23. SPI – Serial Peripheral Interface

23.1. Features

- Full-duplex, Three-wire Synchronous Data Transfer
- Master or Slave Operation
- LSB First or MSB First Data Transfer
- Seven Programmable Bit Rates
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Wake-up from Idle Mode
- Double Speed (CK/2) Master SPI Mode

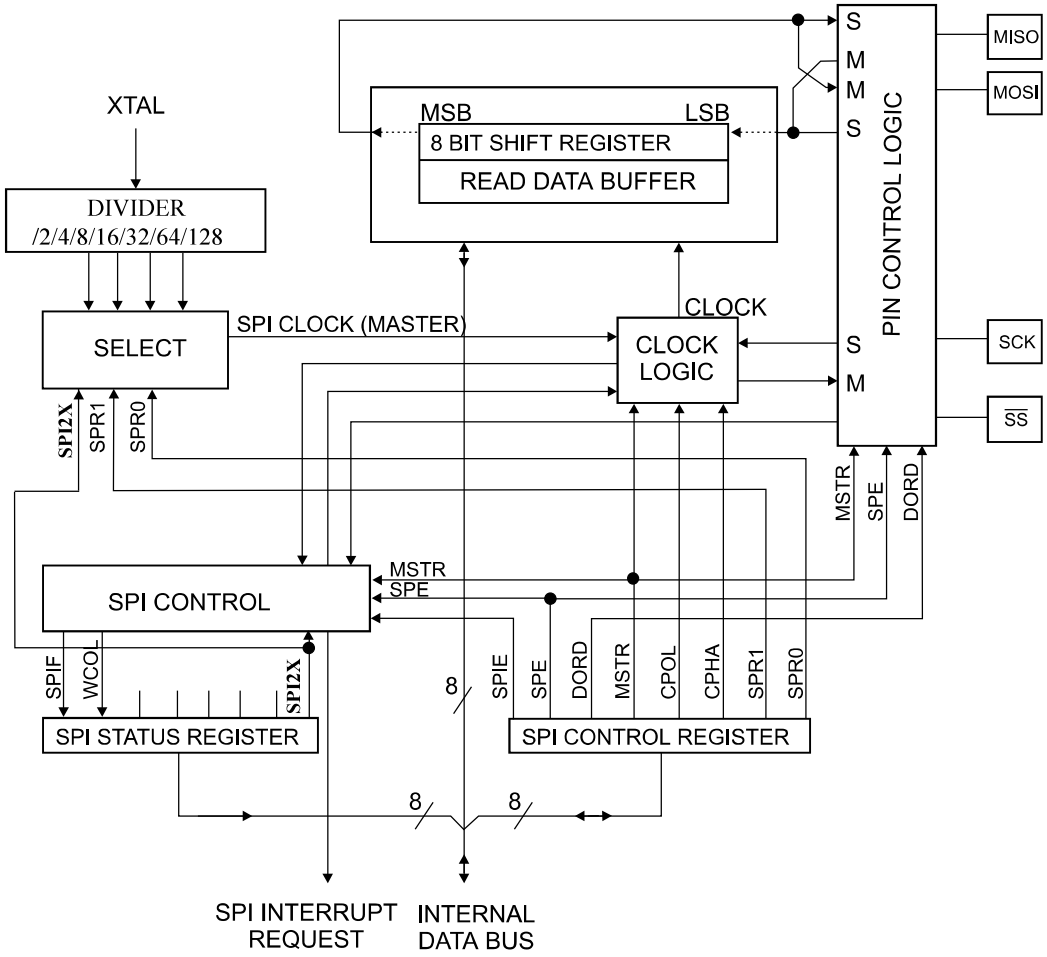
23.2. Overview

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the device and peripheral units, or between several AVR devices.

The USART can also be used in Master SPI mode, please refer to *USART in SPI Mode* chapter.

To enable the SPI module, Power Reduction Serial Peripheral Interface bit in the Power Reduction Register (PRR.PRSPI0) must be written to '0'.

Figure 23-1. SPI Block Diagram



Note: Refer to the pin-out description and the IO Port description for SPI pin placement.

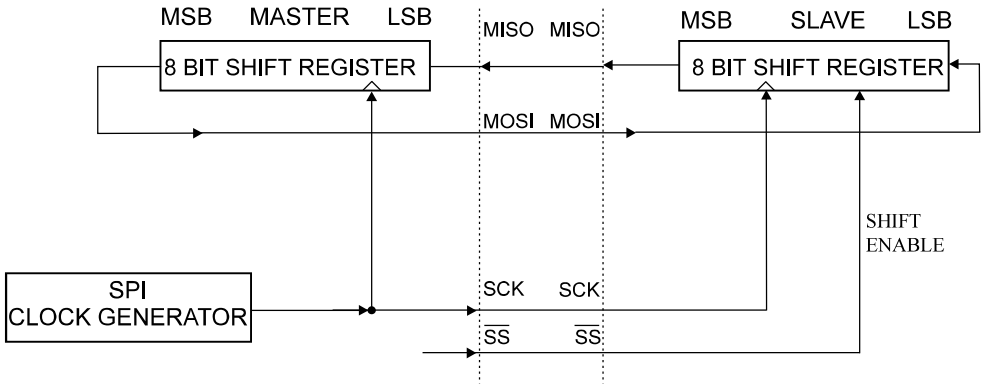
The interconnection between Master and Slave CPUs with SPI is shown in the figure below. The system consists of two shift registers, and a Master Clock generator. The SPI Master initiates the communication cycle when pulling low the Slave Select $\overline{SS}$ pin of the desired Slave. Master and Slave prepare the data to be sent in their respective shift Registers, and the Master generates the required clock pulses on the SCK line to interchange data. Data is always shifted from Master to Slave on the Master Out – Slave In, MOSI, line, and from Slave to Master on the Master In – Slave Out, MISO, line. After each data packet, the Master will synchronize the Slave by pulling high the Slave Select, $\overline{SS}$, line.

When configured as a Master, the SPI interface has no automatic control of the $\overline{SS}$ line. This must be handled by user software before communication can start. When this is done, writing a byte to the SPI Data Register starts the SPI clock generator, and the hardware shifts the eight bits into the Slave. After shifting one byte, the SPI clock generator stops, setting the end of Transmission Flag (SPIF). If the SPI Interrupt Enable bit (SPIE) in the SPCR Register is set, an interrupt is requested. The Master may continue to shift the next byte by writing it into SPDR, or signal the end of packet by pulling high the Slave Select, $\overline{SS}$ line. The last incoming byte will be kept in the Buffer Register for later use.

When configured as a Slave, the SPI interface will remain sleeping with MISO tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the SPI Data Register, SPDR, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. As one byte has been completely shifted, the end of Transmission Flag, SPIF is set. If the SPI Interrupt Enable bit, SPIE, in the SPCR Register is set, an interrupt is requested. The Slave may continue to place new

data to be sent into SPDR before reading the incoming data. The last incoming byte will be kept in the Buffer Register for later use.

Figure 23-2. SPI Master-slave Interconnection



The system is single buffered in the transmit direction and double buffered in the receive direction. This means that bytes to be transmitted cannot be written to the SPI Data Register before the entire shift cycle is completed. When receiving data, however, a received character must be read from the SPI Data Register before the next character has been completely shifted in. Otherwise, the first byte is lost.

In SPI Slave mode, the control logic will sample the incoming signal of the SCK pin. To ensure correct sampling of the clock signal, the minimum low and high periods should be longer than two CPU clock cycles.

When the SPI is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to the table below. For more details on automatic port overrides, refer to the IO Port description.

Table 23-1. SPI Pin Overrides

Pin	Direction, Master SPI	Direction, Slave SPI
MOSI	User Defined	Input
MISO	Input	User Defined
SCK	User Defined	Input
$\overline{SS}$	User Defined	Input

Note: 1. See the IO Port description for how to define the SPI pin directions.

The following code examples show how to initialize the SPI as a Master and how to perform a simple transmission. `DDR_SPI` in the examples must be replaced by the actual Data Direction Register controlling the SPI pins. `DD_MOSI`, `DD_MISO` and `DD_SCK` must be replaced by the actual data direction bits for these pins. E.g. if MOSI is placed on pin PB5, replace `DD_MOSI` with `DDB5` and `DDR_SPI` with `DDRB`.

Assembly Code Example

```
SPI_MasterInit:
   ; Set MOSI and SCK output, all others input
   ldi     r17,(1<<DD_MOSI)|(1<<DD_SCK)
   out     DDR_SPI,r17
   ; Enable SPI, Master, set clock rate fck/16
   ldi     r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
   out     SPCR,r17
   ret
```

```
SPI_MasterTransmit:
   ; Start transmission of data (r16)
   out    SPDR,r16
Wait_Transmit:
   ; Wait for transmission complete
   in     r16, SPSR
   sbrs   r16, SPIF
   rjmp   Wait_Transmit
   ret
```

C Code Example

```
void SPI_MasterInit(void)
{
   /* Set MOSI and SCK output, all others input */
   DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
   /* Enable SPI, Master, set clock rate fck/16 */
   SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0);
}

void SPI_MasterTransmit(char cData)
{
   /* Start transmission */
   SPDR = cData;
   /* Wait for transmission complete */
   while(!(SPSR & (1<<SPIF)))
      ;
}
```

The following code examples show how to initialize the SPI as a Slave and how to perform a simple reception.

Assembly Code Example

```
SPI_SlaveInit:
   ; Set MISO output, all others input
   ldi    r17,(1<<DD_MISO)
   out    DDR_SPI,r17
   ; Enable SPI
   ldi    r17,(1<<SPE)
   out    SPCR,r17
   ret
SPI_SlaveReceive:
   ; Wait for reception complete
   in     r16, SPSR
   sbrs   r16, SPIF
   rjmp SPI_SlaveReceive
   ; Read received data and return
   in     r16,SPDR
   ret
```

C Code Example

```
void SPI_SlaveInit(void)
{
   /* Set MISO output, all others input */
   DDR_SPI = (1<<DD_MISO);
   /* Enable SPI */
   SPCR = (1<<SPE);
}
char SPI_SlaveReceive(void)
{
   /* Wait for reception complete */
   while(!(SPSR & (1<<SPIF)))
      ;
   /* Return Data Register */
   return SPDR;
}
```

Related Links

Pin Descriptions on page 17
USARTSPI - USART in SPI Mode on page 254
PM - Power Management and Sleep Modes on page 62
I/O-Ports on page 97
About Code Examples on page 23

23.3. $\overline{SS}$ Pin Functionality

23.3.1. Slave Mode

When the SPI is configured as a Slave, the Slave Select ($\overline{SS}$) pin is always input. When $\overline{SS}$ is held low, the SPI is activated, and MISO becomes an output if configured so by the user. All other pins are inputs. When $\overline{SS}$ is driven high, all pins are inputs, and the SPI is passive, which means that it will not receive incoming data. The SPI logic will be reset once the SS pin is driven high.

The $\overline{SS}$ pin is useful for packet/byte synchronization to keep the slave bit counter synchronous with the master clock generator. When the $\overline{SS}$ pin is driven high, the SPI slave will immediately reset the send and receive logic, and drop any partially received data in the Shift Register.

23.3.2. Master Mode

When the SPI is configured as a Master (MSTR in SPCR is set), the user can determine the direction of the $\overline{SS}$ pin.

If $\overline{SS}$ is configured as an output, the pin is a general output pin which does not affect the SPI system. Typically, the pin will be driving the $\overline{SS}$ pin of the SPI Slave.

If $\overline{SS}$ is configured as an input, it must be held high to ensure Master SPI operation. If the $\overline{SS}$ pin is driven low by peripheral circuitry when the SPI is configured as a Master with the $\overline{SS}$ pin defined as an input, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The SPIF Flag in SPSR is set, and if the SPI interrupt is enabled, and the I-bit in SREG is set, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that $\overline{SS}$ is driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI Master mode.

23.4. Data Modes

There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize. The following table, summarizes SPCR.CPOL and SPCR.CPHA settings.

Table 23-2. SPI Modes

SPI Mode	Conditions	Leading Edge	Trailing Edge
0	CPOL=0, CPHA=0	Sample (Rising)	Setup (Falling)
1	CPOL=0, CPHA=1	Setup (Rising)	Sample (Falling)

SPI Mode	Conditions	Leading Edge	Trailing Edge
2	CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)
3	CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)

The SPI data transfer formats are shown in the following figure.

Figure 23-3. SPI Transfer Format with CPHA = 0

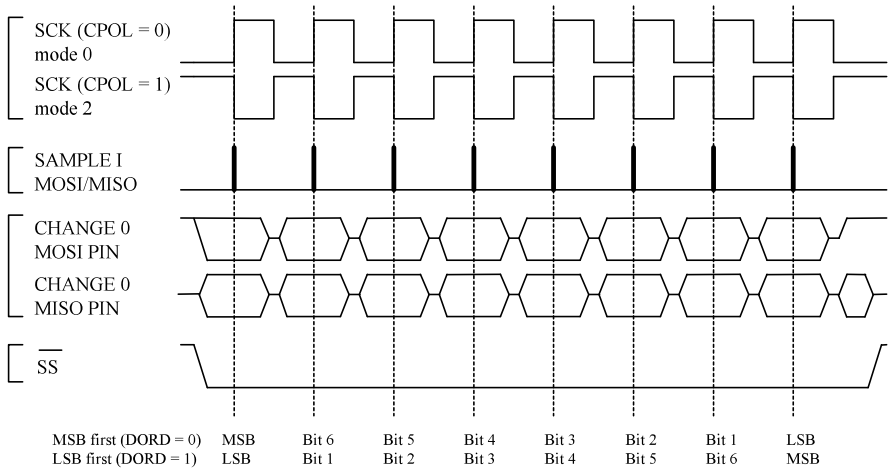
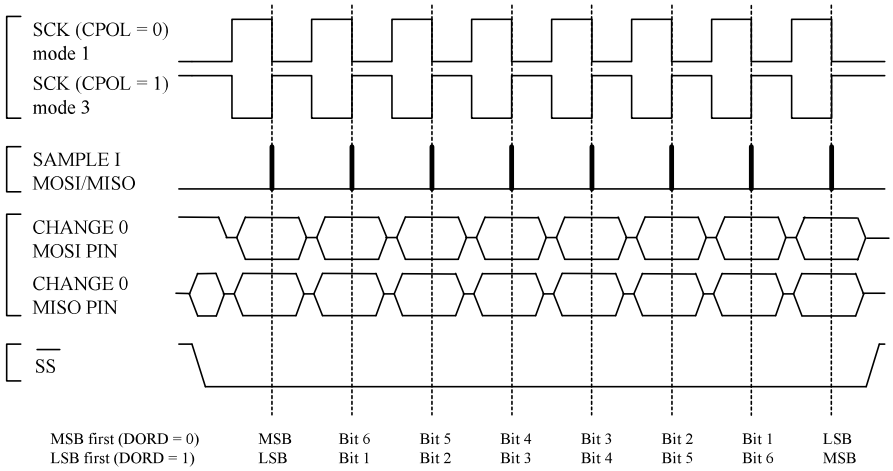


Figure 23-4. SPI Transfer Format with CPHA = 1



23.5. Register Description

23.5.1. SPI Control Register 0

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: SPCR0
Offset: 0x4C
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2C

Bit	7	6	5	4	3	2	1	0
	SPIE0	SPE0	DORD0	MSTR0	CPOL0	CPHA0	SPR01	SPR00
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – SPIE0: SPI0 Interrupt Enable
This bit causes the SPI interrupt to be executed if SPIF bit in the SPSR Register is set and if the Global Interrupt Enable bit in SREG is set.

Bit 6 – SPE0: SPI0 Enable
When the SPE bit is written to one, the SPI is enabled. This bit must be set to enable any SPI operations.

Bit 5 – DORD0: Data0 Order
When the DORD bit is written to one, the LSB of the data word is transmitted first.

When the DORD bit is written to zero, the MSB of the data word is transmitted first.

Bit 4 – MSTR0: Master/Slave0 Select
This bit selects Master SPI mode when written to one, and Slave SPI mode when written logic zero. If SS is configured as an input and is driven low while MSTR is set, MSTR will be cleared, and SPIF in SPSR will become set. The user will then have to set MSTR to re-enable SPI Master mode.

Bit 3 – CPOL0: Clock0 Polarity
When this bit is written to one, SCK is high when idle. When CPOL is written to zero, SCK is low when idle. Refer to Figure 23-3 and Figure 23-4 for an example. The CPOL functionality is summarized below:

Table 23-3. CPOL0 Functionality

CPOL0	Leading Edge	Trailing Edge
0	Rising	Falling
1	Falling	Rising

Bit 2 – CPHA0: Clock0 Phase
The settings of the Clock Phase bit (CPHA) determine if data is sampled on the leading (first) or trailing (last) edge of SCK. Refer to Figure 23-3 and Figure 23-4 for an example. The CPHA functionality is summarized below:

Table 23-4. CPHA0 Functionality

CPHA0	Leading Edge	Trailing Edge
0	Sample	Setup
1	Setup	Sample

Bits 1:0 – SPR0n: SPI0 Clock Rate Select n [n = 1:0]

These two bits control the SCK rate of the device configured as a Master. SPR1 and SPR0 have no effect on the Slave. The relationship between SCK and the Oscillator Clock frequency f_{osc} is shown in the table below.

Table 23-5. Relationship between SCK and Oscillator Frequency

SPI2X	SPR01	SPR00	SCK Frequency
0	0	0	$f_{osc}/4$
0	0	1	$f_{osc}/16$
0	1	0	$f_{osc}/64$
0	1	1	$f_{osc}/128$
1	0	0	$f_{osc}/2$
1	0	1	$f_{osc}/8$
1	1	0	$f_{osc}/32$
1	1	1	$f_{osc}/64$

23.5.2. SPI Status Register 0

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: SPSR0
Offset: 0x4D
Reset: 0x00
Property: When addressing as I/O Register: address offset is 0x2D

Bit	7	6	5	4	3	2	1	0
	SPIF0	WCOL0						SPI2X0
Access	R	R						R/W
Reset	0	0						0

Bit 7 – SPIF0: SPI Interrupt Flag

When a serial transfer is complete, the SPIF Flag is set. An interrupt is generated if SPIE in SPCR is set and global interrupts are enabled. If $\overline{SS}$ is an input and is driven low when the SPI is in Master mode, this will also set the SPIF Flag. SPIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, the SPIF bit is cleared by first reading the SPI Status Register with SPIF set, then accessing the SPI Data Register (SPDR).

Bit 6 – WCOL0: Write Collision Flag

The WCOL bit is set if the SPI Data Register (SPDR) is written during a data transfer. The WCOL bit (and the SPIF bit) are cleared by first reading the SPI Status Register with WCOL set, and then accessing the SPI Data Register.

Bit 0 – SPI2X0: Double SPI Speed Bit

When this bit is written logic one the SPI speed (SCK Frequency) will be doubled when the SPI is in Master mode (refer to Table 23-5). This means that the minimum SCK period will be two CPU clock periods. When the SPI is configured as Slave, the SPI is only guaranteed to work at fosc/4 or lower.

The SPI interface is also used for program memory and EEPROM downloading or uploading. See *Serial Downloading* for serial programming and verification.

23.5.3. SPI Data Register 0

When addressing I/O Registers as data space using LD and ST instructions, the provided offset must be used. When using the I/O specific commands IN and OUT, the offset is reduced by 0x20, resulting in an I/O address offset within 0x00 - 0x3F.

Name: SPDR0
Offset: 0x4E
Reset: 0xXX
Property: When addressing as I/O Register: address offset is 0x2E

Bit	7	6	5	4	3	2	1	0
	SPID[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 7:0 – SPID[7:0]: SPI Data

The SPI Data Register is a read/write register used for data transfer between the Register File and the SPI Shift Register. Writing to the register initiates data transmission. Reading the register causes the Shift Register Receive buffer to be read.