$3^{η}$ Εργαστηριακή Άσκηση στο Εργαστήριο Μικροεπεξεργαστών

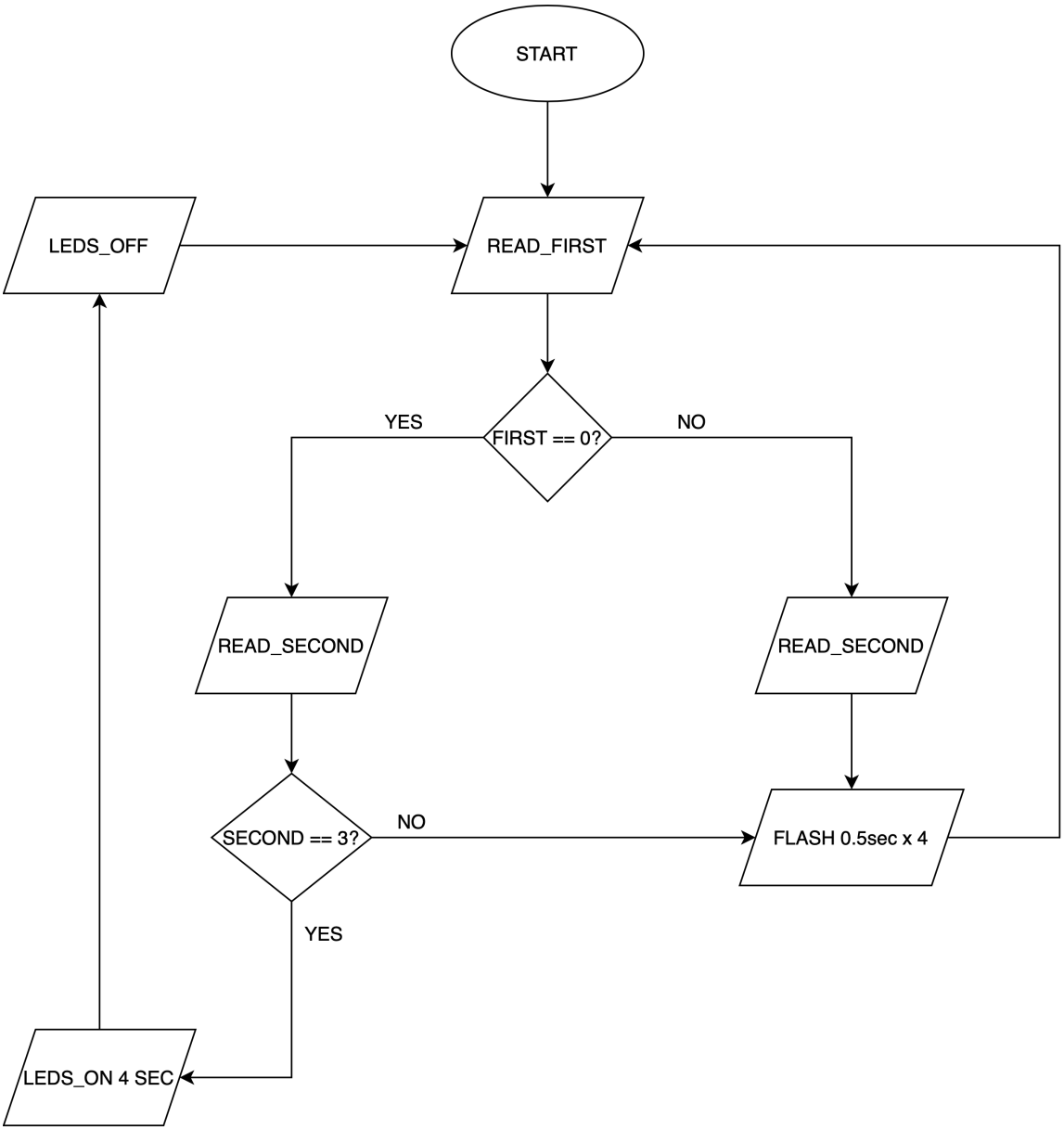
Ομάδα Β 3

Αλέξανδρος Κυριακάκης (03112163),
Ιωάννης Αλεξόπουλος (03117001)

Νοέμβρης 2020

1^{η} Άσκηση

Διάγραμμα Ροής



Code

```
#define F_CPU 8000000   // FREQUENCY OF ATMEGA16
#include <avr/io.h>
#include <util/delay.h>



// MACRO => SET LEDS ON FOR 4sec
#define SUCCESS do {\
 PORTB = 0xFF;\
 _delay_ms(4000);\
 PORTB = 0x00;\
} while (0)

// MACRO => BLINK LEDS ON-OFF FOR 500ms EACH STATE x 4 TIMES = 4 SECONDS
#define BLINK_FAIL do {\
 PORTB = 0xFF;\
 _delay_ms(500);\
 PORTB = 0x00;\
 _delay_ms(500);\
 PORTB = 0xFF;\
 _delay_ms(500);\
 PORTB = 0x00;\
 _delay_ms(500);\
 PORTB = 0xFF;\
 _delay_ms(500);\
 PORTB = 0x00;\
 _delay_ms(500);\
 PORTB = 0xFF;\
 _delay_ms(500);\
 PORTB = 0x00;\
 _delay_ms(500);\
} while (0)

// GLOBAL VARIABLES
unsigned char mem[2],
key_reg[2],
first,second,    // x: 1ST KEY, y: 2ND KEY
flag;      // USED TO CHECK IF FIRST KEY WAS CORRECT

// SCAN ROW(x)
unsigned char scan_row(int i) {   // i = 1,2,3,4
 unsigned char a = ( 1 << 3 ); // SKIP 3 LSB
 a = (a << i);   // SELECT ROW ACCORDING TO FUNCTION INPUT i
 PORTC = a;    // WE SELECT ROW BY SETTING CORRESPONDING BIT TO 1
 _delay_us(500);   // DELAY FOR REMOTE USAGE
 return PINC & 0x0F;  // WE READ THE 4 LSB, '1' INDICATES SWITCH PUSHED
```

```
47  }
48
49  /* FUNCTION TO SWAP LO WITH HO BITS */
50  unsigned char swap(unsigned char x) {
51   return ((x & 0x0F) << 4 | (x & 0xF0) >> 4);
52  }
53
54  /* SCAN ROWS(1..4) *DIFFERENT ORDER FROM EXERSISE DOCUMENT*
55  * FIRST ROW: PC4->PC0: 1,  PC4->PC1: 2,  PC4->PC2: 3, PC4->PC3: A
56  * SECOND ROW: PC5->PC0: 4,  PC5->PC1: 5,  PC5->PC2: 6, PC5->PC3: B
57  * THIRD ROW: PC6->PC0: 7,  PC6->PC1: 8,  PC6->PC2: 9, PC6->PC3: C
58  * FOURTH ROW: PC7->PC0: *,  PC7->PC1: 0,  PC7->PC2: #, PC7->PC3: D
59  */
60  void scan_keypad() {
61   unsigned char i;
62
63   // check row 1, 0b0001-ROW CORRESPONDING TO PC4
64   i = scan_row(1);
65   key_reg[1] = swap(i); //key_reg[1] = first_row(4 MSB)-0000
66
67   // check row 2, 0b0010-ROW CORRESPONDING TO PC5
68   i = scan_row(2);
69   key_reg[1] += i;  //key_reg[1] = first_row(4 MSB)-second_row(4 LSB)
70
71   // check row 3, 0b0100-ROW CORRESPONDING TO PC6
72   i = scan_row(3);
73   key_reg[0] = swap(i); //key_reg[0] = third_row(4 MSB) -0000
74
75   // check row 4, 0b1000-ROW CORRESPONDING TO PC7
76   i = scan_row(4);
77   key_reg[0] += i;  //key_reg[0] = third_row(4 MSB)-fourth_row(4 LSB)
78   PORTC = 0x00;   // added for remote usage
79  }
80
81  int scan_keypad_rising_edge() {
82   // CHECK KEYPAD
83   scan_keypad();      // RETURNS RESULTS IN key_reg
84   // ADD TEMPORARY VARIABLES
85   unsigned char tmp_keypad[2];
86   tmp_keypad[0] = key_reg[0];   //tmp_keypad HOLD ACQUIRED DATA FROM SCAN_KEYPAD()
87   tmp_keypad[1] = key_reg[1];
88
89   _delay_ms(0x15);     // APOFYGH SPINTHIRISMOU
90
91
92   scan_keypad();
93   key_reg[0] &= tmp_keypad[0];  // APPORIPSE TIS TIMES POU EMFANISAN SPINTHIRISMO
94   key_reg[1] &= tmp_keypad[1];
```

```
95
96   tmp_keypad[0] = mem[0];     // BRING LAST STATE OF SWITCHES FROM RAM TO tmp_keypad
97   tmp_keypad[1] = mem[1];
98
99   mem[0] = key_reg[0];     // STORE NEW KEYPAD STATE IN RAM FOR FUTURE CALL
100  mem[1] = key_reg[1];
101
102
103  key_reg[0] &= ~tmp_keypad[0];  // FIND KEYPAD SWITCHES THAT HAVE JUST BEEN
     ↪  PRESSED
104  key_reg[1] &= ~tmp_keypad[1];
105
106  return (key_reg[0] || key_reg[1]); // 16 BIT VALUE INDICATING FRESHLY PRESSED
     ↪  SWITCHES - RETURNS 0 IF NO SWITCH PRESSED
107 }
108
109 /* CONVERT VALUE TO ASCII CODE *CHECK COMMENT ABOVE SCAN_KEYPAD FOR CORRESPONDENCE
110 * key_reg[0] = third_row(4 MSB)-fourth_row(4 LSB)
111 * key_reg[1] = first_row(4 MSB)-second_row(4 LSB)
112 * LSB -> MSB == LEFT -> RIGHT IN KEYPAD */
113 unsigned char keypad_to_ascii() {
114  if (key_reg[0] & 0x01)
115  return '*';
116
117  if (key_reg[0] & 0x02)
118  return '0';
119
120  if (key_reg[0] & 0x04)
121  return '#';
122
123  if (key_reg[0] & 0x08)
124  return 'D';
125
126  if (key_reg[0] & 0x10)
127  return '7';
128
129  if (key_reg[0] & 0x20)
130  return '8';
131
132  if (key_reg[0] & 0x40)
133  return '9';
134
135  if (key_reg[0] & 0x80)
136  return 'C';
137
138  if (key_reg[1] & 0x01)
139  return '4';
140
```

```
  if (key_reg[1] & 0x02)
  return '5';

  if (key_reg[1] & 0x04)
  return '6';

  if (key_reg[1] & 0x08)
  return 'B';

  if (key_reg[1] & 0x10)
  return '1';

  if (key_reg[1] & 0x20)
  return '2';

  if (key_reg[1] & 0x40)
  return '3';

  if (key_reg[1] & 0x80)
  return 'A';

  // Nothing Found
  return 0;
 }

 int main(void) {

  DDRB = 0xFF;         // PORTB => OUTPUT
  DDRC = 0xF0;         // KEYPAD: PORTC[7:4] => OUTPUT, PORTC[3:0] => INPUT

  while (1) {
   MAIN_L:

   mem[0] = 0;     // INITIALIZE RAM
   mem[1] = 0;
   PORTB = 0;
   flag = 0;

   while (1) {

    // GET FIRST DIGIT
    if (scan_keypad_rising_edge()) {
     first = keypad_to_ascii();
     break;
    }
   }

   // IF INPUT EQUAL WITH EXPECTED KEY SET FLAG
```

```
  if (first == '0')
  flag = 1;

   // GET SECOND DIGIT
  while (1) {
   if (scan_keypad_rising_edge()) {
    second = keypad_to_ascii();
    scan_keypad_rising_edge();  // EXTRA CALL ADDED FOR REMOTE USAGE
    break;
   }
  }

  // IF INPUT NOT EQUAL WITH EXPECTED KEY OR FLAG NOT SET INDICATING FIRST DIGIT
   ↪  WRONG ->  WRONG_INPUT
  if (second != '3' || (!flag)) { goto WRONG_INPUT; }

  // SUCCESSFUL

  SUCCESS;
  goto MAIN_L;


  WRONG_INPUT:
  BLINK_FAIL;
 }
 return 0;
}
```

2^{η} Άσκηση

```
; ---- Αρχή τμήματος δεδομένων
.DSEG
_tmp_: .byte 2
; ---- Τέλος τμήματος δεδομένων

.CSEG
.include "m16def.inc"

.def temp=r20
.def cnt=r21
.macro SET_LEDS_ON
; MACRO: SET ALL LEDS OF PORTA TO ON
; AFFECTED REGISTER:
ser r18
out PORTB,r18
.endm

```

```
18
19  .macro SET_LEDS_OFF
20  ; MACRO: SET ALL LEDS OF PORTA TO ON
21  ; AFFECTED REGISTER: r20
22  clr r20
23  out PORTB,r20
24  .endm
25
26  .org 0x00
27  rjmp init
28
29  init:
30  clr temp
31  ; initialization  stack pointer
32  ldi r24, low(RAMEND)
33   out SPL, r24
34
35  ldi r24, high(RAMEND)
36   out SPH, r24
37
38   ser r24 ; r24 = FF
39   out DDRB, r24 ; initialize  port b
40   out DDRD, r24 ; and d for output
41
42   ldi r24, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4) ; θέτει ως εξόδους τα
     ↪  4 MSB
43   out DDRC, r24 ; της θύρας PORTC
44
45
46   first_digit:
47   ldi r24,0xf0 ; pernaw asso se ola ta pliktra
48   rcall scan_keypad_rising_edge_sim ; elegxw tis eksodous
49   clr r22 ; arxikopoiw sto 0
50   or r22, r24 ; ta grafw ola ekei gia na dv an exw allages
51   or r22, r25 ;
52   cpi r22,0 ; an einai 0 shmainei den exw allages kai aksana elegxw
53   breq first_digit
54   ; password  = 03 -> r25 = 0 + r24 = 2 and then r24 = 0 r25 = 0b1000000
55   cpi r25,0 ; elegxw gia to 0
56   brne wrong_first
57   cpi r24,2
58   brne wrong_first
59   rjmp second_digit
60
61   wrong_first:
62   ldi r21,1       ; flag that indicates first digit was incorrect
63
64   second_digit:
```

```
65  ldi r24,0xf0 ; pernaw asso se ola ta pliktra
66  rcall scan_keypad_rising_edge_sim ; elegxw tis eksodous
67  clr r22 ; arxikopoiw sto 0
68  or r22, r24 ; ta grafw ola ekei gia na dv an exw allages
69  or r22, r25
70  cpi r22,0
71  breq second_digit
72  ;r24 = 0 r25 = 0b1000000
73  cpi r21,1
74  breq wrong_passwd
75  cpi r24,0
76  brne wrong_passwd
77  cpi r25,0x40
78  brne wrong_passwd
79  ; an ftasw mexri edw tote exw swsto kwdiko ara
80
81  right_passwd: ; kanei ta 4-sec flashes kai grafei sthn othoni
82  rcall scan_keypad_rising_edge_sim ; extra call for remote usage
83  SET_LEDS_ON  ; ALL LEDS ON (MACRO)
84  ; print  WELCOME 03
85  rcall lcd_init_sim
86  ldi r24,'W'
87  rcall lcd_data_sim ; αποστολή ενός byte δεδομένων στον ελεγκτή της οθόνης lcd
88  ldi r24,'E'
89  rcall lcd_data_sim
90  ldi r24,'L'
91  rcall lcd_data_sim
92  ldi r24,'C'
93  rcall lcd_data_sim
94  ldi r24,'O'
95  rcall lcd_data_sim
96  ldi r24,'M'
97  rcall lcd_data_sim
98  ldi r24,'E'
99  rcall lcd_data_sim
100 ldi r24,' '
101 rcall lcd_data_sim
102 ldi r24,'0'
103 rcall lcd_data_sim
104 ldi r24,'3'
105 rcall lcd_data_sim
106 ldi r24,low(4000)
107 ldi r25,high(4000)
108 rcall wait_msec   ; DELAY 4 SECONDS (MACRO)
109 SET_LEDS_OFF
110
111 rjmp first_digit
```

```
112
113 wrong_passwd:
114 rcall scan_keypad_rising_edge_sim ; extra call for remote usage
115 ; print "ALARM ON"
116 rcall lcd_init_sim
117 ldi r24,'A'
118 rcall lcd_data_sim ; αποστολή ενός byte δεδομένων στον ελεγκτή της οθόνης lcd
119 ldi r24,'L'
120 rcall lcd_data_sim
121 ldi r24,'A'
122 rcall lcd_data_sim
123 ldi r24,'R'
124 rcall lcd_data_sim
125 ldi r24,'M'
126 rcall lcd_data_sim
127 ldi r24,' '
128 rcall lcd_data_sim
129 ldi r24,'O'
130 rcall lcd_data_sim
131 ldi r24,'N'
132 rcall lcd_data_sim
133 ldi cnt,0x04   ; iterate 4 times
134 L1:
135 SET_LEDS_ON   ; set leds on (MACRO)
136 ldi r24,low(500)
137 ldi r25,high(500)
138 rcall wait_msec   ; delay 0.5sec (MACRO)
139
140 SET_LEDS_OFF   ; set leds off (MACRO)
141 ldi r24,low(500)
142 ldi r25,high(500)
143 rcall wait_msec
144
145 dec cnt    ; cnt--
146 cpi cnt, 0x0   ; (cnt == 0) ?
147 brne L1    ; if cnt != 0 goto L1
148
149 rjmp first_digit
150
151
152
153
154
155 scan_row_sim:
156 out PORTC, r25 ; η αντίστοιχη γραμμή τίθεται στο λογικό '1'
157 push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
158 push r25 ; λειτουργία του προγραμματος απομακρυσμένης
```

```
159   ldi r24,low(500) ; πρόσβασης
160  ldi r25,high(500)
161   rcall wait_usec
162   pop r25
163   pop r24 ; τέλος τμήμα κώδικα
164   nop
165   nop ; καθυστέρηση για να προλάβει να γίνει η αλλαγή κατάστασης
166   in r24, PINC ; επιστρέφουν οι θέσεις (στήλες) των διακοπτών που είναι πιεσμένοι
167   andi r24 ,0x0f ; απομονώνονται τα 4 LSB όπου τα '1' δείχνουν που είναι πατημένοι
168   ret ; οι διακόπτες
169
170   scan_keypad_sim:
171   push r26 ; αποθήκευσε τους καταχωρητές r27:r26 γιατι τους
172   push r27 ; αλλάζουμε μέσα στην ρουτίνα
173   ldi r25 , 0x10 ; έλεγξε την πρώτη γραμμή του πληκτρολογίου (PC4: 1 2 3 A)
174   rcall scan_row_sim
175   swap r24 ; αποθήκευσε το αποτέλεσμα
176   mov r27, r24 ; στα 4 msb του r27
177   ldi r25 ,0x20 ; έλεγξε τη δεύτερη γραμμή του πληκτρολογίου (PC5: 4 5 6 B)
178   rcall scan_row_sim
179   add r27, r24 ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r27
180   ldi r25 , 0x40 ; έλεγξε την τρίτη γραμμή του πληκτρολογίου (PC6: 7 8 9 C)
181   rcall scan_row_sim
182   swap r24 ; αποθήκευσε το αποτέλεσμα
183   mov r26, r24 ; στα 4 msb του r26
184   ldi r25 ,0x80 ; έλεγξε την τέταρτη γραμμή του πληκτρολογίου (PC7: * 0 # D)
185   rcall scan_row_sim
186   add r26, r24 ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r26
187   movw r24, r26 ; μετέφερε το αποτέλεσμα στους καταχωρητές r25:r24
188   clr r26 ; προστέθηκε για την απομακρυσμένη πρόσβαση
189   out PORTC,r26 ; προστέθηκε για την απομακρυσμένη πρόσβαση
190   pop r27 ; επανάφερε τους καταχωρητές r27:r26
191   pop r26
192   ret
193
194   scan_keypad_rising_edge_sim:
195   push r22 ; αποθήκευσε τους καταχωρητές r23:r22 και τους
196   push r23 ; r26:r27 γιατι τους αλλάζουμε μέσα στην ρουτίνα
197   push r26
198   push r27
199   rcall scan_keypad_sim ; έλεγξε το πληκτρολόγιο για πιεσμένους διακόπτες
200   push r24 ; και αποθήκευσε το αποτέλεσμα
201   push r25
202   ldi r24 ,15 ; καθυστέρησε 15 ms (τυπικές τιμές 10-20 msec που καθορίζεται από τον
203   ldi r25 ,0 ; κατασκευαστή του πληκτρολογίου ( χρονοδιάρκεια σπινθηρισμών)
204   rcall wait_msec
205   rcall scan_keypad_sim ; έλεγξε το πληκτρολόγιο ξανά και απόρριψε
206   pop r23 ; όσα πλήκτρα εμφανίζουν σπινθηρισμό
```

```
207   pop r22
208   and r24 ,r22
209   and r25 ,r23
210   ldi r26 ,low(_tmp_) ; φόρτωσε την κατάσταση των διακοπτών στην
211   ldi r27 ,high(_tmp_) ; προηγούμενη κλήση της ρουτίνας στους r27:r26
212   ld r23 ,X+
213   ld r22 ,X
214   st X ,r24 ; αποθήκευσε στη RAM τη νέα κατάσταση
215   st -X ,r25 ; των διακοπτών
216   com r23
217   com r22 ; βρες τους διακόπτες που έχουν μόλις πατηθεί
218   and r24 ,r22
219   and r25 ,r23
220   pop r27 ; επανάφερε τους καταχωρητές r27:r26
221   pop r26 ; και r23:r22
222   pop r23
223   pop r22
224   ret
225
226   keypad_to_ascii_sim:
227   push r26 ; αποθήκευσε τους καταχωρητές r27:r26 γιατι τους
228   push r27 ; αλλάζουμε μέσα στη ρουτίνα
229   movw r26 ,r24 ; λογικό '1' στις θέσεις του καταχωρητή r26 δηλώνουν
230   ; τα παρακάτω σύμβολα και αριθμούς
231   ldi r24 ,'*'
232   ; r26
233   ;C 9 8 7 D # 0 *
234   sbrc r26 ,0
235   rjmp return_ascii
236   ldi r24 ,'0'
237   sbrc r26 ,1
238   rjmp return_ascii
239   ldi r24 ,'#'
240   sbrc r26 ,2
241   rjmp return_ascii
242   ldi r24 ,'D'
243   sbrc r26 ,3 ; αν δεν είναι '1'παρακάμπτει την ret, αλλιώς (αν είναι '1')
244   rjmp return_ascii ; επιστρέφει με τον καταχωρητή r24 την ASCII τιμή του D.
245   ldi r24 ,'7'
246   sbrc r26 ,4
247   rjmp return_ascii
248   ldi r24 ,'8'
249   sbrc r26 ,5
250   rjmp return_ascii
251   ldi r24 ,'9'
252   sbrc r26 ,6
253   rjmp return_ascii ;
```

```
254   ldi r24 ,'C'
255   sbrc r26 ,7
256   rjmp return_ascii
257   ldi r24 ,'4' ; λογικό '1' στις θέσεις του καταχωρητή r27 δηλώνουν
258   sbrc r27 ,0 ; τα παρακάτω σύμβολα και αριθμούς
259   rjmp return_ascii
260   ldi r24 ,'5'
261   ;r27
262   ;A 3 2 1 B 6 5 4
263   sbrc r27 ,1
264   rjmp return_ascii
265   ldi r24 ,'6'
266   sbrc r27 ,2
267   rjmp return_ascii
268   ldi r24 ,'B'
269   sbrc r27 ,3
270   rjmp return_ascii
271   ldi r24 ,'1'
272   sbrc r27 ,4
273   rjmp return_ascii ;
274   ldi r24 ,'2'
275   sbrc r27 ,5
276   rjmp return_ascii
277   ldi r24 ,'3'
278   sbrc r27 ,6
279   rjmp return_ascii
280   ldi r24 ,'A'
281   sbrc r27 ,7
282   rjmp return_ascii
283   clr r24
284   rjmp return_ascii
285   return_ascii:
286   pop r27 ; επανάφερε τους καταχωρητές r27:r26
287   pop r26
288   ret
289
290   write_2_nibbles_sim:
291   push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
292   push r25 ; λειτουργία του προγραμματος απομακρυσμένης
293   ldi r24 ,low(6000) ; πρόσβασης
294  ldi r25 ,high(6000)
295   rcall wait_usec
296   pop r25
297   pop r24 ; τέλος τμήμα κώδικα
298   push r24 ; στέλνει τα 4 MSB
299   in r25, PIND ; διαβάζονται τα 4 LSB και τα ξαναστέλνουμε
300   andi r25, 0x0f ; για να μην χαλάσουμε την όποια προηγούμενη κατάσταση
```

```
301   andi r24, 0xf0 ; απομονώνονται τα 4 MSB και
302   add r24, r25 ; συνδυάζονται με τα προϋπάρχοντα 4 LSB
303   out PORTD, r24 ; και δίνονται στην έξοδο
304   sbi PORTD, PD3 ; δημιουργείται παλμός Enable στον ακροδέκτη PD3
305   cbi PORTD, PD3 ; PD3=1 και μετά PD3=0
306   push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
307   push r25 ; λειτουργία του προγραμματος απομακρυσμένης
308   ldi r24 ,low(6000) ; πρόσβασης
309  ldi r25 ,high(6000)
310   rcall wait_usec
311   pop r25
312   pop r24 ; τέλος τμήμα κώδικα
313   pop r24 ; στέλνει τα 4 LSB. Ανακτάται το byte.
314   swap r24 ; εναλλάσσονται τα 4 MSB με τα 4 LSB
315   andi r24 ,0xf0 ; που με την σειρά τους αποστέλλονται
316   add r24, r25
317   out PORTD, r24
318   sbi PORTD, PD3 ; Νέος παλμός Enable
319   cbi PORTD, PD3
320   ret
321
322   lcd_data_sim:
323   push r24
324   push r25
325   sbi PORTD,PD2
326   rcall write_2_nibbles_sim
327   ldi r24,43
328   ldi r25,0
329   rcall wait_usec
330   pop r25
331   pop r24
332   ret
333
334   lcd_command_sim:
335   push r24 ; αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
336   push r25 ; αλλάζουμε μέσα στη ρουτίνα
337   cbi PORTD, PD2 ; επιλογή του καταχωρητή εντολών (PD2=0)
338   rcall write_2_nibbles_sim ; αποστολή της εντολής και αναμονή 39μsec
339   ldi r24, 39 ; για την ολοκλήρωση της εκτέλεσης της από τον ελεγκτή της lcd.
340   ldi r25, 0 ; ΣΗΜ.: υπάρχουν δύο εντολές, οι clear display και return home,
341   rcall wait_usec ; που απαιτούν σημαντικά μεγαλύτερο χρονικό διάστημα.
342   pop r25 ; επανάφερε τους καταχωρητές r25:r24
343   pop r24
344   ret
345
346   lcd_init_sim:
347   push r24 ; αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
348   push r25 ; αλλάζουμε μέσα στη ρουτίνα
```

```
349
350   ldi r24, 40 ; Όταν ο ελεγκτής της lcd τροφοδοτείται με
351   ldi r25, 0 ; ρεύμα εκτελεί την δική του αρχικοποίηση.
352   rcall wait_msec ; Αναμονή 40 msec μέχρι αυτή να ολοκληρωθεί.
353   ldi r24, 0x30 ; εντολή μετάβασης σε 8 bit mode
354   out PORTD, r24 ; επειδή δεν μπορούμε να είμαστε βέβαιοι
355   sbi PORTD, PD3 ; για τη διαμόρφωση εισόδου του ελεγκτή
356   cbi PORTD, PD3 ; της οθόνης, η εντολή αποστέλλεται δύο φορές
357   ldi r24, 39
358   ldi r25, 0 ; εάν ο ελεγκτής της οθόνης βρίσκεται σε 8-bit mode
359   rcall wait_usec ; δεν θα συμβεί τίποτα, αλλά αν ο ελεγκτής έχει διαμόρφωση
360   ; εισόδου 4 bit θα μεταβεί σε διαμόρφωση 8 bit
361   push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
362   push r25 ; λειτουργία του προγραμματος απομακρυσμένης
363   ldi r24,low(1000) ; πρόσβασης
364  ldi r25,high(1000)
365   rcall wait_usec
366   pop r25
367   pop r24 ; τέλος τμήμα κώδικα
368   ldi r24, 0x30
369   out PORTD, r24
370   sbi PORTD, PD3
371   cbi PORTD, PD3
372   ldi r24,39
373   ldi r25,0
374   rcall wait_usec
375   push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
376   push r25 ; λειτουργία του προγραμματος απομακρυσμένης
377   ldi r24 ,low(1000) ; πρόσβασης
378  ldi r25 ,high(1000)
379   rcall wait_usec
380   pop r25
381   pop r24 ; τέλος τμήμα κώδικα
382   ldi r24,0x20 ; αλλαγή σε 4-bit mode
383   out PORTD, r24
384   sbi PORTD, PD3
385   cbi PORTD, PD3
386   ldi r24,39
387   ldi r25,0
388   rcall wait_usec
389   push r24 ; τμήμα κώδικα που προστίθεται για τη σωστή
390   push r25 ; λειτουργία του προγραμματος απομακρυσμένης
391   ldi r24 ,low(1000) ; πρόσβασης
392  ldi r25 ,high(1000)
393   rcall wait_usec
394   pop r25
395   pop r24 ; τέλος τμήμα κώδικα
396   ldi r24,0x28 ; επιλογή χαρακτήρων μεγέθους 5x8 κουκίδων
```

```
397   rcall lcd_command_sim ; και εμφάνιση δύο γραμμών στην οθόνη
398   ldi r24,0x0c ; ενεργοποίηση της οθόνης, απόκρυψη του κέρσορα
399   rcall lcd_command_sim
400   ldi r24,0x01 ; καθαρισμός της οθόνης
401   rcall lcd_command_sim
402   ldi r24, low(1530)
403  ldi r25, high(1530)
404   rcall wait_usec
405   ldi r24 ,0x06 ; ενεργοποίηση αυτόματης αύξησης κατά 1 της διεύθυνσης
406   rcall lcd_command_sim ; που είναι αποθηκευμένη στον μετρητή διευθύνσεων και
407   ; απενεργοποίηση της ολίσθησης ολόκληρης της οθόνης
408   pop r25 ; επανάφερε τους καταχωρητές r25:r24
409   pop r24
410   ret
411   wait_msec:
412   push r24    ; 2 κύκλοι (0.250 μsec)
413   push r25    ; 2 κύκλοι
414   ldi r24 , low(998)  ; φόρτωσε τον καταχ. r25:r24 με 998 (1 κύκλος - 0.125 μsec)
415   ldi r25 , high(998)  ; 1 κύκλος (0.125 μsec)
416   rcall wait_usec   ; 3 κύκλοι (0.375 μsec), προκαλεί συνολικά καθυστέρηση 998.375
      ↪ μsec
417   pop r25     ; 2 κύκλοι (0.250 μsec)
418   pop r24     ; 2 κύκλοι
419   sbiw r24 , 1   ; 2 κύκλοι
420   brne wait_msec   ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
421   ret      ; 4 κύκλοι (0.500 μsec)
422
423   wait_usec:
424   sbiw r24 ,1   ; 2 κύκλοι (0.250 μsec)
425   nop     ; 1 κύκλος (0.125 μsec)
426   nop     ; 1 κύκλος (0.125 μsec)
427   nop     ; 1 κύκλος (0.125 μsec)
428   nop     ; 1 κύκλος (0.125 μsec)
429   brne wait_usec  ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
430   ret     ; 4 κύκλοι (0.500 μsec)
```