/*Code as a PDF File, Nathan Phipps, please note I didn't attach a hardware setup or pictures, since the setup and pin assignments were provided by the lab. */

```
#include <avr/io.h>                //adding library
int dev_addr = 72;                 //Address of TC74A0 Temp Sensor
#define freq          16000000    //16 MHz
#define scl           100000      //SCL 100k, twi frequency

void setup() {
  Serial.begin(9600);
}

void loop() {
  int Tempurature = read_temp(dev_addr);     //Setting int Temperature = to function "read_temp" and calling in the TC74A0 address
  Serial.print("Temperature: ");           //Prints "Temperature: " on serial monitor.
  Serial.print(Tempurature);               //Calls Temperature Function into print statement.
  Serial.print("C");                       //Prints "C" on serial monitor.
  Serial.print("\n");                      //Creates new line.
  delay(1000);                            //Delay 1 second.
}

/* Read from Temperature Sensor */
int read_temp(int dev_addr) {

  unsigned char n = 1;

  i2c_init();                          //Calls init i2c function, initialize TWI for Master mode
  i2c_start();                         //Calls start i2c function, transmit start condition
  i2c_write(0b10010001);               //Calls write i2c function, tranmit SLA + R(1), note that an extra bit is tacked on to the address for the device, the extra bit tacked on is a 1.
  i2c_read();                          //Calls read i2c function, read one byte of data
  i2c_stop();                          //Calls stop i2c function, transmit stop

  return TWDR; //Returns temperature

}

/*************************************************************************
 Initialization of the I2C bus interface.
```

```
*****************************************************************************/
void i2c_init(void)
{
  TWSR = 0; //No Prescale
  TWBR = ((freq/scl)-16)/2; //Operations to achieve (16Mhz/100k) - 16 /2, thus achieving proper clock settings
  TWCR = TWCR | 0X04; //TWEN Enable
}



/*****************************************************************************
 Issues a start condition and sends address and transfer direction.
 return 0 = device accessible, 1= failed to access device
*****************************************************************************/
unsigned char i2c_start(void)
{
  TWCR = (1 << TWINT) | (1 << TWSTA) | (1 << TWEN); //Sends start
  while ((TWCR & (1 << TWINT)) == 0); //Waits for transmission, waits for TWINT flag to be set, indicating transmission.

  if((TWSR != 0x08) && (TWSR != 0x10)) //Checks value of TWI status register, if status isn't Start or repeated start then go to the error_output function.
  {
   return 1; //returns a 1 for the error.
  }

  if(TWSR != 0x18) //If status different from "SLA+W transmitted, ACK received " and " SLA+R transmitted, ACK received ", return 1 otherwise return 0
  {
    return 1; //returns a 1 for the error.
  }

  else
  {
   return 0; //returns 0;
  }

}



/*****************************************************************************
 Terminates the data transfer and releases the I2C bus
```

```
*****************************************************************************/
void i2c_stop()
{
  TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWSTO); // Sends stop

}

/*****************************************************************************
 Send one byte to I2C device

 Input:    byte to be transfered
 Return:   0 write successful
           1 write failed
*****************************************************************************/
unsigned char i2c_write(unsigned char DATA)
{
  TWDR = DATA;
  TWCR = (1 << TWINT) | (1 << TWEN); //Configure TWCR
  while ((TWCR & (1 << TWINT)) == 0); //Waits until data transmitted

  if((TWSR != 0x08) && (TWSR != 0x10)) //If status different from "data transmitted, ACK received", return 1 otherwise return 0
  {
    return 1; //returns a 1 for the error.
  }

  else
  {
   return 0;
  }

}

/*****************************************************************************
 Read one byte from the I2C device, read is followed by a stop condition

 Return:  byte read from I2C device
*****************************************************************************/
```

```
int i2c_read()

{

    //The code below allows us to read one byte

    TWCR = (1 << TWINT) | (1 << TWEN); //TWCR configured

    while ((TWCR & (1 << TWINT)) == 0 ); //Waits until transmission completes

    return TWDR; //Returns temperature



}
```