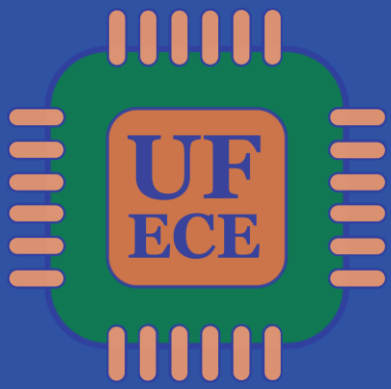
# Microprocessor Applications

## *XMEGA: SPI*




UF UNIVERSITY of FLORIDA

*Dr. Eric Schwartz*
*Christopher Cr*
*Wesley Piard*

1



## XMEGA SPI Specifications

The XMEGA has a total of four Serial Peripheral Interface (**SPI**) modules, one on each of Ports C, D, E, and F.

- Each SPI module on the XMEGA consists of four pins:
  - $\overline{\textbf{SS}}$: Slave Select
  - **MOSI**: Master-out Slave-in
  - **MISO**: Master-in Slave-out
  - **SCK**: Serial Clock
- 8-bit data
- LSb or MSb first
- The naming convention for labeling each individual module is SPI*p*, where *p* specifies the port:
  - *p* : {C, D, E, F}
  - For example, **Port F** has a SPI module labeled **SPIF**.

# Alternate Pin Function Table

- Section 33.2 of doc8385 has a table of alternate pin functions for each port.
- The table for Port F is shown here. See the column labeled "SPIF."

| PORT F | PIN # | INTERRUPT | TCF0 | TCF1 | USARTF0 | USARTF1 | SPIF | TWIF |
|---|---|---|---|---|---|---|---|---|
| GND | 43 | | | | | | | |
| VCC | 44 | | | | | | | |
| PF0 | 45 | SYNC | OC0A | | | | | SDA |
| PF1 | 46 | SYNC | OC0B | | XCK0 | | | SCL |
| PF2 | 47 | SYNC/ASYNC | OC0C | | RXD0 | | | |
| PF3 | 48 | SYNC | OC0D | | TXD0 | | | |
| PF4 | 49 | SYNC | | OC1A | | | $\overline{SS}$ | |
| PF5 | 50 | SYNC | | OC1B | | XCK1 | MOSI | |
| PF6 | 51 | SYNC | | | | RXD1 | MISO | |
| PF7 | 52 | SYNC | | | | TXD1 | SCK | |

doc8385
Table 33-6



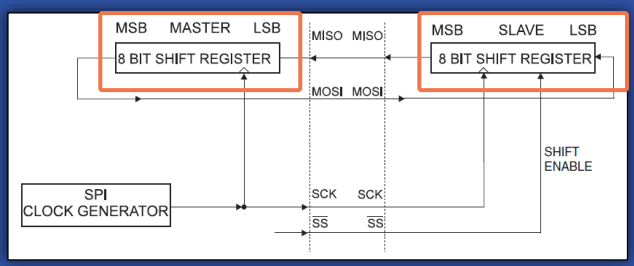




# Block Diagram

Here is the block diagram for an SPI module.

- An 8-bit shift register exists in the master device as well as any slave device(s).
- Each shift register is controlled/synchronized by the clock (SCK) signal that is controlled by the master device.
- The $\overline{SS}$ signal is what enables the slave device and the SCK signal is what clocks or shifts in each bit of data.




doc8331
Figure 22-1

# Overall Functionality – Master Mode

Master Mode

- The SPI module will initiate data transfers and generate the clock signal.
- If there are multiple slave devices on the bus, I/O pins will need to be used as a chip selects for *each* slave device.
- The master device typically pulls the chip select signal low (for the respective slave device) and then begins the data transmission.
  - Note that for simple applications where a single slave is on the bus, *it may be possible to* permanently enable the slave's chip select. This, however, is **all dependent on the slave**, as not all slave devices can operate this way.
- After all the data has been shifted out, the master sets the chip select high again (if a chip select is used).





# Overall Functionality – Slave Mode

Slave Mode

- The SPI module will wait in an idle state, unless the $\overline{SS}$ pin is pulled low.
- If the $\overline{SS}$ pin is pulled low, the internal shift register will be enabled, allowing data to be shifted in on each configured clock edge.
- If the $\overline{SS}$ pin is not pulled low, any information on the data or serial clock lines will be ignored.

# Pin Functionality (Master Mode) – Slave Select

- The XMEGA's Slave Select ($\overline{SS}$) pin of each SPI module is only *required* to be used when the SPI module is operating in slave mode.
- In master mode, the $\overline{SS}$ pin may be used as a chip select to enable slave devices, but in reality, any available I/O pin(s) can be utilized.
- If the $\overline{SS}$ pin is not used, and another I/O pin is used instead, **great care must be taken to ensure that the $\overline{SS}$ pin is not pulled low**, or the SPI module will revert to slave mode.
  - If the $\overline{SS}$ pin is not configured as an output, a pull-up resistor should be connected, since by default the pin is an input.

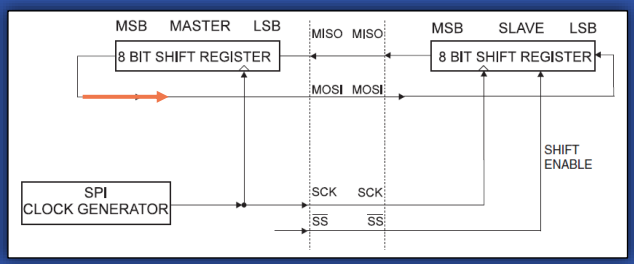






# Pin Functionality (Master Mode) – MOSI

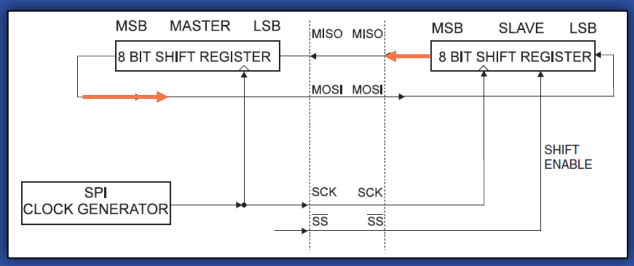
- The MOSI pin must be configured as an output when a SPI module is to be configured in master mode.
  - A bit from the shift register will be shifted out via this pin at each relevant clock edge.




doc8331 Figure 22-1

# Pin Functionality (Master Mode) –MISO

- The MISO pin will be overridden by the SPI module and configured as an input for master mode.
  - Each clock edge in which a bit is shifted out via MOSI, a bit is shifted in via MISO.




doc8331 Figure 22-1

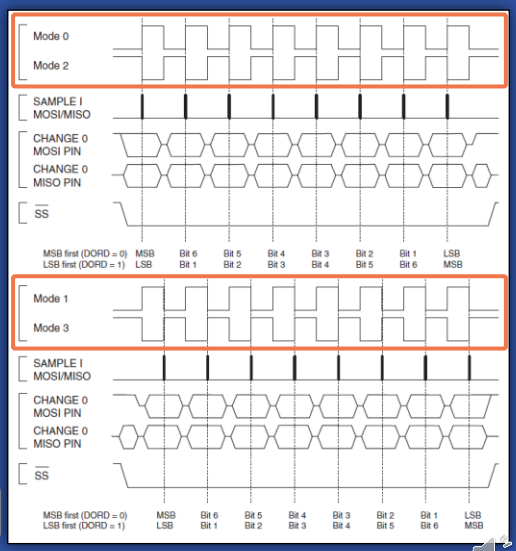






# Pin Functionality (Master Mode) – Serial Clock (SCK)

- The SCK pin must be configured as an output when a SPI module is to be configured in master mode.
  - The XMEGA supports the four standard clocking modes that most SPI devices support.
  - Each of these clock modes correspond to a unique combination of clock polarity and phase.
  - For communication to be successful, the master and slave devices on the same bus must operate using the same mode.




doc8331 Figure 22-2

# SPI Initialization Procedure (Master Mode)

- For master mode, SPI should be initialized as follows:

1) Configure the MOSI and SCK pins as outputs. (*PORTx_DIR*)
2) Choose, if applicable, an available I/O pin to use as a chip select (one for each slave device on the bus). Each of these I/O pins should be configured as an output with a high voltage upon initialization. Additionally, if the $\overline{SS}$ pin is not used, ensure that it is either set as an output, or an input with a pull-up resistor. (*PORTx_OUT, PORTx_DIR*)
3) Set the order in which each byte is transmitted (LSb vs MSb first). (*CTRL*)
4) Configure the "transfer mode" (clock phase and polarity). (*CTRL*)
5) Set the serial clock frequency. (*CTRL*)
6) Enable master mode. (*CTRL*)
7) Enable the SPI module. (*CTRL*)
8) Optional: If SPI interrupts are desired, choose a desired interrupt priority level at the SPI peripheral level. (*INTCTRL*)





# SPI Initialization Procedure (Master Mode)

- For master mode, SPI should be initialized as follows:

1) Configure the MOSI and SCK pins as outputs. (*PORTx_DIR*)
2) Choose, if applicable, an available I/O pin to use as a chip select (one for each slave device on the bus). Each of these I/O pins should be configured as an output with a high voltage upon initialization. Additionally, if the $\overline{SS}$ pin is not used, ensure that it is either set as an output, or an input with a pull-up resistor. (*PORTx_OUT, PORTx_DIR*)
3) Set the order in which each byte is transmitted (LSb vs MSb first). (*CTRL*)
4) Configure the "transfer mode" (clock phase and polarity). (*CTRL*)
5) Set the serial clock frequency. (*CTRL*)
6) Enable master mode. (*CTRL*)
7) Enable the SPI module. (*CTRL*)
8) Optional: If SPI interrupts are desired, choose a desired interrupt priority level at the SPI peripheral level. (*INTCTRL*)

13

# Pin I/O Initializations (Master Mode)

- In master mode, the **MOSI** and **SCK** pins must be configured as **outputs**.
- Each chip select I/O pin must be configured as an output with (in most cases) a high voltage value upon initialization.
- Note that in some instances, a chip select signal may not be required, such as when there is only a single slave device on the bus and said slave device's chip select pin can be permanently enabled, i.e., tied to 0V.
  - This may not always be possible, since some slave devices require the falling and rising edges of the chip select signal to function properly.



14

# SPI Initialization Procedure (Master Mode)

- For master mode, SPI should be initialized as follows:

1) ~~Configure the MOSI and SCK pins as outputs. (*PORTx_OUT*)~~
2) ~~Choose, if applicable, an available I/O pin to use as a chip select (one for each slave device on the bus). Each of these I/O pins should be configured as an output with a high voltage upon initialization. Additionally, if the $\overline{SS}$ pin is not used, ensure that it is either set as an output, or an input with a pull-up resistor. (*PORTx_OUT, PORTx_DIR*)~~
3) Set the order in which each byte is transmitted (LSb vs MSb first). (*CTRL*)
4) Configure the "transfer mode" (clock phase and polarity). (*CTRL*)
5) Set the serial clock frequency. (*CTRL*)
6) Enable master mode. (*CTRL*)
7) Enable the SPI module. (*CTRL*)
8) Optional: If SPI interrupts are desired, choose a desired interrupt priority level at the SPI peripheral level. (*INTCTRL*)

# Initialization – Data Order

- The Data Order (**DORD**) bit determines the order in which the bits of each data byte are shifted in and out.
  - When DORD = 1, the least-significant bit (LSb) is interchanged first.
  - When DORD = 0, the most-significant bit (MSb) is interchanged first.

CTRL

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CLK2X | ENABLE | DORD | MASTER | MODE[1:0] | | PRESCALER[1:0] | |

doc8331
Section 22.7.1





# SPI Initialization Procedure (Master Mode)

- For master mode, SPI should be initialized as follows:

1) ~~Configure the MOSI and SCK pins as outputs. (*PORTx_OUT*)~~
2) ~~Choose, if applicable, an available I/O pin to use as a chip select (one for each slave device on the bus). Each of these I/O pins should be configured as an output with a high voltage upon initialization. Additionally, if the $\overline{SS}$ pin is not used, ensure that it is either set as an output, or an input with a pull-up resistor. (*PORTx_OUT*, *PORTx_DIR*)~~
3) ~~Set the order in which each byte is transmitted (LSb vs MSb first). (*CTRL*)~~
4) Configure the "transfer mode" (clock phase and polarity). (*CTRL*)
5) Set the serial clock frequency. (*CTRL*)
6) Enable master mode. (*CTRL*)
7) Enable the SPI module. (*CTRL*)
8) Optional: If SPI interrupts are desired, choose a desired interrupt priority level at the SPI peripheral level. (*INTCTRL*)

# Initialization – Transfer Mode

- The Transfer Mode (**MODE[1:0]**) bitfield determines the phase and polarity of the clock signal.
  - The clock phase determines which edge of the clock signal data is sampled.
  - The clock polarity determines the idle state of the clock signal.

| 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 |

doc8331 Table 22-2

CTRL

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CLK2X | ENABLE | DORD | MASTER | MODE[1:0] | | PRESCALER[1:0] | |

doc8331 Section 22.7.1



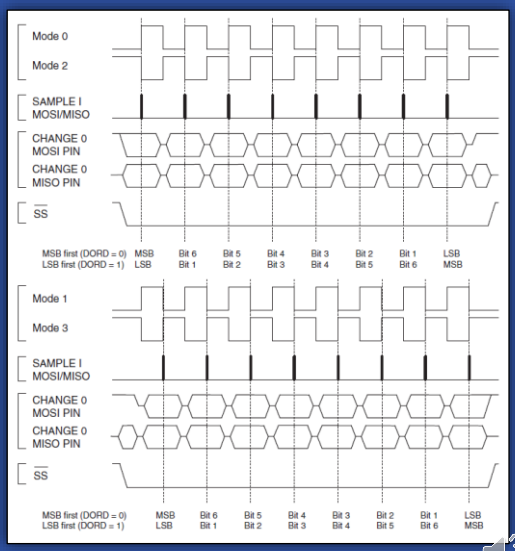




# Initialization – Transfer Mode

- The table below in conjunction with the diagram to the right is used to select a transfer mode.

doc8331 Figure 22-2




doc8331 Table 22-2

| 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 |

# SPI Initialization Procedure (Master Mode)

- For master mode, SPI should be initialized as follows:
  1) ~~Configure the MOSI and SCK pins as outputs. (*PORTx_OUT*)~~
  2) ~~Choose, if applicable, an available I/O pin to use as a chip select (one for each slave device on the bus). Each of these I/O pins should be configured as an output with a high voltage upon initialization. Additionally, if the $\overline{SS}$ pin is not used, ensure that it is either set as an output, or an input with a pull-up resistor. (*PORTx_OUT, PORTx_DIR*)~~
  3) ~~Set the order in which each byte is transmitted (LSb vs MSb first). (*CTRL*)~~
  4) ~~Configure the "transfer mode" (clock phase and polarity). (*CTRL*)~~
  5) Set the serial clock frequency. (*CTRL*)
  6) Enable master mode. (*CTRL*)
  7) Enable the SPI module. (*CTRL*)
  8) Optional: If SPI interrupts are desired, choose a desired interrupt priority level at the SPI peripheral level. (*INTCTRL*)







# Initialization – Clock Frequency

- The **CLK2X** bit and the Clock Prescaler (**PRESCALER[1:0]**) bitfield determine the serial clock frequency. These bits only have an effect in master mode.
  - A prescaler of 4, 16, 64, or 128 is applied to the peripheral clock to generate the serial clock frequency.
  - Optionally, the CLK2X bit can be set to double the generated frequency.
  - This produces a total of eight different frequencies for a given peripheral clock 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 | $Clk_{PER}/64$ |

doc8331 Table 22-2

CTRL

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CLK2X | ENABLE | DORD | MASTER | MODE[1:0] | | PRESCALER[1:0] | |

doc8331 Section 22.7.1

21

# SPI Initialization Procedure (Master Mode)

- For master mode, SPI should be initialized as follows:
  1) ~~Configure the MOSI and SCK pins as outputs. (*PORTx_OUT*)~~
  2) ~~Choose, if applicable, an available I/O pin to use as a chip select (one for each slave device on the bus). Each of these I/O pins should be configured as an output with a high voltage upon initialization. Additionally, if the $\overline{SS}$ pin is not used, ensure that it is either set as an output, or an input with a pull-up resistor. (*PORTx_OUT, PORTx_DIR*)~~
  3) ~~Set the order in which each byte is transmitted (LSb vs MSb first). (*CTRL*)~~
  4) ~~Configure the "transfer mode" (clock phase and polarity). (*CTRL*)~~
  5) ~~Set the serial clock frequency. (*CTRL*)~~
  6) Enable master mode. (*CTRL*)
  7) Enable the SPI module. (*CTRL*)
  8) Optional: If SPI interrupts are desired, choose a desired interrupt priority level at the SPI peripheral level. (*INTCTRL*)



22

# Initialization – Master Mode and SPI Enable

- The **MASTER** bit determines whether the SPI module operates in master or slave mode.
  - When MASTER = 1, the SPI module will operate in master mode.
  - When MASTER = 0, the SPI module will operate in slave mode.
- The **ENABLE** bit enables the SPI module. This bit must be set for any data transmission or reception to occur.

CTRL

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CLK2X | ENABLE | DORD | MASTER | MODE[1:0] | | PRESCALER[1:0] | |

doc8331
Section 22.7.1

# SPI Initialization Procedure (Master Mode)

- For master mode, SPI should be initialized as follows:
  1) ~~Configure the MOSI and SCK pins as outputs. (*PORTx_OUT*)~~
  2) ~~Choose, if applicable, an available I/O pin to use as a chip select (one for each slave device on the bus). Each of these I/O pins should be configured as an output with a high voltage upon initialization. Additionally, if the $\overline{SS}$ pin is not used, ensure that it is either set as an output, or an input with a pull-up resistor. (*PORTx_OUT, PORTx_DIR*)~~
  3) ~~Set the order in which each byte is transmitted (LSb vs MSb first). (*CTRL*)~~
  4) ~~Configure the "transfer mode" (clock phase and polarity). (*CTRL*)~~
  5) ~~Set the serial clock frequency. (*CTRL*)~~
  6) ~~Enable master mode. (*CTRL*)~~
  7) ~~Enable the SPI module. (*CTRL*)~~
  8) Optional: If SPI interrupts are desired, choose a desired interrupt priority level at the SPI peripheral level. (*INTCTRL*)







# Initialization – Interrupts

- The Interrupt Level (**INTLVL[1:0]**) bitfield enables the SPI interrupt and sets its priority level, as per the PMIC specifications.

doc8331
Table 12-1

| Interrupt level configuration | Group configuration | Description |
|---|---|---|
| 00 | OFF | Interrupt disabled. |
| 01 | LO | Low-level interrupt |
| 10 | MED | Medium-level interrupt |
| 11 | HI | High-level interrupt |

```
; SPIF interrupt vector
.equ SPIF_INT_vect = 236
```

INTCTRL

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | – | – | – | INTLVL[1:0] | |

doc8331
Section 22.7.2

# SPI Status Register – Interrupt Flag

IF (Bit 7)

- This flag is set when a serial transfer is complete.
- Because data is transmitted and received simultaneously with SPI, this has two implications:
  - A full byte of data has been transmitted
  - A full byte of data has been received

STATUS

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| IF | WRCOL | – | – | – | – | – | – |

doc8331
Section 22.7.3





# SPI Data Register

In master mode, the DATA register is used to initiate a data transmission.

- The byte written to the DATA register is shifted out via the MOSI line.
  - As data is shifted out, data is shifted in as well.
- The most recent successfully-received byte is read from this register.

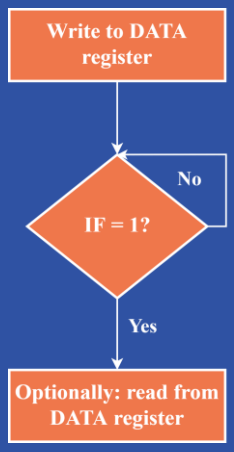
DATA

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DATA[7:0] | | | | | | | |

doc8331
Section 22.7.4

# SPI Data Transmission and Reception (Master Mode)

- To transmit or receive a byte in master mode, data must be written to the DATA register.
  - Yes, to *read* data from a slave device, the master device must *write* data!
- Poll the STATUS register until the IF is set.
  - This indicates the data written to the DATA register was fully transmitted, and the data that was received from a slave device is now available in the DATA register.
- Finally, if desired, the data can be read from the DATA register.
- The interrupt flag will be cleared automatically if either of the following conditions are met:
  - The DATA register is read **after** the STATUS register is read.
  - The SPI interrupt vector is executed.








# Conclusion

- The XMEGA's SPI system is simple, and it can be initialized with almost a single register.
- Even though the system is rather simple, things can still go wrong! It is necessary to study the datasheets for any slave devices in addition to the XMEGA's manual.
- Section 22 of the XMEGA AU manual (doc8331) contains all the details that you may need to learn regarding the SPI system.