Goals:
The goal of this lab is to interface the SPI chip to the AVR microcontroller. Follow the schematics and the sample code provided to construct this lab.

Equipment Usage
For this lab the following equipment will be used:

- Atmega328P
- DS1307 (or equivalent)
- 32.768KHz Crystal Oscillator
- Lithium Battery

Background:
I2C:
Inter-Integrated Circuit is another serial communication protocol used to transfer data between two ICs. In AVR I2C is a two wire interface relying on the Serial Data (SDA) and Serial Clock (SCL) pins. Like the SPI protocol, I2C supports master-slave relation among devices. One of the main design advantages is that ICs can be removed and added without affecting the other ICs in the system.

AVR Registers for Two-Wire Interface (TWI)

TWAR (TWI Slave Address Register) - holds the 7-bit Slave address which the TWI will respond when programmed as a Slave Transmitter or Receiver (not needed in Master Mode.)

TWBR (TWI Bit Rate Register) - bit rate generator is a frequency divider which generates the SCL clock frequency in the Master mode

TWCR (TWI Control Register) - used to control the various operations of the TWI. Examples include start and stop conditions, write collision, and Receiver acknowledge

TWDR (TWI Data Register) - Contains the next byte to be transmitted or last byte that was received

TWSR (TWI Status Register) - Reflects the status of the TWI logic
```c
#include <avr/io.h>

// I2C functions
void i2c_stop()
{
    TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);
}

void i2c_write(unsigned char data)
{
    TWDR = data;
    TWCR = (1<<TWINT)|(1<<TWEN);
    while(!(TWCR & (1<<TWINT)));
}

void i2c_start(void)
{
    TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
    while (!(TWCR & (1<<TWINT)));
}

void i2c_init(void)
{
    TWBR = 0x00; // set prescaler bits to 0
    TWBR = 0x47; // SCL freq. is 50k for XTAL = 8M
    TWCR = 0x04; // enable TWI module
}

int main(void)
{
    // I2C initialization
    i2c_init(); // initialize I2C module
    i2c_start(); // transmit START condition
    i2c_write(0x11010000); // set register pointer to 7
    i2c_write(0x00); // set value of location 7 to 0
    i2c_stop(); // transmit STOP condition

    for (int k = 0; k<100; k++); // wait for a short time

    i2c_start(); // transmit START condition
    i2c_write(0x11010000); // address DS1307 for write
    i2c_write(0x04); // set register pointer to 4
    i2c_write(0x19); // set date to 0x10 = 19 BCD
    i2c_write(0x10); // set month to 0x10 = 10 BCD
    i2c_write(0x10); // set year to 0x10 = 09 BCD

    while(1);
    return 0; // stop here
}

int main(void)
{
    // I2C initialization
    i2c_init(); // initialize I2C module
    i2c_start(); // transmit START condition
    i2c_write(0x11010000); // set register pointer to 7
    i2c_write(0x00); // set value of location 7 to 0
    i2c_stop(); // transmit STOP condition

    for (int k = 0; k<100; k++); // wait for a short time

    i2c_start(); // transmit START condition
    i2c_write(0x11010000); // address DS1307 for write
    i2c_write(0x00); // set register pointer to 4
    i2c_write(0x55); // set date to 0x10 = 19 BCD
    i2c_write(0x58); // set month to 0x10 = 10 BCD
    i2c_write(0b00011110); // set hour = 16 in 24 hours mode
    i2c_stop(); // transmit STOP condition

    while(1);
    return 0; // stop here
}
```
**Lab Experiments:**

**Experiment 1:** Program the current date and time onto the DS1307 chip. Use the microcontroller to read the current date and time and display it to the screen every 30 seconds (use USART/UART interface.)

![Circuit Diagram](attachment:image.png)

**Post-Lab Deliverables:**

1) Submit your completed project report including your working code.
2) Submit a PCB layout Design of your circuit using Altium