ECED3204 – Lab #7

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 TWI (i.e. I2C) 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
- 24LC08B EEPOM
- 2x 2.2K Resistor

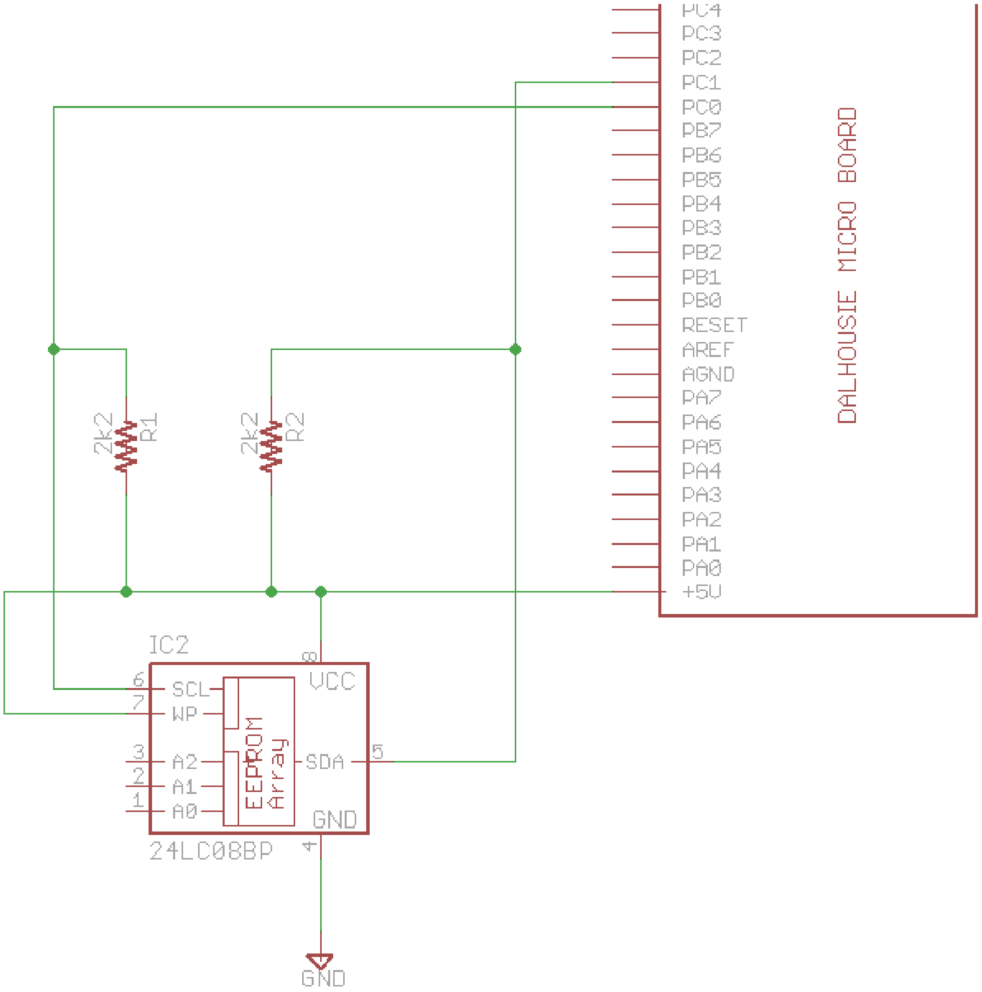
Background

Inter-IC (I^2C or I2C) is a common communications format for many devices. Note Atmel calls this interface the Two-Wire Interface (TWI) to avoid licensing rules around using the trademarked I2C name.

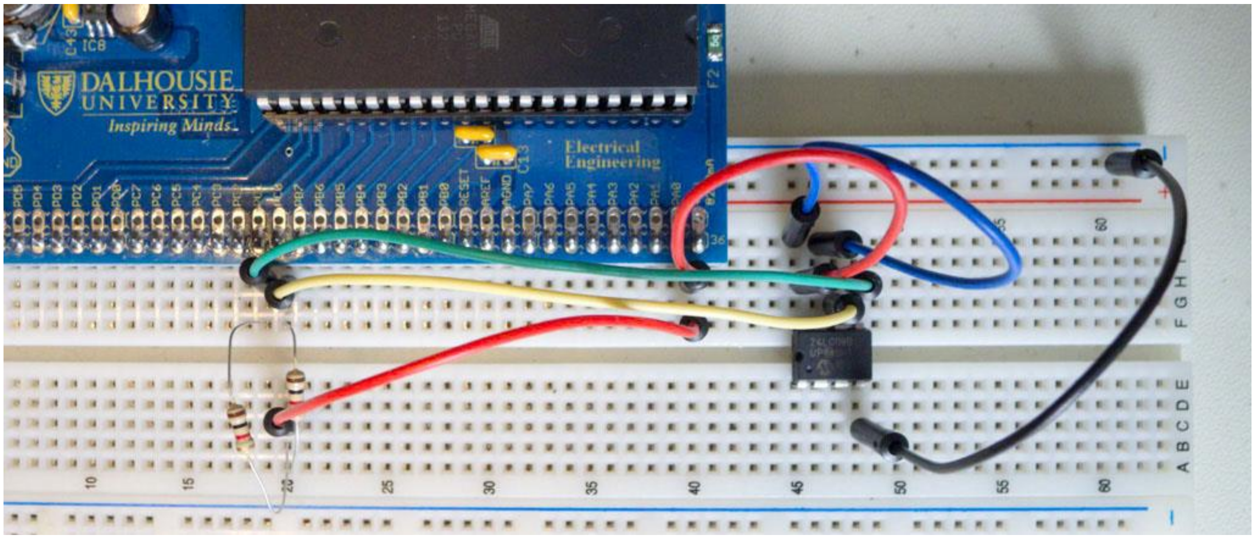
See the course textbook for information – the I2C / TWI chapter includes information on interfacing to these EEPROM devices. This lab assumes you have read that chapter!

Procedure

1. Build the following circuit using the programmed 24LC08B device:



Which might look like this:



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

```
#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;
}


/* Generic I2C Routines */

void TWI_Start(void)
{
   TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN);
   loop_until_bit_is_set(TWCR, TWINT);
}

void TWI_Stop(void)
{
   TWCR = (1<<TWINT)|(1<<TWSTO)|(1<<TWEN);
   loop_until_bit_is_clear(TWCR, TWSTO);
}

void TWI_sendByte(uint8_t cx)
{
   TWDR = cx;
   TWCR = (1<<TWINT)|(1<<TWEN);
   loop_until_bit_is_set(TWCR, TWINT);
}

uint8_t TWI_readByte(char sendAck)
{
   if(sendAck){
          TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWEA);
   } else {
          TWCR = (1<<TWINT)|(1<<TWEN);
   }
   loop_until_bit_is_set(TWCR, TWINT);
   return TWDR;
}

uint8_t TWI_status(void)
{
   return TWSR & 0xF8;
}

/* EEPROM Specific Routines - NO error handling done! */

void writePoll(uint8_t SLA)
{
   char busy = 1;
   while(busy){
          TWI_Start();
          TWI_sendByte(SLA);
          if(TWI_status() == 0x18){
                //OK
                busy = 0;
          }
   }
}

void writeByteEE(uint8_t SLA, uint8_t addr, uint8_t data)
{
   TWI_Start();
   TWI_sendByte(SLA);
   TWI_sendByte(addr);
   TWI_sendByte(data);
   TWI_Stop();
   writePoll(SLA);
}

uint8_t readByteEE(uint8_t SLA, uint8_t addr)
{
   uint8_t tmp;
   TWI_Start();
   TWI_sendByte(SLA);
   TWI_sendByte(addr);
   TWI_Start();
   TWI_sendByte(SLA | 0x01);
   tmp = TWI_readByte(0);
   TWI_Stop();
   return tmp;
}

//You can extend these to have error handling - see
http://www.embedds.com/programming-avr-i2c-interface/
//for example

#define EEPROM_ADDR 0xA0

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

   //~50 kHz I2C frequency (slower than normal)
   TWBR = 132;
   TWCR = 1<<TWEN;
   TWSR = 0;
   uint16_t addr = 105;
   printf("Read address 0x%02x = %02x\n", addr, readByteEE(EEPROM_ADDR,
addr));

}}
```

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.