Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2022-2023

ΑΘΗΝΑ, 7 Δεκεμβρίου 2022

7η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"
Αισθητήρας Θερμοκρασίας DS1820 στην κάρτα ntuAboard_G1

Αναφορά 7ης Εργαστηριακής Άσκησης

Ραπτόπουλος Πέτρος (el19145)
Σαφός Κωνσταντίνος (el19172)

Ζήτημα 7.1:

Παρακάτω φαίνεται ο κώδικας σε C που υλοποιεί τα ζητούμενα της άσκησης.

Η ορθή λειτουργία των προγραμμάτων έχει ελεγχθεί στο περιβάλλον προσομοίωσης MPLAB X, καθώς και στην αναπτυξιακή πλακέτα του εργαστηρίου.

Σημείωση: Ο ακριβής τρόπος λειτουργίας του προγράμματος υποδεικνύεται μέσω σχολίων σε εντολές του κώδικα.

Κώδικας σε C

```c
uint8_t one_wire_reset(void) {
    DDRD = 0b00010000; //PD4 output
    PORTD = PORTD & 0b11101111; //PD4 = 0;
    _delay_us(480); //480 usec reset pulse

    DDRD = DDRD & 0b11101111; //set PD4 as input
    PORTD = PORTD & 0b11101111; // disable pull-up
    _delay_us(100); //wait 100 usec for connected devices to transmit the presence pulse

    uint8_t state = PIND;
    _delay_us(380);

    if((state & 0x10) == 0x00)
        return 0x01;
    return 0x00;
    // returns 1 if PD4 = 0
    // if a connected device is detected(PD4=0) return 1 else return 0
}

uint8_t one_wire_receive_bit(void) {
    DDRD = 0b00010000; //PD4 output
    PORTD = PORTD & 0b11101111; //PD4 = 0;
    _delay_us(2); //time slot 2 usec

    DDRD = 0b11101111; //set PD4 as input
    PORTD = PORTD & 0b11101111; // disable pull-up
    _delay_us(10);

    uint8_t state = PIND;
    _delay_us(49); //delay 49 usec to meet the standards

    if((state & 0x10) == 0x10)
        return 0x01;
    return 0x00; // returns 1 if PD4 is 1
}

void one_wire_transmit_bit(uint8_t data) {
    DDRD = 0b00010000; //PD4 output
    PORTD = PORTD & 0b11101111; //PD4 = 0;
    _delay_us(2);

    PORTD = (PORTD & 0xEF) | ((data << 4) & 0x10);
    // alter the PD4 of current PORTD's state and make it 0.
    // then bitwise or with data
    // we want to change only the PD4 bit
    _delay_us(58); //wait 58 usec for connected device to sample the line

    DDRD = DDRD & 0b11101111; //set PD4 as input
    PORTD = PORTD & 0b11101111; // disable pull-up
    _delay_us(1); //recovery time 1 usec
}

uint8_t one_wire_receive_byte(void) { //starts from LSB
    uint8_t data = 0;
    for (int i = 0; i < 8; i++) {
        uint8_t x = one_wire_receive_bit(); //x[0] holds the received bit
        data = data >> 1;
        if (x == 0x01)
            data = data | 0x80;
    }
    return data;
}
```

```
void one_wire_transmit_byte(uint8_t data) { //starts from LSB
    for (int i = 0; i < 8; i++) {
        one_wire_transmit_bit(data);
        data = data >> 1;
    }
}

int read_temp(void) { // the value returned is the temperature
                      // with 0.0625 degrees decision (DS18B20 sensor)
    if (one_wire_reset() == 0x01) {
        one_wire_transmit_byte(0xCC); //0xCC skip device choice

        one_wire_transmit_byte(0x44); //0x44 start temperature reading

        while (one_wire_receive_bit() == 0x00); //wait for reading completion

        one_wire_reset();             // new reset of the device
        one_wire_transmit_byte(0xCC); //0xCC skip device choice

        one_wire_transmit_byte(0xBE); //0xBE read temperature

        uint8_t temperature = one_wire_receive_byte();
        uint8_t sign = one_wire_receive_byte();
        return (((uint16_t)sign) << 8) + (uint16_t)temperature;
        // combine the two 8bits numbers to one comprised of 16bits
    } else return 0x8000;
}
```

Ζήτημα 7.2:

Παρακάτω φαίνεται ο κώδικας σε C που υλοποιεί τα ζητούμενα της άσκησης.

Η ορθή λειτουργία των προγραμμάτων έχει ελεγχθεί στο περιβάλλον προσομοίωσης MPLAB X, καθώς και στην αναπτυξιακή πλακέτα του εργαστηρίου.

Σημείωση:

(α) Ο ακριβής τρόπος λειτουργίας του προγράμματος υποδεικνύεται μέσω σχολίων σε εντολές του κώδικα.

(β) Οι βοηθητικές συναρτήσεις που αφορούν τον αισθητήρα θερμοκρασίας δεν ξαναπαρουσιάζονται παρακάτω. Ταυτίζονται με αυτές του προηγούμενου ζητήματος.

Κώδικας σε C

```
#define F_CPU 16000000UL
#include <math.h>
#include <util/delay.h>
#include <avr/io.h>

//-------------------------------------------------------------LCD SCREEN-------

// send one byte divided into 2 (4 bit) parts

void write_2_nibbles(char data) {
    char pinState = PIND; // read 4 LSB and resend them
    // in order not to alter any previous state
    PORTD = (pinState & 0x0F) | (data & 0xF0) | (1 << PD3); // send 4MSB
    PORTD &= (0xFF) & (0 << PD3); // set PD3 to zero, lcd enable pulse
    PORTD = (pinState & 0x0F) | ((data << 4) & 0xF0) | (1 << PD3); // send 4LSB
    PORTD &= (0xFF) & (0 << PD3); // set PD3 to zero, lcd enable pulse
}
// send one byte of data to the lcd display

void lcd_data(char data) {
    PORTD |= (1 << PD2); // select data register
    write_2_nibbles(data);
    _delay_us(100);
}
// send one byte of instuction to the lcd display

void lcd_command(char data) {
    PORTD &= (0xFF) & (0 << PD2); // select command register
    write_2_nibbles(data);
    _delay_us(100);
}
```

```
void lcd_init() {
    _delay_ms(40); // lcd init procedure
    PORTD = 0x30 | (1 << PD3); // 8 bit mode
    PORTD &= (0xFF) & (0 << PD3); // set PD3 to zero, lcd enable pulse
    _delay_us(100);
    PORTD = 0x30 | (1 << PD3); // 8 bit mode
    PORTD &= (0xFF) & (0 << PD3); // set PD3 to zero, lcd enable pulse
    _delay_us(100);
    PORTD = 0x20 | (1 << PD3); // change to 4 bit mode
    PORTD &= (0xFF) & (0 << PD3); // set PD3 to zero, lcd enable pulse
    _delay_us(100);
    lcd_command(0x28); // select character size 5x8 dots and two line display
    lcd_command(0x0c); // enable lcd, hide cursor
    lcd_command(0x01); // clear display
    _delay_us(5000);
    lcd_command(0x06); // enable auto increment of address, disable shift of the display
}

int main(void) {

    DDRD = 0xFF;
    lcd_init(); // init lcd
    _delay_ms(2); // wait for lcd init

    while (1) {
        lcd_command(0x08); // disable lcd
        uint16_t temperature = read_temp();
        DDRD = 0xFF;
        lcd_init(); // init lcd
        _delay_ms(2); // wait for lcd init
        if (temperature == 0x8000) { //NO DEVICE
            lcd_data('N');
            _delay_ms(100);

            lcd_data('O');
            _delay_ms(100);

            lcd_data(' ');
            _delay_ms(100);

            lcd_data('D');
            _delay_ms(100);

            lcd_data('E');
            _delay_ms(100);

            lcd_data('V');
            _delay_ms(100);

            lcd_data('I');
            _delay_ms(100);

            lcd_data('C');
            _delay_ms(100);

            lcd_data('E');
            _delay_ms(100);
        } else {
            if ((temperature & 0x8000) == 0x1000) { //negative
                lcd_data(0x2D); //-
                _delay_ms(100);
                temperature = ~temperature; // 2's complement
                temperature += 1;
            } else {
                lcd_data(0x2B); //+
                _delay_ms(100);
            }
            // now variable temperature holds the absolute value
            // of the temperature read by the sensor
```

```
            uint8_t intTemp = (temperature >> 4); // integer part of temperature
            char hund = (intTemp / 100); // hundreds part of temperature
            if (hund != 0) {
                lcd_data(hund + 0x30);
                _delay_ms(100);
            }

            char tens = ((intTemp % 100) / 10); // tens part of temperature
            if ((tens != 0) || (hund != 0)) {
                lcd_data(tens + 0x30);
                _delay_ms(100);
            }

            char ones = (intTemp % 10);      //ones part of temperature
            lcd_data(ones + 0x30);
            _delay_ms(100);

            lcd_data(0x2E); // dot
            _delay_ms(100);

            // remember that temperature holds the value of temperature
            // with 0.0625 degrees precision. That means that
            // the variable temperature must be multiplied by 0.0625 in order
            // to hold the integer part of the temperature
            // multiply that by 10 in order the last digit (of the dec number) to be the
            // previous first decimal. take the mod of 10.

            char firstDecimal = ((int) (temperature*0.0625 * 10) % 10); // 1st decimal
            lcd_data(firstDecimal + 0x30);
            _delay_ms(100);

            char secondDecimal = ((int) (temperature*0.0625 * 100) % 10); // 2nd decimal
            lcd_data(secondDecimal + 0x30);
            _delay_ms(100);

            char thirdDecimal = ((int) (temperature*0.0625 * 1000) % 10); // 3rd decimal
            lcd_data(thirdDecimal + 0x30);
            _delay_ms(100);

            lcd_data('o'); //degrees
            _delay_ms(100);
            lcd_data('C');
            _delay_ms(100);
        }
        _delay_ms(500); // wait for lcd init
        lcd_command(0x01); // clear display
        _delay_us(5000);
    }
}
```