

Základy ovládania LCD displejov kompatibilných s radičom HD44780

Dnes tu máme tretiu a zároveň poslednú časť o ovládaní znakových displejov, v ktorej si ukážeme, ako tieto displeje ovládať v 4 - bitovom móde pomocou mikrokontrolérov.

Čo budeme potrebovať?

Snáď nikoho neprekvapím a dokonca aj poteším, keď napíšem, že presne to isté ako sme potrebovali pri ovládaní displejov pomocou 8 - bitového módu v [predchádzajúcej časti tohto mini seriálu](#).

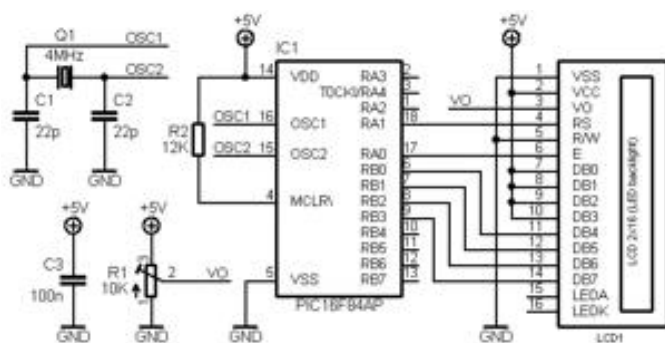
Príprava:

Predpokladám, že po predošlom "hraní sa" s 8 - bitovým módom displeja ste súčiastky nechali zapojené na kontaktnom poli ako ja a nevytrhali ste ich hneď von.

Ak áno, zmena hardvéru z 8 - bitového módu na 4 - bitový bude veľmi jednoduchá. Stačí odpojiť drôtičky pripojené na vývody DB0 až DB3 displeja od mikrokontroléru, pripojiť ich ku kladnému pólu napájacieho napätia a drôtičky pripojené na vývody DB4 až DB7 odpojiť od vrchnej štvorice pinov RB4 až RB7 a pripojiť ku spodnej štvorici pinov portu B, čiže RB0 - RB3.

Ak nie, tak si to všetko zasa môžete postaviť späť podľa nasledujúcej schémy.

Keď sme s prípravou hardvéru hotový, môžeme sa opäť vrhnúť na program, ktorý sa od programu pre 8 - bitový mód, rovnako ako schéma zapojenia nebude veľmi odlišovať.



Program pre mikrokontrolér:

Po hardvérovej stránke sa 8 - bitový a 4 - bitový mód odlišujú len počtom a prepojením dátových vodičov.

Čo z toho vyplýva pre náš program?

Vyplýva z toho veľmi dobrá správa a tou je, že nám stačí prepísať iba dve funkcie **Lcd_write()**, **Lcd_init()** a niektoré symbolické konštanty v hlavičkovom súbore (init_value, function_set). Ostatné funkcie zostávajú v pôvodnej forme ako pri 8 - bitovom móde.

Začneme tým, že sa pozrieme do vývojového diagramu pre 4 - bitový mód. Vidíme, že v hlavičkovom súbore potrebujeme zmeniť hodnotu konštanty init_value na 0x03. Ďalej hodnotu konštanty function_set na 0x28, pretože teraz pracujeme vo 4 - bitovom móde bit DL sa bude rovnáť nule. Poslednou vecou, ktorú v hlavičkovom súbore zmeníme bude, že si zavedieme novú symbolickú konštantu s názvom set_direction a priradíme jej hodnotu 0xF0. Túto konštantu budeme potrebovať pri inicializácii dátového portu. Keďže teraz využívame len jeho 4 - spodné bity (piny), nastavíme ako výstupy iba tie a ostatné ponecháme ako vstupy.

1.

```
#ifndef _LCD_h_
```
2.

```
#define _LCD_h_
```
- 3.

```
4.
    // define LCD interface

5.
    #define LCD_DATA PORTB           // DB0-DB7

6.
    #define LCD_DATA_DIR TRISB

7.

8.
    #define LCD_RS RA1              // RS pin

9.
    #define LCD_RS_DIR TRISA1

10.

11.
    // #define LCD_RW xx             // if R/W not connect to GND

12.
    // #define LCD_RW_DIR xx

13.

14.
    #define LCD_EN RA0              // EN pin

15.
    #define LCD_EN_DIR TRISA0

16.

17.
    // #define LCD_BL xx             // backlight on/off

18.
    // #define LCD_BL_DIR xx

19.

20.
    // define display settings

21.
    #define function_set 0x28        /* 0 0 1 DL N F 0 0 */

22.
    // Function set

23.
    // bit2 set 0 - dot             format 5x8

24.
```

```
//          1 - dot format 5x11

25.          // bit3 set 0 - 1 line          mode

26.          //          1 - 2 line mode

27.          // bit4 set 0 - 4 bit          mode

28.          //          1 - 8 bit mode

29.

30.          //#define cur_disp_shift 0x?? /* 0 0 0 1 S/C R/L 0 0 */

31.          // Cursor or display          shift

32.          // bit2 set 0 - shift          left

33.          //          1 - shift right

34.          // bit3 set 0 - cursor          shift

35.          //          1 - display shift

36.

37.          #define display_control 0x0C /* 0 0 0 0 1 D C B */

38.          // Display          control

39.          // bit0 set 0 - blink          OFF

40.          //          1 - blink ON

41.          // bit1 set 0 - cursor          OFF

42.          //          1 - cursor ON

43.          // bit2 set 0 - display          OFF

44.          //          1 - display ON
```

```
45.
46. #define entry_mode_set 0x06 /* 0 0 0 0 0 1 I/D SH */
47. // Entry mode set
48. // bit0 set 0 - entire shift OFF
49. // 1 - entire shift ON
50. // bit1 set 0 - decrement mode
51. // 1 - increment mode
52.
53. // define other constants
54. #define output 0
55. #define init_value 0x03
56. #define set_direction 0xF0
57.
58. // function prototypes
59. void Lcd_strobe(void);
60. void Lcd_write(unsigned char byte);
61. void Lcd_putch(unsigned char ch);
62. void Lcd_puts(const char *s);
63. void Lcd_clear(void);
64. void Lcd_ret_home(void);
```

```
65. void Lcd_goto_pos(unsigned char row, unsigned char pos);

66. void Lcd_init(void);

67.

68. #endif

69.
```

A teraz ku zmenám v prvej spomínanej funkcii **Lcd_write()**.

Oneskorenú funkciu ponecháme. Pri ďalších úpravách si musíme uvedomiť, že v tejto chvíli už využívame len spodné 4 piny portu a to RB0 až RB3, ktoré teraz zodpovedajú dátovým vodičom DB4 až DB7.

Príkazový alebo dátový bajt musíme teraz poslať na dvakrát a to tak, že prvé pošleme horné 4 bity a potom spodné 4 bity.

Pre budúcnosť je dôležité si uvedomiť, že táto funkcia je závislá na hardvérovej konfigurácii (my sme pripojili vodiče DB4 - DB7 k spodným štyrom pinom portu B, rovnako sme ich ale mohli zapojiť ku vrchným štyrom pinom tohto portu a potom by ale funkcia Lcd_write(), symbolické konštanty init_value a set_direction vyzerali trochu inak, ako to už nechám na vás).

```
1. void Lcd_write(unsigned char byte)

2. {

3.     DelayUs(43);

4.     LCD_DATA = ((byte>>4)&0x0F);

5.     Lcd_strobe();

6.     LCD_DATA = (byte&0x0F);

7.     Lcd_strobe();

8. }
```

S funkciou Lcd_write() sme hotový a teraz si podáme funkciu pre inicializáciu displeja.

Najprv inicializujeme dátový port a radiace piny mikrokontroléra.

Opäť nahliadneme do vývojového diagramu v nám z minulej časti známom datasheete rodiny BC1602A, teraz však do časti

12-5.5.2 4-bit interface. Vidíme z neho, že opäť musíme minimálne 15ms čakať a následne poslať na dátové vodiče 3 krát rovnakú kombináciu log. úrovní (0 0 1 1) s určitými oneskoreniami, ako to bolo pri 8 - bitovom móde, v ktorom sa displej momentálne ešte nachádza.

Ďalšou kombináciou log. úrovní (0 0 1 0) privedených na dátové vodiče dáme radiču displeja pokyn aby sa "prepol" do 4 - bitového módu. Následne na to, pošleme radiču informácie o tom ako chceme nakonfigurovať displej (podobne ako pri 8 bitovom - móde) a tým je inicializácia skončená.

```
1. void Lcd_init(void)
2. {
3.     LCD_DATA_DIR = set_direction;
4.     LCD_RS = LCD_EN = 0;
5.     LCD_RS_DIR = LCD_EN_DIR = output;
6.
7.     DelayMs(15);
8.     LCD_DATA = init_value;
9.     Lcd_strobe();
10.    DelayMs(5);
11.    LCD_DATA = init_value;
12.    Lcd_strobe();
13.    DelayUs(200);
14.    LCD_DATA = init_value;
15.    Lcd_strobe();
16.    LCD_DATA = 0x02;
17.    Lcd_strobe();
```

```
18.    Lcd_write(function_set);

19.    Lcd_write(display_control);

20.    Lcd_clear();

21.    Lcd_write(entry_mode_set);

22. }
```

Keďže ostatné funkcie sa zhodujú s funkciami používanými v 8 - bitovom móde, môžeme si smelo napísať nejaký ten hlavný program a vypísať si niečo na displeje.

Aby sme sa neopakovali tak pre zmenu si na dvojriadkovom displeji vypíšeme text "Zdravi vas mikrozone.eu" a na štvorriadkovom text "Mikrozone.eu najcerstvejsie informacie z oblasti mcu".

Ako už možno sami tušíte, keďže sme menili len funkcie Lcd_write() a Lcd_init(), v hlavnom programe veľa zmien nenastane. Jediné zmeny budú v textoch ktoré chceme vypísať na displeje.

```
1.    #include //

2.    #include "delay.h"

3.    #include "lcd.h"

4.

5.    __CONFIG(UNPROTECT&PWRTEN&WDTDIS&XT);

6.

7.    const unsigned char *text1      = "Zdravi Vas";

8.    const unsigned char *text2      = "mikrozone.eu";

9.    const unsigned char *text3      = "Mikrozone.eu";

10.   const unsigned char *text4      = "najcerstvejsie";

11.   const unsigned char *text5      = "informacie z";
```

```
12.     const unsigned char *text6      = "oblasti mcu.";
13.
14.     void main(void)
15.     {
16.         Lcd_init();
17.
18.         // 2 line display
19.         Lcd_goto_pos(1, 3);
20.         Lcd_puts(text1);
21.         Lcd_goto_pos(2, 2);
22.         Lcd_puts(text2);
23.
24.         // 4 line display
25.         /*
26.         Lcd_goto_pos(1, 2);
27.         Lcd_puts(text3);
28.         Lcd_goto_pos(2, 1);
29.         Lcd_puts(text4);
30.         Lcd_goto_pos(3, 2);
31.         Lcd_puts(text5);
```

```
32.    Lcd_goto_pos(4, 2);

33.    Lcd_puts(text6);

34.    */

35.

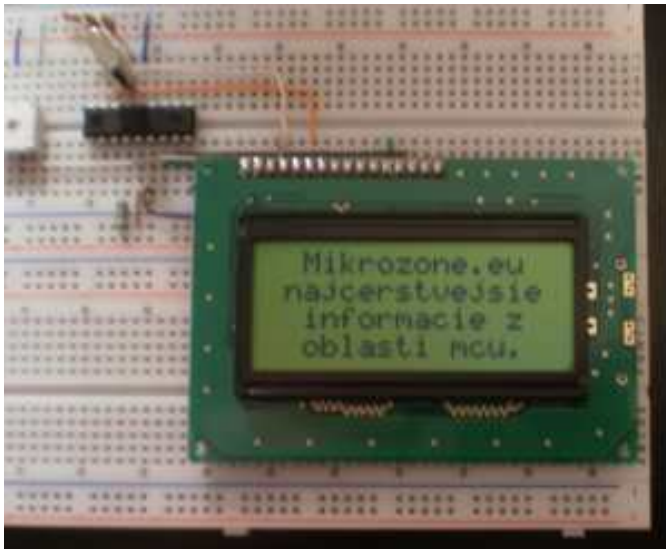
36.    while(1);

37.    }
```

Firmvér mikrokontroléru bol opäť skompilovaný s pomocou HI-TECH C Compiler for PIC10/12/16 MCU's ver. 9.70 pod MPLAB IDE ver. 8.50.

Kompletné zdrojové súbory ako aj .hex súbory vhodné priamo na nahratie do mikrokontroléru môžete nájsť v prílohách. Rovnako ako v predchádzajúcej časti ani teraz si neodpustím ukážky textov zobrazených na dvoj a štvorriadkovom displeji, ktoré nás sprevádzali dvomi tretinami tohto miniseriálu.





Týmto riadkom sa s vami mini seriál o znakových LCD displejoch lúči a ja vám želám, aby vás tie vaše displeje poslúchali minimálne tak ako mňa moje.

Použitá literatúra a zdroje:

- [1] [Datasheet displeja BC1602HYPNEH](#)
- [2] [Datasheet rodiny displejov BC1602A](#)
- [3] [Datasheet displeja MC1604B-SYR časť.1](#)
- [4] [Datasheet displeja MC1604B-SYR časť.2](#)
- [5] [Datasheet mikrokontroléra PIC16F84A](#)
- [6] [MPLAB IDE](#)
- [7] [HI-TECH C for the PIC10/12/16 MCU Family](#)

[<- 2. časť](#)