ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εργαστήριο Μικροϋπολογιστών

3η Εργαστηριακή Άσκηση

Ονοματεπώνυμα: Μπέτζελος Χρήστος, Γιαννιός Γεώργιος-Ταξιάρχης
Α.Μ. : 031 16 067, 031 16 156
Ομάδα : Α 16
Εξάμηνο: 7ο

Ζήτημα 4.1

Υλοποιήσαμε ένα πρόγραμμα σε assembly για την επικοινωνία του μικροελεγκτή με τον αισθητήρα θερμοκρασίας DS1820 που θα αποτελεί τον οδηγό συσκευής για αυτόν. Ο οδηγός αυτός περιλάμβανε μόνο τις στοιχειώδεις λειτουργίες, όπως στην περίπτωση της αναπτυξιακής πλακέτας EasyAVR6, όπου γνωρίζουμε εκ των προτέρων ότι στον ακροδέκτη PA4 είναι συνδεδεμένος μόνο ένας αισθητήρας. Οι εντολές αυτές μπορούν να χωριστούν σε εντολές μνήμης, με τις οποίες επιλέγεται η συσκευή με την οποία θα γίνει η επικοινωνία, και εντολές λειτουργίας, με τις οποίες υλοποιούνται λειτουργίες που έχουν σχέση με τη μέτρηση θερμοκρασίας. Ενδιαμέσως, ο αισθητήρας βρίσκεται σε κατάσταση χαμηλής κατανάλωσης ισχύος και επανέρχεται σε κατάσταση λειτουργίας με μια εντολή αρχικοποίησης. Ειδικότερα να γράφτηκε ρουτίνα που επέστρεφε στον διπλό καταχωρητή r25:r24 την τιμή της θερμοκρασίας και σε περίπτωση που δεν υπάρχει συνδεδεμένη συσκευή επέστρεφε την τιμή 0x8000. Ποιο αναλυτικά η ρουτίνα περιλάμβανε τα εξής: Αρχικοποίηση της συσκευής και έλεγχος αν υπάρχει συνδεδεμένη συσκευή (one_wire_reset). Αποστολή της εντολή 0xCC, η οποία παρακάμπτει την επιλογή συσκευής από διάδρομο πολλών συσκευών, μια και γνωρίζουμε ότι υπάρχει μόνο μία. Στη συνέχεια, αποστέλλεται η εντολή 0x44, που ξεκινάει μια μέτρηση θερμοκρασίας. Το DS1820 σηματοδοτεί ότι τερματίστηκε η μετατροπή (μόλις η μέτρηση θερμοκρασίας ολοκληρωθεί) με την αποστολή-απάντηση ενός bit με τιμή ‘1’, που ελέγχεται με την εκτέλεση της υπορουτίνας one_wire_receive_bit και την εντολή sbrs r24 , 0. Στο επόμενο βήμα, ο αισθητήρας αρχικοποιείται ξανά («ξυπνάει» από την κατάσταση χαμηλής κατανάλωσης ισχύος) και γίνεται ξανά ο έλεγχος αν υπάρχει συνδεδεμένη συσκευή (one_wire_reset). Αποστέλλεται η εντολή 0xCC και τέλος, με την εντολή 0xBE γίνεται ανάγνωση των 16 bit της μετρημένης θερμοκρασίας, με κλίση της υπορουτίνας one_wire_receive_byte δύο (2) φορές και αποθήκευση της τιμής στο ζεύγος καταχωρητών r25:r24. Η τιμή που διαβάζεται, σύμφωνα με όσα αναφέρονται στις σελίδες 3 και 4 του τεχνικού εγχειριδίου, είναι ένας αριθμός σε μορφή συμπληρώματος ως προς 2. Τα 8 λιγότερα σημαντικά bit αντιστοιχούν σε θερμοκρασία βαθμών Κελσίου, με ακρίβεια 0.5C ανά bit, και τα 8 περισσότερο σημαντικά αποτελούν, στην προκαθορισμένη λειτουργία του αισθητήρα, επέκταση προσήμου. Για παράδειγμα, ο αριθμός 0x0032 αντιστοιχεί σε θερμοκρασία 25C, ο 0xff92 σε -55C και ο 0xffff σε - 0.5C. Η ρουτίνα συμπληρώνεται από μια εντολή που επιστρέφει την τιμή 0x8000 σε περίπτωση που δεν υπάρχει συσκευή συνδεδεμένη στον ακροδέκτη PA4. Με βάση τη παραπάνω ρουτίνα δόθηκε πρόγραμμα που απεικονίζει συνεχώς την θερμοκρασία σε C (με περικοπή του κλασματικού μέρους) και σε μορφή συμπληρώματος ως προς 2 στην θύρα PORTB.

Λύση 4.1 (Assembly)

```
#include "m16def.inc"

start:

;Αρχικοποίηση δείκτη στοίβας

 ldi r24 , low(RAMEND)
 out SPL , r24
 ldi r24 , high(RAMEND)
 out SPH , r24

 clr r24
 out DDRA, r24
 ser r24
 out DDRB, r24                    ;PORTB έξοδος

 rcall temperature
 cpi r25, 0x80                    ;Loop No Device
 breq start
;Μόλις διαβάσει θερμοκρασία
´;Έλεγξε αν είναι θετικός - αρνητικός
 cpi r25, 0x00
 brne negative

;Αν είναι θετικός
positive:
 ;Ολίσθηση και πρόσθεση κρατουμένου για στρογγυλοποίηση
 lsr r24
 adc r24,r25
 out PORTB,r24
 rjmp start

negative:
;Υπολογισμός συμπληρώματος ως προς 2
 neg r24
 clr r25
;Ολίσθηση και πρόσθεση κρατουμένου για στρογγυλοποίηση
 lsr r24
 adc r24,r25
 neg r24
 ori r24, 0x80
 out PORTB,r24
 rjmp start
```

```
;Διαδικασία που μετράει θερμοκρασία
;που έγινε με τη βοήθεια υπόδείξεων στην εκφώνηση

temperature:
 rcall one_wire_reset
 sbrs r24, 0
 rjmp no_device

 ldi r24, 0xCC
 rcall one_wire_transmit_byte

 ldi r24, 0x44
 rcall one_wire_transmit_byte
check:
 rcall one_wire_receive_bit
 sbrs r24, 0
 rjmp check

reset:
 rcall one_wire_reset
 sbrs r24, 0
 rjmp no_device

 ldi r24, 0xCC
 rcall one_wire_transmit_byte

 ldi r24, 0xBE
 rcall one_wire_transmit_byte

 rcall one_wire_receive_byte
 mov r16, r24
 rcall one_wire_receive_byte
 mov r25, r24
 mov r24, r16
 ret

no_device:
 ldi r25,0x80
 ldi r24,0x00
 ret

;Έτοιμες συναρτήσεις απο εκφώνση
one_wire_receive_byte:
 ldi r27 ,8
 clr r26
```

```
loop_:
 rcall one_wire_receive_bit
 lsr r26
 sbrc r24 ,0
 ldi r24 ,0x80
 or r26 ,r24
 dec r27
 brne loop_
 mov r24 ,r26
 ret

one_wire_receive_bit:
 sbi DDRA ,PA4
 cbi PORTA ,PA4                ; generate time slot
 ldi r24 ,0x02
 ldi r25 ,0x00
 rcall wait_usec
 cbi DDRA ,PA4                 ; release the line
 cbi PORTA ,PA4
 ldi r24 ,10             ; wait 10 µs
 ldi r25 ,0
 rcall wait_usec
 clr r24                       ; sample the line
 sbic PINA ,PA4
 ldi r24 ,1
 push r24
 ldi r24 ,49             ; delay 49 µs to meet the standards
 ldi r25 ,0              ; for a minimum of 60 µsec time slot
 rcall wait_usec         ; and a minimum of 1 µsec recovery time
 pop r24
 ret

one_wire_transmit_byte:
 mov r26 ,r24
 ldi r27 ,8
_one_more_:
 clr r24
 sbrc r26 ,0
 ldi r24 ,0x01
 rcall one_wire_transmit_bit
 lsr r26
 dec r27
 brne _one_more_
 ret

one_wire_transmit_bit:
```

```
 push r24                    ; save r24
 sbi DDRA ,PA4
 cbi PORTA ,PA4              ; generate time slot
 ldi r24 ,0x02
 ldi r25 ,0x00
 rcall wait_usec
 pop r24 ; output bit
 sbrc r24 ,0
 sbi PORTA ,PA4
 sbrs r24 ,0
 cbi PORTA ,PA4
 ldi r24 ,58             ; wait 58 μsec for the
 ldi r25 ,0              ; device to sample the line
 rcall wait_usec
 cbi DDRA ,PA4               ; recovery time
 cbi PORTA ,PA4
 ldi r24 ,0x01
 ldi r25 ,0x00
 rcall wait_usec
 ret

one_wire_reset:
 sbi DDRA ,PA4               ; PA4 configured for output
 cbi PORTA ,PA4              ; 480 μsec reset pulse
 ldi r24 ,low(480)
 ldi r25 ,high(480)
 rcall wait_usec
 cbi DDRA ,PA4               ; PA4 configured for input
 cbi PORTA ,PA4
 ldi r24 ,100            ; wait 100 μsec for devices
 ldi r25 ,0              ; to transmit the presence pulse
 rcall wait_usec
 in r24 ,PINA            ; sample the line
 push r24
 ldi r24 ,low(380)       ; wait for 380 μsec
 ldi r25 ,high(380)
 rcall wait_usec
 pop r25                     ; return 0 if no device was
 clr r24                     ; detected or 1 else
 sbrs r25 ,PA4
 ldi r24 ,0x01
 ret


;Ρουτίνες χρονοκαθυστέρησης
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
 brne wait_usec
 ret
```

Λύση 4.1 (C)

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>

//Συναρτήσεις Οθόνης

char one_wire_reset(void);
void one_wire_transmit_bit(char);
void one_wire_transmit_byte(char);
char one_wire_receive_bit(void);
char one_wire_receive_byte(void);

//Επιστρέφει την τιμή της θερμοκρασίας σε πινακα 2 θέσεων
char * temperature(void);

void main(void)
{
      while(1){
            SP = RAMEND;
            char *temp;
            DDRB=0xFF; // Αρχικοποίηση PORT B as output

            temp=temperature();
```

```
//Όσο δεν διαβάζεις θερμοκρασία ξαναπάρε μέτρηση
            while (temp[0]==0x00 && temp[1]==0x80){
                  temp=temperature();
            }
//Μόλις πάρεις μέτρηση
            //Αν είναι Θετικός αριθμός
            if(temp[1]==0x00){
                  //Αν το τελευταίο bit ειναι  1 κάνε στρογυλλοποίηση
                  if((temp[0] & 0x01)==1) {
                        temp[0]=temp[0]>>1;
                        temp[0]++;
                  }
                  else temp[0]=temp[0]>>1;
                  //Εμφάνιση σε PORTB
                  PORTB=temp[0];
            }
            else if(temp[1]==0x80){
            //Αν είναι Αρνητικός αριθμός
                  temp[0]=~temp[0];
                  temp[0]++;
                  //Αν το τελευταίο bit ειναι  1 κάνε στρογυλλοποίηση
                  if((temp[0] & 0x01)==1) {
                        temp[0]=temp[0]>>1;
                        temp[0]++;
                  }
                  else temp[0]=temp[0]>>1;

                  temp[0]=~temp[0];
                  temp[0]++;
                  //Εμφάνιση σε PORTB
                  PORTB= temp[0] | 0x80;
            }
      }
}

//Ακολουθούν οι συναρτήσεις-διαδικασίες σε C οι οποίες προέκυψαν με το
ίδιο σκεπτικό όπως δόθηκαν καθ'υπόδειξη σε assembly

char one_wire_reset(){
      PORTA=PORTA | (1<<PA4);
      DDRA=DDRA | (1<<PA4);
      PORTA=PORTA & ~(1<<PA4);

      _delay_us(480);

      DDRA=DDRA & ~(1<<PA4);
      PORTA=PORTA & ~(1<<PA4);
```

```
	_delay_us(100);

	char x=(PINA & (1<<PA4));
	_delay_us(380);

	if(x==0x10) return 0x00;
	else return 0x01;
}

void one_wire_transmit_bit(char x){
	DDRA=DDRA | (1<<PA4);
	PORTA=PORTA & (0<<PA4);
	_delay_us(2);
	if((x & 0x01)==1) PORTA=PORTA | (1<<PA4);
	if((x & 0x01)==0) PORTA=PORTA & (0<<PA4);
	_delay_us(58);
	DDRA=DDRA & (0<<PA4);
	PORTA=PORTA & (0<<PA4);
	_delay_us(1);
}

void one_wire_transmit_byte(char x){
	for(int i=0; i<8; i++){
		if((x & 0x01)==1) one_wire_transmit_bit(0x01);
		else one_wire_transmit_bit(0x00);
		x=x>>1;
	}
}

char one_wire_receive_bit(){
	char x=0x00;
	DDRA=DDRA | (1<<PA4);
	PORTA=PORTA & (0<<PA4);
	_delay_us(2);
	DDRA=DDRA & (0<<PA4);
	PORTA=PORTA & (0<<PA4);
	_delay_us(10);
	if((PINA & 0x10)==0x10) x=0x01;
	_delay_us(49);
	return x;
}

char one_wire_receive_byte(){
	char x, y=0x00;
	for(int i=0; i<8; i++){
		x=one_wire_receive_bit();
		y=y>>1;
		if((x & 0x01)==1) y=y|0x80;
```

```
            else y=y|x;
      }
      return y;
}


char * temperature(){
      static char x[2];
      char check=one_wire_reset();
      //PORTC=check;   correct
      if((check & 0x01)==0) {
            //PORTC=0xF0; correct
            x[0]=0x00;
            x[1]=0x80;
            return x;
      }
      one_wire_transmit_byte(0xCC);
      one_wire_transmit_byte(0x44);
      while((one_wire_receive_bit() & 0x01)==0);

      if((one_wire_reset() & 0x01)==0) {
            x[0]=0x00;
            x[1]=0x80;
            return x;
      }
      one_wire_transmit_byte(0xCC);
      one_wire_transmit_byte(0xBE);
      x[0]=one_wire_receive_byte();
      x[1]=one_wire_receive_byte();

      return x;

}
```

Ζήτημα 4.2.α

Γράψαμε πρόγραμμα σε assembly και σε C που με την χρήση της προηγούμενης ρουτίνας απεικόνιζε τη θερμοκρασία σε C στο LCD display σε δεκαδική τιμή τριών ψηφίων με το πρόσημο (-55C έως +125C). Επίσης στην περίπτωση που δεν υπάρχει συσκευή συνδεδεμένη εμφάνιζε το μήνυμα “NO Device”.

Λύση 4.2.α(Assembly)

```
#include "m16def.inc"

start:
 ;Αρχικοποίηση στοίβας
 ldi r24 , low(RAMEND)
 out SPL , r24
```

```
 ldi r24 , high(RAMEND)
 out SPH , r24

 clr r24
 out DDRA, r24
 ser r24                        ;Αρχικοποίηση αισθητήρα θερμοκρασίας(είσοδος)
 out DDRB, r24

 ldi r24,0xfc                   ;Αρχικοποίηση οθόνης (έξοδος)
 out DDRD, r24

;Όσο δεν διαβάζεις θερμοκρασία
 rcall temperature
 cpi r25, 0x80
 breq no_div
 rjmp check_zero

no_div:
 rcall lcd_init
 ldi r24, 'N'
 rcall lcd_data          ;Τύπωσε  'N'
 ldi r24, 'o'
 rcall lcd_data          ;Τύπωσε  'o'
 ldi r24, ' '
 rcall lcd_data          ;Τύπωσε  ' '
 ldi r24, 'D'
 rcall lcd_data          ;Τύπωσε  'D'
 ldi r24, 'e'
 rcall lcd_data          ;Τύπωσε  'e'
 ldi r24, 'v'
 rcall lcd_data          ;Τύπωσε  'v'
 ldi r24, 'i'
 rcall lcd_data          ;Τύπωσε  'i'
 ldi r24, 'c'
 rcall lcd_data          ;Τύπωσε  'c'
 ldi r24, 'e'
 rcall lcd_data          ;Τύπωσε  'e'
 rjmp start

;Ξεχωριστή περίπτωση 0
check_zero:
 cpi r24,0x00
 brne sign

zero:
 rcall lcd_init
```

```
 ldi r24, '0'
 rcall lcd_data
 rjmp start

;Έλεγχος αν είναι θετικός - αρνητικός
sign:
 cpi r25, 0x00
 brne negative

;Αν είναι θετικός
positive:
 push r24
 rcall lcd_init
 ldi r24, '+'
 rcall lcd_data          ;Τύπωσε  '+'
 pop r24
 lsr r24
 adc r24,r25
 rjmp bcd

negative:
 push r24
 rcall lcd_init
 ldi r24, '-'
 rcall lcd_data          ;Τύπωσε '-'
 pop r24
 neg r24
 clr r25
 lsr r24
 adc r24,r25

;Μέτρημα εκατοντάδων - δεκάδων -μονάδων σύμφωνα με διάγραμμα ροής
;προηγούμενης εργαστηριακής
bcd:
 cpi r24,0x64
 brcc ekat
 ldi r28,'0'
 rjmp deci

ekat:
 ldi r28,'1'
 subi r24,0x64

deci:
 ldi r27,0x00
 cpi r24,0x0A
```

```
 brcc mon
 rjmp end

mon:
 inc r27
 subi r24,0x0A
 cpi r24,0x0A
 brcc mon

end:
 ldi r16,0x30
 add r27,r16
 add r24,r16
 mov r26,r24

;r26 monades
;r27 dekades
;r28 ekatontades

lcd:
 cpi r28,'0'
 brne triple

;Ελεγχος αν είναι τριψήφιος- διψήφιος -μονοψήφιος

double:
 cpi r27,'0'
 breq single
 mov r24, r27
 rcall lcd_data
 mov r24, r26
 rcall lcd_data
 ldi r24, '°'
 rcall lcd_data
 ldi r24, 'C'
 rcall lcd_data
 rjmp start

single:
 mov r24, r26
 rcall lcd_data
 ldi r24, '°'
 rcall lcd_data
 ldi r24, 'C'
 rcall lcd_data
 rjmp start
```

```
triple:
 mov r24, r28
 rcall lcd_data
 mov r24, r27
 rcall lcd_data
 mov r24, r26
 rcall lcd_data
 ldi r24, '°'
 rcall lcd_data
 ldi r24, 'C'
 rcall lcd_data
 rjmp start

;Ακολουθούν οι γνωστές συναρτήσεις
;οθόνης - αισθητήρα
lcd_init:
 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
 ldi r24 ,0x30
 out PORTD ,r24
 sbi PORTD ,PD3
 cbi PORTD ,PD3
 ldi r24 ,39
 ldi r25 ,0
 rcall wait_usec
 ldi r24 ,0x20
 out PORTD ,r24
 sbi PORTD ,PD3
 cbi PORTD ,PD3
 ldi r24 ,39
 ldi r25 ,0
 rcall wait_usec
 ldi r24 ,0x28
 rcall lcd_command
 ldi r24 ,0x0c
 rcall lcd_command
 ldi r24 ,0x01
```

```
 rcall lcd_command
 ldi r24 ,low(1530)
 ldi r25 ,high(1530)
 rcall wait_usec
 ldi r24 ,0x06
 rcall lcd_command
 ret

lcd_data:
 sbi PORTD ,PD2
 rcall write_2_nibbles
 ldi r24 ,43
 ldi r25 ,0
 rcall wait_usec
 ret

lcd_command:
 cbi PORTD ,PD2
 rcall write_2_nibbles
 ldi r24 ,39
 ldi r25 ,0
 rcall wait_usec
 ret

write_2_nibbles:
 push r24
 in r25 ,PIND
 andi r25 ,0x0f
 andi r24 ,0xf0
 add r24 ,r25
 out PORTD ,r24
 sbi PORTD ,PD3
 cbi PORTD ,PD3
 pop r24
 swap r24
 andi r24 ,0xf0
 add r24 ,r25
 out PORTD ,r24
 sbi PORTD ,PD3
 cbi PORTD ,PD3
 ret


temperature:
 rcall one_wire_reset
 sbrs r24, 0
```

```
 rjmp no_device

 ldi r24, 0xCC
 rcall one_wire_transmit_byte

 ldi r24, 0x44
 rcall one_wire_transmit_byte
check:
 rcall one_wire_receive_bit
 sbrs r24, 0
 rjmp check

reset:
 rcall one_wire_reset
 sbrs r24, 0
 rjmp no_device

 ldi r24, 0xCC
 rcall one_wire_transmit_byte

 ldi r24, 0xBE
 rcall one_wire_transmit_byte

 rcall one_wire_receive_byte
 mov r16, r24
 rcall one_wire_receive_byte
 mov r25, r24
 mov r24, r16
 ret

no_device:
 ldi r25,0x80
 ldi r24,0x00
 ret


one_wire_receive_byte:
 ldi r27 ,8
 clr r26
loop_:
 rcall one_wire_receive_bit
 lsr r26
 sbrc r24 ,0
 ldi r24 ,0x80
 or r26 ,r24
 dec r27
```

```
 brne loop_
 mov r24 ,r26
 ret

one_wire_receive_bit:
 sbi DDRA ,PA4
 cbi PORTA ,PA4                ; generate time slot
 ldi r24 ,0x02
 ldi r25 ,0x00
 rcall wait_usec
 cbi DDRA ,PA4                 ; release the line
 cbi PORTA ,PA4
 ldi r24 ,10             ; wait 10 µs
 ldi r25 ,0
 rcall wait_usec
 clr r24                       ; sample the line
 sbic PINA ,PA4
 ldi r24 ,1
 push r24
 ldi r24 ,49             ; delay 49 µs to meet the standards
 ldi r25 ,0              ; for a minimum of 60 µsec time slot
 rcall wait_usec         ; and a minimum of 1 µsec recovery time
 pop r24
 ret

one_wire_transmit_byte:
 mov r26 ,r24
 ldi r27 ,8
_one_more_:
 clr r24
 sbrc r26 ,0
 ldi r24 ,0x01
 rcall one_wire_transmit_bit
 lsr r26
 dec r27
 brne _one_more_
 ret

one_wire_transmit_bit:
 push r24                      ; save r24
 sbi DDRA ,PA4
 cbi PORTA ,PA4                ; generate time slot
 ldi r24 ,0x02
 ldi r25 ,0x00
 rcall wait_usec
 pop r24 ; output bit
```

```
 sbrc r24 ,0
 sbi PORTA ,PA4
 sbrs r24 ,0
 cbi PORTA ,PA4
 ldi r24 ,58            ; wait 58 µsec for the
 ldi r25 ,0             ; device to sample the line
 rcall wait_usec
 cbi DDRA ,PA4                  ; recovery time
 cbi PORTA ,PA4
 ldi r24 ,0x01
 ldi r25 ,0x00
 rcall wait_usec
 ret

one_wire_reset:
 sbi DDRA ,PA4                  ; PA4 configured for output
 cbi PORTA ,PA4                 ; 480 µsec reset pulse
 ldi r24 ,low(480)
 ldi r25 ,high(480)
 rcall wait_usec
 cbi DDRA ,PA4                  ; PA4 configured for input
 cbi PORTA ,PA4
 ldi r24 ,100           ; wait 100 µsec for devices
 ldi r25 ,0             ; to transmit the presence pulse
 rcall wait_usec
 in r24 ,PINA           ; sample the line
 push r24
 ldi r24 ,low(380)      ; wait for 380 µsec
 ldi r25 ,high(380)
 rcall wait_usec
 pop r25                        ; return 0 if no device was
 clr r24                        ; detected or 1 else
 sbrs r25 ,PA4
 ldi r24 ,0x01
 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
 brne wait_usec
 ret
```

Λύση 4.2.α(C)

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>

//Συναρτήσεις αισθητήρα θερμοκρασίας

char one_wire_reset(void);
void one_wire_transmit_bit(char);
void one_wire_transmit_byte(char);
char one_wire_receive_bit(void);
char one_wire_receive_byte(void);
char * temperature(void);

//Συναρτήσεις Οθόνης

void lcd_init (void);
void write_2_nibbles(char);
void lcd_data(char);
void lcd_command(char);

void main(void)
{
      SP = RAMEND;
      char c;
      DDRD=0xFC;              //Αρχικοποίηση Οθόνης για έξοδο
      while(1){
            char *temp;
            DDRA=0x00;

          char sign,metro=0,ekat=0,dec=0,mon=0;
          temp=temperature();
```

```
//Όσο δεν λαμβάνεις κάποια μέτρηση απο αισθητήρα
            while (temp[0]==0x00 && temp[1]==0x80){
                  lcd_init();
                  lcd_data('N');
                  lcd_data('O');
                  lcd_data(' ');
                  lcd_data('D');
                  lcd_data('e');
                  lcd_data('v');
                  lcd_data('i');
                  lcd_data('c');
                  lcd_data('e');
                  temp=temperature();
            }
//Μόλις πάρεις την πρώτη μέτρηση

            //Άν είναι θετικός αριθμός
            if(temp[1]==0x00){
                  //Αν το τελευταίο bit =1->Στρογγυλοποίηση
                  if((temp[0] & 0x01)==1) {
                        temp[0]=temp[0]>>1;
                        temp[0]++;
                  }
                  else temp[0]=temp[0]>>1;
            }
            //Άν είναι αρνητικός αριθμός
            else{
                  temp[0]=~temp[0];
                  temp[0]++;
                  //Αν το τελευταίο bit =1->Στρογγυλοποίηση
                  if((temp[0] & 0x01)==1) {
                        temp[0]=temp[0]>>1;
                        temp[0]++;
                  }
                  else temp[0]=temp[0]>>1;
            }
//Έλεγχος ξεχωριστά της περίπτωσης η θερμοκρασία να είναι 0
            c=temp[0];
            if(c==0)
            {
                  lcd_init();
                  lcd_data('0');
                  lcd_data('°');
                  lcd_data('C');
            }
//Αλλίως αν δέν είναι η θερμοκρασία 0

            else
```

```
        {
                //Αν είναι αρνητική
                if(c>=0x80)
                {
                        sign='-';
                        metro=(0x80-(c & 0x7f));
                }
                //Αν είναι θετική
                else
                {
                        sign='+';
                        metro=c;
                }

                //Αν μέτρο απο 100 έως 125
                if(metro>=100)
                {
                        ekat=1;
                        metro=metro-100;
                }
                //Υπολόγισε δεκάδες αφαιρώντας σε loop 10
                while(metro>=10)
                {
                        dec++;
                        metro=metro-10;
                }
                //Ό,τι περίσσεψε μονάδες

                mon=metro;
//Τύπωσε τα αποτελέσματα στην οθόνη
                lcd_init();
                lcd_data(sign);
                if(ekat==0)
                {
                        if(dec==0)
                        //Αν είναι μονοψήφιος
                        {
                                lcd_data((mon & 0x0f)+0x30);
                                lcd_data('°');
                                lcd_data('C');
                        }
                        else
                        {
                        //Αν είναι διψήφιος
                                lcd_data((dec & 0x0f)+0x30);
                                lcd_data((mon & 0x0f)+0x30);
                                lcd_data('°');
                                lcd_data('C');
```

```
                        }
                  }
                  else
                  {
                  //Αν είναι τριψήφιος
                        lcd_data((ekat & 0x0f)+0x30);
                        lcd_data((dec & 0x0f)+0x30);
                        lcd_data((mon & 0x0f)+0x30);
                        lcd_data('°');
                        lcd_data('C');
                  }
            }
      }
}

//Ακολοθούν οι υλοποιήσεις συναρτήσεων - διαδικασιών για οθόνη -
αισθητήρα

char one_wire_reset(){
      PORTA=PORTA | (1<<PA4);
      DDRA=DDRA | (1<<PA4);
      PORTA=PORTA & ~(1<<PA4);

      _delay_us(480);

      DDRA=DDRA & ~(1<<PA4);
      PORTA=PORTA & ~(1<<PA4);

      _delay_us(100);

      char x=(PINA & (1<<PA4));
      _delay_us(380);

      if(x==0x10) return 0x00;
      else return 0x01;
}

void one_wire_transmit_bit(char x){
      DDRA=DDRA | (1<<PA4);
      PORTA=PORTA & (0<<PA4);
      _delay_us(2);
      if((x & 0x01)==1) PORTA=PORTA | (1<<PA4);
      if((x & 0x01)==0) PORTA=PORTA & (0<<PA4);
      _delay_us(58);
      DDRA=DDRA & (0<<PA4);
      PORTA=PORTA & (0<<PA4);
      _delay_us(1);
}
```

```
void one_wire_transmit_byte(char x){
      for(int i=0; i<8; i++){
            if((x & 0x01)==1) one_wire_transmit_bit(0x01);
            else one_wire_transmit_bit(0x00);
            x=x>>1;
      }
}

char one_wire_receive_bit(){
      char x=0x00;
      DDRA=DDRA | (1<<PA4);
      PORTA=PORTA & (0<<PA4);
      _delay_us(2);
      DDRA=DDRA & (0<<PA4);
      PORTA=PORTA & (0<<PA4);
      _delay_us(10);
      if((PINA & 0x10)==0x10) x=0x01;
      _delay_us(49);
      return x;
}

char one_wire_receive_byte(){
      char x, y=0x00;
      for(int i=0; i<8; i++){
            x=one_wire_receive_bit();
            y=y>>1;
            if((x & 0x01)==1) y=y|0x80;
            else y=y|x;
      }
      return y;
}


char * temperature(){
      static char x[2];
      char check=one_wire_reset();
      //PORTC=check;   correct
      if((check & 0x01)==0) {
            //PORTC=0xF0; correct
            x[0]=0x00;
            x[1]=0x80;
            return x;
      }
      one_wire_transmit_byte(0xCC);
      one_wire_transmit_byte(0x44);
      while((one_wire_receive_bit() & 0x01)==0);
```

```
	if((one_wire_reset() & 0x01)==0) {
		x[0]=0x00;
		x[1]=0x80;
		return x;
	}
	one_wire_transmit_byte(0xCC);
	one_wire_transmit_byte(0xBE);
	x[0]=one_wire_receive_byte();
	x[1]=one_wire_receive_byte();

	return x;

}

void write_2_nibbles(char x)
{
	char y=PIND & 0x0f;
	char x1=x & 0xf0;
	x1=x1+y;
	PORTD=x1;
	PORTD=PORTD | (1<<PD3);
	PORTD=PORTD & (0<<PD3);
	x=x<<4 | x>>4;
	x=x & 0xf0;
	PORTD=x+y;
	PORTD=PORTD | (1<<PD3);
	PORTD=PORTD & (0<<PD3);
}

void lcd_data(char x)
{
	PORTD=PORTD | (1<<PD2);
	write_2_nibbles(x);
	_delay_us(43);
}

void  lcd_command(char x)
{
	PORTD=PORTD | (0<<PD2);
	write_2_nibbles(x);
	_delay_us(43);
}

void lcd_init (void)
{
	_delay_ms(40);
	PORTD=0x30;
	PORTD=PORTD | (1<<PD3);
```

```
        PORTD=PORTD & (0<<PD3);
        _delay_us(39);
        PORTD=0x30;
        PORTD=PORTD | (1<<PD3);
        PORTD=PORTD & (0<<PD3);
        _delay_us(39);
        PORTD=0x20;
        PORTD=PORTD | (1<<PD3);
        PORTD=PORTD & (0<<PD3);
        _delay_us(39);
        lcd_command(0x28);
        lcd_command(0x0c);
        lcd_command(0x01);
        _delay_us(1530);
        lcd_command(0x06);
}
```

Ζήτημα 4.2.β

Για να μπορούμε να ελέγχουμε την απεικόνιση, ανεξάρτητα από ύπαρξη του αισθητήρα DS1820, δώσαμε και μια άλλη μορφή του προγράμματος που λαμβάνει τα 4 Hex ψηφία των καταχωρητών r25:r24 μέσω του δεκαεξαδικού πληκτρολογίου. Για το σκοπό αυτό έγινει μετατροπή στην ρουτίνα keypad_to_ascii ώστε να αναγνωρίζει το '#' ως 'F' και το '*' ως 'E' και να μπορεί το πληκτρολόγιο να χρησιμοποιηθεί ως δεκαεξαδικό. Η νέα αυτή ρουτίνα keypad_to_hex επέστρεφε την δεκαεξαδική τιμή των πλήκτρων συμπεριλαμβανομένων των '#' και '*' που θα θεωρούνται πλέον ως 'F' και 'E' αντίστοιχα (βλ. Υπόδειξη από το Ζήτημα 3.2).

Λύση 4.2.β(Assembly)

```
#include "m16def.inc"

.DSEG
 _tmp_: .byte 2

.CSEG

reset:
 ;Αρχικοποίηση δείκτη στοίβας
 ldi r24 , low(RAMEND)
 out SPL , r24
 ldi r24 , high(RAMEND)
 out SPH , r24

 ldi r24, 0xfc                ;αρχικοποίηση PORTD που συνδέεται
 out DDRD, r24                ;η οθόνη ως έξοδος
```

```
;Θέτουμε ως έξοδο τα 4 MSB του PORTC
 ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
 out DDRC ,r24

first_key:
 ldi r24,0x05            ;5ms σπινθηρισμού
 rcall scan_keypad_rising_edge
 rcall keypad_to_hex
 ;ldi r24,0x08
 cpi r24,0x10            ;αν είναι 0x10 τότε δεν πατήθηκε τίποτα και
 breq first_key          ;ξαναπήγαινε στο first_key
 andi r24,0x0F           ;απομώνωσε τα LSB
 mov r16,r24             ;r16=1ο ψηφίο
 swap r24                        ;LSB -> MSB
 mov r29, r24            ;αποθήκευσε

 ldi r20, 0x30
 cpi r16,0x0A
 brcc is_letter1
 add r16,r20
 rjmp second_key

is_letter1:
 ldi r20, 0x37
 add r16,r20


second_key:
 ldi r24,0x05            ;5ms σπινθηρισμού
 rcall scan_keypad_rising_edge
 rcall keypad_to_hex
 ;ldi r24,0x00
 cpi r24,0x10            ;αν είναι 0x10 τότε δεν πατήθηκε τίποτα και
 breq second_key         ;ξαναπήγαινε στο second_key
 andi r24,0x0F           ;απομώνωσε τα LSB
 mov r17,r24             ;r17=2ο ψηφίο
 add r29,r17             ;πρόσθεσε τον προηγούμενο αριθμό

 ldi r20, 0x30
 cpi r17,0x0A
 brcc is_letter2
 add r17,r20
 rjmp third_key

is_letter2:
 ldi r20, 0x37
```

```
 add r17,r20

third_key:
 ldi r24,0x05            ;5ms σπινθηρισμού
 rcall scan_keypad_rising_edge
 rcall keypad_to_hex
 ;ldi r24,0x00
 cpi r24,0x10            ;αν είναι 0x10 τότε δεν πατήθηκε τίποτα και
 breq third_key          ;ξαναπήγαινε στο third_key
 andi r24,0x0F           ;απομώνωσε τα LSB
 mov r18,r24             ;r18=3ο ψηφίο ascii
 swap r24                        ;LSB -> MSB
 mov r21, r24            ;αποθήκευσε προσωρινά στον r21

 ldi r20, 0x30
 cpi r18,0x0A
 brcc is_letter3
 add r18,r20
 rjmp fourth_key

is_letter3:
 ldi r20, 0x37
 add r18,r20

fourth_key:
 ldi r24,0x05            ;5ms σπινθηρισμού
 rcall scan_keypad_rising_edge
 rcall keypad_to_hex
 ;ldi r24,0x00
 cpi r24,0x10            ;αν είναι 0x10 τότε δεν πατήθηκε τίποτα και
 breq fourth_key         ;ξαναπήγαινε στο fourth_key
 andi r24,0x0F           ;απομώνωσε τα LSB
 mov r19,r24             ;r19=4ο ψηφίο
 add r21,r24             ;πρόσθεσε τον προηγούμενο αριθμό

 ldi r20, 0x30
 cpi r19,0x0A
 brcc is_letter4
 add r19,r20
 rjmp print_keys

is_letter4:
 ldi r20, 0x37
 add r19,r20

;Στον r29,r21 έχουμε την θερμοκρασία και στους r16-r19 τα ψηφία προς εκτύπωση
```

```
print_keys:
 rcall lcd_init
 mov r24,r16
 rcall lcd_data
 mov r24,r17
 rcall lcd_data
 mov r24,r18
 rcall lcd_data
 mov r24,r19
 rcall lcd_data
 ldi r24, '='
 rcall lcd_data
 ldi r24, '>'
 rcall lcd_data

check_device:
 cpi r29, 0x80
 brne check2
 cpi r21, 0x00
 brne error

no_div:
 ldi r24, 'N'
 rcall lcd_data          ;Τύπωσε το 'N'
 ldi r24, 'O'
 rcall lcd_data          ;Τύπωσε το 'o'
 ldi r24, ' '
 rcall lcd_data          ;Τύπωσε το ' '
 ldi r24, 'D'
 rcall lcd_data          ;Τύπωσε το 'D'
 ldi r24, 'e'
 rcall lcd_data          ;Τύπωσε το 'e'
 ldi r24, 'v'
 rcall lcd_data          ;Τύπωσε το 'v'
 ldi r24, 'i'
 rcall lcd_data          ;Τύπωσε το 'i'
 ldi r24, 'c'
 rcall lcd_data          ;Τύπωσε το 'c'
 ldi r24, 'e'
 rcall lcd_data          ;Τύπωσε το 'e'
 rjmp reset

check2:
 cpi r29, 0x00
 breq check_zero
```

```
 cpi r29, 0xFF
 breq negative
 rjmp error

check_zero:
 cpi r21,0x00
 brne positive

zero:
 ldi r24, '0'
 rcall lcd_data
 ldi r24, '°'
 rcall lcd_data
 ldi r24, 'C'
 rcall lcd_data
 rjmp reset

error:
 ldi r24, 'E'
 rcall lcd_data          ;Τύπωσε το 'E'
 ldi r24, 'R'
 rcall lcd_data          ;Τύπωσε το 'R'
 ldi r24, 'R'
 rcall lcd_data          ;Τύπωσε το 'R
 ldi r24, 'O'
 rcall lcd_data          ;Τύπωσε το 'O'
 ldi r24, 'R'
 rcall lcd_data          ;Τύπωσε το 'R'
 rjmp reset

negative:               ;Βρίσκουμε το μέτρο του αρνητικού αριθμού
 neg r21
 cpi r21, 0x00          ;OVERFLOW
 breq error
 clr r29
 lsr r21
 adc r21,r29
 cpi r21,0x38
 brcc error
 ldi r24,'-'
 rcall lcd_data
 rjmp bcd

positive:
 clr r29
 lsr r21
```

```
 adc r21,r29
 cpi r21,0x7e
 brcc error
 ldi r24,'+'
 rcall lcd_data


bcd:
 mov r24,r21
 cpi r24,0x64          ;Σύγκριση μέτρου με το 100
 brcc ekat                   ;Αν είναι >= 100 πήγαινε στο ekat
 ldi r28,'0'           ;Εκατοντάδες (r28) = 0
 rjmp deci

ekat:
 ldi r28,'1'           ;Εκατοντάδες (r28) = 1
 subi r24,0x64

deci:
 ldi r27,0x00          ;Αρχικοποίηση δεκάδων (r27) στο 0
 cpi r24,0x0A          ;Σύγκριση μέτρου με το 10
 brcc mon              ;Αν είναι >= 10 πήγαινε στο mon
 rjmp end

mon:
 inc r27
 subi r24,0x0A
 cpi r24,0x0A          ;Σύγκριση μέτρου με το 10
 brcc mon              ;Αν είναι >= 10 πήγαινε στο mon

;Οπότε τελικά έχουμε στον r28:ekat σε ascii, r27:dec, r26:mon και αρχικός τετραψήφιος
στους r16-r19

end:
 ldi r20,0x30
 add r27,r20           ;Μετατροπή σε ascii
 add r24,r20           ;Μετατροπή σε ascii
 mov r26,r24           ;Μετατροπή σε ascii

 cpi r28,'0'           ;Αν ekat!=0
 brne triple           ;πήγαινε στο triple

double:
 cpi r27,'0'           ;Αν dec==0
 breq single           ;πήγαινε στο single
 mov r24, r27
```

```
 rcall lcd_data          ;Τύπωσε δεκάδες
 mov r24, r26
 rcall lcd_data          ;Τύπωσε μονάδες
 ldi r24, '°'
 rcall lcd_data
 ldi r24, 'C'
 rcall lcd_data
 rjmp reset

single:
 mov r24, r26
 rcall lcd_data          ;Τύπωσε μονάδες
 ldi r24, '°'
 rcall lcd_data
 ldi r24, 'C'
 rcall lcd_data
 rjmp reset

triple:
 mov r24, r28
 rcall lcd_data          ;Τύπωσε εκατοντάδες
 mov r24, r27
 rcall lcd_data          ;Τύπωσε δεκάδες
 mov r24, r26
 rcall lcd_data          ;Τύπωσε μονάδες
 ldi r24, '°'
 rcall lcd_data
 ldi r24, 'C'
 rcall lcd_data
 rjmp reset


 ;ρουτίνα αντιστοίχισης διακοπτών σε hex
keypad_to_hex:
 movw r26 ,r24
 ldi r24 ,0x0E
 sbrc r26 ,0
 ret
 ldi r24 ,0x00
 sbrc r26 ,1
 ret
 ldi r24 ,0x0F
 sbrc r26 ,2
 ret
 ldi r24 ,0x0D
 sbrc r26 ,3
```

```
 ret
 ldi r24 ,0x07
 sbrc r26 ,4
 ret
 ldi r24 ,0x08
 sbrc r26 ,5
 ret
 ldi r24 ,0x09
 sbrc r26 ,6
 ret
 ldi r24 ,0x0C
 sbrc r26 ,7
 ret
 ldi r24 ,0x04
 sbrc r27 ,0
 ret
 ldi r24 ,0x05
 sbrc r27 ,1
 ret
 ldi r24 ,0x06
 sbrc r27 ,2
 ret
 ldi r24 ,0x0B
 sbrc r27 ,3
 ret
 ldi r24 ,0x01
 sbrc r27 ,4
 ret
 ldi r24 ,0x02
 sbrc r27 ,5
 ret
 ldi r24 ,0x03
 sbrc r27 ,6
 ret
 ldi r24 ,0x0A
 sbrc r27 ,7
 ret
 ldi r24, 0x10              ;Αν δεν πατήθηκε τίποτα φόρτωσε το 0x010
 ret

scan_keypad_rising_edge:
 mov r22 ,r24
 rcall scan_keypad
 push r24
 push r25
 mov r24 ,r22
```

```
 ldi r25 ,0
 rcall wait_msec
 rcall scan_keypad
 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
 ret


;Ρουτίνα ελέγχου όλου του πληκτρολογίου
scan_keypad:
 ldi r24 , 0x01          ; έλεγξε την πρώτη γραμμή του πληκτρολογίου
 rcall scan_row
 swap r24                ; αποθήκευσε το αποτέλεσμα
 mov r27 , r24           ; στα 4 msb του r27
 ldi r24 ,0x02           ; έλεγξε τη δεύτερη γραμμή του πληκτρολογίου
 rcall scan_row
 add r27 , r24           ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r27
 ldi r24 , 0x03          ; έλεγξε την τρίτη γραμμή του πληκτρολογίου
 rcall scan_row
 swap r24                ; αποθήκευσε το αποτέλεσμα
 mov r26 , r24           ; στα 4 msb του r26
 ldi r24 ,0x04           ; έλεγξε την τέταρτη γραμμή του πληκτρολογίου
 rcall scan_row
 add r26 , r24           ; αποθήκευσε το αποτέλεσμα στα 4 lsb του r26
 movw r24 , r26          ; μετέφερε το αποτέλεσμα στους καταχωρητές r25:r24
 ret


;Ρουτίνα ελέγχου μιας γραμμής του πληκτρολογίου
scan_row:
 ldi r25 , 0x08          ; αρχικοποίηση με '0000 1000'
back_:
 lsl r25                 ; αριστερή ολίσθηση του '1' τόσες θέσεις
 dec r24                 ; όσος είναι ο αριθμός της γραμμής
```

```
 brne back_
 out PORTC , r25       ; η αντίστοιχη γραμμή τίθεται στο λογικό ‘1’
 nop                   ; καθυστέρηση για να προλάβει να γίνει η αλλαγή κατάστασης
 nop
 in r24 , PINC         ; επιστρέφουν οι θέσεις (στήλες) των διακοπτών που είναι πιεσμένοι
 andi r24 ,0x0f        ; απομονώνονται τα 4 LSB όπου τα ‘1’ δείχνουν που είναι πατημένοι
 ret                   ; οι διακόπτες.

;ρουτίνα αρχικοποίσης και ρυθμίσεων LCD
lcd_init:
 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
 ldi r24 ,0x30
 out PORTD ,r24
 sbi PORTD ,PD3
 cbi PORTD ,PD3
 ldi r24 ,39
 ldi r25 ,0
 rcall wait_usec
 ldi r24 ,0x20
 out PORTD ,r24
 sbi PORTD ,PD3
 cbi PORTD ,PD3
 ldi r24 ,39
 ldi r25 ,0
 rcall wait_usec
 ldi r24 ,0x28
 rcall lcd_command
 ldi r24 ,0x0c
 rcall lcd_command
 ldi r24 ,0x01
 rcall lcd_command
 ldi r24 ,low(1530)
 ldi r25 ,high(1530)
 rcall wait_usec
 ldi r24 ,0x06
 rcall lcd_command
 ret
```

```
;ρουτίνα αποστολής ενός byte δεδομένων στην LCD
lcd_data:
 sbi PORTD ,PD2
 rcall write_2_nibbles
 ldi r24 ,43
 ldi r25 ,0
 rcall wait_usec
 ret

;ρουτίνα αποστολής μιας εντολής στην LCD
lcd_command:
 cbi PORTD ,PD2
 rcall write_2_nibbles
 ldi r24 ,39
 ldi r25 ,0
 rcall wait_usec
 ret

;ρουτίνα αποστολής ενός byte στην LCD
write_2_nibbles:
 push r24
 in r25 ,PIND
 andi r25 ,0x0f
 andi r24 ,0xf0
 add r24 ,r25
 out PORTD ,r24
 sbi PORTD ,PD3
 cbi PORTD ,PD3
 pop r24
 swap r24
 andi r24 ,0xf0
 add r24 ,r25
 out PORTD ,r24
 sbi PORTD ,PD3
 cbi PORTD ,PD3
 ret

;Ρουτίνες χρονοκαθυστέρησης
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
 brne wait_usec
 ret
```

Λύση 4.2.β(C)

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>


//Συναρτήσεις για πληκτρολόγιο (βλ προηγούμενη εργασητριακή)
char scan_row(int);
void scan_keypad(char *);
void scan_keypad_rising_edge(char*, char*);
int keypad_to_hex(char *);

//Συναρτήσεις οθόνης
void lcd_init (void);
void write_2_nibbles(char);
void lcd_data(char);
void lcd_command(char);

void main(void)
{
	SP = RAMEND;
	char prev[2],next[2],c,c1,c2,pressed[4];
	DDRD=0xFC;

	while(1)
	{
		char sign,metro=0,ekat=0,dec=0,mon=0;
		//Πληκτρολόγιο - Έξοδος
		DDRC=(1<<PC7)|(1<<PC6)|(1<<PC5)|(1<<PC4);
//Διάβασε πρώτο πλήκτρο
scan_keypad_rising_edge(prev,next);
		while((c=keypad_to_hex(next))==0x10)
		{
```

```
                scan_keypad_rising_edge(prev,next);
          }

            pressed[0]=c & 0x0f;
//Διάβασε δεύτερο πλήκτρο
            scan_keypad_rising_edge(prev,next);
            while((c=keypad_to_hex(next))==0x10)
            {
                scan_keypad_rising_edge(prev,next);
          }

            pressed[1]=c & 0x0f;
//Διάβασε τρίτο πλήκτρο

            scan_keypad_rising_edge(prev,next);
            while((c=keypad_to_hex(next))==0x10)
            {
                scan_keypad_rising_edge(prev,next);
          }

            pressed[2]=c & 0x0f;
//Διάβασε τέταρτο πλήκτρο
            scan_keypad_rising_edge(prev,next);
            while((c=keypad_to_hex(next))==0x10)
            {
                scan_keypad_rising_edge(prev,next);
          }

            pressed[3]=c & 0x0f;

//Τύπωσε αυτά που δίαβασες
//Για κάθε αριθμό
//Αν είναι χαρακτήρας (ABCDEF)                ->πρόσθεσε 37
//Αν δεν είναι χαρακτήρας (1234567890)        ->πρόσθεσε 30
//Για υπολογισμό ASCII


            lcd_init();
            if(pressed[0]>=10)
            {
                lcd_data(pressed[0]+0x37);
            }
                else
            {
                lcd_data(pressed[0]+0x30);
            }

            if(pressed[1]>=10)
```

```
            {
                  lcd_data(pressed[1]+0x37);
            }

            else
            {
                  lcd_data(pressed[1]+0x30);
            }

            if(pressed[2]>=10)
            {
                  lcd_data(pressed[2]+0x37);
            }

            else
            {
                  lcd_data(pressed[2]+0x30);
            }

            if(pressed[3]>=10)
            {
                  lcd_data(pressed[3]+0x37);
            }

            else
            {
                  lcd_data(pressed[3]+0x30);
            }
//Ως εδώ : Αριθμός είσόδου =>
            lcd_data('=');
            lcd_data('>');


            c1=(pressed[0]<<4)|pressed[1];
            c2=(pressed[2]<<4)|pressed[3];
//Αν δεν βρέθηκε αισθητήρας
            if(c1==0x80 && c2==0x00){
                  lcd_data('N');
                  lcd_data('O');
                  lcd_data(' ');
                  lcd_data('D');
                  lcd_data('e');
                  lcd_data('v');
                  lcd_data('i');
                  lcd_data('c');
                  lcd_data('e');
                  continue;
            }
```

```
//Αλλίως αν δόθηκε τιμή που δεν ανήκει στο αποδεκτό εύρος
//(55-125)
else if(c1!=0x00 && c1!=0xFF){
      lcd_data('E');
      lcd_data('R');
      lcd_data('R');
      lcd_data('O');
      lcd_data('R');
      continue;
}
//Ξεχωριστή περίπτωση που η τιμή είναι 0
if((c2==0) && (c1==0))
{
      lcd_data('0');
      lcd_data('°');
      lcd_data('C');

}
else
//Αν είναι αρνητικός
{
      if(c1 == 0xFF)
      {
            //Ξεχωριστή περίπτωση 0xFF00
            //Υπερχείληση
            if (c2 == 0x00)
            {
                  lcd_data('E');
                  lcd_data('R');
                  lcd_data('R');
                  lcd_data('O');
                  lcd_data('R');
                  continue;
            }
            sign='-';
            metro=~c2;
            metro=metro+0x01;

      }
      else
      //Αν είναι θετικός
      {
            sign='+';
            metro=c2;
      }
      //Αν ξεπερνά το αποδεκτό εύρος
      if(sign=='+' && metro>=251){
            lcd_data('E');
```

```
                lcd_data('R');
                lcd_data('R');
                lcd_data('O');
                lcd_data('R');
                continue;
        }
        else if(sign=='-' && metro>=111){
                lcd_data('E');
                lcd_data('R');
                lcd_data('R');
                lcd_data('O');
                lcd_data('R');
                continue;
        }
        //Στρογγυλοποίηση αν χρειάζεται
        if (metro%2==0) metro=metro/2;
        else metro=(metro/2)+1;




        //Υπολογισμός μονάδων - δεκάδων - εκατοντάδων
        //Με τη βοήθεια του διαγράμματος ροής της 3ης
        //Εργαστηριακής Άσκησης
        if(metro>=100)
        {
                ekat=1;
                metro=metro-100;
        }

        while(metro>=10)
        {
                dec++;
                metro=metro-10;
        }
        mon=metro;

        //Έλεγχος αν το αποτέλεσμα είναι
μονοψήφιο-διψήφιο-τριψήφιο

        lcd_data(sign);
        if(ekat==0)
        {
                if(dec==0)
                {
                        lcd_data((mon & 0x0f)+0x30);
                        lcd_data('°');
```

```
                        lcd_data('C');
                    }
                    else
                    {
                        lcd_data((dec & 0x0f)+0x30);
                        lcd_data((mon & 0x0f)+0x30);
                        lcd_data('°');
                        lcd_data('C');
                    }
                }
                else
                {
                    lcd_data((ekat & 0x0f)+0x30);
                    lcd_data((dec & 0x0f)+0x30);
                    lcd_data((mon & 0x0f)+0x30);
                    lcd_data('°');
                    lcd_data('C');
                }
            }
        }
}
//Υλοποιήσεις συναρτήσεων οθόνης - πληκτρολογίου
void write_2_nibbles(char x)
{
    char y=PIND & 0x0f;
    char x1=x & 0xf0;
    x1=x1+y;
    PORTD=x1;
    PORTD=PORTD | (1<<PD3);
    PORTD=PORTD & (0<<PD3);
    x=x<<4 | x>>4;
    x=x & 0xf0;
    PORTD=x+y;
    PORTD=PORTD | (1<<PD3);
    PORTD=PORTD & (0<<PD3);
}

void lcd_data(char x)
{
    PORTD=PORTD | (1<<PD2);
    write_2_nibbles(x);
    _delay_us(43);
}

void  lcd_command(char x)
{
    PORTD=PORTD | (0<<PD2);
    write_2_nibbles(x);
```

```
	_delay_us(43);
}

void lcd_init (void)
{
	_delay_ms(40);
	PORTD=0x30;
	PORTD=PORTD | (1<<PD3);
	PORTD=PORTD & (0<<PD3);
	_delay_us(39);
	PORTD=0x30;
	PORTD=PORTD | (1<<PD3);
	PORTD=PORTD & (0<<PD3);
	_delay_us(39);
	PORTD=0x20;
	PORTD=PORTD | (1<<PD3);
	PORTD=PORTD & (0<<PD3);
	_delay_us(39);
	lcd_command(0x28);
	lcd_command(0x0c);
	lcd_command(0x01);
	_delay_us(1530);
	lcd_command(0x06);
}


char scan_row(int row)
{
	char x=0x08,a;
	x=x<<row;
	PORTC =x;
	_delay_us(1);
	a=PINC & 0x0F;
	return a;
}

void scan_keypad(char next_st[2])
{
	next_st[0]=0x00;
	next_st[1]=0x00;

	char line=scan_row(1) & 0x0f;
	char temp = line<<4;
	next_st[0]=temp;

	line=scan_row(2) & 0x0f;
	next_st[0]=next_st[0]|line;
```

```
	line=scan_row(3) & 0x0f;
	temp = line<<4;
	next_st[1]=temp;

	line=scan_row(4) & 0x0f;
	next_st[1]=next_st[1]|line;

	return;
}

void scan_keypad_rising_edge(char prev_st[2], char next_st[2]) {

	scan_keypad(next_st);
	char temp[2];

	temp[0] = next_st[0];
	temp[1] = next_st[1];

	_delay_ms(15);

	scan_keypad(next_st);

	next_st[0] = next_st[0] & temp[0];
	next_st[1] = next_st[1] & temp[1];

	temp[0] = ~prev_st[0];
	temp[1] = ~prev_st[1];

	prev_st[0] = next_st[0];
	prev_st[1] = next_st[1];

	next_st[0] = next_st[0] & temp[0];
	next_st[1] = next_st[1] & temp[1];

	return;

}

int keypad_to_hex(char * keys)
{
	if ((keys[1]&0x01)==0x01) return 0x0E;
	if ((keys[1]&0x02)==0x02) return 0x00;
	if ((keys[1]&0x04)==0x04) return 0x0F;
	if ((keys[1]&0x08)==0x08) return 0x0D;
	if ((keys[1]&0x10)==0x10) return 0x07;
	if ((keys[1]&0x20)==0x20) return 0x08;
	if ((keys[1]&0x40)==0x40) return 0x09;
	if ((keys[1]&0x80)==0x80) return 0x0C;
```

```
        if ((keys[0]&0x01)==0x01) return 0x04;
        if ((keys[0]&0x02)==0x02) return 0x05;
        if ((keys[0]&0x04)==0x04) return 0x06;
        if ((keys[0]&0x08)==0x08) return 0x0B;
        if ((keys[0]&0x10)==0x10) return 0x01;
        if ((keys[0]&0x20)==0x20) return 0x02;
        if ((keys[0]&0x40)==0x40) return 0x03;
        if ((keys[0]&0x80)==0x80) return 0x0A;
        return 0x10;
}
```

ΤΕΛΟΣ ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΝΑΦΟΡΑΣ