7η εργαστηριακή άσκηση

Φωτεινή Κυριακοπούλου 03120182
Ιωάννης Μπουφίδης 03120162

Ζήτημα 7.1-2

```c
#define F_CPU 16000000UL

#include<avr/io.h>
#include<avr/interrupt.h>
#include<util/delay.h>

// swap macro
uint8_t swap(uint8_t x) {
    return (uint8_t)((x >> 4) | (x << 4));
}

void write_2_nibbles(uint8_t command) {
    // write MSB nibble
    PORTD = ((PIND & 0X0f) | (command & 0xf0));

    PORTD |= (1 << PD3);  // Enable pulse
    _delay_us(1);
    PORTD &= ~(1 << PD3); // Disable pulse

    // write LSB nibble
    PORTD = ((PIND & 0X0f) | (swap(command) & 0xf0));

    PORTD |= (1 << PD3);
    _delay_us(1);
    PORTD &= ~(1 << PD3);
}

// send a command to lcd's controller (4-bit mode)
void lcd_command(uint8_t command) {
    PORTD &= ~(1 << PD2);  // LCD_RS = 0 => instruction
    write_2_nibbles(command);  // send instruction
    _delay_us(250);  // wait 250us
}

// send data to lcd's controller (4-bit mode)
void lcd_data(uint8_t data) {
    PORTD |= (1 << PD2);  // LCD_RS = 1 => data
    write_2_nibbles(data);  // send instruction
    _delay_us(250);  // wait 250us
}

void lcd_clear_display() {
    lcd_command(0x01);
    _delay_ms(5);
}
```

7η εργαστηριακή άσκηση

```
// lcd monitor set-up
void lcd_init() {
    _delay_ms(200);  // wait 200ms

    // set microcontroller in 4-bit mode

    for (int i=0; i<3; i++) {  // 3 times
        PORTD = 0x30;  // switch to 8-bit mode
        PORTD |= (1 << PD3);  // Enable pulse
        _delay_us(1);
        PORTD &= ~(1 << PD3); // Disable pulse
        _delay_us(250);
    }

    PORTD = 0x20;  // switch to 4-bit mode
    PORTD |= (1 << PD3);  // Enable pulse
    _delay_us(1);
    PORTD &= ~(1 << PD3); // Disable the enable pulse
    _delay_us(250);

    lcd_command(0x28);   // 5x8 dots, 2 lines
    lcd_command(0x0c);   // display on, cursor-blinking off
    lcd_clear_display(); // clear display
    lcd_command(0x06);   // increase address, no display shift
}

int one_wire_reset() {
    DDRD |= (1 << PD4);     // set PD4 as output
    PORTD &= (0 << PD4);
    _delay_us(480);         // 480 usec reset pulse

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

    uint8_t portd = PIND;   // read PORTD
    _delay_us(380);         // wait for 380 usec

    // if a connected device is detected (PD4=0) return 1
    if ((portd & (1 << PD4)) == 0) return 1;
    // else return 0
    else return 0;
}

uint8_t one_wire_receive_bit() {
    DDRD |= (1 << PD4);     // set PD4 as output
    PORTD &= (0 << PD4);
    _delay_us(2);           // time slot 2 usec

    DDRD &= (0 << PD4);     // set PD4 as input
    PORTD &= (0 << PD4);    // disable pull-up
    _delay_us(10);          // wait 10 usec
```

```
    uint8_t portd = PIND;
    _delay_us(49);          // delay 49 usec to meet the standards
    return ((portd & 0x10) >> 4);  // return PD4
}

void one_wire_transmit_bit(uint8_t tr_bit) {
    DDRD |= (1 << PD4);     // set PD4 as output
    PORTD &= (0 << PD4);
    _delay_us(2);           // time slot 2 usec

    // if transmitted bit is '1', set PD4
    if (tr_bit == 1) PORTD |= (1 << PD4);
    else PORTD &= (0 << PD4);
    _delay_us(58);  // wait 58 usec for connected device to sample the line

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

uint8_t one_wire_receive_byte() {
    uint8_t byte = 0, b = 0;
    for (int i = 0; i < 8; i++) {
        byte >>= 1;                 // shift received byte right
        b = one_wire_receive_bit();
        if (b == 1) byte |= 0x80;   // if received bit is '1', set byte's MSB
    }
    return byte;
}

void one_wire_transmit_byte(uint8_t byte) {
    uint8_t tr_bit;
    for (int i=0; i<8; i++) {
        tr_bit = byte & 0x01;
        one_wire_transmit_bit(tr_bit);    // transmit byte's LSB
        byte >>= 1;                       // shift transmitted byte right
    }
}

int temperature() {
    if (one_wire_reset() == 0) return 0x8000; //no device connected
    one_wire_transmit_byte(0xCC); //skip device selection
    one_wire_transmit_byte(0x44); //start measurement

    uint8_t finished = 0;
    while (!finished) { //wait for measurement to be completed
        finished = one_wire_receive_bit();
    }
    if (one_wire_reset() == 0) return 0x8000;
    one_wire_transmit_byte(0xCC);

    //read 16-bit value of measurement
```

```
    one_wire_transmit_byte(0xBE);
    uint8_t temp_low = one_wire_receive_byte(); //low-byte
    uint8_t temp_high = one_wire_receive_byte(); //high-byte

    uint16_t value = 0;
    //if sign bits are set (negative temperature), set the 12th bit of the 16-bit
    //value
    if ((temp_high & 0xF8) == 0xF8) value = 1;
    //combine the low and high bytes
    value <<=3;
    value |= (temp_high & 0x07);
    value <<= 8;
    value |= temp_low;

    return value;
}

void lcd(uint16_t temp) {
    //initialize LCD screen
    DDRD |= 0xFF;
    lcd_init();
    _delay_ms(100);

    //if 12th bit not set - positive temperature
    if((temp & 0x0800) != 0x0100) lcd_data('+');
    //else negative temperature
    else {
        lcd_data('-');
        temp = ~(temp) + 1; //2's complement
        //2's complement -> sign bit set to '0'
    }

    //compute the fractional part (bits 0-3)
    float dec = 0;
    for (int i=4; i>0; i--) {
        if ((temp & 0x01) == 0x01) dec += 1/pow(2,i);
        temp >>= 1;
    }

    //display hundreds
    int h = temp/100;
    if (h != 0) lcd_data(h + '0');
    //display tens
    int t = (temp%100)/10;
    if (t!=0 || h!=0) lcd_data(t + '0');
    //display ones
    int o = temp%10;
    lcd_data(o + '0');

    lcd_data('.');
    //if there's a fractional part
    if (dec != 0) {
        int digit = 0;
```

```
        for (int i = 0; i < 3; i++) {
            digit = dec*10;
            dec = dec*10 - (float)digit;
            if (dec >= 0.5 && i == 2) digit++; //round last digit
            lcd_data(digit + '0'); //display each digit
        }
    }
    else {
        for (int i=0; i<3; i++) lcd_data('0'); //else display zeros
    }
}

char no_dev[] = {'N', 'O', ' ', 'D', 'e', 'v', 'i', 'c', 'e'};
char zero_deg[] = {'0', '.', '0', '0', '0'};

//display 'NO Device'
void dev_msg() {
    DDRD |= 0xFC;
    lcd_init();
    _delay_ms(100);
    for (int i=0; i<9; i++) {
        lcd_data(no_dev[i]);
    }
}

//display '0.000'
void zero_msg() {
    DDRD |= 0xFC;
    lcd_init();
    _delay_ms(100);
    for (int i=0; i<5; i++) {
        lcd_data(zero_deg[i]);
    }
}

void main(void) {
    uint16_t temp;
    while(1) {
        temp = temperature();             //get output of temperature sensor
        if (temp == 0x8000) dev_msg();    //there's no device connected
        //display temperature value
        else if (temp == 0) zero_msg();
        else lcd(temp);
        _delay_ms(750);
    }
}
```