Ιωάννης Γεωργακόπουλος - 03111512
Νικόλαος Μαρούσης - 03111025

Εργαστήριο Μικροϋπολογιστών

8η Εργαστηριακή Άσκηση

Άσκηση 1

Η άσκηση αφορά την κατασκευή της ρουτίνας **read_temp**, η οποία διαβάζει σειριακά την θερμοκρασία σε **συμπλήρωμα ως προς 2** από τον αισθητήρα, την απεικονίζει σε **συμπλήρωμα ως προς 1** στην θύρα εξόδου Α και την **επιστρέφει σε συμπλήρωμα ως προς 2** στους καταχωρητές r25:r24.

Καθώς η άσκηση 1 ζητά μόνο την κατασκευή της ρουτίνας, θα την παραθέσουμε μαζί με τον υπόλοιπο κώδικα της άσκησης 2.

Άσκηση 2

Το πρόγραμμα της άσκησης 2 μπορεί να τροποποιηθεί για να λειτουργεί με 2 (αμοιβαία αποκλειόμενους) τρόπους:

1. **Διάβασμα θερμοκρασίας από τον αισθητήρα**
   Σε αυτή την περίπτωση ανατρέχουμε στο σχόλιο ΤΜΗΜΑ 1 και κάνουμε uncomment το τμήμα κώδικα ανάμεσα στις οριζόντιες γραμμές. Για να αποφύγουμε την ασταθή εκτύπωση της εξόδου, έχουμε ρυθμίσει τη συχνότητα λήψης μετρήσεων στα 200msec.

2. **Διάβασμα θερμοκρασίας από το πληκτρολόγιο**
   Σε αυτή την περίπτωση ανατρέχουμε στο σχόλιο ΤΜΗΜΑ 2 και κάνουμε uncomment το τμήμα κώδικα ανάμεσα στις οριζόντιες γραμμές. Ο χρήστης εισάγει την θερμοκρασία σε **συμπλήρωμα ως προς 2**, ξεκινώντας από τα MSB της. Εδώ σημειώνουμε ότι δεν γίνεται έλεγχος για την ορθή είσοδο της επέκτασης προσήμου της θερμοκρασίας, αυτή βαραίνει τον χρήστη.

   Στη συνέχεια, η εφαρμογή τυπώνει την αντίστοιχη έξοδο στην οθόνη για 2 δευτερόλεπτα και ύστερα επιστρέφει σε λειτουργία αναμονής εισόδου.

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

.CSEG
.org 0x0
	rjmp reset

; Εισαγωγή βοηθητικών βιβλιοθηκών.
.INCLUDE "wait.asm"
.INCLUDE "one_wire.asm"
.INCLUDE "screen.asm"
.INCLUDE "hex_keyboard.asm"

.def temp = r18
.def first = r19

reset:
	; Αρχικοποίηση μεταβλητής _tmp_ .
	ldi temp, 0
	ldi r26, low(_tmp_)
	ldi r27, high(_tmp_)
	st X+, temp
	st X, temp

	; Δημιουργία της Στοίβας.
	ldi r24, LOW(RAMEND)
	out SPL, r24
	ldi r25, HIGH(RAMEND)
	out SPH, r25

	; Αρχικοποίηση της PORTC για το πληκτρολόγιο.
	ldi temp, 0xf0
	out DDRC, temp

	; Αρχικοποίηση της PORTD για την οθόνη LCD.
	ldi temp, 0xff
	out DDRD, temp
	out DDRB, temp

	; Αρχικοποίηση οθόνης.
	rcall lcd_init
```

```
; ΤΜΗΜΑ 1
; Εκτύπωση θερμοκρασίας από τον αισθητήρα.
; Για ενεργοποίηση uncomment από εδώ
; ------------------------------------------------------------------------- ;
start:
      rcall read_temp                ; Ανάγνωση θερμοκρασίας από τον
      rcall print_some               ; αισθητήρα και εκτύπωση της.
      ldi r24, low(200)
      ldi r25, high(200)             ; Αναμονή 200msec μέχρι την επόμενη
      rcall wait_msec                ; μέτρηση.
      rjmp start
; ------------------------------------------------------------------------- ;
; εως εδώ.


; ΤΜΗΜΑ 2
; Εκτύπωση θερμοκρασίας από το πληκτρολόγιο.
; Για ενεργοποίηση uncomment από εδώ
; ------------------------------------------------------------------------- ;
; start:
;     ldi r20, 0
;     ldi r21, 0
;
;     ldi r24, 0x01                  ; Καθαρισμός της οθόνης.
;     rcall lcd_command
;     ldi r24, low(1530)
;     ldi r25, high(1530)
;     rcall wait_usec
;
;     rcall read_hex                 ; Ανάγνωση των 2 MSB και αποθήκευσή τους
;     mov r21, r24                   ; στον καταχωρητή r21.
;     swap r21
;     rcall read_hex
;     add r21, r24
;     rcall read_hex                 ; Ανάγνωση των 2 LSB και αποθήκευσή τους
;     mov r20, r24                   ; στον καταχωρητή r20.
;     swap r20
;     rcall read_hex
;     add r20, r24
;
;     mov r24, r20                   ; r25:r24 = r21:r20.
;     mov r25, r21
;
;     rcall print_some               ; Εκτύπωση της θερμοκρασίας
;
;     ldi r24, low(2000)             ; και επιστροφή σε λειτουργία εισόδου
;     ldi r25, high(2000)            ; μετά από 2 δευτερόλεπτα.
;     rcall wait_msec
;
;     rjmp start
; ------------------------------------------------------------------------- ;
; εως εδώ.
```

```
; Βοηθητική συνάρτηση που διαβάζει ένα ASCII δεκαεξαδικό ψηφίο και επιστρέφει
; στον r24 την τιμή του.
read_hex:
	ldi r24, 10
	rcall scan_keypad_rising_edge	; Διάβασμα από το πληκτρολόγιο.
	rcall keypad_to_hex		; Μετατροπή σε ASCII hex ψηφίο.
	cpi r24, 0
	breq read_hex			; Επανάληψη όσο δεν διαβάστηκε κάτι.
	subi r24, '0'
	cpi r24, 10
	brlo finished
	subi r24, 7			; Για ψηφία 'A'-'F' απαιτείται επιπλέον
					; επεξεργασία
finished:
	ret

; Ρουτίνα που επιστρέφει την μετρούμενη θερμοκρασία στο ζεύγος r25:r24
; σε δυαδική μορφή συμπληρώματος ως προς 1. Αν εμφανιστεί οποιοδήποτε σφάλμα,
; επιστρέφουμε την τιμή 8000.
read_temp:
	rcall one_wire_reset		; Αρχικοποίηση συσκευής.
	sbrs r24, 0			; Αν δε βρεθεί συσκευή (r24=0),
	rjmp no_device			; επιστρέφουμε με τιμή 8000.

	ldi r24, 0xCC			; Αγνοούμε τον έλεγχο για πολλές συσκευές.
	rcall one_wire_transmit_byte
	ldi r24, 0x44			; Ζητάμε να ξεκινήσει η μέτρηση της
	rcall one_wire_transmit_byte	; θερμοκρασίας.

check_finished:
	rcall one_wire_receive_bit	; Ελέγχουμε αν έχει τελειώσει η μετατροπή
	sbrs r24, 0			; της θερμοκρασίας (r24=1), αλλιώς
	rjmp check_finished		; περιμένουμε.

	rcall one_wire_reset		; Αρχικοποιούμε και πάλι τη συσκευή, γιατί
					; μετά τη μέτρηση επανέρχεται σε κατάσταση
					; χαμηλής κατανάλωσης ισχύος.
	sbrs r24, 0			; Αν αποσυνδέθηκε επιστρέφουμε με τιμή 8000.
	rjmp no_device

	ldi r24, 0xCC			; Αγνοούμε τον έλεγχο για πολλές συσκευές.
	rcall one_wire_transmit_byte
	ldi r24, 0xBE			; Ζητάμε να γίνει ανάγνωση.
	rcall one_wire_transmit_byte


	rcall one_wire_receive_byte	; Διαβάζουμε τα 2 bytes της θερμοκρασίας.
	push r24
	rcall one_wire_receive_byte
	mov r25, r24			; Στο τέλος έχουμε r25:r24 = high:low
	pop r24
```

```
	mov first, r24			; Μεταφέρουμε στον first την τιμή της
	sbrc r25, 0			; θερμοκρασίας
	dec first			; Τη μετατρέπουμε σε συμπλήρωμα ως προς 1,
					; αν το πρόσημό της είναι αρνητικό.
	ser temp			; Η θύρα Α τίθεται ως θύρα εξόδου
	out DDRA, temp

	out PORTA, first		; Εκτύπωση αποτελέσματος σε συμπλήρωμα ως
					; προς 1 στην PORTA.
	ret				; Επιστροφή της θερμοκρασίας στους r25:r24.

no_device:				; Επιστροφή με r25:r24 = 0x8000 σε περίπτωση
	ldi r25, 0x80			; σφάλματος.
	ldi r24, 0x00
	ret

; Βοηθητική ρουτίνα που δέχεται μια τιμή θερμοκρασίας στους καταχωρητές r25:r24
; και την τυπώνει κατάλληλα.
print_some:
	cpi r25, 0x80			; Έλεγχος αν r25:r24 = 0x8000...
	brne ok
	cpi r24, 0x00
	brne ok
	rcall print_error		; και εκτύπωση μηνύματος σφάλματος, σε
	ret				; τέτοια περίπτωση.
ok:
	rcall print_temp		; Αλλιώς εκτύπωση θερμοκρασίας
	ret

; Βοηθητική συνάρτηση εκτύπωσης μηνύματος σφάλματος.
print_error:
	ldi r24, 0x01			; Καθαρισμός της οθόνης.
	rcall lcd_command
	ldi r24, low(1530)
	ldi r25, high(1530)
	rcall wait_usec
	ldi r24, 'N'			; Τύπωμα "NO Device".
	rcall lcd_data
	ldi r24, 'O'
	rcall lcd_data
	ldi r24, ' '
	rcall lcd_data
	ldi r24, 'D'
	rcall lcd_data
	ldi r24, 'e'
	rcall lcd_data
	ldi r24, 'v'
	rcall lcd_data
	ldi r24, 'i'
	rcall lcd_data
	ldi r24, 'c'
	rcall lcd_data
```

```
	ldi r24, 'e'
	rcall lcd_data
	ret

; Βοηθητική συνάρτηση εκτύπωσης θερμοκρασίας.
print_temp:
	mov temp, r24			; Ο temp περιέχει την τιμή της θερμοκρασίας.
	push r25

	ldi r24, 0x0c			; Ενεργοποίηση της οθόνης, χωρίς κέρσορα
	rcall lcd_command
	ldi r24, 0x01			; Καθαρισμός της οθόνης.
	rcall lcd_command
	ldi r24, low(1530)
	ldi r25, high(1530)
	rcall wait_usec

	pop r25				; Επαναφορά του r25...
	ldi r24, '+'
	sbrc r25, 0
	ldi r24, '-'
	push r25
	rcall lcd_data			; και εκτύπωση προσήμου

	pop r25				; Επαναφορά του r25.
	sbrc r25, 0			; Αν ο αριθμός είναι αρνητικός (r25=0xff),
	dec temp			; τότε υπολογίζουμε το συμπλήρωμά του ως
	sbrc r25, 0			; προς 2.
	com temp

	lsr temp			; Αγνοούμε το κλασματικό ψηφίο. Αν θέλαμε
					; να το λάβουμε υπόψιν, θα μπορούσαμε να
					; θέσουμε ένα flag βάσει του carry μετά το
					; right shift.

	ldi first, 0			; Flag που συμβολίζει αν έχει τυπωθεί το
	ldi r24, 0			; πρώτο ψηφίο που μπορεί να τυπώθει.

	cpi temp, 100			; Εκτύπωση εκατοντάδων.
	brlo decades
	ldi first, 1
	ldi r24, '1'
	rcall lcd_data
	subi temp, 100

	ldi r24, 0			; Αρχικοποίηση μετρητή δεκάδων.
decades:
	cpi temp, 10			; Υπολογισμός δεκάδων.
	brlo print_dec
	inc r24
	subi temp, 10
	rjmp decades
```

```
print_dec:                          ; Εκτύπωση δεκάδων,
      cpi r24, 0                    ; εκτός αν
      sbrs first, 0                 ; δεν έχει τυπωθεί το πρώτο ψηφίο
      breq digit                    ; και οι δεκάδες είναι ίσες με μηδέν.
      subi r24, -48                 ; ASCIIοποίηση δεκάδων
      rcall lcd_data                ; και εκτ΄ύπωσή τους.
digit:
      mov r24, temp
      subi r24, -48                 ; ASCIIοποίηση μονάδων
      rcall lcd_data                ; και εκτύπωσή τους.

      ldi r24, 0xb2                 ; Εκτύπωση ο
      rcall lcd_data
      ldi r24, 'C'                  ; Εκτύπωση C
      rcall lcd_data

      ret
```

Στη συνέχεια παραθέτουμε τις βοηθητικές βιβλιοθήκες που χρησιμοποιήσαμε.

wait.asm

```
;; ΑΡΧΗ ΡΟΥΤΙΝΩΝ ΧΡΟΝΟΚΑΘΥΣΤΕΡΗΣΗΣ ;;
;; Προκαλεί καθυστέρηση r25:r24 msec.
wait_msec:
      push r24                      ; 2 κύκλοι (0.250 μsec)
      push r25                      ; 2 κύκλοι
      ldi r24,  low(998)            ; φόρτωσε τον καταχ. r25:r24 με 998
                                    ; (1 κύκλος - 0.125 μsec)
      ldi r25,  high(998)           ; 1 κύκλος (0.125 μsec)
      rcall wait_usec               ; 3 κύκλοι (0.375 μsec), προκαλεί συνολικά
                                    ; καθυστέρηση 998.375 μsec
      pop r25                       ; 2 κύκλοι (0.250 μsec)
      pop r24                       ; 2 κύκλοι
      sbiw r24,  1                  ; 2 κύκλοι
      brne wait_msec                ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
      ret                           ; 4 κύκλοι (0.500 μsec)

;; Προκαλεί καθυστέρηση r25:r24 μsec.
wait_usec:
      sbiw r24, 1                   ; 2 κύκλοι (0.250 μsec)
      nop                           ; 1 κύκλος (0.125 μsec)
      nop                           ; 1 κύκλος (0.125 μsec)
      nop                           ; 1 κύκλος (0.125 μsec)
      nop                           ; 1 κύκλος (0.125 μsec)
      brne wait_usec                ; 1 ή 2 κύκλοι (0.125 ή 0.250 μsec)
      ret                           ; 4 κύκλοι (0.500 μsec)
;; ΤΕΛΟΣ ΡΟΥΤΙΝΩΝ ΧΡΟΝΟΚΑΘΥΣΤΕΡΗΣΗΣ ;;;; ΤΕΛΟΣ ΡΟΥΤΙΝΩΝ ΧΡΟΝΟΚΑΘΥΣΤΕΡΗΣΗΣ ;;
```

one_wire.asm

```
; File Name: one_wire.asm
; Title: one wire protocol
; Target mcu: atmega16
; Development board: easyAVR6
; Assembler: AVRStudio assembler
; Description:

; This file includes routines implementing the one wire protocol over the PA4 pin
; of the microcontroller.
; Dependencies: wait.asm

; Routine: one_wire_receive_byte
; Description:
; This routine generates the necessary read
; time slots to receives a byte from the wire.
; return value: the received byte is returned in r24.
; registers affected: r27:r26 ,r25:r24
; routines called: one_wire_receive_bit
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

; Routine: one_wire_receive_bit
; Description:
; This routine generates a read time slot across the wire.
; return value: The bit read is stored in the lsb of r24.
; if 0 is read or 1 if 1 is read.
; registers affected: r25:r24
; routines called: wait_usec
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

; Routine: one_wire_transmit_byte
; Description:
; This routine transmits a byte across the wire.
; parameters:
; r24: the byte to be transmitted must be stored here.
; return value: None.
; registers affected: r27:r26 ,r25:r24
; routines called: one_wire_transmit_bit
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

; Routine: one_wire_transmit_bit
; Description:
; This routine transmits a bit across the wire.
; parameters:
; r24: if we want to transmit 1
; then r24 should be 1, else r24 should
; be cleared to transmit 0.
; return value: None.
; registers affected: r25:r24
; routines called: wait_usec
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

; Routine: one_wire_reset
; Description:
; This routine transmits a reset pulse across the wire
; and detects any connected devices.
; parameters: None.
; return value: 1 is stored in r24
; if a device is detected, or 0 else.
; registers affected r25:r24
; routines called: wait_usec
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
```

screen.asm

```
;; ΑΡΧΗ ΡΟΥΤΙΝΩΝ ΟΘΟΝΗΣ ;;
write_2_nibbles:
	push r24 			; στέλνει τα 4 MSB
	in r25, PIND 			; διαβάζονται τα 4 LSB και τα ξαναστέλνουμε
	andi r25, 0x0f 			; για να μην χαλάσουμε την όποια προηγούμενη
	andi r24, 0xf0 			; κατάσταση απομονώνονται τα 4 MSB και
	add r24, r25 			; συνδυάζονται με τα προϋπάρχοντα 4 LSB
	out PORTD, r24 			; και δίνονται στην έξοδο
	sbi PORTD, PD3 			; δημιουργείται παλμός Enable στον ακροδέκτη
	cbi PORTD, PD3 			; PD3, PD3=1 και μετά PD3=0
	pop r24 			; στέλνει τα 4 LSB. Ανακτάται το byte.
	swap r24 			; εναλλάσσονται τα 4 MSB με τα 4 LSB
	andi r24, 0xf0 			; που με την σειρά τους αποστέλλονται
	add r24, r25
	out PORTD, r24
	sbi PORTD, PD3 			; Νέος παλμός Enable
	cbi PORTD, PD3
	ret

lcd_data:
	sbi PORTD, PD2 			; επιλογή του καταχωρήτη δεδομένων (PD2=1)
	rcall write_2_nibbles 		; αποστολή του byte
	ldi r24, 43 			; αναμονή 43μsec μέχρι να ολοκληρωθεί η λήψη
	ldi r25, 0 			; των δεδομένων από τον ελεγκτή της lcd
	rcall wait_usec
	ret

lcd_command:
	cbi PORTD, PD2 			; επιλογή του καταχωρητή εντολών (PD2=0)
	rcall write_2_nibbles 		; αποστολή της εντολής και αναμονή 39μsec
	ldi r24, 39 			; για την ολοκλήρωση της εκτέλεσης της από
					; τον ελεγκτή της lcd.
	ldi r25, 0 			; ΣΗΜ.: υπάρχουν δύο εντολές, οι clear
					; display και return home,
	rcall wait_usec 		; που απαιτούν σημαντικά μεγαλύτερο χρονικό
					; διάστημα.
	ret

lcd_init:
	ldi r24, 40 			; Όταν ο ελεγκτής της lcd τροφοδοτείται με
	ldi r25, 0 			; ρεύμα εκτελεί την δική του αρχικοποίηση.
	rcall wait_msec 		; Αναμονή 40 msec μέχρι αυτή να ολοκληρωθεί.
	ldi r24, 0x30 			; εντολή μετάβασης σε 8 bit mode
	out PORTD, r24 			; επειδή δεν μπορούμε να είμαστε βέβαιοι
	sbi PORTD, PD3 			; για τη διαμόρφωση εισόδου του ελεγκτή
	cbi PORTD, PD3 			; της οθόνης, η εντολή αποστέλλεται δύο φορές
	ldi r24, 39
	ldi r25, 0 			; εάν ο ελεγκτής της οθόνης βρίσκεται σε
	rcall wait_usec 		; 8-bit mode δεν θα συμβεί τίποτα, αλλά αν ο
```

```
                                        ; ελεγκτής έχει διαμόρφωση εισόδου 4 bit
                                        ; θα μεταβεί σε διαμόρφωση 8 bit
        ldi r24, 0x30
        out PORTD, r24
        sbi PORTD, PD3
        cbi PORTD, PD3
        ldi r24, 39
        ldi r25, 0
        rcall wait_usec
        ldi r24, 0x20                   ; αλλαγή σε 4-bit mode
        out PORTD, r24
        sbi PORTD, PD3
        cbi PORTD, PD3
        ldi r24, 39
        ldi r25, 0
        rcall wait_usec
        ldi r24, 0x28                   ; επιλογή χαρακτήρων μεγέθους 5x8 κουκίδων
        rcall lcd_command               ; και εμφάνιση δύο γραμμών στην οθόνη
        ldi r24, 0x0e                   ; ενεργοποίηση της οθόνης, εμφάνιση του
        rcall lcd_command               ; κέρσορα
        ldi r24, 0x01                   ; καθαρισμός της οθόνης
        rcall lcd_command
        ldi r24, low(1530)
        ldi r25, high(1530)
        rcall wait_usec
        ldi r24, 0x06                   ; ενεργοποίηση αυτόματης αύξησης κατά 1 της
        rcall lcd_command               ; διεύθυνσης που είναι αποθηκευμένη στον
                                        ; μετρητή διευθύνσεων και απενεργοποίηση της
                                        ; ολίσθησης ολόκληρης της οθόνης
        ret
;; ΤΕΛΟΣ ΡΟΥΤΙΝΩΝ ΟΘΟΝΗΣ ;;
```

hex_keyboard

```
;; ΑΡΧΗ ΡΟΥΤΙΝΩΝ ΠΛΗΚΤΡΟΛΟΓΙΟΥ ;;
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
      nop                            ; καθυστέρηση για να προλάβει να γίνει η
                                     ; αλλαγή κατάστασης
      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.


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

scan_keypad_rising_edge:
      mov r22, r24                   ; αποθήκευσε το χρόνο σπινθηρισμού στον r22
      rcall scan_keypad              ; έλεγξε το πληκτρολόγιο για πιεσμένους
      push r24                       ; διακόπτες και αποθήκευσε το αποτέλεσμα
      push r25
      mov r24, r22                   ; καθυστέρησε r22 ms(τυπικές τιμές 10-20 msec
      ldi r25, 0                     ; που καθορίζεται από τον κατασκευαστή του
                                     ; πληκτρολογίου – χρονοδιάρκεια σπινθηρισμών)
      rcall wait_msec
      rcall scan_keypad              ; έλεγξε το πληκτρολόγιο ξανά και
      pop r23                        ; απόρριψε όσα πλήκτρα εμφανίζουν
      pop r22                        ; σπινθηρισμό
```

```
       and r24, r22
       and r25, r23
       ldi r26, low(_tmp_)          ; φόρτωσε την κατάσταση των διακοπτών στην
       ldi r27, high(_tmp_)         ; προηγούμενη κλήση της ρουτίνας στους
                                    ; r27:r26
       ld r23, X+
       ld r22, X
       st X, r24                    ; αποθήκευσε στη RAM τη νέα κατάσταση
       st -X, r25                   ; των διακοπτών
       com r23
       com r22                      ; βρες τους διακόπτες που έχουν «μόλις»
       and r24, r22                 ; πατηθεί
       and r25, r23
       ret

keypad_to_hex:                      ; λογικό ‘1’ στις θέσεις του καταχωρητή r26
       movw r26, r24                ; δηλώνουν τα παρακάτω σύμβολα και αριθμούς
       ldi r24, 'E'                 ; Τροποποίηση: * -> E
       sbrc r26, 0
       ret
       ldi r24, '0'
       sbrc r26, 1
       ret
       ldi r24, 'F'                 ; Τροποποίηση: # -> F
       sbrc r26, 2
       ret
       ldi r24, 'D'
       sbrc r26, 3                  ; αν δεν είναι ‘1’παρακάμπτει την ret, αλλιώς
       ret                          ; (αν είναι ‘1’) επιστρέφει με τον καταχωρητή
                                    ; r24 την ASCII τιμή του D.
       ldi r24, '7'
       sbrc r26, 4
       ret
       ldi r24, '8'
       sbrc r26, 5
       ret
       ldi r24, '9'
       sbrc r26, 6
       ret
       ldi r24, 'C'
       sbrc r26, 7
       ret
       ldi r24, '4'                 ; λογικό ‘1’ στις θέσεις του καταχωρητή r27
       sbrc r27, 0                  ; δηλώνουν τα παρακάτω σύμβολα και αριθμούς
       ret
       ldi r24, '5'
       sbrc r27, 1
       ret
       ldi r24, '6'
       sbrc r27, 2
       ret
       ldi r24, 'B'
```

```
        sbrc r27, 3
        ret
        ldi r24, '1'
        sbrc r27, 4
        ret
        ldi r24, '2'
        sbrc r27, 5
        ret
        ldi r24, '3'
        sbrc r27, 6
        ret
        ldi r24, 'A'
        sbrc r27, 7
        ret
        clr r24
        ret
;; ΤΕΛΟΣ ΡΟΥΤΙΝΩΝ ΠΛΗΚΤΡΟΛΟΓΙΟΥ ;;
```