ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ – 7ο ΕΞΑΜΗΝΟ

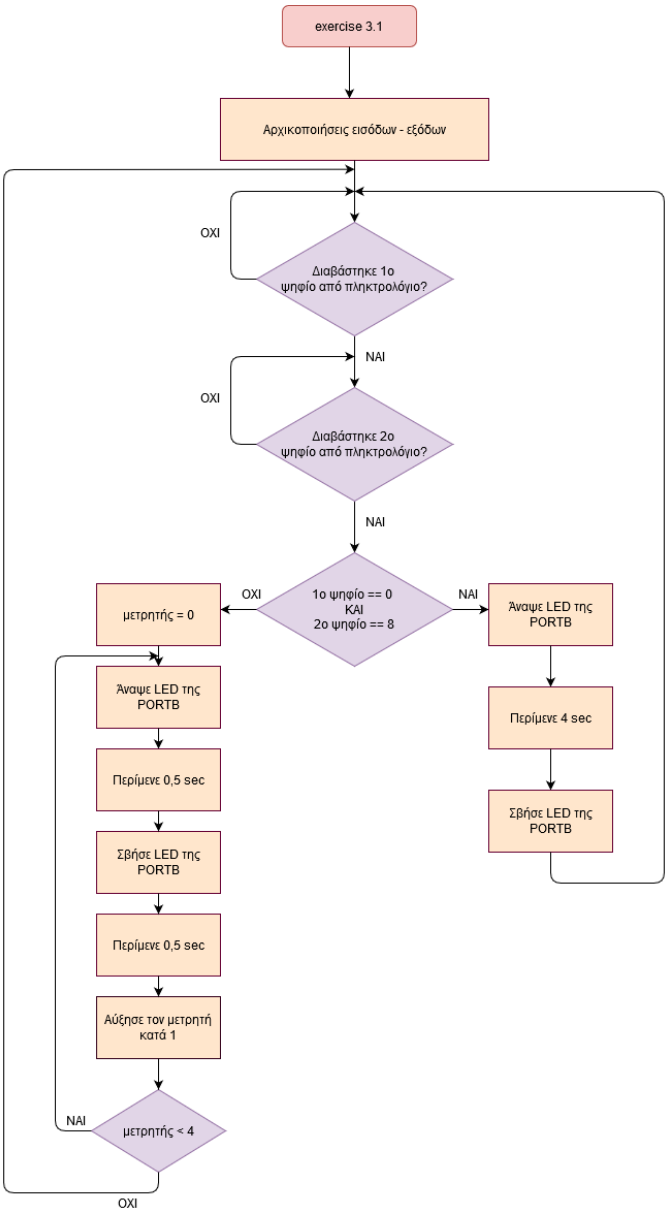
Δημητρίου Αγγελική – ΑΜ: 03117106

Τζομάκα Αφροδίτη – ΑΜ: 03117107

Ομάδα Β8

3η Εργαστηριακή Άσκηση

ΖΗΤΗΜΑ 2.1



Αρχικά γράφτηκαν σε c οι δοθείσες ρουτίνες που αφορούν στον διάβασμα από το πληκτρολόγιο.

Η λογική υλοποίησης της άσκησης φαίνεται στο διπλανό διάγραμμα ροής. Ξεκινάμε αρχικοποιώντας το PORTB ως έξοδο για το άναμμα των LEDs και το PORTC ως έξοδο (PC4-PC7) και είσοδο (PC0-PC3) όπως ορίζει ο αλγόριθμος διαβάσματος από το πληκτρολόγιο.

Στη συνέχεια αν πατήθηκε κάτι από το πληκτρολόγιο το διαβάζουμε παίρνοντας έτσι τους 2 πρώτους αριθμούς. Ελέγχουμε αν είναι 0 και 8 αντίστοιχα και αν ναι ανάβουμε το PORTB για 4 sec σταθερά, ενώ σε αντίθετη περίπτωση το αναβοσβήνουμε για επίσης 4 sec.

Το πρόγραμμα είναι συνεχούς λειτουργίας.

```c
#define F_CPU   8000000UL       //frequency of easyAVR

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

int _tmp_ = 0x0000;

int scan_row_sim(char row_no)
{
       PORTC = (0x08<<row_no);   //pick correct row in PC4-PC7
       _delay_us(500);           //wait 500 usec for remote access

       asm volatile("nop");      //compiler shouldn't move these nops they're crucial
       asm volatile("nop");
       return (PINC&0x0F);       //4 switches in LSBs
}

int scan_keypad_sim()
{
       int keys_pressed = scan_row_sim(4) | (scan_row_sim(3)<<4) |
(scan_row_sim(2)<<8) | (scan_row_sim(1)<<12); //for rows 1(PC4) to 4(PC7) scan row
                                             //put consecutive results in one var
       PORTC=0x00;              //for remote access
       return keys_pressed;
}

int scan_keypad_rising_edge_sim()
{
       int scan1 = scan_keypad_sim();
       _delay_ms(15);
       int scan2 = scan_keypad_sim();
       int result = scan1 & scan2;             //scan twice with 10-20 ms in between
                                               //to eliminate debounce flaws
       int final_result = result & ~(_tmp_);   //get different switches pressed
                                               //comparing to old state
       _tmp_=result;                           //update state of switches
       return final_result;
}

char keypad_to_ascii(int pressed_key)
{
       if((pressed_key&0x0001)==0x0001) return '*';
       if((pressed_key&0x0002)==0x0002) return '0';
       if((pressed_key&0x0004)==0x0004) return '#';
       if((pressed_key&0x0008)==0x0008) return 'D';
       if((pressed_key&0x0010)==0x0010) return '7';
       if((pressed_key&0x0020)==0x0020) return '8';
       if((pressed_key&0x0040)==0x0040) return '9';
       if((pressed_key&0x0080)==0x0080) return 'C';
       if((pressed_key&0x0100)==0x0100) return '4';
       if((pressed_key&0x0200)==0x0200) return '5';
       if((pressed_key&0x0400)==0x0400) return '6';
       if((pressed_key&0x0800)==0x0800) return 'B';
       if((pressed_key&0x1000)==0x1000) return '1';
       if((pressed_key&0x2000)==0x2000) return '2';
       if((pressed_key&0x4000)==0x4000) return '3';
       if((pressed_key&0x8000)==0x8000) return 'A';
       return 0x00;
}

int main(void)
{
```

```
        DDRC=0xF0;  //PC4-PC7 -row detection are outputs
                    //PC3-PC0 inputs -column detection (for keypad)
        DDRB=0xFF;  //PORTB is output
        char number1,number2;

        while (1)
        {
                _tmp_=0x0000; //reset tmp before reading the 2 nums
                //loop until sth is read
                while((number2=keypad_to_ascii(scan_keypad_rising_edge_sim())) == 0x00);
                while((number1=keypad_to_ascii(scan_keypad_rising_edge_sim())) == 0x00);
                if(number1 == '8' && number2 == '0')
                {
                        PORTB=0xFF;
                        _delay_ms(4000);
                        scan_keypad_rising_edge_sim(); //for remote access
                        PORTB=0x00;
                }
                else
                {
                        int cnt=0;
                        while(cnt<4)  //4*1secs
                        {
                                PORTB=0xFF;
                                _delay_ms(500);
                                PORTB=0x00;
                                _delay_ms(500);
                                scan_keypad_rising_edge_sim(); //for remote access
                                cnt++;
                        }
                }

        }
}
```

ΖΗΤΗΜΑ 2.2

Ξεκινάμε αρχικοποιώντας όπως και πριν προσθέτοντας αυτήν την φορά και την αρχικοποίηση της στοίβας (πριν από οποιαδήποτε κλήση ρουτίνας) και του PORTD ως έξοδο για την ορθή λειτουργία της οθόνης LCD. Η λειτουργία του προγράμματος είναι ίδια με πριν με την διαφορά της προσθήκης εντολών ελέγχου και γραψίματος στην οθόνη. Πιο συγκεκριμένα, καλούμε πριν το κυρίως πρόγραμμα την συνάρτηση που ρυθμίζει την οθόνη στις αρχικές της συνθήκες. Επίσης σε κάθε μία από τις περιπτώσεις εξόδου αρχικά εμφανίζουμε με τις κατάλληλες κλήσεις ρουτινών το αντίστοιχο μήνυμα και μόλις παρέλθει το ζητούμενο χρονικό διάστημα «καθαρίζουμε» την οθόνη για να είναι έτοιμη για την επόμενη εγγραφή.

```
.DSEG
 _tmp_: .byte 2

.CSEG
.include "m16def.inc"
.org  0x0
  rjmp   reset

reset:
  ldi r24,low(RAMEND)           ;Initialize stack pointer
  out SPL,r24
  ldi r24,high(RAMEND)
  out SPH,r24

  ser r24
  out DDRD, r24                 ;Set PORTD as output for LCD
  out DDRB, r24                 ;Set PORTB as output for LEDS
  clr r24
  ldi r24,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
  out DDRC,r24                  ;Set the 4 MSBs as output for keyboard
  clr r24
  rcall lcd_init_sim


main:
  rcall scan_keypad_rising_edge_sim     ;Read the first num
  rcall keypad_to_ascii_sim
  cpi   r24,0x00                        ;If 0 is returned then nothing was given
  breq  main                            ;Check again
  mov   r16,r24                         ;Store the first number at r16
read_second:
  rcall scan_keypad_rising_edge_sim     ;Read the second number
  rcall keypad_to_ascii_sim             ;Same logic as the first
  cpi   r24,0x00
  breq  read_second
  mov   r17,r24             ;Store the second number at r17

  cpi   r16,0x30            ;Check the 1st number (hex code for ascii '0' is 30)
  breq  check1
wrong:
  rcall alarm_on
  jmp   end
check1:
  cpi   r17,0x38            ;Check the 2nd number
  breq  success
  jmp   wrong
success:
  rcall welcome
end:
  jmp main


alarm_on:
;Message displayed on lcd
    ldi r24,'A'
       rcall lcd_data_sim
       ldi r24,'L'
       rcall lcd_data_sim
       ldi r24,'A'
       rcall lcd_data_sim
       ldi r24,'R'
       rcall lcd_data_sim
```

```
       ldi r24,'M'
       rcall lcd_data_sim
       ldi r24,' '
       rcall lcd_data_sim
       ldi r24,'O'
       rcall lcd_data_sim
       ldi r24,'N'
       rcall lcd_data_sim
;Blink session
    ldi r19,4                          ;Initialize counter for blinking at r19
rep:
    ser   r18
    out   PORTB,r18
    ldi   r24,low(500)
    ldi   r25,high(500)
    rcall wait_msec                    ;Leds ON for 0.5msecs
;Needed for remote access
    rcall scan_keypad_rising_edge_sim
    clr   r18
    out   PORTB,r18
    ldi   r24,low(500)
    ldi   r25,high(500)
    rcall wait_msec                    ;Leds OFF for 0.5msecs
;Needed for remote access
    rcall scan_keypad_rising_edge_sim
    dec   r19                          ;Cnt--
    brne  rep

    clr r25
    clr r18
    out PORTB,r18                      ;Turn leds and lcd OFF and return
    ldi r24, 0x01
    rcall lcd_command_sim
    ldi r24, low(1530)
    ldi r25, high(1530)
    rcall wait_usec
    ret

welcome:
;Message displayed on lcd
       ldi r24,'W'
       rcall lcd_data_sim
       ldi r24,'E'
       rcall lcd_data_sim
       ldi r24,'L'
       rcall lcd_data_sim
       ldi r24,'C'
       rcall lcd_data_sim
       ldi r24,'O'
       rcall lcd_data_sim
       ldi r24,'M'
       rcall lcd_data_sim
       ldi r24,'E'
       rcall lcd_data_sim
       ldi r24,' '
       rcall lcd_data_sim
       ldi r24,'0'
       rcall lcd_data_sim
       ldi r24,'8'
       rcall lcd_data_sim
;Blink session
    ser r18
```

```
    out PORTB,r18
    ldi r24,low(4000)
    ldi r25,high(4000)
    rcall wait_msec                         ;Leds ON for 4secs
    clr r24
    clr r25
    clr r18
    out PORTB,r18                           ;Turn leds and lcd OFF and return
       ldi r24, 0x01
       rcall lcd_command_sim
       ldi r24, low(1530)
       ldi r25, high(1530)
       rcall wait_usec
    ret

scan_keypad_rising_edge_sim:
    push r22
       push r23
       push r26
       push r27
       rcall scan_keypad_sim
       push r24
       push r25
       ldi r24, 15
       ldi r25, 0
       rcall wait_msec
       rcall scan_keypad_sim
       pop r23
       pop r22
       and r24 ,r22
       and r25 ,r23
       ldi r26 ,low(_tmp_)
       ldi r27 ,high(_tmp_)
       ld r23 ,X+
       ld r22 ,X
       st X ,r24
       st -X ,r25
       com r23
       com r22
       and r24 ,r22
       and r25 ,r23

       pop r27
       pop r26
       pop r23
       pop r22
       ret


scan_row_sim:
       out PORTC , r25
       ;remote access cseg
       push r24
       push r25
       ldi r24, low(500)
       ldi r25, high(500)
       rcall wait_usec
       pop r25
       pop r24
       ;end remote access cseg
       nop
       nop
```

```
        in r24 , PINC
        andi r24 ,0x0f
        ret



scan_keypad_sim:
        push r26
        push r27
        ldi r25 , 0x10
        rcall scan_row_sim
        swap r24
        mov r27, r24
        ldi r25 ,0x20
        rcall scan_row_sim
        add r27, r24
        ldi r25 , 0x40
        rcall scan_row_sim
        swap r24
        mov r26, r24
        ldi r25 ,0x80
        rcall scan_row_sim
        add r26, r24
        movw r24, r26
        clr r26
        out    PORTC,r26
        pop r27
        pop r26
        ret



keypad_to_ascii_sim:
        push r26
        push r27
        movw r26 ,r24
        ldi r24 ,'*'
        sbrc r26 ,0
        rjmp return_ascii
        ldi r24 ,'0'
        sbrc r26 ,1
        rjmp return_ascii
        ldi r24 ,'#'
        sbrc r26 ,2
        rjmp return_ascii
        ldi r24 ,'D'
        sbrc r26 ,3
        rjmp return_ascii
        ldi r24 ,'7'
        sbrc r26 ,4
        rjmp return_ascii
        ldi r24 ,'8'
        sbrc r26 ,5
        rjmp return_ascii
        ldi r24 ,'9'
        sbrc r26 ,6
        rjmp return_ascii ;
        ldi r24 ,'C'
        sbrc r26 ,7
        rjmp return_ascii
        ldi r24 ,'4'
        sbrc r27 ,0
```

```
        rjmp return_ascii
        ldi r24 ,'5'
        sbrc r27 ,1
        rjmp return_ascii
        ldi r24 ,'6'
        sbrc r27 ,2
        rjmp return_ascii
        ldi r24 ,'B'
        sbrc r27 ,3
        rjmp return_ascii
        ldi r24 ,'1'
        sbrc r27 ,4
        rjmp return_ascii ;
        ldi r24 ,'2'
        sbrc r27 ,5
        rjmp return_ascii
        ldi r24 ,'3'
        sbrc r27 ,6
        rjmp return_ascii
        ldi r24 ,'A'
        sbrc r27 ,7
        rjmp return_ascii
        clr r24
        rjmp return_ascii
return_ascii:
        pop r27
        pop r26 ret

write_2_nibbles:
        ;remote access cseg
        push r24
        push r25
        ldi r24, low(6000)
        ldi r25, high(6000)
        rcall wait_usec
        pop r25
        pop r24
        ;end remote access cseg
        push r24
        in r25 ,PIND
        andi r25 ,0x0f
        andi r24 ,0xf0
        add r24 ,r25
        out PORTD ,r24
        sbi PORTD ,PD3
        cbi PORTD ,PD3
        ;remote access cseg
        push r24
        push r25
        ldi r24, low(6000)
        ldi r25, high(6000)
        rcall wait_usec
        pop r25
        pop r24
        ;end remote access cseg
        pop r24
        swap r24
        andi r24 ,0xf0
        add r24 ,r25
        out PORTD ,r24
        sbi PORTD ,PD3
        cbi PORTD ,PD3
```

```
        ret

lcd_data_sim:
        push r24
        push r25
        sbi PORTD ,PD2
        rcall write_2_nibbles
        ldi r24 ,43
        ldi r25 ,0
        rcall wait_usec
        pop r25
        pop r24
        ret

lcd_command_sim:
        push r24
        push r25
        cbi PORTD, PD2
        rcall write_2_nibbles
        ldi r24, 39
        ldi r25, 0
        rcall wait_usec
        pop r25
        pop r24
        ret

lcd_init_sim:
        push r24
        push r25
        ldi r24 ,40
        ldi r25 ,0
        rcall wait_msec

        ldi r24 ,0x30
        out PORTD ,r24
        sbi PORTD ,PD3
        cbi PORTD ,PD3
        ldi r24 ,39
        ldi r25 ,0
        rcall wait_usec

        ;remote access cseg
        push r24
        push r25
        ldi r24, low(1000)
        ldi r25, high(1000)
        rcall wait_usec
        pop r25
        pop r24
        ;end remote access cseg
        ldi r24 ,0x30
        out PORTD ,r24
        sbi PORTD ,PD3
        cbi PORTD ,PD3
        ldi r24 ,39
        ldi r25 ,0
        rcall wait_usec
        ;remote access cseg
        push r24
        push r25
        ldi r24, low(1000)
        ldi r25, high(1000)
```

```
        rcall wait_usec
        pop r25
        pop r24
        ;end remote access cseg

        ldi r24 ,0x20
        out PORTD ,r24
        sbi PORTD ,PD3
        cbi PORTD ,PD3
        ldi r24 ,39
        ldi r25 ,0
        rcall wait_usec
        ;remote access cseg
        push r24
        push r25
        ldi r24, low(1000)
        ldi r25, high(1000)
        rcall wait_usec
        pop r25
        pop r24
        ;end remote access cseg

        ldi r24 ,0x28
        rcall lcd_command_sim
        ldi r24 ,0x0c
        rcall lcd_command_sim
        ldi r24 ,0x01
        rcall lcd_command_sim
        ldi r24 ,low(1530)
        ldi r25 ,high(1530)
        rcall wait_usec

        ldi r24 ,0x06
        rcall lcd_command_sim

        pop r25
        pop r24
        ret

wait_msec:
        push r24
        push r25
        ldi r24 , low(998)
        ldi r25 , high(998)
        rcall wait_usec
        pop r25
        pop r24
        sbiw r24 , 1
        brne wait_msec
        ret
wait_usec:
        sbiw r24 ,1
        nop
        nop
        nop
        nop
        nop
        brne wait_usec
        ret
```