Mikroprocesorové praktikum **FJFI ČVUT v Praze**

Úloha 09 – Sériové komunikační rozhraní I2C

Jméno: **Kateřina Hrnečková** Měřeno: **17.3.2023** Klasifikace:

1 Pracovní úkoly

1. Seznamte se modulem Master Synchronous Serial Port (MSSP).
2. Vytvořte program v jazyce C, který bude přes sériové rozhraní komunikovat s vybraným zařízením.
3. Program bude obsahovat tyto funkce:
   - void I2C_Init(void), případně void SPI_Init(void)
   - uint8_t I2C_Receive(uint8_t address), případně uint8_t SPI_Receive(void)
   - void I2C_Send(uint8_t address, uint8_t data), případně void SPI_Send(uint8_t data)
4. Program přeložte a vložte do paměti mikrokontroléru.

2 Vypracování

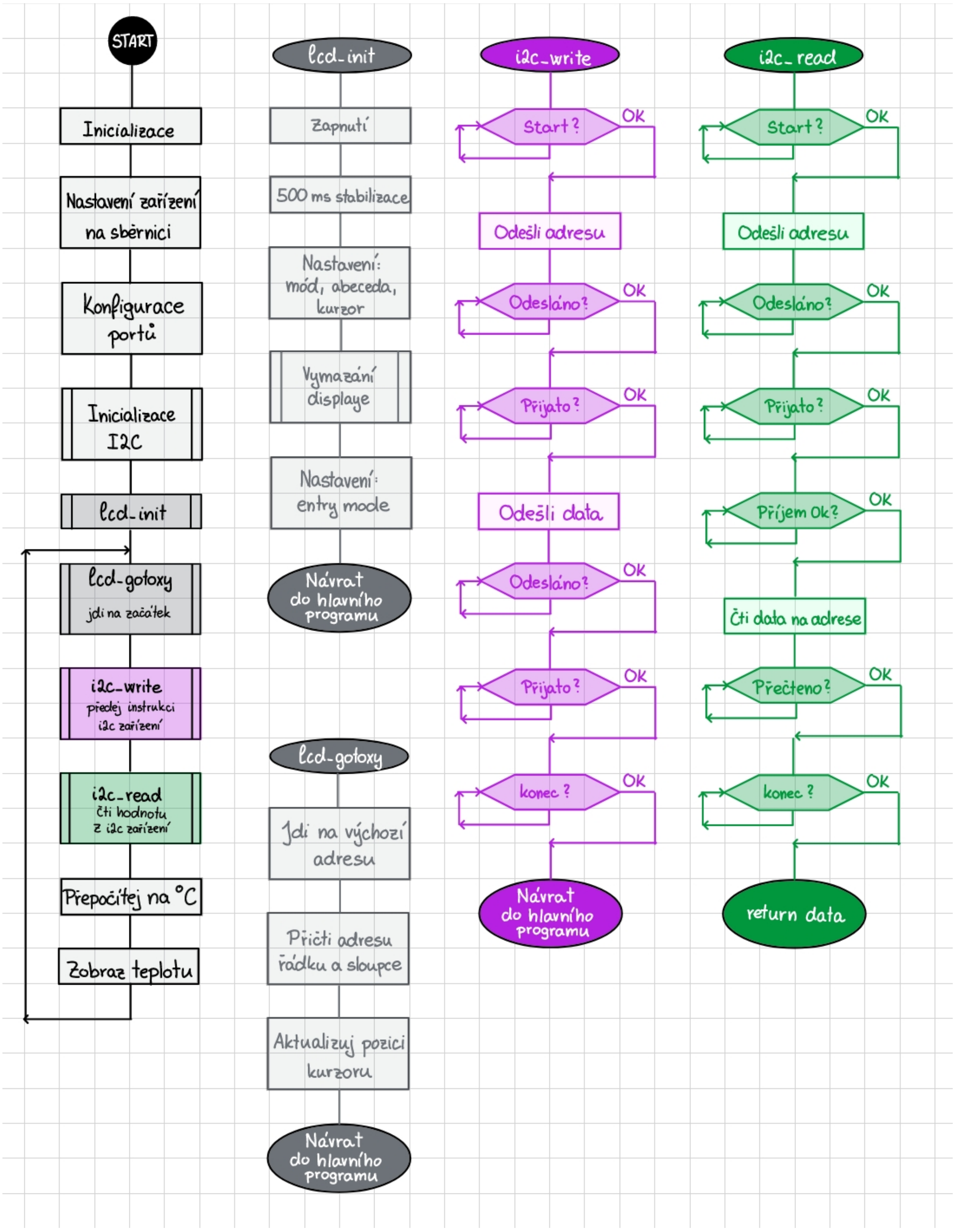
Řešená úloha: Zobrazení naměřené teploty.

```
#include <xc.h>
#include <stdint.h>
#include <stdio.h>
#include <pic16f877a.h>

#define _XTAL_FREQ 3276800                // Frekvence krystalu

// CONFIG
#pragma config FOSC = XT        // Oscillator Selection bits (XT oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR disabled)
#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data EEPROM code protection of
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write protection off; all prog
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection off)

#define RS PORTEbits.RE0
#define RW PORTEbits.RE1
#define EN PORTEbits.RE2

#define DATA PORTD


void __interrupt() preruseni(void)
{
    //...
}
```

```
void i2c_init()
{
        TRISCbits.TRISC4 = 1;        // data IN = 1
        TRISCbits.TRISC3 = 0;        // clock OUT = 0

        SSPCON = 0b00101000; // MSSP Control Register (SSPCON)
        // SSPCON2
        SSPSTAT = 0b10000000; // MSSP Status Register (SSPSTAT)
        /*
                SSPCON, SSPCON2 and SSPSTAT are the control
                and status registers in I2C mode operation. The
                SSPCON and SSPCON2 registers are readable and
                writable. The lower six bits of the SSPSTAT are
                read-only. The upper two bits of the SSPSTAT are
                read/write.
        */
        SSPADD = 7;

}
void i2c_write(uint8_t addr, uint8_t dat)
{
        SEN = 1;
        while(SEN != 0)                 // start condition
                ;

        SSPBUF = addr << 1; // I2C 7bit address, 0 na konci = write
        SSPIF = 0;
        while(SSPIF != 1)        // odeslano??
                ;
        while(ACKSTAT != 0)        // prijato??
                ;

        SSPBUF = dat;                // control data
        SSPIF = 0;
        while(SSPIF != 1)        // odeslano??
                ;
        while(ACKSTAT != 0)        // prijato??
                ;

        PEN = 1;
        while(PEN != 0)
                ;
}

uint8_t i2c_read(uint8_t addr)
{
        /*
         Takto naimplementovana funkce vraci vzdy jen
         prvni namerenou hodnotu a necte kontrolni soucet.
         Pro vycteni teploty i vlhkosti je nutne tuto funkci zavolat
         dvakrat, pokazde s prikazem cteni zadaneho udaje jako prvniho.

         Funkce je uzpusobena na CLOCK Stretching Enabled.
         */
        uint8_t dat;
        SEN = 1;
        while(SEN != 0)                        // start condition
                ;

        SSPBUF = (addr << 1)+1;        // I2C 7bit address, 1 na konci = read
        SSPIF = 0;
        while(SSPIF != 1)                // odeslano??
                ;
        while(ACKSTAT != 0)                // prijato??
                ;

        RCEN = 1;
        while(RCEN != 0)
                ;
        dat = SSPBUF;

        ACKDT = 1;
        ACKEN = 1;
        while(ACKEN != 0)
                ;

        PEN = 1;
        while(PEN != 0)
                ;
        return dat;
}


/* ============================ DISPLAY ============================== */
void lcd_clr (void)                // vymazani displeje
{
        EN = 0;
        RS = 0;
        RW = 0;

        DATA = 0b00000000;
        EN = 1;
        EN = 0;
        DATA = 0b00010000;
        EN = 1;
        EN = 0;
        __delay_ms(10);
}

void lcd_init (void)        // inicializace displeje
{
        EN = 0;
        RW = 0;
        RS = 0;
        // wait for power stabilization 500 ms
        __delay_ms(500);
```

```
// function set
/*DATA = 0b00100000;
EN = 1;
EN = 0;*/
__delay_ms(10);
DATA = 0b00100000;        // dle manualu
EN = 1;
EN = 0;
__delay_ms(10);
DATA = 0b00100000;        // 2x zopakovane = 4-bit mode
EN = 1;
EN = 0;
__delay_ms(10);
/*
        N F FT1 FT0
 *                N = pocet radku ("1"=2, "0"=1)
 *                F = velikost znaku ("1"=5x10, "0"=5x8)
 *                FT znakova tabulka:        "00" = ENGLISH_JAPANESE - default
 *                                                                  "01" = WESTERN_EUROPEAN_1
 *                                                                  "10" = ENGLISH_RUSSIAN
 *                                                                  "11" = WESTERN_EUROPEAN_2
 */
DATA = 0b10000000;
EN = 1;
EN = 0;

// check busy flag
__delay_ms(10);

// display ON/OFF control
DATA = 0b00000000;
EN = 1;
EN = 0;
/*
        1 D C B
 *                D = display ON/OFF ("1"=ON)
 *                C = cursor display ON/OFF ("1"=ON)
 *                B = blinking ON/OFF ("1"=ON)
 */
DATA = 0b10110000;
EN = 1;
EN = 0;

// check busy flag
__delay_ms(10);

// display clear
lcd_clr();

// check busy flag
__delay_ms(10);

// return home
DATA = 0b00000000;
EN = 1;
```

```
        EN = 0;
        DATA = 0b00100000;
        EN = 1;
        EN = 0;

        // check busy flag
        __delay_ms(10);

        // entry mode set
        DATA = 0b00000000;
        EN = 1;
        EN = 0;
        /*
                0 1 I/D S/H
         *                I/D = Increment/decrement bit ("1 = incr")
         *                S = Shift entire display control bit ("0"=disable)
         */
        DATA = 0b01100000;
        EN = 1;
        EN = 0;

        // check busy flag
        __delay_ms(10);

        // initialization end
        // display ON/OFF control
        DATA = 0b00000000;
        EN = 1;
        EN = 0;
        /*
                1 D C B
         *                D = display ON/OFF ("1"=ON)
         *                C = cursor display ON/OFF ("1"=ON)
         *                B = blinking ON/OFF ("1"=ON)
         */
        DATA = 0b11000000;
        EN = 1;
        EN = 0;
        __delay_ms(10);
}

void lcd_send (char znak)        // odeslani znaku na displej
{
        RS = 1;
        DATA = znak;
        EN = 1;
        EN = 0;
        DATA = znak<<4;                 // musim posilat po 4bitovych castech
        EN = 1;
        EN = 0;
        __delay_ms(1);
}

void lcd_gotoxy (uint8_t z, uint8_t r) // nastaveni pozice pro vypis na displeji
{
        RS = 0;
        uint8_t ADDR = 0b10000000;                // zaklad
        ADDR = ADDR+z-1 + (r-1)*0x40;        // prictu radek a sloupec
        DATA = ADDR;
        EN = 1;
        EN = 0;
        DATA = ADDR<<4;
        EN = 1;
        EN = 0;
        __delay_ms(5);
}

void putch(char data)
{
        lcd_send(data);
}

/*================================================================*/
void main(void)
{
        uint8_t ADcode;
        uint8_t addr = 0b1001000;                // 7bit adresa zarizeni
        uint8_t CONTROL = 0b00000001;                // cti teplotu prvni
        double T;
        double VADC;
        double Rx;


        /*================================================================*/
    ADCON1 = 0b11000010;

        // PORT E na digitální

        // E i D na výstup (TRIS)
        TRISE = 0; //vystup = 0
        TRISD = 0;
        /*================================================================*/
        /*        Dalsi blbinky potrebny k tomu, aby to sprvne fungovalo,
                protoze z nejakyho duvodu kdyz jsou tyhle porty zapnuty,
                rusi to komunikaci s displayem...*/
        // nebo tak neco
        TRISBbits.TRISB0 = 0;
        TRISBbits.TRISB1 = 0;
        TRISBbits.TRISB2 = 0;
        TRISBbits.TRISB4 = 0;
        TRISCbits.TRISC1 = 0;

        PORTBbits.RB0 = 1;
        PORTBbits.RB1 = 1;
        PORTBbits.RB2 = 1;
        PORTBbits.RB4 = 1;
        PORTCbits.RC1 = 1;
        /*================================================================*/

        i2c_init();
        lcd_init();

        while(1)
        {
                lcd_gotoxy(1,1);

                i2c_write(addr, CONTROL);
                ADcode = i2c_read(addr);

                VADC = (5.0/256)*ADcode;
                Rx = (VADC*1000)/(5-VADC);
                T = -0.0071*Rx+46.5;

                printf("T=%.1f",T);
                __delay_ms(300);

        }


        return;
}
```



Obr. 1: Diagram vypracování úlohy – Zobrazení naměřené teploty.