x0
0x14	Interrupt Enable Register	SPI_IER	Write-only	–
0x18	Interrupt Disable Register	SPI_IDR	Write-only	–
0x1C	Interrupt Mask Register	SPI_IMR	Read-only	0x0
0x20–0x2C	Reserved	–	–	–
0x30 + (CS_number * 0x04)	Chip Select Register	SPI_CSR	Read/Write	0x0
0x40–0x48	Reserved	–	–	–
0x4C–0xE0	Reserved	–	–	–
0xE4	Write Protection Mode Register	SPI_WPMR	Read/Write	0x0
0xE8	Write Protection Status Register	SPI_WPSR	Read-only	0x0
0xEC–0xF8	Reserved	–	–	–
0xFC	Reserved	–	–	–

41.8.1 SPI Control Register

Name: SPI_CR

Address: 0x40008000 (0), 0x40058000 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
15	14	13	12	11	10	9	8
–	–	–	REQCLR	–	–	–	–
7	6	5	4	3	2	1	0
SWRST	–	–	–	–	–	SPIDIS	SPIEN

This register can only be written if the WPCREN bit is cleared in the SPI Write Protection Mode Register.

- **SPIEN: SPI Enable**

0: No effect.

1: Enables the SPI to transfer and receive data.

- **SPIDIS: SPI Disable**

0: No effect.f

1: Disables the SPI.

All pins are set in Input mode after completion of the transmission in progress, if any.

If a transfer is in progress when SPIDIS is set, the SPI completes the transmission of the shifter register and does not start any new transfer, even if the SPI_THR is loaded.

Note: If both SPIEN and SPIDIS are equal to one when the SPI_CR is written, the SPI is disabled.

- **SWRST: SPI Software Reset**

0: No effect.

1: Reset the SPI. A software-triggered hardware reset of the SPI interface is performed.

The SPI is in Slave mode after software reset.

- **REQCLR: Request to Clear the Comparison Trigger**

0: No effect.

1: Restarts the comparison trigger to enable SPI_RDR loading.

- **LASTXFER: Last Transfer**

0: No effect.

1: The current NPCS is deasserted after the character written in TD has been transferred. When SPI_CSRx.CSAAT is set, the communication with the current serial peripheral can be closed by raising the corresponding NPCS line as soon as TD transfer is completed.

Refer to Section 41.7.3.5 “Peripheral Selection” for more details.

41.8.2 SPI Mode Register

Name: SPI_MR

Address: 0x40008004 (0), 0x40058004 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
DLYBCS							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	PCS			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	–	–	–
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LLB	–	WDRBT	MODFDIS	–	PCSDEC	PS	MSTR

This register can only be written if the WPEN bit is cleared in the SPI Write Protection Mode Register.

- **MSTR: Master/Slave Mode**

0: SPI is in Slave mode

1: SPI is in Master mode

- **PS: Peripheral Select**

0: Fixed Peripheral Select

1: Variable Peripheral Select

- **PCSDEC: Chip Select Decode**

0: The chip select lines are directly connected to a peripheral device.

1: The four NPCS chip select lines are connected to a 4-bit to 16-bit decoder.

When PCSDEC = 1, up to 15 chip select signals can be generated with the four NPCS lines using an external 4-bit to 16-bit decoder. The chip select registers define the characteristics of the 15 chip selects, with the following rules:

SPI_CSR0 defines peripheral chip select signals 0 to 3.

SPI_CSR1 defines peripheral chip select signals 4 to 7.

SPI_CSR2 defines peripheral chip select signals 8 to 11.

SPI_CSR3 defines peripheral chip select signals 12 to 14.

- **MODFDIS: Mode Fault Detection**

0: Mode fault detection enabled

1: Mode fault detection disabled

- **WDRBT: Wait Data Read Before Transfer**

0: No Effect. In Master mode, a transfer can be initiated regardless of the SPI_RDR state.

1: In Master mode, a transfer can start only if the SPI_RDR is empty, i.e., does not contain any unread data. This mode prevents overrun error in reception.

- **LLB: Local Loopback Enable**

0: Local loopback path disabled.

1: Local loopback path enabled.

LLB controls the local loopback on the data shift register for testing in Master mode only (MISO is internally connected on MOSI).

- **PCS: Peripheral Chip Select**

This field is only used if fixed peripheral select is active (PS = 0).

If SPI_MR.PCSDEC = 0:

PCS = xxx0	NPCS[3:0] = 1110
PCS = xx01	NPCS[3:0] = 1101
PCS = x011	NPCS[3:0] = 1011
PCS = 0111	NPCS[3:0] = 0111
PCS = 1111	forbidden (no peripheral is selected)

(x = don’t care)

If SPI_MR.PCSDEC = 1:

NPCS[3:0] output signals = PCS.

- **DLYBCS: Delay Between Chip Selects**

This field defines the delay between the inactivation and the activation of NPCS. The DLYBCS time guarantees nonoverlapping chip selects and solves bus contentions in case of peripherals having long data float times.

If DLYBCS is lower than 6, six peripheral clock periods are inserted by default.

Otherwise, the following equation determines the delay:

$$\text{Delay Between Chip Selects} = \frac{\text{DLYBCS}}{f_{\text{peripheral clock}}}$$

41.8.3 SPI Receive Data Register

Name: SPI_RDR

Address: 0x40008008 (0), 0x40058008 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	PCS			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RD							

- **RD: Receive Data**

Data received by the SPI Interface is stored in this register in a right-justified format. Unused bits are read as zero.

- **PCS: Peripheral Chip Select**

In Master mode only, these bits indicate the value on the NPCS pins at the end of a transfer. Otherwise, these bits are read as zero.

Note: When using Variable Peripheral Select mode (PS = 1 in SPI_MR), it is mandatory to set the SPI_MR.WDRBT bit if the PCS field must be processed in SPI_RDR.

41.8.4 SPI Transmit Data Register

Name: SPI_TDR

Address: 0x4000800C (0), 0x4005800C (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	LASTXFER

23	22	21	20	19	18	17	16
–	–	–	–	PCS			

15	14	13	12	11	10	9	8
TD							

7	6	5	4	3	2	1	0
TD							

- **TD: Transmit Data**

Data to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

- **PCS: Peripheral Chip Select**

This field is only used if variable peripheral select is active (PS = 1).

If SPI_MR.PCSDEC = 0:

PCS = xxx0	NPCS[3:0] = 1110
PCS = xx01	NPCS[3:0] = 1101
PCS = x011	NPCS[3:0] = 1011
PCS = 0111	NPCS[3:0] = 0111
PCS = 1111	forbidden (no peripheral is selected)

(x = don't care)

If SPI_MR.PCSDEC = 1:

NPCS[3:0] output signals = PCS.

- **LASTXFER: Last Transfer**

0: No effect

1: The current NPCS is deasserted after the transfer of the character written in TD. When SPI_CSRx.CSAAT is set, the communication with the current serial peripheral can be closed by raising the corresponding NPCS line as soon as TD transfer is completed.

This field is only used if variable peripheral select is active (SPI_MR.PS = 1).

41.8.5 SPI Status Register

Name: SPI_SR

Address: 0x40008010 (0), 0x40058010 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	SPIENS
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	SFERR	–	UNDES	TXEMPTY	NSSR
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	OVRES	MODF	TDRE	RDRF

- **RDRF: Receive Data Register Full (cleared by reading SPI_RDR)**

0: No data has been received since the last read of SPI_RDR.

1: Data has been received and the received data has been transferred from the shift register to SPI_RDR since the last read of SPI_RDR.

- **TDRE: Transmit Data Register Empty (cleared by writing SPI_TDR)**

0: Data has been written to SPI_TDR and not yet transferred to the shift register.

1: The last data written in the SPI_TDR has been transferred to the shift register.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to 1.

- **MODF: Mode Fault Error (cleared on read)**

0: No mode fault has been detected since the last read of SPI_SR.

1: A mode fault occurred since the last read of SPI_SR.

- **OVRES: Overrun Error Status (cleared on read)**

0: No overrun has been detected since the last read of SPI_SR.

1: An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the shift register since the last read of the SPI_RDR.

- **NSSR: NSS Rising (cleared on read)**

0: No rising edge detected on NSS pin since the last read of SPI_SR.

1: A rising edge occurred on NSS pin since the last read of SPI_SR.

- **TXEMPTY: Transmission Registers Empty (cleared by writing SPI_TDR)**

0: As soon as data is written in SPI_TDR.

1: SPI_TDR and internal shift register are empty. If a transfer delay has been defined, TXEMPTY is set after the end of this delay.

- **UNDES: Underrun Error Status (Slave mode only) (cleared on read)**

0: No underrun has been detected since the last read of SPI_SR.

1: A transfer starts whereas no data has been loaded in SPI_TDR.

- **SFERR: Slave Frame Error (cleared on read)**

0: There is no frame error detected for a slave access since the last read of SPI_SR.

1: In Slave mode, the Chip Select raised while the character defined in SPI_CSR0.BITS was not complete.

- **SPIENS: SPI Enable Status**

0: SPI is disabled.

1: SPI is enabled.

41.8.6 SPI Interrupt Enable Register

Name: SPI_IER

Address: 0x40008014 (0), 0x40058014 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	OVRES	MODF	TDRE	RDRF

This register can only be written if the WPITEN bit is cleared in the SPI Write Protection Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Enables the corresponding interrupt.

- **RDRF: Receive Data Register Full Interrupt Enable**

- **TDRE: SPI Transmit Data Register Empty Interrupt Enable**

- **MODF: Mode Fault Error Interrupt Enable**

- **OVRES: Overrun Error Interrupt Enable**

- **NSSR: NSS Rising Interrupt Enable**

- **TXEMPTY: Transmission Registers Empty Enable**

- **UNDES: Underrun Error Interrupt Enable**

41.8.7 SPI Interrupt Disable Register

Name: SPI_IDR

Address: 0x40008018 (0), 0x40058018 (1)

Access: Write-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	OVRES	MODF	TDRE	RDRF

This register can only be written if the WPITEN bit is cleared in the SPI Write Protection Mode Register.

The following configuration values are valid for all listed bit names of this register:

0: No effect.

1: Disables the corresponding interrupt.

- **RDRF: Receive Data Register Full Interrupt Disable**

- **TDRE: SPI Transmit Data Register Empty Interrupt Disable**

- **MODF: Mode Fault Error Interrupt Disable**

- **OVRES: Overrun Error Interrupt Disable**

- **NSSR: NSS Rising Interrupt Disable**

- **TXEMPTY: Transmission Registers Empty Disable**

- **UNDES: Underrun Error Interrupt Disable**

41.8.8 SPI Interrupt Mask Register

Name: SPI_IMR

Address: 0x4000801C (0), 0x4005801C (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
–	–	–	–	–	UNDES	TXEMPTY	NSSR
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	OVRES	MODF	TDRE	RDRF

The following configuration values are valid for all listed bit names of this register:

0: The corresponding interrupt is not enabled.

1: The corresponding interrupt is enabled.

- **RDRF: Receive Data Register Full Interrupt Mask**

- **TDRE: SPI Transmit Data Register Empty Interrupt Mask**

- **MODF: Mode Fault Error Interrupt Mask**

- **OVRES: Overrun Error Interrupt Mask**

- **NSSR: NSS Rising Interrupt Mask**

- **TXEMPTY: Transmission Registers Empty Mask**

- **UNDES: Underrun Error Interrupt Mask**

41.8.9 SPI Chip Select Register

Name: SPI_CSRx [x=0..3]

Address: 0x40008030 (0), 0x40058030 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
DLYBCT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DLYBS							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SCBR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BITS				CSAAT	CSNAAT	NCPHA	CPOL

This register can only be written if the WPEN bit is cleared in the SPI Write Protection Mode Register.

Note: SPI_CSRx must be written even if the user wants to use the default reset values. The BITS field is not updated with the translated value unless the register is written.

- **CPOL: Clock Polarity**

0: The inactive state value of SPCK is logic level zero.

1: The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce the required clock/data relationship between master and slave devices.

- **NCPHA: Clock Phase**

0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured. NCPHA is used with CPOL to produce the required clock/data relationship between master and slave devices.

- **CSNAAT: Chip Select Not Active After Transfer (Ignored if CSAAT = 1)**

0: The Peripheral Chip Select Line does not rise between two transfers if the SPI_TDR is reloaded before the end of the first transfer and if the two transfers occur on the same chip select.

1: The Peripheral Chip Select Line rises systematically after each transfer performed on the same slave. It remains inactive after the end of transfer for a minimal duration of:

$$\frac{DLYBCS}{f_{\text{peripheral clock}}}$$ (If field DLYBCS is lower than 6, a minimum of six periods is introduced.)

- **CSAAT: Chip Select Active After Transfer**

0: The Peripheral Chip Select Line rises as soon as the last transfer is achieved.

1: The Peripheral Chip Select Line does not rise after the last transfer is achieved. It remains active until a new transfer is requested on a different chip select.

- **BITS: Bits Per Transfer**

(See the note below the SPI_CSR bitmap.)

The BITS field determines the number of data bits transferred. Reserved values should not be used.

Value	Name	Description
0	8_BIT	8 bits for transfer
1	9_BIT	9 bits for transfer
2	10_BIT	10 bits for transfer
3	11_BIT	11 bits for transfer
4	12_BIT	12 bits for transfer
5	13_BIT	13 bits for transfer
6	14_BIT	14 bits for transfer
7	15_BIT	15 bits for transfer
8	16_BIT	16 bits for transfer
9	–	Reserved
10	–	Reserved
11	–	Reserved
12	–	Reserved
13	–	Reserved
14	–	Reserved
15	–	Reserved

- **SCBR: Serial Clock Bit Rate**

In Master mode, the SPI Interface uses a modulus counter to derive the SPCK bit rate from the peripheral clock. The bit rate is selected by writing a value from1 to 255 in the SCBR field. The following equation determines the SPCK bit rate:

SCBR = $f_{\text{peripheral clock}}$ / SPCK Bit Rate

Programming the SCBR field to 0 is forbidden. Triggering a transfer while SCBR is at 0 can lead to unpredictable results.

If BRSRCCLK = 1 in SPI_MR, SCBR must be programmed with a value greater than 1.

At reset, SCBR is 0 and the user has to program it at a valid value before performing the first transfer.

Note: If one of the SCBR fields in SPI_CSRx is set to 1, the other SCBR fields in SPI_CSRx must be set to 1 as well, if they are used to process transfers. If they are not used to transfer data, they can be set at any value.

- **DLYBS: Delay Before SPCK**

This field defines the delay from NPCS falling edge (activation) to the first valid SPCK transition.

When DLYBS = 0, the delay is half the SPCK clock period.

Otherwise, the following equation determines the delay:

DLYBS = Delay Before SPCK × $f_{\text{peripheral clock}}$

- **DLYBCT: Delay Between Consecutive Transfers**

This field defines the delay between two consecutive transfers with the same peripheral without removing the chip select. The delay is always inserted after each transfer and before removing the chip select if needed.

When DLYBCT = 0, no delay between consecutive transfers is inserted and the clock keeps its duty cycle over the character transfers.

Otherwise, the following equation determines the delay:

DLYBCT = Delay Between Consecutive Transfers × $f_{\text{peripheral clock}}$ / 32

41.8.10 SPI Write Protection Mode Register

Name: SPI_WPMR

Address: 0x400080E4 (0), 0x400580E4 (1)

Access: Read/Write

31	30	29	28	27	26	25	24
WPKEY							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
WPKEY							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
WPKEY							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	WPCREN	WPITEN	WPEN

- **WPEN: Write Protection Enable**

0: Disables the write protection if WPKEY corresponds to 0x535049 (“SPI” in ASCII)

1: Enables the write protection if WPKEY corresponds to 0x535049 (“SPI” in ASCII)

- **WPITEN: Write Protection Interrupt Enable**

0: Disables the write protection on Interrupt registers if WPKEY corresponds to 0x535049.

1: Enables the write protection on Interrupt registers if WPKEY corresponds to 0x535049.

- **WPCREN: Write Protection Control Register Enable**

0: Disables the write protection on the Control register if WPKEY corresponds to 0x535049.

1: Enables the write protection on the Control register if WPKEY corresponds to 0x535049.

- **WPKEY: Write Protection Key**

Value	Name	Description
0x535049	PASSWD	Writing any other value in this field aborts the write operation of the WPEN bit. Always reads as 0.

See Section 41.7.5 “Register Write Protection” for the list of registers that can be write-protected.

41.8.11 SPI Write Protection Status Register

Name: SPI_WPSR

Address: 0x400080E8 (0), 0x400580E8 (1)

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
–	–	–	–	–	–	–	–
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
WPVSRC							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
–	–	–	–	–	–	–	WPVS

- **WPVS: Write Protection Violation Status**

0: No write protection violation has occurred since the last read of SPI_WPSR.

1: A write protection violation has occurred since the last read of SPI_WPSR. If this violation is an unauthorized attempt to write a protected register, the associated violation is reported into field WPVSRC.

- **WPVSRC: Write Protection Violation Source**

When WPVS = 1, WPVSRC indicates the register address offset at which a write access has been attempted.