

| | | |
|-----------------------------------|---|----------------------|
| Maciej Byczko Bartosz Matysiak | Prowadzący: dr inż. Jacek Mazurkiewicz | Numer ćwiczenia 3 |
| Cz 13:15 TN | Temat ćwiczenia: Wyświetlacz LCD | Ocena: |
| Grupa: B | Data wykonania: 25 Marca 2021 | |

1 Zadanie 1

1.1 Polecenie

Przygotować program wiążący guziki przypięte do P3 z wyświetlaczem LCD. W kodzie programu przygotowujemy 4 różne łańcuchy znaków przypisane do każdego z 4 guzików. Wyświetlanie danego tekstu następuje po naciśnięciu związanego z nim guzika. Program działa w pętli, a jednoczesne naciśnięcie dwóch skrajnych guzików powoduje wyjście z programu.

1.2 Rozwiązanie

```

1  ljmp start
2
3  LCDstatus equ 0FF2EH      ; adres do odczytu gotowosci LCD
4  LCDcontrol equ 0FF2CH     ; adres do podania bajtu sterujacego LCD
5  LCDdataWR equ 0FF3DH     ; adres do podania kodu ASCII na LCD
6
7  // bajty sterujace LCD, inne dostepne w opisie LCD na stronie WWW
8  #define HOME 0x80 // put cursor to second line
9  #define INITDISP 0x38 // LCD init (8-bit mode)
10 #define HOM2 0xc0 // put cursor to second line
11 #define LCDON 0x0e // LCD on, cursor off, blinking off
12 #define CLEAR 0x01 // LCD display clear
13
14 org 0100h
15
16 // deklaracja tekstów
17 text_button: db "Wcisnieto \\/\\" , 00
18 text_button1: db "Przycisk -> 1 " , 00
19 text_button2: db "Przycisk -> 2 " , 00
20 text_button3: db "Przycisk -> 3 " , 00
21 text_button4: db "Przycisk -> 4 " , 00
22 text_exit: db "1+4 aby wyjsc " , 00
23 text_end: db "Do widzenia ;) " , 00
24 text_test: db "wypisz 16 znakow" , 00; ciag zawierajacy
    dokładnie 16 znakow
25 text_buttons: db "2124812489" , 00
26
27 // macro do wprowadzenia bajtu sterujacego na LCD
28 LCDcntrlWR MACRO x      ; x - parametr wywolania macra - bajt
    sterujacy

```

```

29     LOCAL loop      ; LOCAL oznacza ze etykieta loop moze sie
                        powtorzyc w programie
30 loop: MOV  DPTR,#LCDstatus ; DPTR zaladowany adresem statusu
31     MOVX A,@DPTR      ; pobranie bajtu z biezacym statusem LCD
32     JB  ACC.7,loop    ; testowanie najstarszego bitu akumulatora
33                        ; - wskazuje gotowosc LCD
34     MOV  DPTR,#LCDcontrol ; DPTR zaladowany adresem do podania bajtu
                        sterujacego
35     MOV  A, x         ; do akumulatora trafia argument wywolania macra,
                        bajt sterujacy
36     MOVX @DPTR,A      ; bajt sterujacy podany do LCD - zadana akcja
                        widoczna na LCD
37     ENDM
38
39 // macro do wypisania znaku ASCII na LCD, znak ASCII przed
                        wywolaniem macra ma byc w A
40 LCDcharWR MACRO
41     LOCAL tutu      ; LOCAL oznacza ze etykieta tutu moze sie
                        powtorzyc w programie
42     PUSH ACC        ; odlozenie biezacej zawartosci akumulatora na
                        stos
43 tutu: MOV  DPTR,#LCDstatus ; DPTR zaladowany adresem statusu
44     MOVX A,@DPTR      ; pobranie bajtu z biezacym statusem LCD
45     JB  ACC.7,tutu    ; testowanie najstarszego bitu akumulatora
46                        ; - wskazuje gotowosc LCD
47     mov  83h, 06h     ; DPH - 83h, r6 - 06h czyli MOV DPH, R6
48     mov  82h, 07h     ; DPL - 82h, r7 - 07h czyli MOV DPL, R7
49     ;MOV  DPTR,#LCDdataWR ; DPTR zaladowany adresem do podania
                        bajtu sterujacego
50     POP  ACC          ; w akumulatorze ponownie kod ASCII znaku na LCD
51     MOVX @DPTR,A      ; kod ASCII podany do LCD - znak widoczny na
                        LCD
52     ENDM
53
54 // macro do inicjalizacji wyswietlacza - bez parametrów
55 init_LCD MACRO
56     LCDcntrlWR #INITDISP ; wywołanie macra LCDcntrlWR -
                        inicjalizacja LCD
57     LCDcntrlWR #CLEAR    ; wywołanie macra LCDcntrlWR - czyszczenie
                        LCD
58     LCDcntrlWR #LCDON    ; wywołanie macra LCDcntrlWR - konfiguracja
                        kursora
59     ENDM
60
61 // wypisz zadany tekst
62 write_str MACRO x
63     mov  dptr, x
64     acall putstrLCD
65     ENDM
66
67 // funkcja wypisania znaku

```

```
68 putcharLCD: LCDcharWR
69     ret
70
71 //funkcja wypisania lancucha znaków
72 putstrLCD:  mov r7, #30h ; DPL ustawiony tak by byl w DPTR adres
              FF30H
73 nextchar:  clr a
74           movc a, @a+dptr
75           jz koniec
76           push dph
77           push dpl
78           acall putcharLCD
79           pop dpl
80           pop dph
81           inc r7 ; dzieki temu mozliwa inkrementacja DPTR
82           inc dptr
83           sjmp nextchar
84 koniec:  ret
85
86 // funkcja opóznienia
87 delay:  mov r0, #15H
88 one:    mov r1, #0FFH
89 dwa:    mov r2, #0FFH
90 trzy:   djnz r2, trzy
91         djnz r1, dwa
92         djnz r0, one
93         ret
94
95 // program główny
96 start:  init_LCD
97
98         mov r6, #0FFH ; adres LCDdataWR equ 0FF3DH jest w parze R6–R7
99         mov r7, #30H
100
101         LCDcntrlWR #CLEAR
102         write_str #text_test
103         LCDcntrlWR #HOM2
104         write_str #text_exit
105         jmp select
106         // przez krótki zasięg jump musimy podzielic sekcje
           wypisywania na 2 czesci
107
108         // czesc pierwsza wypisywania
109         push_button1;; gdy zostal wcisniety przycisk 1
110         LCDcntrlWR #CLEAR; wyczysc wyswietlacz
111         write_str #text_button; wypisz tekst
112         LCDcntrlWR #HOM2; przesun kursor na kolejna linie
113         write_str #text_button1; wypisz tekst
114         jmp select; skocz to "panelu" wyboru
115
116         push_button2:
```

```
117     LCDcntrlWR #CLEAR
118     write_str #text_button
119     LCDcntrlWR #HOM2
120     write_str #text_button2
121     jmp select
122
123     select:
124
125     clr c ; wyczysc zmienna c
126     orl c, p3.2; jezeli przycisk 1 jest wcisniety, daj zmiennej c
        = 1
127     anl c, p3.5; jezeli przycisk 5 jest wcisniety i p3.2 byl
        wcisniety, c = 1
128     jc loop_exit; jezeli c = 1, przejdz do zakonczenia programu
129
130     mov a, p3; przenies wcisnieta wartosc do akumulatora i skocz
        do wybranej opcji
131     jnb acc.2, push_button1; wcisnieto przycisk 1
132     jnb acc.3, push_button2; wcisnieto przycisk 2
133     jnb acc.4, push_button3; wcisnieto przycisk 3
134     jnb acc.5, push_button4; wcisnieto przycisk 4
135     jmp select; gdy zaden przycisk nie zostal wybrany skocz na
        poczatek petli
136
137     // czesc druga wypisywania
138     push_button3:
139         LCDcntrlWR #CLEAR
140         write_str #text_button
141         LCDcntrlWR #HOM2
142         write_str #text_button3
143         jmp select
144
145     push_button4:
146         LCDcntrlWR #CLEAR
147         write_str #text_button
148         LCDcntrlWR #HOM2
149         write_str #text_button4
150         jmp select
151
152     loop_exit:
153         LCDcntrlWR #CLEAR
154         write_str #text_end; wypisz pozegnanie
155     nop
156     nop
157     nop
158     jmp $
159     end start
```

2 Zadanie 2

2.1 Polecenie

Przygotować program wyświetlający na LCD przygotowany w programie łańcuch znaków o długości znacząco przekraczającej 16 znaków. Tekst jest wyświetlany tak by w pierwszej linii LCD pokazanych zostało 16 znaków, po czym następuje automatyczne przejście do drugiej linii, gdzie wyświetlany kolejne 16 znaków. Następnie pojawia się pauza, a po jej zakończeniu kasujemy bieżącą zawartość wyświetlacza i znów w pierwszej linii wyświetlamy następne 16 znaków tekstu, w drugiej - kolejne 16 znaków tekstu, pauza, itd. Akcja dobiega końca gdy zostanie takimi etapami wyświetlony cały przygotowany łańcuch znaków.

2.2 Rozwiązanie na mikrokontroler fizyczny

```

1  ljmp start
2
3  LCDstatus equ 0FF2EH          ; adres do odczytu gotowosci LCD
4  LCDcontrol equ 0FF2CH         ; adres do podania bajtu sterujacego LCD
5  LCDdataWR equ 0FF2DH         ; adres do podania kodu ASCII na LCD
6
7  // bajty sterujace LCD, inne dostepne w opisie LCD na stronie WWW
8  #define HOME      0x80        // put cursor to first line
9  #define INITDISP  0x38        // LCD init (8-bit mode)
10 #define HOM2      0xc0        // put cursor to second line
11 #define LCDON      0x0e        // LCD nn, cursor off, blinking off
12 #define CLEAR      0x01        // LCD display clear
13
14 org 0100H
15
16 // deklaracje tekstów
17 text: db "Orka oceaniczna – gatunek ssaka z rodziny delfinowatych
        (Delphinidae). Największy przedstawiciel delfinowatych, jedyny
        przedstawiciel rodzaju Orcinus.",00
18
19 // macro do wprowadzenia bajtu sterujacego na LCD
20 LCDcntrlWR MACRO x            ; x – parametr wywołania macra – bajt
    sterujacy
21         LOCAL loop            ; LOCAL oznacza ze etykieta loop moze
        sie powtórzyć w programie
22 loop: MOV DPTR, #LCDstatus     ; DPTR załadowany adresem statusu
23         MOVX A, @DPTR          ; pobranie bajtu z bieżącym statusem LCD
24         JB ACC.7, loop         ; testowanie najstarszego bitu
        akumulatora
25         ; – wskazuje gotowosc LCD
26         MOV DPTR, #LCDcontrol ; DPTR załadowany adresem do podania
        bajtu sterujacego
27         MOV A, x               ; do akumulatora trafia argument
        wywołania –macrabajt sterujacy
28         MOVX @DPTR, A         ; bajt sterujacy podany do LCD – zadana
        akcja widoczna na LCD
29 ENDM

```

```

30
31 // macro do wypisania znaku ASCII na LCD, znak ASCII przed
    wywołaniem macra ma być w A
32 LCDcharWR MACRO
33     LOCAL tutu                ; LOCAL oznacza że etykieta tutu może
        się powtórzyć w programie
34     PUSH ACC                  ; odłożenie bieżącej zawartości
        akumulatora na stos
35 tutu: MOV DPTR, #LCDstatus    ; DPTR załadowany adresem statusu
36     MOVX A, @DPTR             ; pobranie bajtu z bieżącym statusem LCD
37     JB ACC.7, tutu            ; testowanie najstarszego bitu
        akumulatora
38                                ; – wskazuje gotowość LCD
39     MOV DPTR, #LCDdataWR      ; DPTR załadowany adresem do podania
        bajtu sterującego
40     POP ACC                   ; w akumulatorze ponownie kod ASCII
        znaku na LCD
41     MOVX @DPTR, A             ; kod ASCII podany do LCD – znak widoczny
        na LCD
42     ENDM
43
44 // macro do inicjalizacji wyświetlacza – bez parametrów
45 init_LCD MACRO
46     LCDcntrlWR #INITDISP      ; wywołanie macra LCDcntrlWR –
        inicjalizacja LCD
47     LCDcntrlWR #CLEAR         ; wywołanie macra LCDcntrlWR –
        czyszczenie LCD
48     LCDcntrlWR #LCDON         ; wywołanie macra LCDcntrlWR –
        konfiguracja kursora
49     ENDM
50
51 // funkcja opóźnienia
52 delay: mov r0, #15H
53 one:   mov r1, #0FFH
54 dwa:   mov r2, #0FFH
55     trzy: djnz r2, trzy
56     djnz r1, dwa
57     djnz r0, one
58     ret
59
60 // funkcja wypisania znaku
61 putcharLCD: LCDcharWR
62     ret
63
64 //funkcja wypisania łańcucha znaków
65 putstrLCDin2Lines:
66     mov r7, #10H              ;licznik pomocniczy
67     push dph                  ;makro LCDcntrlWR modyfikuje wartość dptr,
        dlatego trzeba ją odłożyć na stos...
68     push dpl
69     LCDcntrlWR #HOME          ;ustaw kursor na początku pierwszej linii

```

```
70     pop dpl          ;... i nastepnie przywrocic
71     pop dph
72 nextcharinFirstLine:    ;petla wypisujaca znaki w pierwszej linii
73     clr a
74     movc a, @a+dptra
75     jz koniec          ;Dopoki sa jakies znaki do wypisania...
76     push dph
77     push dpl
78     acall putcharLCD   ;... wypisuj je...
79     pop dpl
80     pop dph
81     inc dptra          ;... i skacz na poczatek petli
82     djnz r7, nextcharinFirstLine
83                     ; Jesli pierwsza linia wyswietlacz sie skonczyla...
84     mov r7, #10H       ;...ustaw znów licznik pomocniczy...
85     push dph
86     push dpl
87     LCDcntlWR #HOM2    ;... i przejdź do drugiej linii...
88     pop dpl
89     pop dph
90 nextcharinSecondLine:  ;...by w analogiczny sposób wypisać znaki
    wlasnie tam
91     clr a
92     movc a, @a+dptra
93     jz koniec
94     push dph
95     push dpl
96     acall putcharLCD
97     pop dpl
98     pop dph
99     inc dptra
100    djnz r7, nextcharinSecondLine
101                    ;po wyjściu z drugiej petli odczekaj pewien rozsądny
    czas
102    acall delay
103    push dph
104    push dpl
105    LCDcntlWR #CLEAR;wyczyść ekran przed przystąpieniem do
    powtornego cyklu wypisywania
106    pop dpl
107    pop dph
108    sjmp putstrLCDin2Lines ;skocz na początek dużej petli
109 koniec: ret
110
111 // program główny
112 start:  init_LCD
113
114     mov dptra, #text
115     acall putstrLCDin2Lines
116     acall delay
117
```

```

118  nop
119  nop
120  nop
121  jmp $
122  end start

```

2.3 Alternatywne rozwiązanie do symulatora

```

1  ljmp start
2
3  LCDstatus equ 0FF2EH      ; adres do odczytu gotowosci LCD
4  LCDcontrol equ 0FF2CH     ; adres do podania bajtu sterujacego LCD
5  LCDdataWR equ 0FF3DH     ; adres do podania kodu ASCII na LCD
6
7  // bajty sterujace LCD, inne dostepne w opisie LCD na stronie WWW
8  #define HOME 0x80 // put cursor to second line
9  #define INITDISP 0x38 // LCD init (8-bit mode)
10 #define HOM2 0xc0 // put cursor to second line
11 #define LCDON 0x0e // LCD nn, cursor off, blinking off
12 #define CLEAR 0x01 // LCD display clear
13
14 org 0100h
15
16 // deklaracja tekstu
17 text_orka: db "Orka oceaniczna – gatunek ssaka z rodziny
18             delfinowatych (Delphinidae). Najwiekszy przedstawiciel
19             delfinowatych, jedyny przedstawiciel rodzaju Orcinus.",00
20
21 // macro do wprowadzenia bajtu sterujacego na LCD
22 LCDcntrlWRR MACRO x      ; x – parametr wywolania macra – bajt
23             sterujacy
24             LOCAL loop    ; LOCAL oznacza ze etykieta loop moze sie
25                             powtorzyc w programie
26 loop: MOV DPTR,#LCDstatus ; DPTR zaladowany adresem statusu
27       MOVX A,@DPTR        ; pobranie bajtu z biezacym statusem LCD
28       JB ACC.7,loop        ; testowanie najstarszego bitu akumulatora
29                             ; – wskazuje gotowosc LCD
30       MOV DPTR,#LCDcontrol ; DPTR zaladowany adresem do podania bajtu
31                             sterujacego
32       MOV A,x              ; do akumulatora trafia argument wywolania macra,
33                             bajt sterujacy
34       MOVX @DPTR,A         ; bajt sterujacy podany do LCD – zadana akcja
35                             widoczna na LCD
36       ENDM
37
38 // macro do wypisania znaku ASCII na LCD, znak ASCII przed
39 // wywolaniem macra ma byc w A
40 LCDcharWR MACRO
41     LOCAL tutu            ; LOCAL oznacza ze etykieta tutu moze sie
42                             powtorzyc w programie
43     PUSH ACC               ; odlozenie biezacej zawartosci akumulatora na
44                             stos

```



```

35 tutu: MOV DPTR, #LCDstatus ; DPTR zaladowany adresem statusu
36 MOVX A, @DPTR ; pobranie bajtu z biezacym statusem LCD
37 JB ACC.7, tutu ; testowanie najstarszego bitu akumulatora
38 ; - wskazuje gotowosc LCD
39 mov 83h, 06h ; DPH - 83h, r6 - 06h czyli MOV DPH, R6
40 mov 82h, 07h ; DPL - 82h, r7 - 07h czyli MOV DPL, R7
41 ;MOV DPTR, #LCDdataWR ; DPTR zaladowany adresem do podania
bajtu sterujacego
42 POP ACC ; w akumulatorze ponownie kod ASCII znaku na LCD
43 MOVX @DPTR, A ; kod ASCII podany do LCD - znak widoczny na
LCD
44 ENDM
45
46 // macro do inicjalizacji wyswietlacza - bez parametrów
47 init_LCD MACRO
48 LCDcntrlWR #INITDISP ; wywołanie macra LCDcntrlWR -
inicjalizacja LCD
49 LCDcntrlWR #CLEAR ; wywołanie macra LCDcntrlWR - czyszczenie
LCD
50 LCDcntrlWR #LCDON ; wywołanie macra LCDcntrlWR - konfiguracja
kursora
51 ENDM
52
53 // wypisz zadany tekst
54 write_str MACRO x ; x - adres poczatku wypisywanego tekstu
55 mov dptr, x
56 acall putstrLCD
57 ENDM
58
59 // funkcja wypisania znaku
60 putcharLCD: LCDcharWR
61 ret
62
63 //funkcja wypisania lancucha znaków
64 putstrLCD:
65 mov r5, #10h ; licznik pomocniczy
66 mov r7, #30h ; DPL ustawiony tak by byl w DPRT adres FF30H
67 nextchar: clr a
68 movc a, @a+dptr
69 jz koniec
70 push dph ; odlozenie na stos, gdyz putcharLCD modyfikuje
dptr
71 push dpl
72 acall putcharLCD
73 pop dpl
74 pop dph
75 inc r7 ; dzieki temu mozliwa inkrementacja DPTR (jako
wskaznika zrodlowego)
76 inc dptr ; inkrementacja dptr (jako wskaznika docelowego)
77 djnz r5, nextchar
78 secondLine:

```

```
79      mov r5, #10h ; licznik pomocniczy
80      mov r7, #66h ; DPL ustawiony tak by byl w DPRT adres FF66H
81 nextcharsecondLine: clr a
82      movc a, @a+dptra
83      jz koniec
84      push dph
85      push dpl
86      acall putcharLCD
87      pop dpl
88      pop dph
89      inc r7
90      inc dptra
91      djnz r5, nextcharsecondline
92      acall delay ;wywołanie opóźnienia
93      acall clearMEM ;wywołanie czyszczenia pamieci
94      sjmp putstrLCD
95
96      koniec: ret
97
98 // funkcja opóźnienia
99      delay: mov r0, #15H
100     one:  mov r1, #0FFH
101     dwa:  mov r2, #0FFH
102     trzy: djnz r2, trzy
103           djnz r1, dwa
104           djnz r0, one
105           ret
106
107     clearMEM:
108         push dph
109         push dpl
110         mov r5, #10h
111         mov r7, #30h
112         clearNextChar:
113             mov dph, r6
114             mov dpl, r7
115             clr a
116             movx @dptra, a
117             inc r7
118             djnz r5, clearNextChar
119
120         mov r5, #10h
121         mov r7, #66h
122         clearNextCharSecondLine:
123             mov dph, r6
124             mov dpl, r7
125             clr a
126             movx @dptra, a
127             inc r7
128             djnz r5, clearNextCharSecondLine
129
```

```
130     pop dpl
131     pop dph
132     ret
133
134 // program główny
135 start: init_LCD
136
137     mov r6, #0FFH ; adres LCDdataWR equ 0FF3DH jest w parze R6–R7
138     mov r7, #30H
139
140     write_str #text_orka ; napis wypisze sie w lokacja pamieci: X
141                       :0x00FF30 – 0x00FF3F (pierwsza linia) oraz X:0x00FF66 – 0
142                       x00FF75 (druga linia)
143
144     nop
145     nop
146     jmp $
147     end start
```