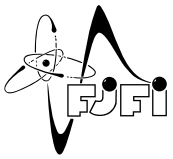
Mikroprocesorové praktikum **FJFI ČVUT v Praze**

Úloha 07 – Paralelní komunikační rozhraní – ovládání znakového LCD displeje



Jméno: **Kateřina Hrnečková** Měřeno: **3.3.2023** Klasifikace:

---

1 Pracovní úkoly

1. Seznamte se se znakovým LCD displejem GDM1602 řízeným kontrolérem S6A0069.
2. Vytvořte následující funkce pro ovládání displeje:
   - lcd_init pro inicializaci displeje
   - lcd_clr pro vymazání displeje
   - lcd_send pro odeslání znaku na displej
   - lcd_gotoxy pro nastavení pozice pro výpis na displeji
3. Napište program v jazyce C využívající LCD displej.
4. Program přeložte a vložte do paměti mikrokontroléru.

2 Vypracování

Řešená úloha: Zobrazování napětí na trimerech.

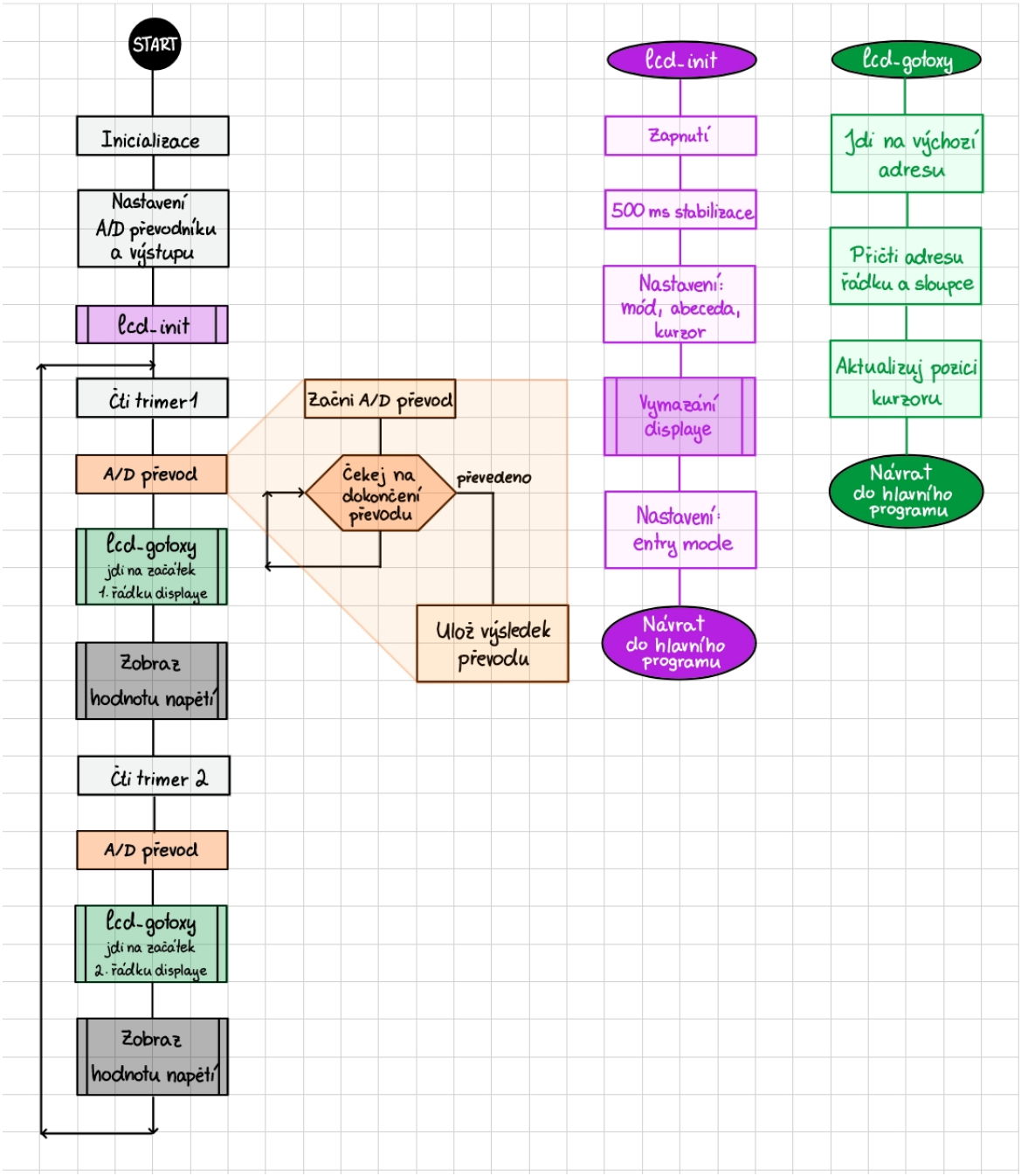
```
#include <xc.h>
#include <stdint.h>
#include <stdio.h>
#include <pic16f877a.h>

#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 progr
#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 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)
{
        /*=====================================================================*/
        TRISAbits.TRISA2 = 1;                // nastaveni A/D prevodniku
    ADCON1 = 0b11000010;
    ADCON0 = 0b10010001;
        /*=====================================================================*/
        // PORT E na digitální

        // E i D na výstup (TRIS)
        TRISE = 0; //vystup = 0
```

```
        TRISD = 0;
        uint16_t vysledekAD = 0;

        lcd_init();

        while(1)
        {
                ADCON0 = 0b10010001;                // ctu TRIM1
                __delay_ms(1);
                ADCON0bits.GO = 1;
                while(ADCON0bits.GO != 0)        // cekam na dokonceni A/D preodu
                        ;

                vysledekAD = (ADRESH << 8) + ADRESL;
                vysledekAD = vysledekAD*5;
                lcd_gotoxy (1,1);                        // kurzor na zacatek

                printf("TR1:%d    ", vysledekAD);


                ADCON0 = 0b10011001;                // ctu TRIM1
                __delay_ms(1);

                ADCON0bits.GO = 1;
                while(ADCON0bits.GO != 0)
                        ;

                vysledekAD = (ADRESH << 8) + ADRESL;
                vysledekAD = vysledekAD*5;
                lcd_gotoxy (1,2);                        // kurzor na 2. radek

                printf("TR2:%d    ", vysledekAD);

                __delay_ms(200);

        }

        return;
}
```



Obr. 1: Diagram vypracování úlohy – Zobrazování napětí na trimerech.