# 22. SPI – Serial Peripheral Interface

## 22.1 Features

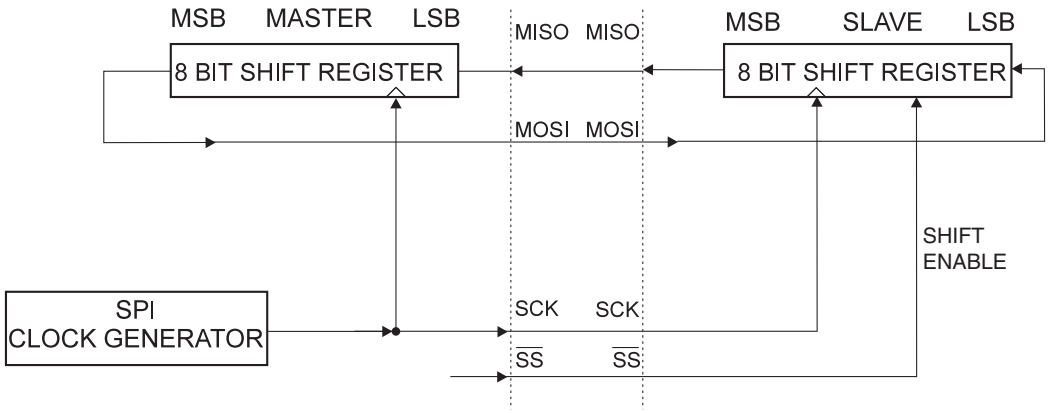
- Full-duplex, three-wire synchronous data transfer
- Master or slave operation
- Lsb first or msb first data transfer
- Eight programmable bit rates
- Interrupt flag at the end of transmission
- Write collision flag to indicate data collision
- Wake up from idle sleep mode
- Double speed master mode

## 22.2 Overview

The Serial Peripheral Interface (SPI) is a high-speed synchronous data transfer interface using three or four pins. It allows fast communication between an XMEGA device and peripheral devices or between several microcontrollers. The SPI supports full-duplex communication.

A device connected to the bus must act as a master or slave.The master initiates and controls all data transactions. The interconnection between master and slave devices with SPI is shown in Figure 22-1 on page 273. The system consists of two shift registers and a master clock generator. The SPI master initiates the communication cycle by pulling the slave select ($\overline{SS}$) signal low for 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 are always shifted from master to slave on the master output, slave input (MOSI) line, and from slave to master on the master input, slave output (MISO) line. After each data packet, the master can synchronize the slave by pulling the $\overline{SS}$ line high.

**Figure 22-1. SPI master-slave interconnection.**




The SPI module is unbuffered in the transmit direction and single 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, a received character must be read from the DATA register before the next character has been completely shifted in. Otherwise, the first byte will be lost.

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

When the SPI module is enabled, the data direction of the MOSI, MISO, SCK, and $\overline{SS}$ pins is overridden according to Table 22-1 on page 274. The pins with user-defined direction must be configured from software to have the correct direction according to the application.

**Table 22-1. SPI pin override and directions.**

| Pin | Master mode | Slave mode |
|---|---|---|
| MOSI | User defined | Input |
| MISO | Input | User defined |
| SCK | User defined | Input |
| $\overline{SS}$ | User defined | Input |

## 22.3 Master Mode

In master mode, the SPI interface has no automatic control of the $\overline{SS}$ line. If the $\overline{SS}$ pin is used, it must be configured as output and controlled by user software. If the bus consists of several SPI slaves and/or masters, a SPI master can use general purpose I/O pins to control the $\overline{SS}$ line to each of the slaves on the bus.

Writing a byte to the DATA register starts the SPI clock generator and the hardware shifts the eight bits into the selected slave. After shifting one byte, the SPI clock generator stops and the SPI interrupt flag is set. The master may continue to shift the next byte by writing new data to the DATA register, or can signal the end of the transfer by pulling the $\overline{SS}$ line high. The last incoming byte will be kept in the buffer register.

If the $\overline{SS}$ pin is not used and is configured as input, it must be held high to ensure master operation. If the $\overline{SS}$ pin is set as input and is being driven low, the SPI module will interpret this as another master trying to take control of the bus. To avoid bus contention, the master will take the following action:

1. The master enters slave mode.
2. The SPI interrupt flag is set.

## 22.4 Slave Mode

In slave mode, the SPI module will remain sleeping with the MISO line tri-stated as long as the $\overline{SS}$ pin is driven high. In this state, software may update the contents of the DATA register, but the data will not be shifted out by incoming clock pulses on the SCK pin until the $\overline{SS}$ pin is driven low. If $\overline{SS}$ is driven low, the slave will start to shift out data on the first SCK clock pulse. When one byte has been completely shifted, the SPI interrupt flag is set. The slave may continue placing new data to be sent into the DATA register before reading the incoming data. The last incoming byte will be kept in the buffer register.

When $\overline{SS}$ is driven high, the SPI logic is reset, and the SPI slave will not receive any new data. Any partially received packet in the shift register will be dropped.

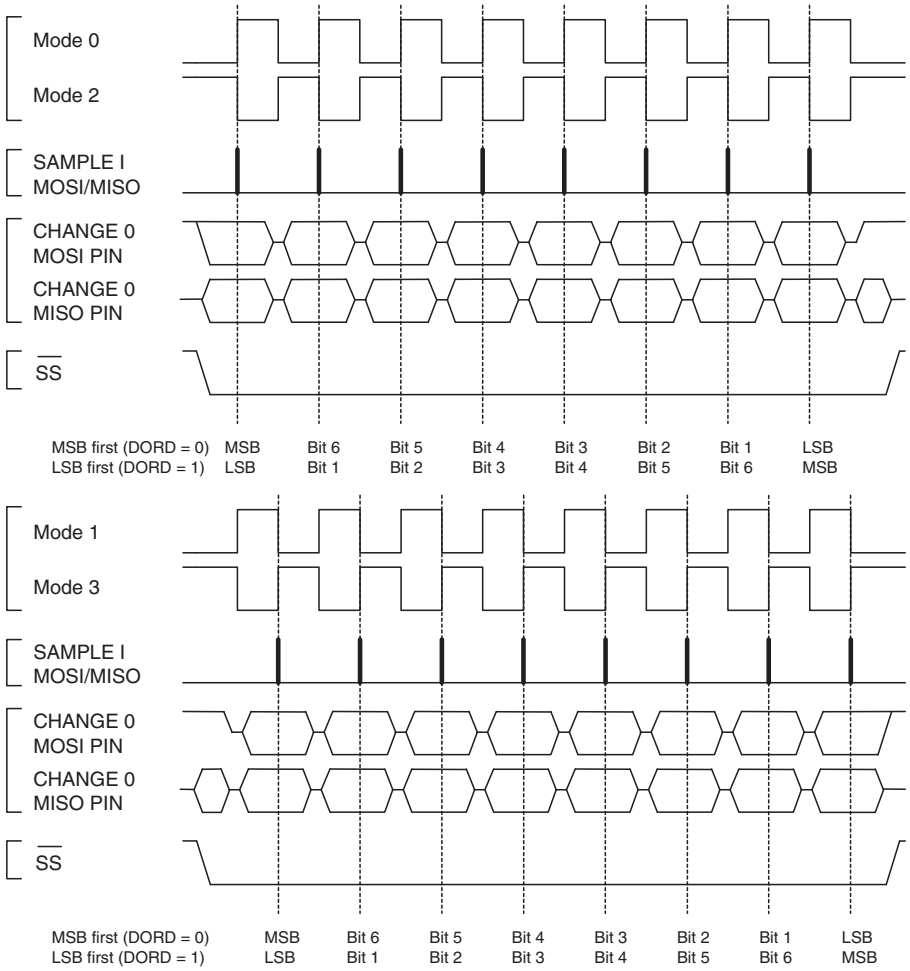
As the $\overline{SS}$ pin is used to signal the start and end of a transfer, it is also useful for doing packet/byte synchronization, keeping the slave bit counter synchronous with the master clock generator.

## 22.5 Data Modes

There are four combinations of SCK phase and polarity with respect to serial data. The SPI data transfer formats are shown in Figure 22-2. Data bits are shifted out and latched in on opposite edges of the SCK signal, ensuring sufficient time for data signals to stabilize.

The leading edge is the first clock edge of a clock cycle. The trailing edge is the last clock edge of a clock cycle.

**Figure 22-2. SPI transfer modes.**




## 22.6 DMA Support

DMA support on the SPI module is available only in slave mode. The SPI slave can trigger a DMA transfer as one byte has been shifted into the DATA register. It is possible, however, to use the XMEGA USART in SPI mode and then have DMA support in master mode. For details, refer to “USART in Master SPI Mode” on page 291.

## 22.7 Register Description

### 22.7.1 CTRL – Control register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| +0x00 | CLK2X | ENABLE | DORD | MASTER | MODE[1:0] | | PRESCALER[1:0] | |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- **Bit 7 – CLK2X: Clock Double**
  When this bit is set, the SPI speed (SCK frequency) will be doubled in master mode (see Table 22-3 on page 291).

- **Bit 6 – ENABLE: Enable**
  Setting this bit enables the SPI module. This bit must be set to enable any SPI operations.

- **Bit 5 – DORD: Data Order**
  DORD decides the data order when a byte is shifted out from the DATA register. When DORD is written to one, the least-significant bit (lsb) of the data byte is transmitted first, and when DORD is written to zero, the most-significant bit (msb) of the data byte is transmitted first.

- **Bit 4 – MASTER: Master Select**
  This bit selects master mode when written to one, and slave mode when written to zero. If $\overline{SS}$ is configured as an input and driven low while master mode is set, master mode will be cleared.

- **Bit 3:2 – MODE[1:0]: Transfer Mode**
  These bits select the transfer mode. The four combinations of SCK phase and polarity with respect to the serial data are shown in Table 22-2 on page 276. These bits decide whether the first edge of a clock cycle (leading edge) is rising or falling, and whether data setup and sample occur on the leading or trailing edge.
  When the leading edge is rising, the SCK signal is low when idle, and when the leading edge is falling, the SCK signal is high when idle.

**Table 22-2. SPI transfer modes.**

| MODE[1:0] | Group configuration | Leading edge | Trailing edge |
|---|---|---|---|
| 00 | 0 | Rising, sample | Falling, setup |
| 01 | 1 | Rising, setup | Falling, sample |
| 10 | 2 | Falling, sample | Rising, setup |
| 11 | 3 | Falling, setup | Rising, sample |

- **Bits 1:0 – PRESCALER[1:0]: Clock Prescaler**
  These two bits control the SPI clock rate configured in master mode. These bits have no effect in slave mode. The relationship between SCK and the peripheral clock frequency ( $clk_{PER}$) is shown in Table 22-3 on page 277.

**Table 22-3. Relationship between SCK and the peripheral clock ($Clk_{PER}$) frequency.**

| CLK2X | PRESCALER[1:0] | SCK frequency |
|---|---|---|
| 0 | 00 | $Clk_{PER}/4$ |
| 0 | 01 | $Clk_{PER}/16$ |
| 0 | 10 | $Clk_{PER}/64$ |
| 0 | 11 | $Clk_{PER}/128$ |
| 1 | 00 | $Clk_{PER}/2$ |
| 1 | 01 | $Clk_{PER}/8$ |
| 1 | 10 | $Clk_{PER}/32$ |
| 1 | 11 | $Clk_{PER}/64$ |

### 22.7.2 INTCTRL – Interrupt Control register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| +0x01 | – | – | – | – | – | – | INTLVL[1:0] | |
| Read/Write | R | R | R | R | R | R | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- **Bit 7:2 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- **Bit 1:0 – INTLVL[1:0]: Interrupt Level**
  These bits enable the SPI interrupt and select the interrupt level, as described in “Interrupts and Programmable Multilevel Interrupt Controller” on page 131. The enabled interrupt will be triggered when IF in the STATUS register is set.

### 22.7.3 STATUS – Status register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| +0x02 | IF | WRCOL | – | – | – | – | – | – |
| Read/Write | R | R | R | R | R | R | R | R |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- **Bit 7 – IF: Interrupt Flag**
  This flag is set when a serial transfer is complete and one byte is completely shifted in/out of the DATA register. If $\overline{SS}$ is configured as input and is driven low when the SPI is in master mode, this will also set this flag. IF is cleared by hardware when executing the corresponding interrupt vector. Alternatively, the IF flag can be cleared by first reading the STATUS register when IF is set, and then accessing the DATA register.

- **Bit 6 – WRCOL: Write Collision Flag**
  The WRCOL flag is set if the DATA register is written during a data transfer. This flag is cleared by first reading the STATUS register when WRCOL is set, and then accessing the DATA register.

- **Bit 5:0 – Reserved**
  These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

### 22.7.4 DATA – Data register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| +0x03 | DATA[7:0] | | | | | | | |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The DATA register is used for sending and receiving data. Writing to the register initiates the data transmission, and the byte written to the register will be shifted out on the SPI output line. Reading the register causes the shift register receive buffer to be read, returning the last byte successfully received.

## 22.8 Register summary

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Page |
|---|---|---|---|---|---|---|---|---|---|---|
| +0x00 | CTRL | CLK2X | ENABLE | DORD | MASTER | MODE[1:0] | | PRESCALER[1:0] | | 276 |
| +0x01 | INTCTRL | – | – | – | – | – | – | INTLVL[1:0] | | 277 |
| +0x02 | STATUS | IF | WRCOL | – | – | – | – | – | – | 277 |
| +0x03 | DATA | DATA[7:0] | | | | | | | | 278 |

## 22.9 Interrupt vector summary

**Table 22-4. SPI interrupt vector and its offset word address.**

| Offset | Source | Interrupt Description |
|---|---|---|
| 0x00 | SPI_vect | SPI interrupt vector |