7η Εργαστηριακή Άσκηση
Εργαστήριο Μικροϋπολογιστών

Ανδρέας Στάμος (03120018), Κωνσταντίνος Πίκουλας (03120112)

Δεκέμβριος 2023

Περιεχόμενα

1 Ζήτημα 1 – Ανάπτυξη συναρτήσεων επικοινωνίας 1-Wire με το θερμόμετρο

Αρχικά εκτελούμε `one_wire_reset` προκειμένου να θέσουμε το θερμόμετρο σε κατάσταση λειτουργίας. Έπειτα στέλνουμε εντολή (`0xCC`)πως απευθυνόμαστε στην μοναδική συνδεδεμένη συσκεύη (παρακάμπτωντας την αποστολή διεύθυνσης έτσι), και έπειτα στέλνουμε εντολή θερμομέτρησης (`0x44`). Μέχρι να ολοκληρωθεί η θερμομέτρηση, το θερμόμετρο μας στέλνει 0, αν ζητήσουμε να στείλει ένα bit (με την `one_wire_receive_bit`). Μόλις μας στείλει 1, η θερμομέτρηση έχει ολοκληρωθεί.

Έπειτα θέτουμε ξανά την συσκεύη σε κατάσταση λειτουργίας εκ νέου με `one_wire_reset`, την επιλέγουμε στέλνοντας `0xCC`, και στέλνουμε εντολή ανάγνωσης της μνήμης της θερμομέτρου (`0xBE`). Τέλος, διαβάζουμε 2 bytes (με `one_wire_receive_byte`), πρώτα το LSB και έπειτα το MSB (little-endian) και τα ενώνουμε σε μία προσημασμένη 16bit μεταβλητή (int16_t).

Ο κώδικας έχει διαχωριστεί σε χωριστά αρχεία, με σκοπό να είναι modular.

Ο κώδικας για την επικοινωνία με το θερμόμετρο βρίσκεται στο αρχείο `thermometer.c` που υπάρχει στο παράρτημα Δ′, με την αντίστοιχη C επικεφαλίδα να βρίσκεται στο παράρτημα Α′.

Ο κώδικας για την One Wire επικοινωνία – χρησιμοποιείται από το `thermometer.c` βρίσκεται στο αρχείο `onewire.c` που υπάρχει στο παράρτημα Δ′, με την αντίστοιχη C επικεφαλίδα να βρίσκεται στο παράρτημα Γ′.

2 Ζήτημα 2 – Απεικόνιση θερμοκρασίας στην LCD οθόνη

Ο κώδικας για την επικοινωνία με την LCD οθόνη συντάχθηκε σε προηγούμενη εργασία και παρατίθεται εδώ ξανά. Πιο συγκεκριμένα:

- `lcd.c` στο παράρτημα ΣΤ′ και `lcd.h` στο παράρτημα Ε′ – για την επικοινωνία με την οθόνη. Πρέπει κατά την μεταγλώττιση του `lcd.c` να γίνει `define` η macro `LCD_PORTD` ή `LCD_PCA9555` ανάλογα το πώς επιθυμούμε να γίνεται η επικοινωνία. Εδώ επιθυμούμε μέσω PCA9555 διότι το PD4 χρησιμοποιείται για την 1-wire επικοινωνία με το θερμόμετρο.

- `twi_pca9555.c` στο παράρτημα Η′ και `twi_pca9555.h` στο παράρτημα Ζ′ για την επικοινωνία TWI και επιπρόσθετα για την επικοινωνία μέσω TWI με το ολοκληρωμένο PCA9555.

Αρχικά διαβάζουμε την θερμοκρασία από το θερμόμετρο με την συνάρτηση που αναπτύχθηκε στο Ζήτημα 1. Αν δεν ανιχνευθεί συσκευή, εκτυπώνουμε `NO Device` στην οθόνη. Διαφορετικά εκτυπώνουμε την θερμοκρασία με 3 δεκαδικά ψηφία. Αρχικά μετατρέπουμε τον fixed-point αριθμό που μας δίνει το θερμόμετρο (4 bits κλασματικό μέρος και 8 bits ακέραιο), σε αριθμό κινητής υποδιαστολής τύπου `float` και στην συνέχεια με χρήση της συνάρτησης `snprintf` μετατρέπουμε τον αριθμό αυτό σε συμβολοσειρά. Τέλος εκτυπώνουμε την συμβολοσειρά στην οθόνη. Δόθηκε έμφαση στην σωστή ερμηνείας των αρνητικών αριθμών (δίνονται από το θερμόμετρο ως συμπλήρωμα ως προς 2). Τέλος, προκειμένου η οθόνη να μην κάνει "blink", ανανενώνουμε το μήνυμα στην οθόνη μόνο αν η ανάγνωση θερμοκρασίας επιστρέψει καινούρια τιμή σε σχέση με την προηγούμενη φορά που εκτυπώθηκε θερμοκρασία στην οθόνη. Αξίζει, επίσης να επισημανθεί, πως για να λειτουργήσει η snprintf με float, πρέπει να γίνει link με την σωστή snprintf (η default δεν περιέχει λειτουργικότητα αριθμών κινητής υποδιαστολής για οικονομία χώρου), και αυτό φαίνεται παρακάτω στο `Makefile`.

Ο κώδικας του ζητήματος είναι ο εξής:

```
#include "../common/thermometer.h"
#include "../common/lcd.h"
#include "../common/onewire.h"

#include<stdio.h>

int main(void) {
    const char nodevmsg[] = "NO device";

    lcd_init();

    int16_t t = 0xffff;
    while(1) {
        int16_t t_new = get_temperature();
        //if the same message is to be displayed, dont redisplay it.
        if (t == t_new) continue;
        t = t_new;

        if (t == 0x8000) {
            //no device
            lcd_clear_display();
            for (int8_t i=0; i<sizeof(nodevmsg)-1; ++i) lcd_data(nodevmsg[i]);
            continue;
        }

        uint8_t is_neg = t<0;
        if (t<0) t=-t;

        float t_fl = ((int16_t) (t >> 4)) + ((float)(t & 0xf))/16.0;
        if (is_neg) t_fl = -t_fl;

        char message[15];
        int8_t length;
        length = snprintf(message, sizeof(message), "%+.3f""\xdf""C", t_fl);
        lcd_clear_display();
        for (int8_t i=0; i<length && i<sizeof(message); ++i) lcd_data(message[i]);
    }
}

```

Τέλος, επειδή η μεταγλώττιση και η εγκατάσταση στην πλακέτα γίνεται χωρίς το MPLAB X, αλλά με μεταγλώττιση με AVR-GCC και εγκατάσταση με avrdude στην μνήμη προγράμματος παρατίθεται το σχετικό `Makefile`:

Ο φάκελος `pack` που αναφέρεται είναι οι σχετικές βιβλιοθήκες με τις αντίστοιχες επικεφαλίδες για τον ATmega328PB, καθώς δεν υποστηρίζεται από default από τον avr-gcc. Αρκεί κανείς να κατεβάσει από την σχετική ιστοσελίδα της Microchip https://packs.download.microchip.com/ το σχετικό pack για την ATmega Series και να το αποσυμπιέσει μέσα στον φάκελο `pack`.

```
MCU = atmega328pb
F_CPU = 16000000UL

CC = avr-gcc
LD = avr-ld
OBJCOPY = avr-objcopy

CFLAGS = -Wall -Wextra -Bpack/gcc/dev/atmega328pb -Ipack/include -mmcu=$(MCU) -DF_CPU=$(F_CPU) -Wl,--gc-sections -Os -ffunction-sections -fdata-sections -fshort-enums -funsigned-char -funsigned-bitfields -gdwarf-3  -Wl,-u,vfprintf -lprintf_flt -lm
LDFLAGS = -Bpack/gcc/dev/atmega328pb -mmcu=$(MCU)


all: main

../common/lcd.o: CFLAGS += -DLCD_PCA9555

main: ../common/thermometer.o ../common/onewire.o ../common/lcd.o ../common/twi_pca9555.o

main.hex: main
	avr-objcopy -O ihex main main.hex

install: main.hex
	avrdude -p m328pb -c xplainedmini -U flash:w:main.hex

clean:
	rm -f main main.hex main.o ../common/keypad.o ../common/twi_pca9555.o ../common/lcd.o ../common/thermometer.o ../common/onewire.o

.PHONY: all install clean

```

Α΄ Αρχείο thermometer.h

```
#ifndef __THERMOMETER_H
#define __THERMOMETER_H

#include<stdint.h>

int16_t get_temperature(void);

#endif
```

Β΄ Αρχείο thermometer.c

```
#include "onewire.h"
#include "thermometer.h"

int16_t get_temperature(void) {
        if (!one_wire_reset()) return 0x8000L;
        one_wire_transmit_byte(0xCC);
        one_wire_transmit_byte(0x44);
        while (!one_wire_receive_bit());
        if (!one_wire_reset()) return 0x8000L;
        one_wire_transmit_byte(0xCC);
        one_wire_transmit_byte(0xBE);

        int16_t ret;
        ret = one_wire_receive_byte(); //LSB byte
        ret |= one_wire_receive_byte() << 8; //MSB byte

        return ret;
}

```

Γ΄ Αρχείο onewire.h

```
#ifndef __ONEWIRE_H
#define __ONEWIRE_H

#include<stdint.h>

uint8_t one_wire_reset(void);
uint8_t one_wire_receive_bit(void);
void one_wire_transmit_bit(uint8_t);
uint8_t one_wire_receive_byte(void);
void one_wire_transmit_byte(uint8_t);

#endif

```

Δ΄ Αρχείο onewire.c

```
#include<stdint.h>
#include<avr/io.h>
#include<util/delay.h>

#include "onewire.h"

#define SETOUTPUT() do { DDRD |= (uint8_t) 1U<<4; } while (0)
#define OUTPUT_0() do { PORTD &= (uint8_t) ~(1U<<4); } while (0)
```

```
#define OUTPUT_1() do { PORTD |= (uint8_t) 1U<<4; } while (0)
#define OUTPUT(x) OUTPUT_##x()

#define SETINPUT() do { DDRD &= (uint8_t) ~(1U<<4); PORTD &= (uint8_t) ~(1U<<4); } while (0)
#define READ() ((PIND >> 4) & 1U)

uint8_t one_wire_reset(void) {
        SETOUTPUT();
        OUTPUT(0);
        _delay_us(480);
        SETINPUT();
        _delay_us(100);
        uint8_t ret = (READ() == 0);
        _delay_us(380);
        return ret;
}

uint8_t one_wire_receive_bit(void) {
        SETOUTPUT();
        OUTPUT(0);
        _delay_us(2);
        SETINPUT();
        _delay_us(10);
        uint8_t ret = READ();
        _delay_us(49);
        return ret;
}

void one_wire_transmit_bit(uint8_t datum) {
        SETOUTPUT();
        OUTPUT(0);
        _delay_us(2);
        if (datum & 1) OUTPUT(1);
        else OUTPUT(0);
        _delay_us(58);
        SETINPUT();
        _delay_us(1);
}

uint8_t one_wire_receive_byte(void) {
        uint8_t ret = 0;
        for (int8_t i=0; i<8; ++i) ret |= one_wire_receive_bit() << i;
        return ret;
}

void one_wire_transmit_byte(uint8_t data) {
        for (int8_t i=0; i<8; ++i) {
                one_wire_transmit_bit(data & 1);
                data >>= 1;
        }
}

```

Ε΄ Αρχείο lcd.h

```
#ifndef __LCD_H
#define __LCD_H

#include<stdint.h>

```

```
void lcd_data(uint8_t data);
void lcd_command(uint8_t command);
void lcd_goto_line1(void);
void lcd_goto_line2(void);
void lcd_clear_display(void);
void lcd_init(void);

#endif
```

ΣΤ' Αρχείο lcd.c

```
#include<util/delay.h>

#include "lcd.h"

#define NOP() { asm volatile ("nop"); }

#define RS_DATA 1
#define RS_COMMAND 0

#if defined(LCD_PORTD) && defined(LCD_PCA9555)
#error "Both LCD_PORTD and LCD_PCA9555 are defined. Only one should be defined."
#elif !defined(LCD_PORTD) && !defined(LCD_PCA9555)
#error "Neither LCD_PORTD nor LCD_PCA9555 is defined. One of them must be defined."
#endif

#ifdef LCD_PORTD
#include<avr/io.h>
#define WRITE_LCD(x) PORTD = (uint8_t) (x)
#endif

#ifdef LCD_PCA9555
#include "twi_pca9555.h"
#define WRITE_LCD(x) PCA9555_0_write(REG_OUTPUT_0, (uint8_t) (x))
#endif

static void write_2_nibbles(uint8_t command, uint8_t type) {
        WRITE_LCD((command & 0xf0) | type<<2 | 1U<<3);
        NOP();
        NOP();
        WRITE_LCD((command & 0xf0) | type<<2);

        WRITE_LCD(((command & 0x0f) << 4) | type<<2 | 1U<<3);
        NOP();
        NOP();
        WRITE_LCD(((command & 0x0f) << 4) | type<<2);
}

void lcd_data(uint8_t data) {
        write_2_nibbles(data, RS_DATA);
        _delay_ms(0.250);
}

void lcd_command(uint8_t command) {
        write_2_nibbles(command, RS_COMMAND);
        _delay_ms(0.250);
}

void lcd_goto_line1(void) {
```

```
        lcd_command((1U<<7) | 0x00); //set ddram address to 0x00
}

void lcd_goto_line2(void) {
        lcd_command((1U<<7) | 0x40); //set ddram address to 0x40
}

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

void lcd_init(void) {

#ifdef LCD_PCA9555
        twi_init();
        //configure pca9555_io0 as output
        PCA9555_0_write(REG_CONFIGURATION_0, 0x00);
#endif

#ifdef LCD_PORTD
        DDRD = 0xFF;
#endif

        _delay_ms(200); // wait for screen to initialize
        for(uint8_t i=0; i<3; ++i) {
                WRITE_LCD(0x38); // set 8 bit mode 3 times
                NOP();
                NOP();
                WRITE_LCD( 0x30);
                _delay_ms(0.250);
        }

        // switch to 4bit mode
        WRITE_LCD(0x28);
        NOP();
        NOP();
        WRITE_LCD(0x20);
        _delay_ms(0.250);

        lcd_command(0x28);
        lcd_command(0x0c);
        lcd_clear_display();
}

```

Z′ Αρχείο twi_pca9555.h

```
#ifndef __TWI_PCA9555_H
#define __TWI_PCA9555_H

#include<stdint.h>

#define REG_INPUT_0                0
#define        REG_INPUT_1                 1
#define        REG_OUTPUT_0                2
#define REG_OUTPUT_1                3
#define REG_POLARITY_INV_0        4
#define REG_POLARITY_INV_1        5
#define REG_CONFIGURATION_0        6
```

```
#define REG_CONFIGURATION_1        7

void twi_init(void);
unsigned char twi_start(uint8_t address);
void twi_stop(void);
unsigned char twi_write(uint8_t data);
unsigned char twi_readAck(void);
unsigned char twi_readNak(void);
void twi_start_wait(uint8_t address);

void PCA9555_0_write(uint8_t reg, uint8_t value);
uint8_t PCA9555_0_read(uint8_t reg);

#endif
```

Η΄ Αρχείο twi_pca9555.c

```
#include<avr/io.h>
#include<util/twi.h>
#include<stdint.h>

#include "twi_pca9555.h"

#define PCA9555_0_ADDRESS 0x40

#define SCL_CLOCK 100000L
#define TWBR0_VALUE ((F_CPU/SCL_CLOCK-16)/2)

#define TW0_STATUS (TWSR0 & 0xF8)


void twi_init(void) {
        TWSR0 = 0; //prescaler = 1
        TWBR0 = TWBR0_VALUE;
}

unsigned char twi_start(uint8_t address) {
        //7 msb of address are address and lsb is set if R and not set if W

        TWCR0 = 1<<TWSTA | 1<<TWINT | 1<<TWEN; //START
        while (!(TWCR0 & (1<<TWINT))); //wait till START transmitted
        if (TW0_STATUS != TW_START && TW0_STATUS != TW_REP_START) return 1; //failed

        TWDR0 = address; //SLA_W or SLA_R
        TWCR0 = 1<<TWINT | 1<<TWEN;

        while (!(TWCR0 & (1<<TWINT))); //wait till SLA transmitted

        //if SLA_R
        if (address & 0x01) {
                if (TW0_STATUS != TW_MR_SLA_ACK) return 1; //failed
        } else {
        //if SLA_W
                if (TW0_STATUS != TW_MT_SLA_ACK) return 1; //failed
        }

        return 0;
}

```

```
43  void twi_stop(void) {
44          TWCR0 = 1<<TWSTO | 1<<TWINT | 1<<TWEN; //STOP
45          while (!(TWCR0 & (1<<TWSTO))); //wait till STOP transmitted
46  }
47
48  unsigned char twi_write(uint8_t data) {
49          TWDR0 = data;
50          TWCR0 = 1<<TWINT | 1<<TWEN;
51          while (!(TWCR0 & (1<<TWINT))); //wait till transmitted
52          if (TW0_STATUS != TW_MR_DATA_ACK) return 1;
53          return 0;
54  }
55
56  unsigned char twi_readAck(void) {
57          TWCR0 = 1<<TWINT | 1<<TWEA | 1<<TWEN;
58          while (!(TWCR0 & (1<<TWINT))); //wait till received
59          return TWDR0;
60  }
61
62  unsigned char twi_readNak(void) {
63          TWCR0 = 1<<TWINT | 1<<TWEN;
64          while (!(TWCR0 & (1<<TWINT))); //wait till received
65          return TWDR0;
66  }
67
68  void twi_start_wait(uint8_t address) {
69          while (twi_start(address));
70  }
71
72  void PCA9555_0_write(uint8_t reg, uint8_t value) {
73          twi_start_wait(PCA9555_0_ADDRESS | TW_WRITE);
74          twi_write(reg); //what if PCA9555 NACKs?
75          twi_write(value);
76          twi_stop();
77  }
78
79  uint8_t PCA9555_0_read(uint8_t reg) {
80          uint8_t ret_val;
81          twi_start_wait(PCA9555_0_ADDRESS | TW_WRITE);
82          twi_write(reg); //what if PCA9555 NACKs?
83          twi_start(PCA9555_0_ADDRESS | TW_READ); //what if fails? should we wait instead?
84          ret_val = twi_readNak();
85          twi_stop();
86          return ret_val;
87  }
88
```