Lab Report #4

I2C Communication

Jonathan Boryla

ELEC 2520 : Embedded Systems Engineering II

Instructor: Chao Liu

April 25, 2018

# Part 1: I2C

### I2C\_TC74.ino

#include <LiquidCrystal.h>

#define START 0x08

#define R\_START 0x10

#define MT\_SLA\_ACK 0x18

#define MT\_DATA\_ACK 0x28

#define MR\_SLA\_ACK 0x40

#define device\_address 0x48

int Tempurature;

const int rs=12, en=11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {

Serial.begin(9600);

lcd.begin(16, 2);

Tempurature = read\_temp(device\_address);

lcd.print("Start Temp:");

lcd.print(Tempurature);

lcd.print("C");

}

void loop() {

Tempurature = read\_temp(device\_address); //call read\_temp function to begin i2c retrieval of temperature. Save return value to "tempurature"

Serial.print("Temperature: "); //print temperature formatted to

Serial.print(Tempurature);

Serial.print("C");

Serial.print("\n");

lcd.setCursor(0, 1);

lcd.print("Current Temp:");

lcd.print(Tempurature);

lcd.print("C");

delay(1000);

}

/\* Read from Temperature Sensor \*/

int read\_temp(int address) {

/\* Implemtement this function by using functions below:

i2c\_init();

i2c\_start();

i2c\_write();

i2c\_read();

i2c\_stop();

And return the value of temperture

\*/

i2c\_init();

int address\_write = address << 1;

i2c\_start(address\_write);

i2c\_write(0x00);

i2c\_stop();

int address\_read = (address << 1) + 1;

i2c\_start(address\_read);

int c = i2c\_read();

return c;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Initialization of the I2C bus interface.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void i2c\_init(void)

{

/\*

1. no prescaler

2. SCL clock: 100 kHz clock

\*/

TWSR=0;

TWBR=72;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Issues a start condition and sends address and transfer direction.

return 0 = device accessible, 1= failed to access device

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char i2c\_start(unsigned char address)

{

/\*

1. send START condition

2. wait until transmission completed

3. check value of TWI Status Register. Mask prescaler bits. If status different from "start" and "repeated start", return 1

4. send device address, configure TWCR

5. wail until transmission completed and ACK/NACK has been received

6. check value of TWI Status Register. Mask prescaler bits. If status different from "SLA+W transmitted, ACK received " and " SLA+R transmitted, ACK received ", return 1 otherwise return 0

\*/

TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN); //Send START condition

while (!(TWCR & (1<<TWINT))); //Wait until transmission completed

if (((TWSR & 0xF8) != START) && ((TWSR & 0xF8) != R\_START)) //Check value of TWI status register (masked prescaler bits), if status is different from "start" or "repeated start", return 1

return 1;

TWDR = address; //send device address

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

while (!(TWCR & (1<<TWINT))); //wait until transmission completed and ACK/NACK has ben received

if (((TWSR & 0xF8) != MT\_SLA\_ACK) && ((TWSR & 0xF8) != MR\_SLA\_ACK)) //If status different from "SLA+W transmitted, ACK received " and " SLA+R transmitted, ACK received ", return 1 otherwise return 0

return 1;

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Terminates the data transfer and releases the I2C bus

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void i2c\_stop(void)

{

/\*

1. send stop condition

2. wait until stop condition is executed and bus released

\*/

TWCR = (1 << TWINT) | (1 << TWSTO) | (1 << TWEN);

while (!(TWCR & (1 << TWSTO)));

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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 )

{

/\*

1. send data to the previously addressed device

2. wait until transmission completed

3. check value of TWI Status Register. Mask prescaler bits. If status different from "data transmitted, ACK received", return 1 otherwise return 0

\*/

TWDR = data; //send data to previously addressed device

TWCR = (1 << TWINT) | (1 << TWEN); //Clear TWINT to transmit address

while (!(TWCR & (1<<TWINT))); //wait until transmission completed

if ((TWSR & 0xF8) != MT\_DATA\_ACK) //check value of TWSR (prescaler bits masked. If status different from "data transmitted, ACK received", return 1 otherwise return 0

return 1;

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(void)

{

/\*

1. configure TWCR

2. wait until transmission completed

3. return temperature data

\*/

TWCR = (1 << TWINT) | (1 << TWEN) | (1 << TWEA);

while (!(TWCR & (1 << TWINT)));

return TWDR;

}