Lab 7: Communicating with Other Devices

Objective: Using communication hardware to communicate with other devices.

Description of Lab 7:

In many applications, a separate hardware device will be used to perform a special function. In order to work with this device, some type of communication protocol is needed to move data from the device to the microprocessor. We have already been using a protocol (RS-232), when we send serial data from the Arduino to the Serial Monitor.

When communicating with small electronic devices, a more common protocol, known as Serial Peripheral Interface (SPI). The SPI protocol is a three wire interface with a Serial ClocK (SCK), a Master In-Slave Out (MISO) and a Master-Out Slave-In (MOSI) line. The layout of these signals are shown in Figure 7-1, and the names are very descriptive of their function. With all the different devices, the hardware on most microprocessors is built to support a variety of clocking schemes. The various clocking schemes are shown in Figure 7-2.

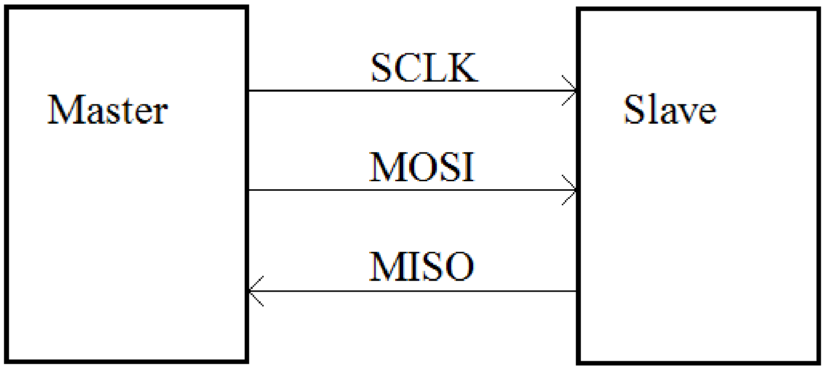

Figure 7-1. Signals for SPI Interface.

In the lab, a set of SPI based thermocouples are available which will read the temperature and report it over an SPI interface. Figure 7-3, shows the clocking of the data from the thermocouple. Note this will correspond with the settings of CPOL = 0 and CPHA = 0 (**MODE 0** – See table 7-1). Now the question is what should the timing of SCK be? In other words, how do we make the system not toggle the clock to quickly?

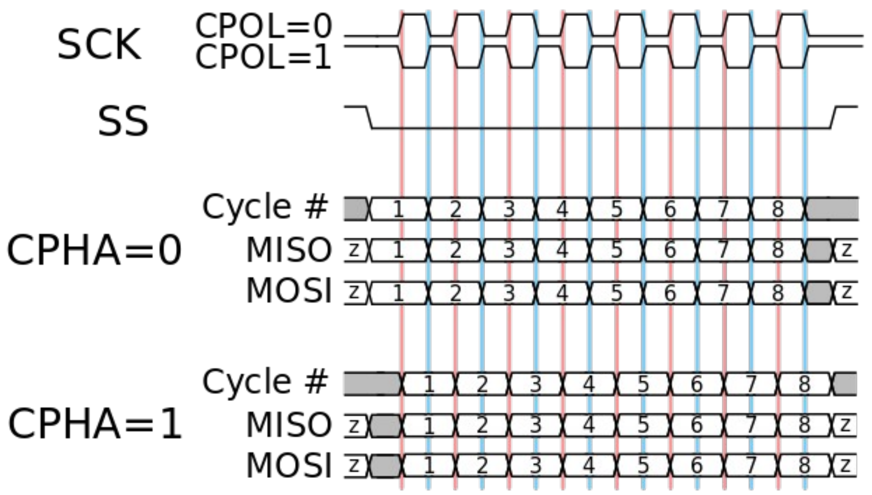

Figure 7-2. SPI Timing for Various Parameter Settings.

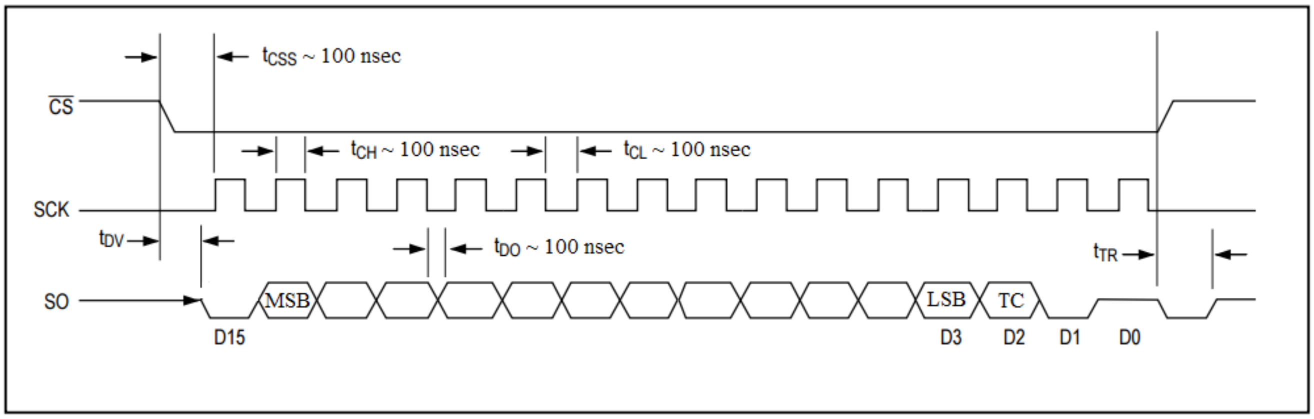

Figure 7-3. Clocking for the SPI Based Thermocouple.

Now recall that the clock on our microprocessor is 16MHz, which means its cycle time (or period) is 1/ 16M = 62.5 nanoseconds. Thus we will need to "divide" the clocking frequency by 2, to 8 MHz (125 nanosecond period).
At this point we need to look at the software that Arduino provides for talking over the SPI. It has two setup functions we should call to have the SPI hardware properly work with the Max6675. These are setClockDivider(divider) and setDataMode(Mode). The divider values are defined for Arduino in the form SPI_CLOCK_DIVx, where x is 2, 4, ... 128. Thus based on our previous discussion we should set this to SPI_CLOCK_DIV2. The mode is based on the values in Table 7-1. The following link provides a good example of how to setup the SPI settings.

https://www.arduino.cc/en/Tutorial/SPITransaction

For this lab we will be wiring the thermocouple system to our Arduino. The use of the breadboards and digi-designers in the lab will be demonstrated in class. But one very important point to be noted is that the SPI on the Arduino only works on specific pins (SCK = Pin 13, MISO = Pin 12, MOSI = Pin 11). Now pin 11 is used by the LCD, but since our thermocouple does not receive data, we will not have to use the MOSI.

Table 7-1: Modes for SPI Hardware

Mode	Clock Polarity (CPOL)	Clock Phase (CPHA)
SPI_MODE0	0	0
SPI_MODE1	0	1
SPI_MODE2	1	0
SPI_MODE3	1	1

Finally the data that comes back from the thermocouple will be two bytes. In earlier versions of the software the SPI code only read one byte at a time, requiring the merging of the bytes after reading. Fortunately we now have functions that can read 16 bit words, so no merging is necessary, however it still needs to be translated into temperature. Looking at the data sheet for the Max6675, we can see that the temperature is encoded into bits 14 to 3. Bit 15 is zero and bits 2, 1 and 0 have other functions. Thus if we were to read in the data, the twelve bits of temperature data could be pulled out as follows.

```
int TempBits = ( SPI.transfer16(0) >> 3 ) & 0x0fff; //Shift over 3, then mask off upper bits
```

The number in TempBits can be translated into a Celsius temperature realizing the 0 in TempBits, is 0 degrees Celsius, and each bit represents **0.25 degrees Celsius**, or

```
float Temperature = 0.25 * (float) TempBits; // Convert to degrees Celsius
float Temperature = 1.8*Temperature + 32.0;  // Convert to degrees Fahrenheit.
```

Program Structure for Arduino:

```
float ReadTemperature()
    Set CS LOW
    Read two bytes over SPI.
    Set CS HIGH
    convert temperature to Fahrenheit
    return temperature

SetUp:
        Set CS as an output
        Configure SPI hardware
        Configure Serial

Loop:
        if 1 seconds has passed,
                ReadTemperature();
                Transmit temperature over Serial
        end
```

Lab Assignment:

Prelab: Write the program described in the program section and upload it to the webpage.

Lab 7:

1) Hook up the thermocouple circuit to your Arduino, program and demonstrate the program to your instructor.

Questions:

1) How did you test the thermocouple in the lab?

2) Based on the calculation used to compute the temperature, what is the resolution in degrees Fahrenheit?

3) What is the range of temperatures that this sensor can measure? Realize that 0 from the chip is zero degrees Celsius, so what is the maximum value it can read?