ECE3411 – Spring 2019

Lecture 10

SPI: Serial Peripheral Interface

Sung Yeul Park
Department of Electrical & Computer Engineering
University of Connecticut
Email: sung_yeul.park@uconn.edu

With the help of:

www.wikipedia.org

ATmega328P Datasheet

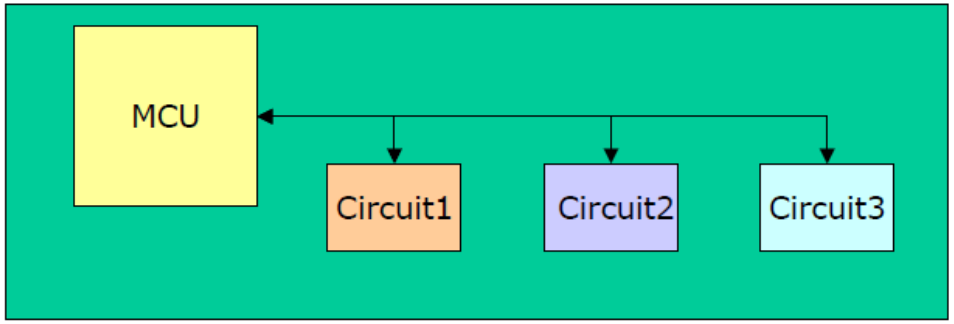
Copied from ECE3411 – Fall 2015,
by Marten van Dijk and Syed Kamran Haider

Bus and Communication Interfaces

- Parallel Bus Systems
  - Processor Buses – AVR etc.
  - Industrial Buses
    - VMEbus
    - CompactPCI
    - PC/104
    - …
- Serial Local Buses
  - SPI
  - MicroWire
  - I2C
  - 1-Wire
- Serial Lines (1 to 1, 1 to N)
  - UART
  - RS-232C
  - RS-422
  - USB
- Networks (N to M)
  - CAN
  - RS-485
  - LAN/Ethernet
- Wireless Communication
  - IR/IrDA
  - ISM
  - WiFi
  - Bluetooth
  - Zigbee

Serial synchronous interfaces

- Local serial interconnection of microcontrollers and peripheral circuits/functions
- Required features:
  - Low complexity
  - Low to medium data rate
  - Small physical footprint/few pins
  - Short distances
  - Low cost
- Most MCUs have built-in peripheral units for communicating with external circuits, e.g. ATmegaAVR (SPI and TWI (I2C))
- Great abundance of different types of peripheral circuits that implements synchronous serial interfaces (Flash, EEPROM, AD, DA, RTC, Display drivers, sensors etc.)



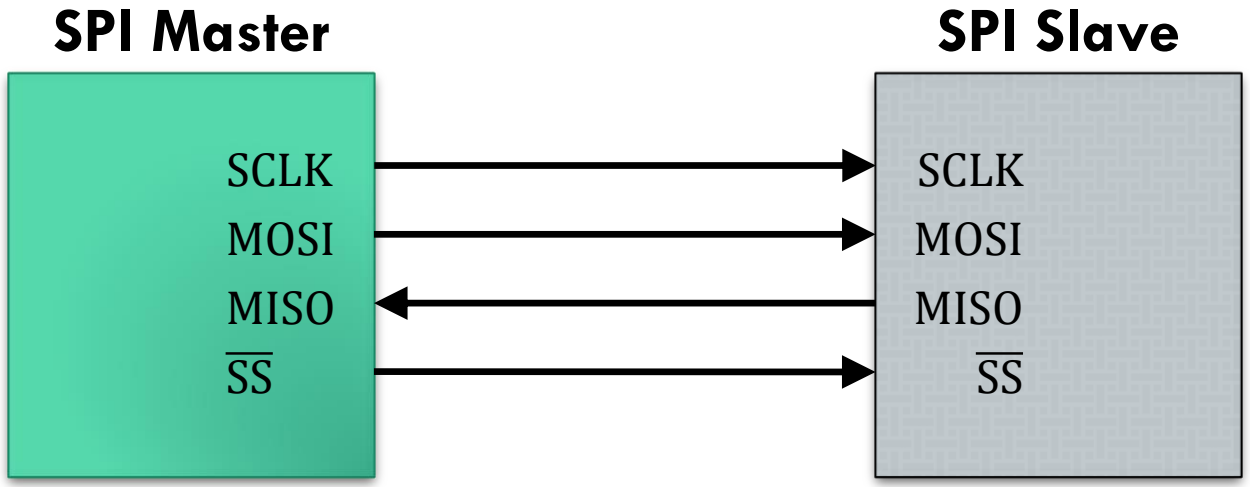
Top Results for: sensor SPI

Embedded - Microcontrollers (1741 items)	Integrated Circuits (ICs)
Motion Sensors - Accelerometers (571 items)	Sensors, Transducers
Pressure Sensors, Transducers (551 items)	Sensors, Transducers
Data Acquisition - Analog to Digital Converters (ADC) (433 items)	Integrated Circuits (ICs)
Evaluation Boards - Sensors (431 items)	Development Boards, Kits, Programmers
Temperature Sensors - Analog and Digital Output (355 items)	Sensors, Transducers
Motion Sensors - IMUs (Inertial Measurement Units) (253 items)	Sensors, Transducers
Interface - Sensor, Capacitive Touch (212 items)	Integrated Circuits (ICs)
Motion Sensors - Gyroscopes (109 items)	Sensors, Transducers
Position Sensors - Angle, Linear Position Measuring (107 items)	Sensors, Transducers
Magnetic Sensors - Linear, Compass (ICs) (101 items)	Sensors, Transducers
Specialized Sensors (59 items)	Sensors, Transducers
PMIC - Thermal Management (45 items)	Integrated Circuits (ICs)
Optical Sensors - Ambient Light, IR, UV Sensors (43 items)	Sensors, Transducers
Humidity, Moisture Sensors (41 items)	Sensors, Transducers

SPI: Serial Peripheral Interface

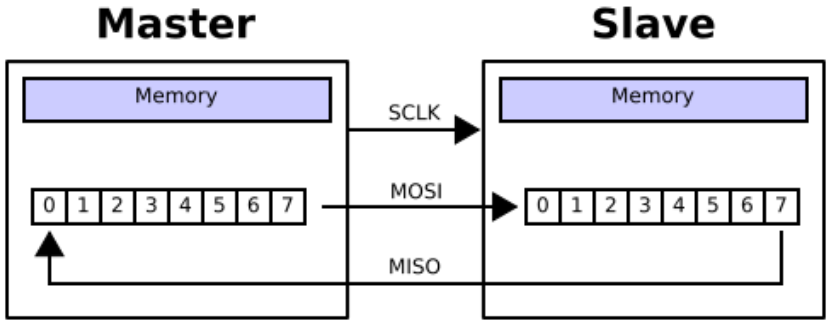
The SPI bus specifies four logic signals:

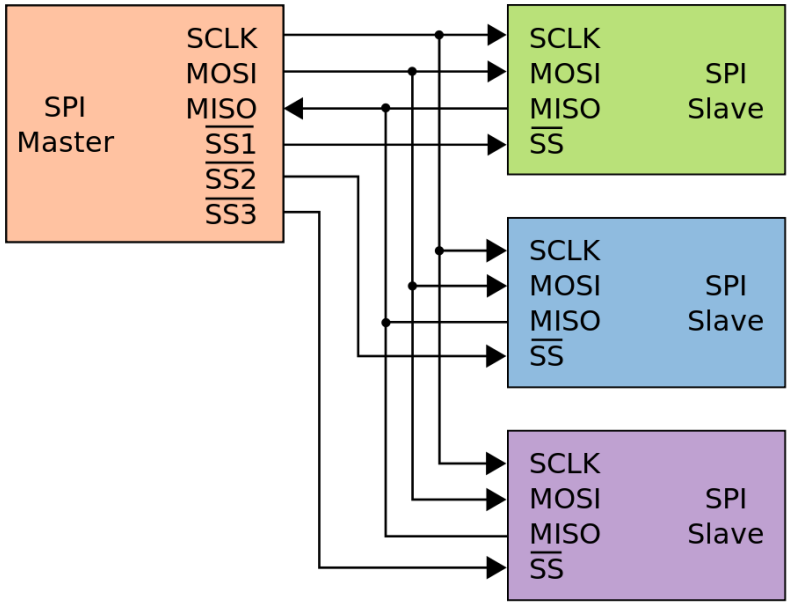
- SCLK : Serial Clock (output from master).
- MOSI : Master Output, Slave Input (output from master).
- MISO : Master Input, Slave Output (output from slave).
- SS : Slave Select (active low, output from master).



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

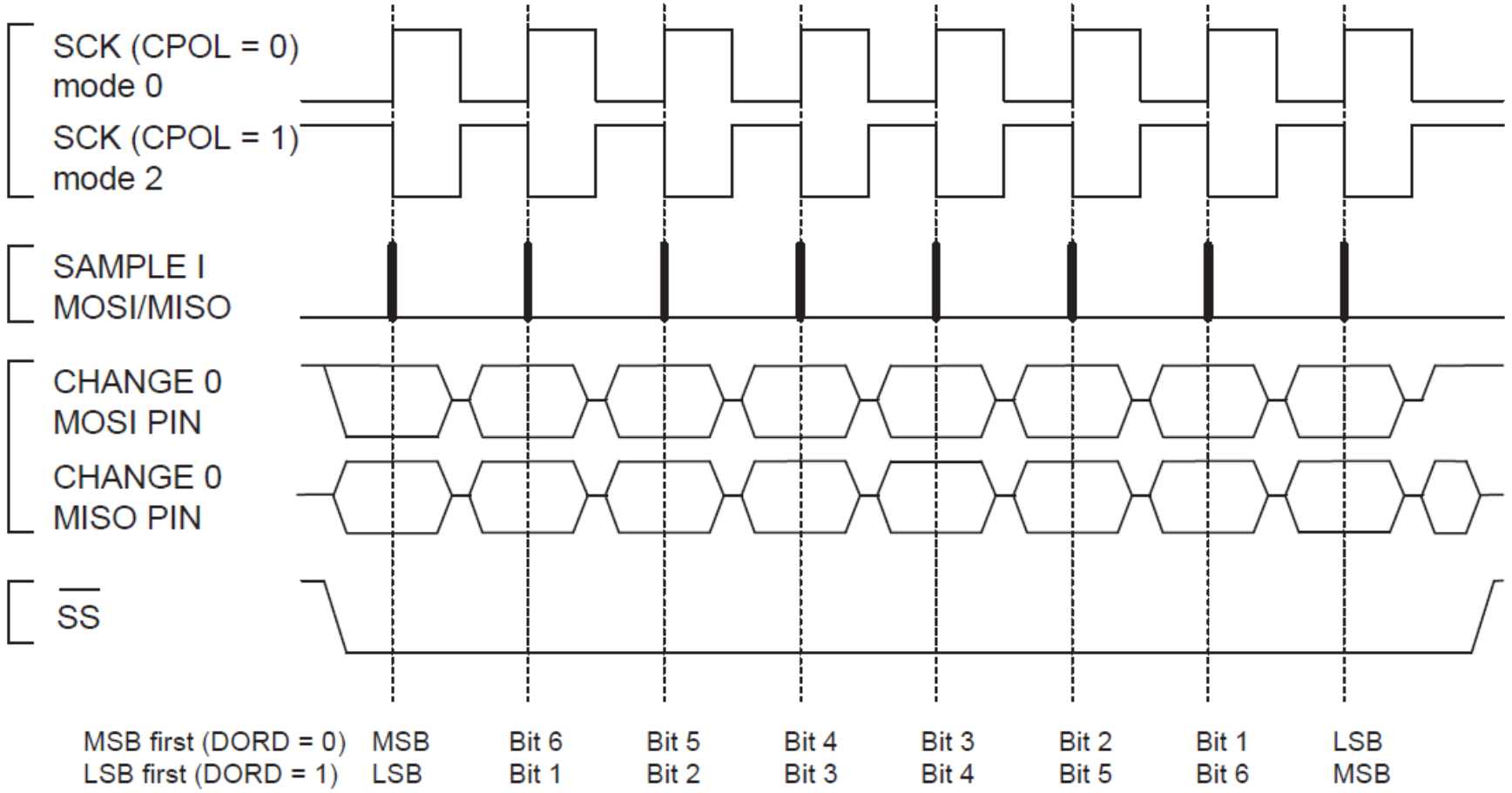
- There are four combinations of SCK phase and polarity with respect to serial data, which are determined by control bits CPHA and CPOL.

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

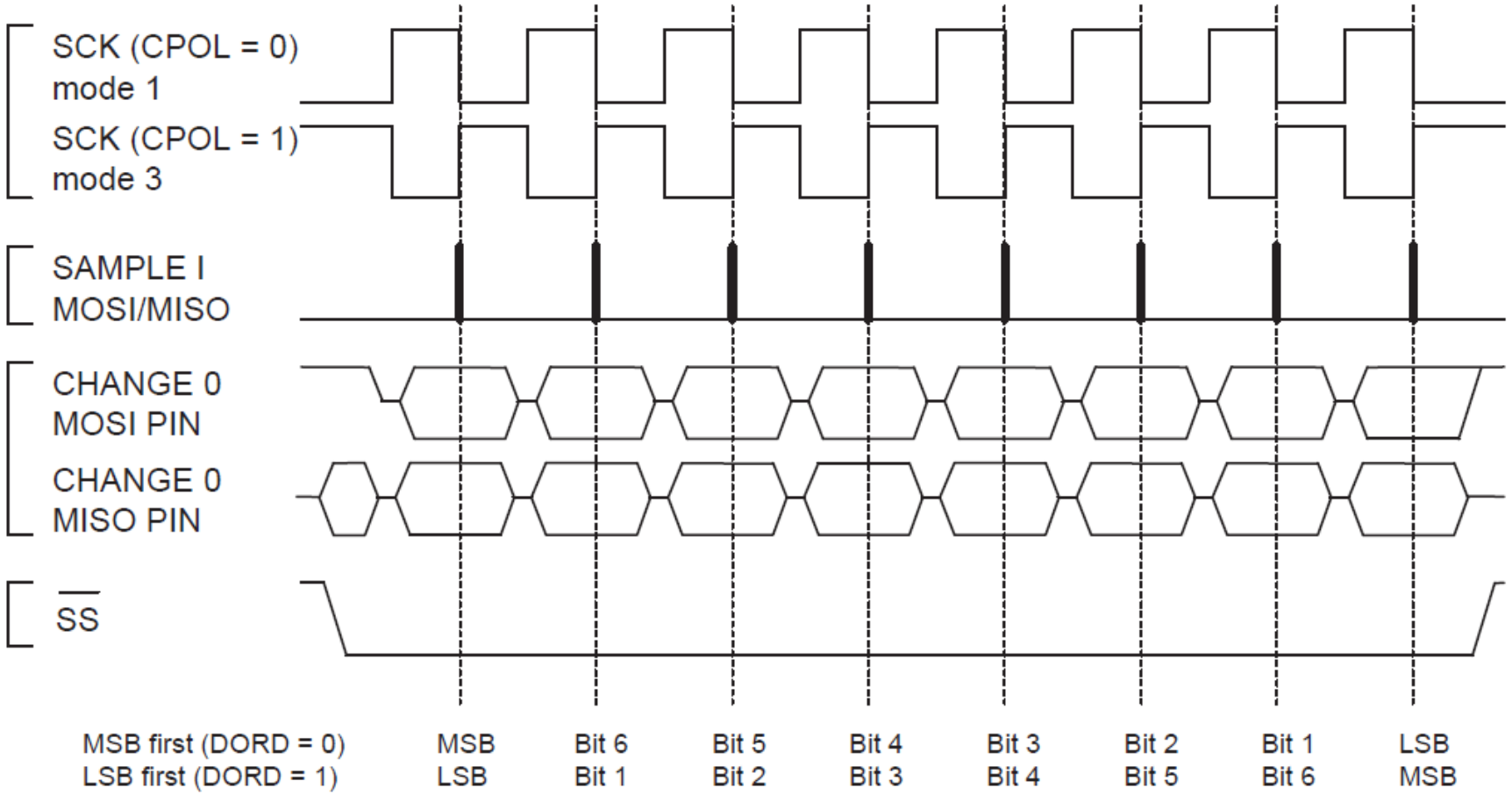
SPI Frame Transfer with CPHA=0

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



SPI Frame Transfer with CPHA=1

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



SPCR Register

23.5.2 SPI Control Register 1

Name: SPCR1
Offset: 0xAC [ID-000004d0]
Reset: 0x00

Bit	7	6	5	4	3	2	1	0
	SPIE1	SPE1	DORD1	MSTR1	CPOL1	CPHA1	SPR1 [1:0]	
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 5 – DORD1 Data1 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 – MSTR1 Master/Slave1 Select
This bit selects the Master SPI mode when written to one, and the 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 the SPI Master mode.

SPCR Register

Bits 1:0 – SPR1 [1:0] SPI1 Clock Rate Select
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 23-8. Relationship Between SCK and Oscillator Frequency

SPI2X	SPR1[1]	SPR1[0]	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	$f_{osc}/64$

SPI Master Example

```
#define SPI_DDR  DDRB
#define SPI_SS   2
#define SPI_MOSI 3
#define SPI_MISO 4
#define SPI_SCK  5
```

```
void SPI_MasterInit(void)
{
        /* Set SS, MOSI and SCK output, all others input */
        SPI_DDR = (1<<SPI_SS) | (1<<SPI_MOSI) | (1<<SPI_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
}
```

SPI Slave Example

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

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

MicroWire (μWire)

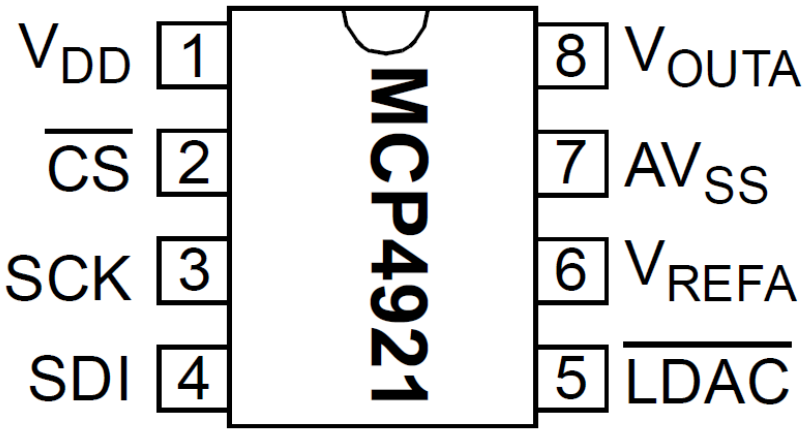
- Essentially a subset of SPI
- SPI mode 0 → (CPOL, CPHA) = (0, 0)
- Often found in half duplex "three-wire mode"
- Common bi-directional serial data line → only three wires needed (SIO, SCLK, CS)
- Used in e.g. RTCs (real-time clocks) and serial EEPROMs

DAC: Digital to Analog Converter

We use an external DAC for this lab: MCP4921

- 12 bit resolution.
- SPI interface.

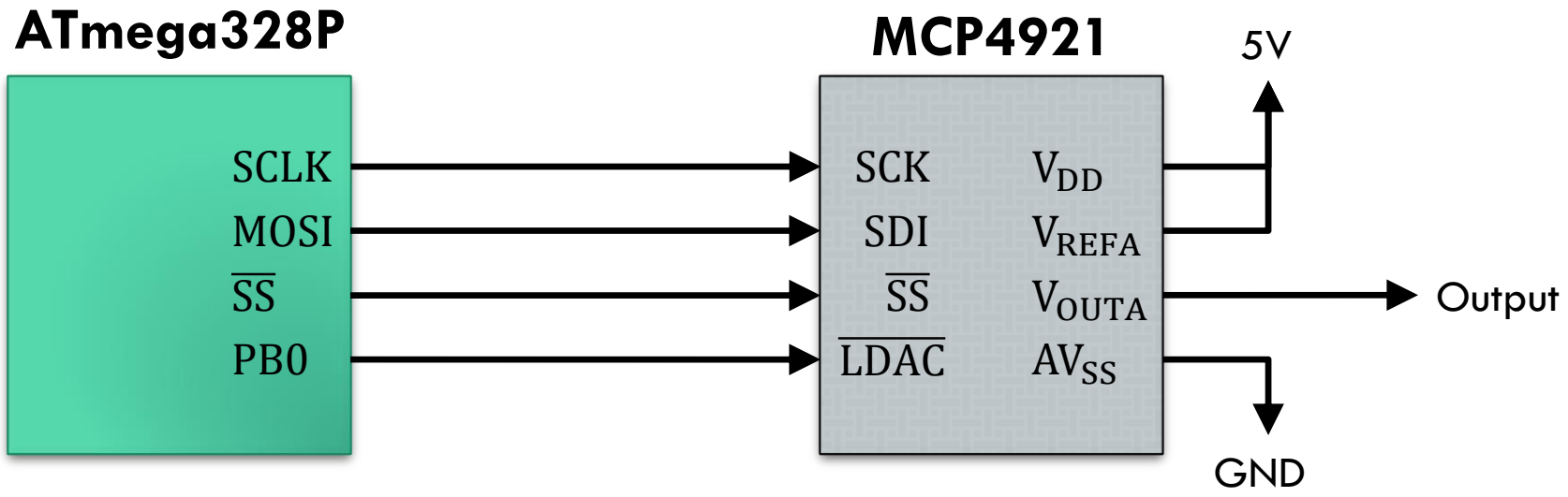
1	V_{DD}	Positive Power Supply Input (2.7V to 5.5V)
2	CS	Chip Select Input. (SPI Slave Select)
3	SCK	SPI Serial Clock Input
4	SDI	SPI Serial Data Input (MOSI)
5	LDAC	Synchronization input used to transfer DAC settings from serial latches to the output latches.
6	V_{REFA}	DAC_A Voltage Input (AV_{SS} to V_{DD})
7	AV_{SS}	Analog ground
8	V_{OUTA}	DAC_A Output



DAC SPI Interface

MCP4921 acts as SPI Slave and only receives data → MISO is not connected.

- Connect the ATmega328P with MCP4921 as shown in the figure below.
- Notice that LDAC pin also needs to be connected to a GPIO pin on ATmega328P
  - Defaults high
  - Assert low when sending data



DAC SPI Frame Format

- MCP4921 receives a 16-bit word from the MCU in two 8-bit SPI transactions
- The format of the 16-bit frame containing 4 command and 12 data bits is shown below
- CPOL=0, CPHA=0, MSB first

REGISTER 5-1: WRITE COMMAND REGISTER

Upper Half:

W-x	W-x	W-x	W-0	W-x	W-x	W-x	W-x
$\overline{A}$/B	BUF	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8
bit 15							bit 8

Lower Half:

W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
D7	D6	D5	D4	D3	D2	D1	D0
bit 7							bit 0

DAC Command Bits

- The upper 4 bits of the 16 bit word are DAC command bits.
- The description of the 16 bit frame bits is as follows:

bit 15 **A/B:** DAC_A or DAC_B Select bit

1 = Write to DAC_B

0 = Write to DAC_A

bit 14 **BUF:** V_{REF} Input Buffer Control bit

1 = Buffered

0 = Unbuffered

bit 13 **$\overline{GA}$:** Output Gain Select bit

1 = 1x ($V_{OUT} = V_{REF} * D/4096$)

0 = 2x ($V_{OUT} = 2 * V_{REF} * D/4096$)

bit 12 **$\overline{SHDN}$:** Output Power Down Control bit

1 = Output Power Down Control bit

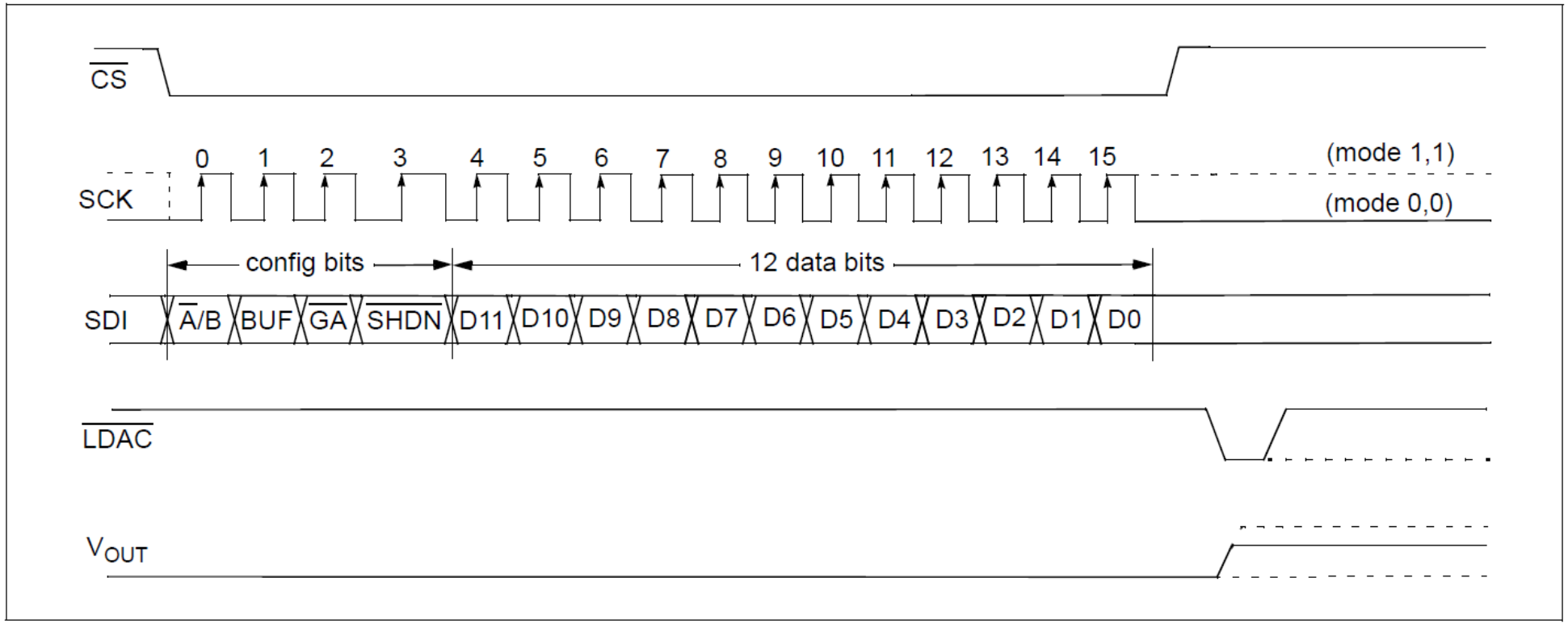
0 = Output buffer disabled, Output is high impedance

bit 11-0 **D11:D0:** DAC Data bits

12 bit number "D" which sets the output value. Contains a value between 0 and 4095.

DAC SPI Interface Timing

- The figure below shows the timing of one SPI transaction (command + data) between the MCU and DAC.
- You need to implement the same timing through SPI interface on ATmega328P.



Conn

- Connect the board as in Fig.1.
- You need to connect 12 bit DAC (MCP4921) as SPI device
- You could use 2 potentionmeter
  - 1 for ADC
  - 1 for contrast control
- For SPI clarification use datasheet of SPI and Atmega328P

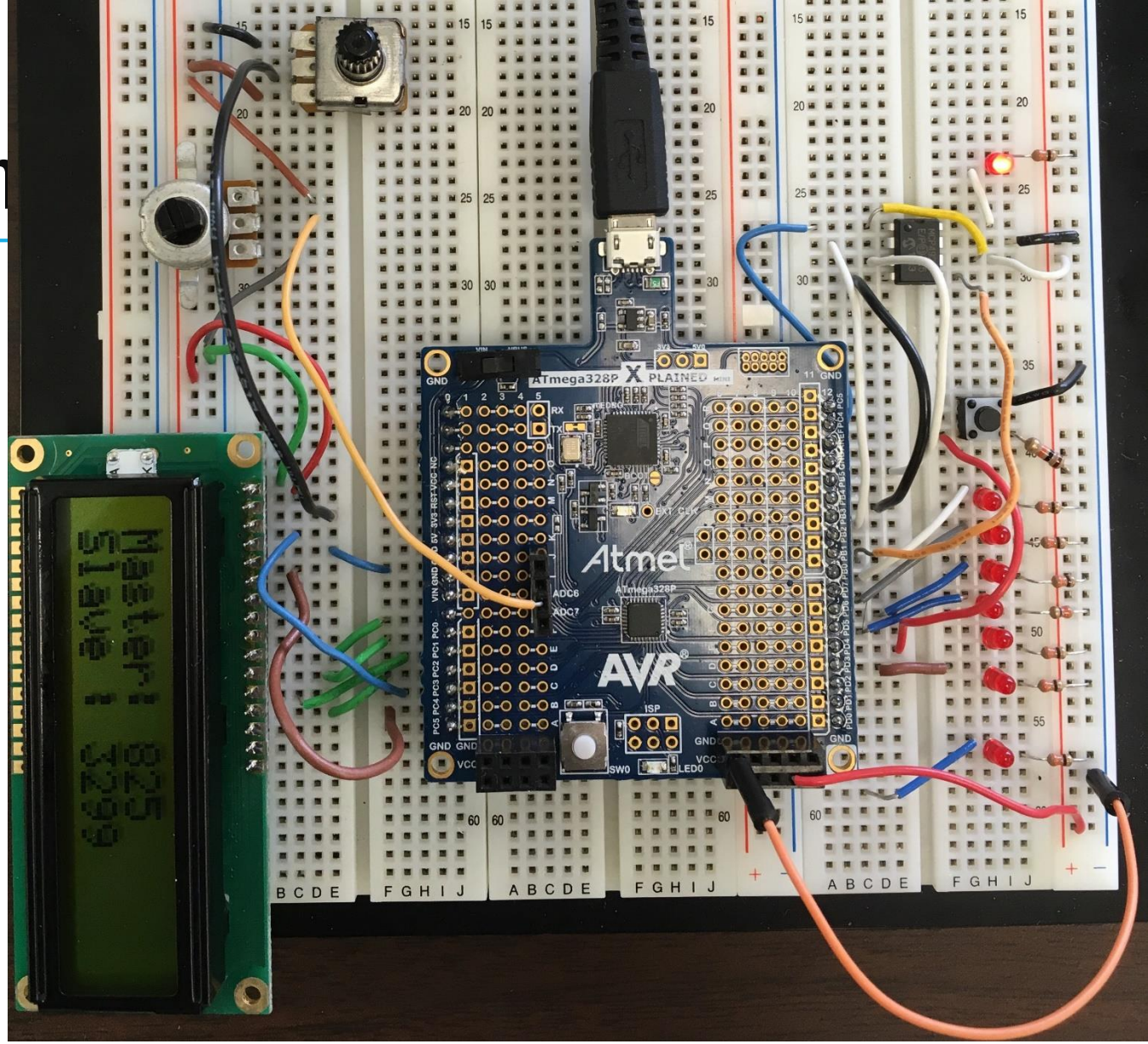
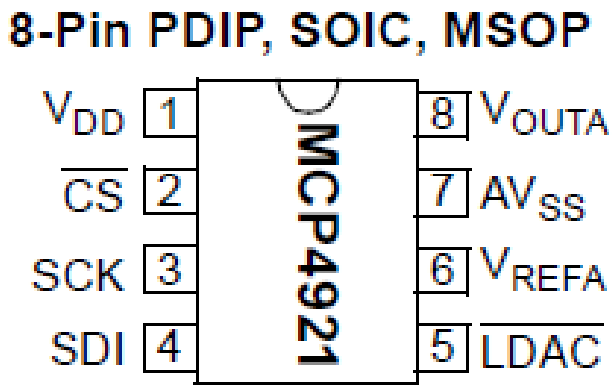
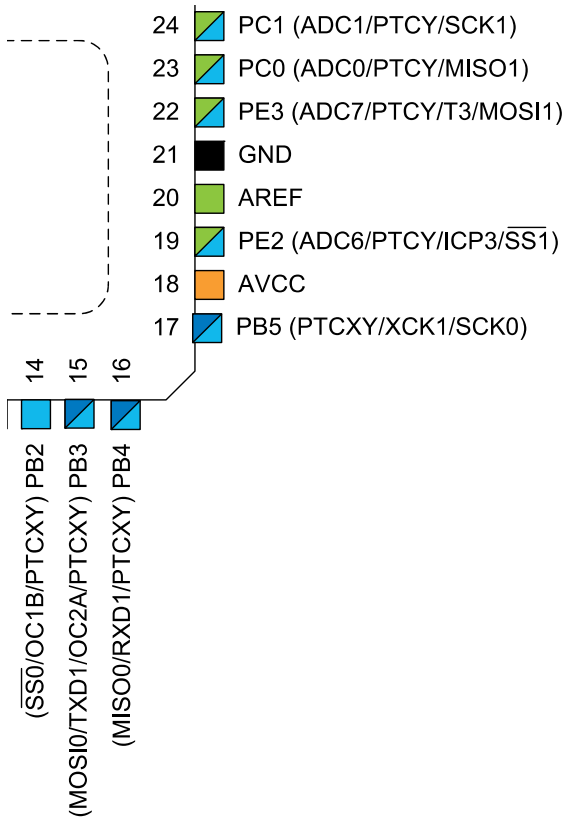


Fig1. Connections for SPI

Pins


8-Pin PDIP, SOIC, MSOP
VDD 1
CS 2
SCK 3
SDI 4
MCP4921
8 VOUTA
7 AVSS
6 VREFA
5 LDAC


24 PC1 (ADC1/PTCY/SCK1)
23 PC0 (ADC0/PTCY/MISO1)
22 PE3 (ADC7/PTCY/T3/MOSI1)
21 GND
20 AREF
19 PE2 (ADC6/PTCY/ICP3/SS1)
18 AVCC
17 PB5 (PTCXY/XCK1/SCK0)
14 15 16
(SS0/OC1B/PTCXY) PB2
(MOSI0/TXD1/OC2A/PTCXY) PB3
(MISO0/RXD1/PTCXY) PB4

Important Notes

- On Xplained Mini, ATmega328PB 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 ATmega328PB.
- 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 ATmega328PB.
  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 ATmega328PB at least once. This will enable the DWEN fuse before the SPI pins MOSI and MISO are shorted together.

Task: Controlling LED Glow

Write a simple program to control the glow of a LED using DAC.

In particular:

- Configure the SPI in Master mode.
- Read a potentiometer's voltage through ADC every 100ms (full 10 bit resolution).
- Normalize the 10-bit ADC reading to a 12-bit digital value for DAC.
- Transmit the 4-bit command and 12-bit data value to DAC over SPI.
- Don't forget to generate a LOW pulse at LDAC pin after transmission.
- Print the ADC's and DAC's readings on LCD.