ECED3204 – Lab #6

STUDENT NAME(s): _____ .

STUDENT NUMBER(s): B00 _____ .

Pre-Lab Information

It is recommended that you read this entire lab ahead of time. Doing so will save you considerable time during the lab, as you will be required to write some simple C code during this lab!

Objective

- Use the SPI module in the AVR Mega microcontroller
- Interface to an EEPROM

Required Materials

- Microprocessor Module with Programmer
- Breadboard
- USB Cable
- Power Supply
- Computer with Atmel Studio 6.2 and Programmer Utility installed
- 25LC080 EEPOM
- 1K Resistor

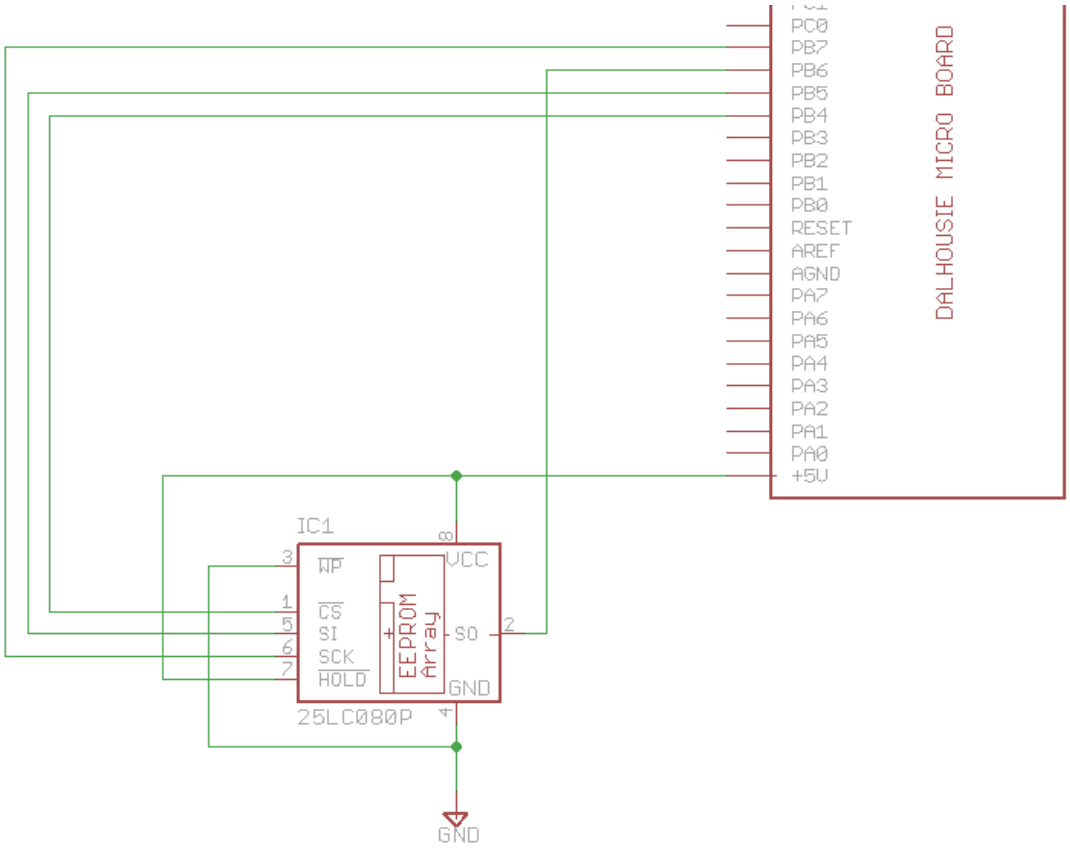
Background

Serial Peripheral Interface (SPI) is used by many devices. This lab will have you dump the contents of a SPI-connected EEPROM device.

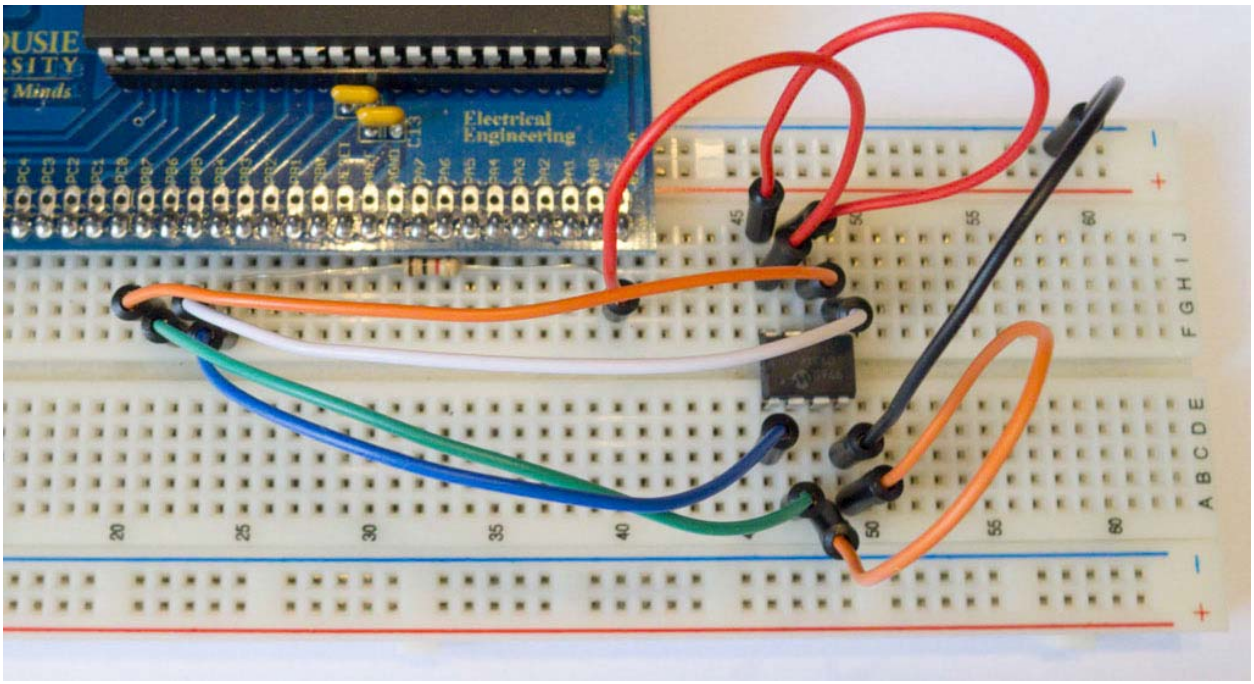
See the course textbook for information – the SPI chapter includes information on interfacing to these EEPROM devices. This lab assumes you have read that chapter!

Procedure

1. Build the following circuit using the programmed 25LC080P device:



Which might look like this:



2. Start a new C/C++ project (see Lab #1 for details), write the following code into it:

```
#include <stdio.h>
#include <avr/io.h>
#include <avr/pgmspace.h>

static int uart_putchar(char c, FILE *stream);
static int uart_getchar(FILE *stream);
FILE mystdout = FDEV_SETUP_STREAM(uart_putchar, NULL, _FDEV_SETUP_WRITE);
FILE mystdin = FDEV_SETUP_STREAM(NULL, uart_getchar, _FDEV_SETUP_READ);

static int uart_putchar(char c, FILE *stream)
{
   loop_until_bit_is_set(UCSR0A, UDRE0);
   UDR0 = c;
   return 0;
}

static int uart_getchar(FILE *stream)
{
   loop_until_bit_is_set(UCSR0A, RXC0); /* Wait until data exists. */
   return UDR0;
}

void init_uart(void)
{
   UCSR0B = (1<<RXEN0) | (1<<TXEN0);
   UBRR0 = 7;
   stdout = &mystdout;
   stdin = &mystdin;
}

#define CS_HIGH() PORTB |= 1<<4
#define CS_LOW()  PORTB &= ~(1<<4)

#define INST_WREN  0x06 /* Write Enable */
#define INST_WRDI  0x04 /* Disable writes */
#define INST_RDSR  0x05 /* Read status/config register */
#define INST_WRSR  0x01 /* Write status/config register */
#define INST_READ  0x03 /* Read data from memory */
#define INST_WRITE 0x02 /* Write data to memory */

uint8_t spi_readwrite(uint8_t data)
{
   SPDR = data;
   loop_until_bit_is_set(SPSR, SPIF);
   return SPDR;
}

int main(void)
{
   init_uart();
   printf_P(PSTR("System Booted, built %s on %s\n"), __TIME__, __DATE__);

   //Set CS as output, other lines set automatically
   DDRB |= (1<<4) | (1<<5) | (1<<7);
   CS_HIGH();

   //Slowest possible SPI clock (easiest with long wires)
   SPCR = (1<<SPE) | (1<<MSTR) | (1<<SPR1) | (1<<SPR0);
```

```
    uint8_t data;

    CS_LOW();
    spi_readwrite(INST_RDSR);
    data = spi_readwrite(0);
    CS_HIGH();

    printf("Status register: %02x\n", data);

    uint8_t addr = 101;

    CS_LOW();
    spi_readwrite(INST_READ);
    spi_readwrite(0);
    spi_readwrite(addr);
    data = spi_readwrite(0);
    CS_HIGH();

    printf("Address %d: %02x\n", addr, data);
}
```

3. The EEPROM has been programmed with the following information:

   Address 00: Secret Byte
   Address 01: Secret Byte
   …
   Address 98: Secret Byte
   Address 99: Secret Byte
   Address 100: 0x00
   Address 101: 0x01
   Address 102: 0x02
   Address 103: 0x03
   Address 104: 0xDE
   Address 105: 0xAD
   Address 106: 0xBE
   Address 107: 0xEF

   Check your setup is working by verifying that address 100-107 have the expected values. You can use the printf() setup (see Lab #5 for getting this working) to dump these values.

4. Print the byte value corresponding to the last two digits of your banner number. For example if your student ID was B00123456, you would print the value stored at address **56**. This secret value will be used to verify your lab report.