ECE3411 – Fall 2016

Lab 6b.

SPI: Serial Peripheral Interface

Marten van Dijk, Chenglu Jin
Department of Electrical & Computer Engineering
University of Connecticut
Email: {marten.van_dijk, chenglu.jin}@uconn.edu

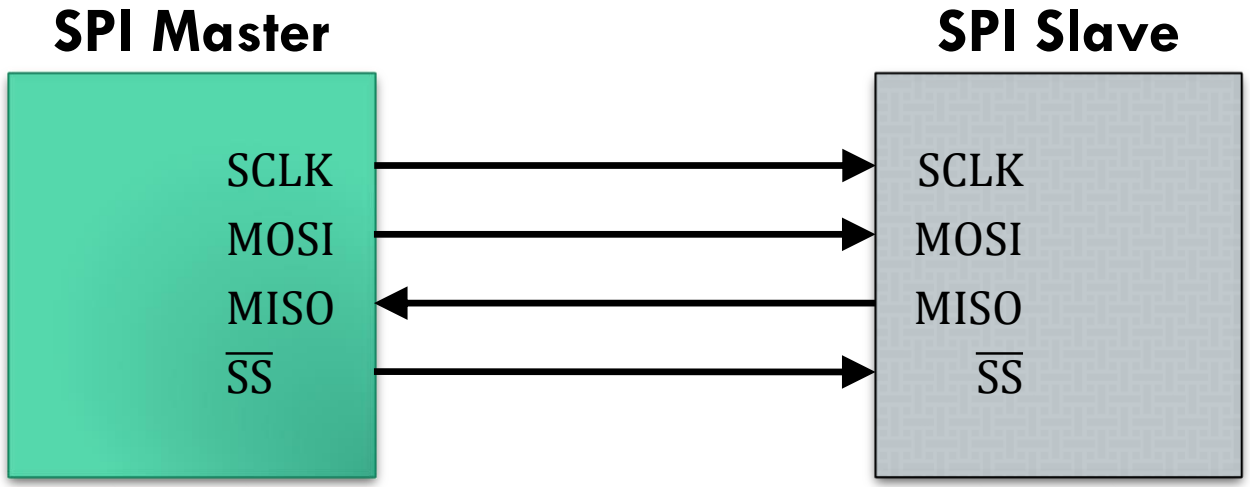
Copied from Lab 6c, ECE3411 – Fall 2015, by
Marten van Dijk and Syed Kamran Haider
With the help of:
www.wikipedia.org
ATmega328P Datasheet

UCONN

SPI: Serial Peripheral Interface

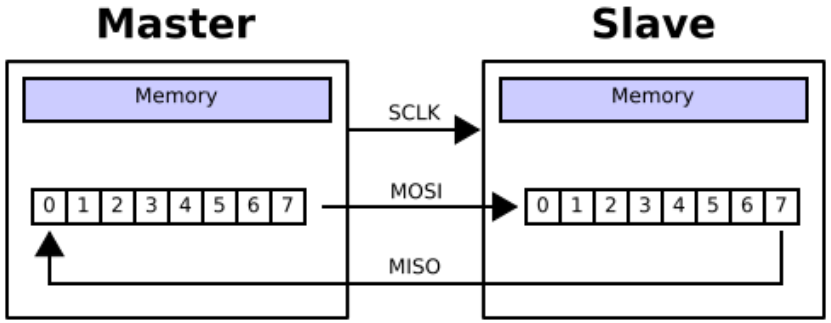
The SPI bus specifies four logic signals:

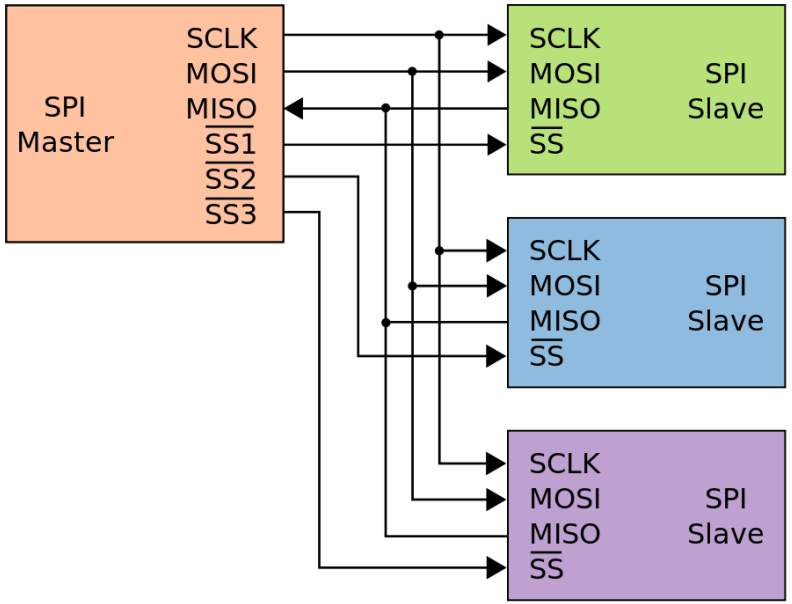
- SCLK : Serial Clock (output from master).
- MOSI : Master Output, Slave Input (output from master).
- MISO : Master Input, Slave Output (output from slave).
- SS : Slave Select (active low, output from master).



SPI Master & Slaves

- The SPI bus can operate with a single Master device and with one or more Slave devices.
- In case of multiple slaves, the master selects the slave device with a logic 0 on the select (SS) line.
- During each SPI clock cycle, the master sends a bit on the MOSI line and the slave reads it, while the slave sends a bit on the MISO line and the master reads it.
- This sequence is maintained even when only one-directional data transfer is intended.





SPI Data Modes

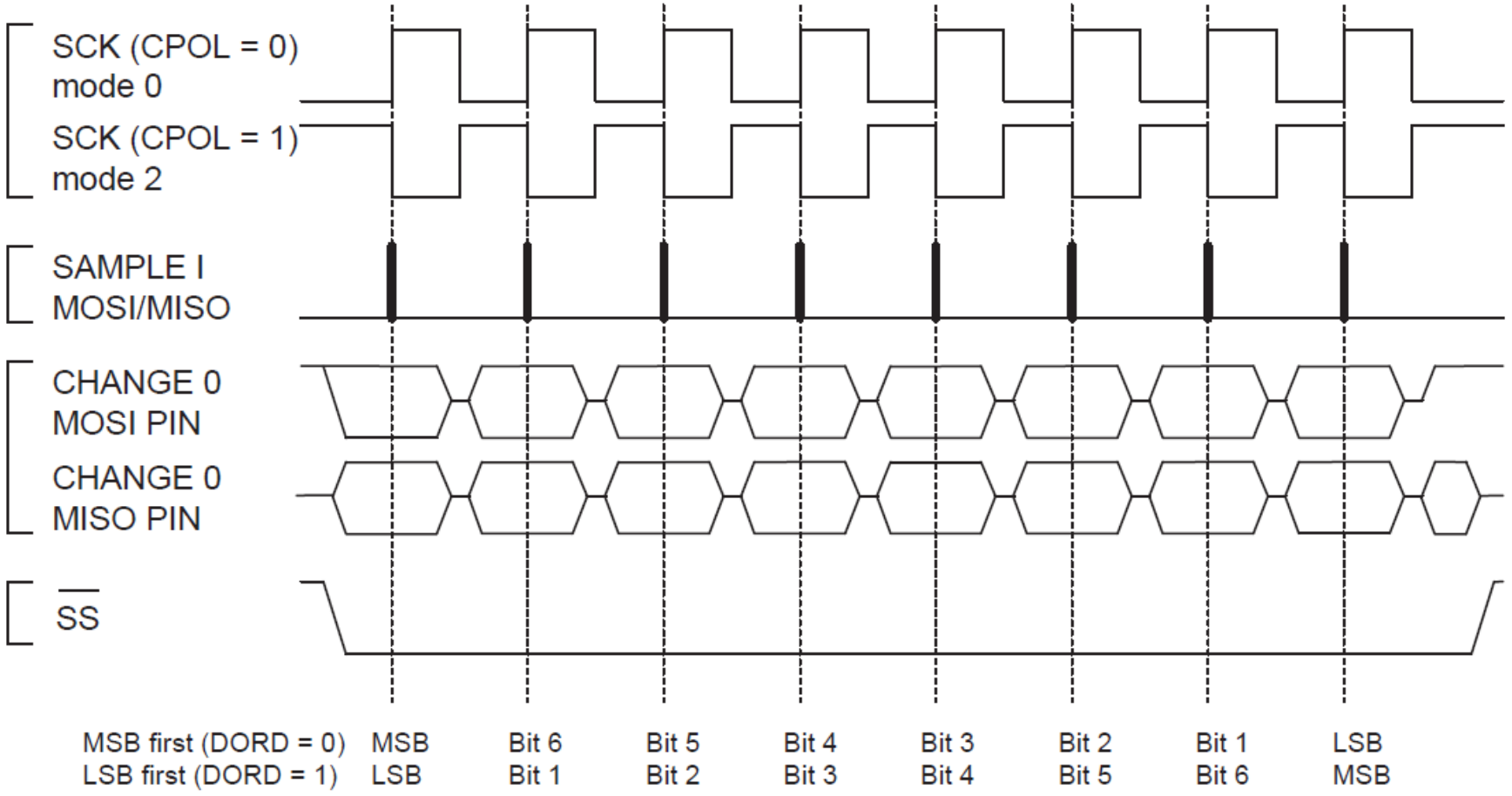
- There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL.

Table 18-2. 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)
2	CPOL=1, CPHA=0	Sample (Falling)	Setup (Rising)
3	CPOL=1, CPHA=1	Setup (Falling)	Sample (Rising)

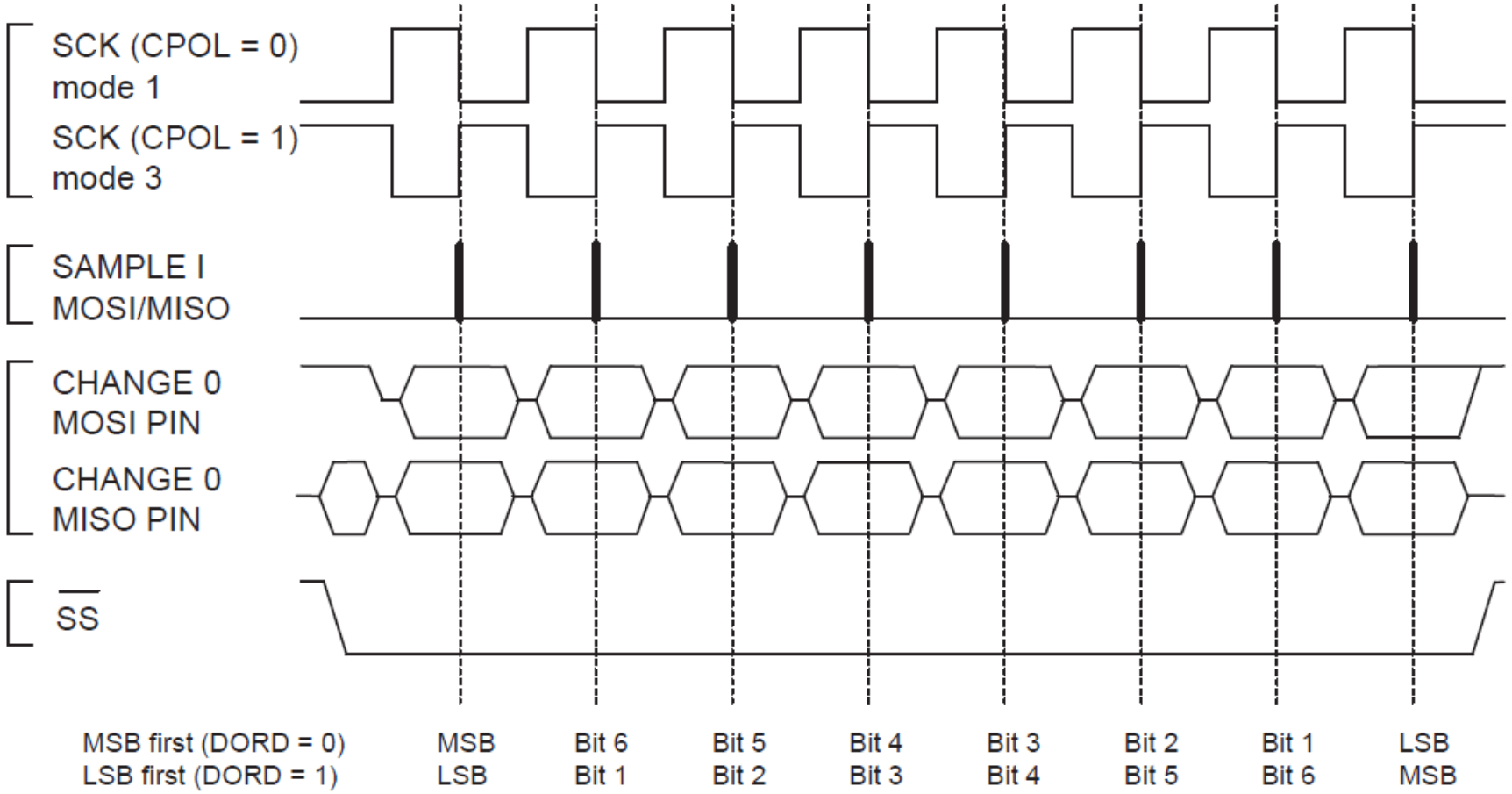
SPI Frame Transfer with CPHA=0

Figure 18-3. SPI Transfer Format with CPHA = 0



SPI Frame Transfer with CPHA=1

Figure 18-4. SPI Transfer Format with CPHA = 1



SPI Master Example

```
void SPI_MasterInit(void)
{
        /* Set SS, MOSI and SCK output, all others input */
        DDR_SPI = (1<<DD_SS) | (1<<DD_MOSI) | (1<<DD_SCK);
        /* Enable SPI, Master, set clock rate fck/128 */
        SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR1) | (1<<SPR0);
}
```

```
uint8_t SPI_Master_Transceiver(uint8_t cData)
{
        PORTB &= ~(1<<SPI_SS);          // Pull Slave_Select low
        SPDR = cData;                   // Start transmission
        while( !(SPSR & (1<<SPIF)) );   // Wait for transmission complete
        PORTB |= (1<<SPI_SS);           // Pull Slave Select High
        return SPDR;                    // Return received data
}
```

Note:
DDR_SPI in the examples must be replaced by the actual Data Direction Register controlling the SPI pins.
DD_SS, 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 PB3, replace DD_MOSI with DDB3 and DDR_SPI with DDRB.
SPI_SS should be replaced with actual bit position of SS pin in the port corresponding to SPI pins.

SPI Slave Example

```
void SPI_SlaveInit(void)
{
        /* Set MISO output, all others input */
        DDR_SPI = (1<<DD_MISO);
        /* Enable SPI */
        SPCR = (1<<SPE);
}
```

```
uint8_t SPI_SlaveReceive(void)
{
        /* Wait for reception complete */
        while(!(SPSR & (1<<SPIF)));
        /* Return Data Register */
        return SPDR;
}
```

Note:
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.

Important Notes

- On Xplained Mini, ATmega328P is programmed by ATmega32U4
- Programming in ISP mode and/or enabling/disabling fuses uses SPI bus.
- If you program in ISP mode, you'll need to restart Atmel Studio every time you program the ATmega328P.
- Therefore, use debugWire interface to program the ATmega328P for this lab.
- REMEMBER: debugWire interface requires DWEN fuse to be enabled, which is done over SPI bus between ATmega32U4 and ATmega328P.
  Therefore, if you plan to connect MOSI and MISO pins together to perform loopback testing of SPI, do so only after entering into debugWire interface and programming the ATmega328P at least once. This will enable the DWEN fuse before the SPI pins MOSI and MISO are shorted together.

Task1: SPI Loopback Testing

Write a simple program to test SPI in loopback mode. In particular:

- Configure SPI in Master mode
- Read a potentiometer's voltage through ADC every 100ms (only upper 8 bits).
- Transmit the byte containing voltage reading over SPI.
- Loopback the transmitted byte by connecting MOSI and MISO pins together according to the instructions given on previous slide.
- Print on LCD the byte value received over SPI.

Task2: SPI Master Slave Communication

Extend Task1 so that you can exchange voltage readings between yours and your friend's board over SPI bus.

- Configure your board as SPI Master and ask your friend to configure his as SPI Slave.
- Make proper wire connections of SS, SCK, MOSI and MISO pins between the two boards.
- The slave MCU should read its ADC value and write it to SPDR register every 50ms.
- Transmit Master's voltage value every 100ms. This will also result in receiving the last voltage value written in SPDR register in the slave MCU.
- For both Master and Slave, print both transmitted and received values on LCD.

Homework: Use SPI interrupts on both Master and Slave sides for non-blocking SPI communication.