Εργαστήριο Μικροϋπολογιστών,7ο Εξάμηνο

Βικέντιος Βιτάλης el18803

Αναφορά 4ης εργαστηριακής άσκησης

Ομάδα 03

Ζήτημα 4.1

Στον κώδικα χρησιμοποιήθηκαν δύο διακοπές. Η διακοπή υπερχείλισης του Timer1, ώστε να διαβάζεται η τιμή του αισθητήρα CO κάθε 100 msec, και η διακοπή του ADC, ο οποίος ενεργοποιείται κάθε 100 msec (δηλαδή μόλις υπερχειλίσει ο timer1) ώστε να ξεκινήσει μία μετατροπή του αναλογικού σήματος του αισθητήρα CO σε ψηφιακό. Η ψηφιακή τιμή θα μπορεί πλέον να διαβαστεί και να αξιοποιηθεί από το μικροελεγκτή μας. Τόσο στην assembly όσο και στη C, τα led επιλέχθηκαν να ανάβουν με τον εξής τρόπο:

```
; Συγκέντρωση αερίου -> Ψηφιακή τιμή από τον ADC -> Led που πρέπει να ανάψουν
; CCO = 0-35    ppm   -> ADC = 21 - 113          -> *
; CCO = 35-50   ppm   -> ADC = 113 - 153         -> **
; CCO = 50-60   ppm   -> ADC = 153 - 179         -> ***
; CCO = 60-70   ppm   -> ADC = 179 - 205         -> ****
; CCO = 70-85   ppm   -> ADC = 205 - 245         -> *****
; CCO = 85-105  ppm   -> ADC = 245 - 298         -> ******
; CCO = 105-150 ppm   -> ADC = 298 - 417         -> *******
; Χρήση του τύπου από το Data_Sheet του αισθητήρα
; Cx = 1/129*10^4 * (Vgas-Vgas0) = 129*Cx*10^(-4) + Vgas0 = Vgas

.INCLUDE "m16def.inc"


        ; Αρχή τμήματος δεδομένων
        .DSEG
        _tmp_: .byte 2

        ; Τέλος τμήματος δεδομένων
        .CSEG
```

```
.equ startime = 0xFCF3      ; Ορίζουμε την αρχική τιμή από την οποία ξεκινάει η μέτρηση
; του χρονιστή,
; ώστε η υπερχείλισή του να επιτευχθεί μόλις παρέλθουν 100 msec (επιλέγεται συχνότητα
; αύξησης του
; χρονιστή ίση με clock/1024 = 8MHz/1024 = 7812.5Hz, άρα για να παρέλθουν 100msec =
; 0,1 sec
; πρέπει να μετρήσουμε 7812,5*0,1 = 781,25 = 782 αυξήσεις). Ο χρονιστής υπερχειλίζει
; στην τιμή 65536,
; άρα πρέπει η τιμή εκκίνησης να είναι 65536 – 782 = 64754 = 0xFCF3.

        .equ hightime = high(startime) ; Το high μέρος της τιμής
        .equ lowtime = low(startime) ; Το low μέρος της τιμής
        .org 0
        rjmp start
        .org 0x10     ; θέση μνήμης για τη ρουτίνα εξυπηρέτησης διακοπής υπεχείλισης του
        ; Timer1

        rjmp timer1_isr
        .org 0x1c; θέση μνήμης για τη ρουτίνα εξυπηρέτησης διακοπής ADC

        rjmp adc_r

start:


        ldi r24, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4) ; θέτει ως εξόδους
        ; τα 4 MSB
        out DDRC, r24 ; της θύρας PORTC

        ser r16       ; θέτουμε τη θύρα B ως έξοδο (για το χειρισμό των led)
        out DDRB, r16

        ser r24       ; θέτουμε τη θύρα D ως έξοδο (για το χειρισμό της οθόνης)
        out DDRD, r24

        ldi r24, low(RAMEND) ; Αρχικοποίηση του stack pointer
        out SPL, r24
        ldi r24, high(RAMEND)
        out SPH, r24

        ldi r16,(1<<REFS0)          ; Χρήση της Vcc = 5V ως Vref
        out ADMUX,r16               ; MUX4:0 = 00000 ώστε ο ADC  να διαβάσει την
        ; αναλογική τιμή από τη θύρα A0.

        ldi r24 ,(1<<TOIE1)         ; Ενεργοποίηση διακοπής υπερχείλισης του μετρητή
        ; TCNT1
        out TIMSK ,r24

        ldi r24 ,(1<<CS12) | (0<<CS11) | (1<<CS10) ; Αρχικοποίηση για CK/1024
        out TCCR1B ,r24

        ldi r24, hightime           ; Αρχικοποίηση του TCNT1 για υπερχείλιση μετά από
        ; 0.1sec
        out TCNT1H, r24

        ldi r24, lowtime
        out TCNT1L, r24

        sei                         ; Ενεργοποίηση διακοπών
```

```
        rjmp main

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)

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
                                                ; 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)

scan_row_sim:
        out PORTC, r25                  ; Η αντίστοιχη γραμμή τίθεται στο λογικό ‘1’
        push r24                        ; Τμήμα κώδικα που προστίθεται για τη σωστή
        push r25                        ; λειτουργία του προγραμματος απομακρυσμένης
        ldi r24,low(500)                ; πρόσβασης
        ldi r25,high(500)
        rcall wait_usec
        pop r25
        pop r24                         ; Τέλος τμήμα κώδικα
        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_sim:
        push r26                        ; Αποθήκευσε τους καταχωρητές r27:r26 γιατι
        ; τους
        push r27                        ; αλλάζουμε μέσα στην ρουτίνα
        ldi r25 , 0x10                  ; Έλεγξε την πρώτη γραμμή του πληκτρολογίου
        ; (PC4: 1 2 3 A)
        rcall scan_row_sim
        swap r24                ; Αποθήκευσε το αποτέλεσμα
        mov r27, r24            ; στα 4 msb του r27
        ldi r25 ,0x20           ; Έλεγξε τη δεύτερη γραμμή του πληκτρολογίου (PC5: 4
        ; 5 6 B)
        rcall scan_row_sim
        add r27, r24            ; Αποθήκευσε το αποτέλεσμα στα 4 lsb του r27
        ldi r25 , 0x40          ; Έλεγξε την τρίτη γραμμή του πληκτρολογίου
        ; (PC6: 7 8 9 C)
        rcall scan_row_sim
        swap r24                ; Αποθήκευσε το αποτέλεσμα
        mov r26, r24            ; στα 4 msb του r26
```

```
        ldi r25 ,0x80               ; Έλεγξε την τέταρτη γραμμή του πληκτρολογίου (PC7:
        ; * 0 # D)
        rcall scan_row_sim
        add r26, r24                ; Αποθήκευσε το αποτέλεσμα στα 4 lsb του r26
        movw r24, r26               ; Μετέφερε το αποτέλεσμα στους καταχωρητές r25:r24
        clr r26                     ; Προστέθηκε για την
        out PORTC,r26               ; απομακρυσμένη πρόσβαση
        pop r27                     ; Επανάφερε τους καταχωρητές r27:r26
        pop r26
        ret

scan_keypad_rising_edge_sim:
        push r22                ; Αποθήκευσε τους καταχωρητές r23:r22 και τους
        push r23
        push r26                ; r26:r27 γιατι τους αλλάζουμε μέσα στην ρουτίνα
        push r27
        rcall scan_keypad_sim    ; Έλεγξε το πληκτρολόγιο για πιεσμένους διακόπτες
        push r24                ; κι αποθήκευσε το αποτέλεσμα
        push r25
        ldi r24 ,15             ; Καθυστέρησε 15 ms (τυπικές τιμές 10-20 msec
                                ; που καθορίζεται από τον
        ldi r25 ,0              ; κατασκευαστή του πληκτρολογίου -
                                ; χρονοδιάρκεια σπινθηρισμών)
        rcall wait_msec
        rcall scan_keypad_sim       ; Έλεγξε το πληκτρολόγιο ξανά κι απόρριψε
        pop r23                     ; όσα πλήκτρα εμφανίζουν σπινθηρισμό
        pop r22
        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» πατηθεί
        and r24 ,r22
        and r25 ,r23
        pop r27             ; Επανάφερε τους καταχωρητές r27:r26
        pop r26             ; και r23:r22
        pop r23
        pop r22
        ret


keypad_to_ascii_sim:
        push r26            ; Αποθήκευσε τους καταχωρητές r27:r26 γιατί τους
        push r27            ; αλλάζουμε μέσα στη ρουτίνα
        movw r26 ,r24       ; Λογικό ‘1’ στις θέσεις του καταχωρητή r26 δηλώνουν
        ; τα παρακάτω σύμβολα και αριθμούς
        ldi r24 ,'*'
        ; r26
        ; C 9 8 7 D # 0 *
        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   ; Αν δεν είναι ‘1’ παρακάμπτει την ret, αλλιώς (αν είναι ‘1’)
        rjmp return_ascii ; επιστρέφει με τον καταχωρητή r24 την ASCII τιμή του D.
        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'          ; Λογικό ‘1’ στις θέσεις του καταχωρητή r27 δηλώνουν
        sbrc r27 ,0           ; τα παρακάτω σύμβολα κι αριθμούς
        rjmp return_ascii
        ldi r24 ,'5'
        ; r27
        ; A 3 2 1 B 6 5 4
        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       ; Επανάφερε τους καταχωρητές r27:r26
        pop r26
        ret

write_2_nibbles_sim:
        push r24              ; Τμήμα κώδικα που προστίθεται για τη σωστή
        push r25              ; λειτουργία του προγραμματος απομακρυσμένης
        ldi r24 ,low(6000)    ; πρόσβασης
        ldi r25 ,high(6000)
        rcall wait_usec
        pop r25
        pop r24               ; Τέλος τμήμα κώδικα
        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 κατάσταση
        andi r24, 0xf0        ; Απομονώνονται τα 4 MSB,
```

```
        add r24, r25          ; συνδυάζονται με τα προϋπάρχοντα 4 LSB
        out PORTD, r24        ; και δίνονται στην έξοδο
        sbi PORTD, PD3        ; Δημιουργείται παλμός Enable στον ακροδέκτη PD3
        cbi PORTD, PD3        ; PD3 = 1 και μετά PD3 = 0
        push r24              ; Τμήμα κώδικα που προστίθεται για τη σωστή
        push r25              ; λειτουργία του προγραμματος απομακρυσμένης
        ldi r24 ,low(6000)    ; πρόσβασης
        ldi r25 ,high(6000)
        rcall wait_usec
        pop r25
        pop r24               ; Τέλος τμήμα κώδικα
        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_sim:
        push r24              ; Αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
        push r25              ; αλλάζουμε μέσα στη ρουτίνα
        sbi PORTD, PD2        ; Επιλογή του καταχωρητή δεδομένων (PD2 = 1)
        rcall write_2_nibbles_sim ; 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
        pop r25               ; Επανάφερε τους καταχωρητές r25:r24
        pop r24
        ret

lcd_command_sim:
        push r24              ; Αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
        push r25              ; αλλάζουμε μέσα στη ρουτίνα
        cbi PORTD, PD2        ; Επιλογή του καταχωρητή εντολών (PD2 = 0)
        rcall write_2_nibbles_sim ; Αποστολή της εντολής κι 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.
        pop r25               ; επανάφερε τους καταχωρητές r25:r24
        pop r24
        ret

lcd_init_sim:
        push r24 ; Αποθήκευσε τους καταχωρητές r25:r24 γιατί τους
        push r25 ; αλλάζουμε μέσα στη ρουτίνα

        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 8-bit mode
        rcall wait_usec       ; δεν θα συμβεί τίποτα, αλλά αν ο ελεγκτής έχει διαμόρφωση
       ; εισόδου 4 bit θα μεταβεί σε διαμόρφωση 8 bit
        push r24              ; Τμήμα κώδικα που προστίθεται για τη σωστή
        push r25              ; λειτουργία του προγραμματος απομακρυσμένης
```

```
        ldi r24,low(1000)    ; πρόσβασης
        ldi r25,high(1000)
        rcall wait_usec
        pop r25
        pop r24              ; Τέλος τμήμα κώδικα
        ldi r24, 0x30
        out PORTD, r24
        sbi PORTD, PD3
        cbi PORTD, PD3
        ldi r24,39
        ldi r25,0
        rcall wait_usec
        push r24             ; Τμήμα κώδικα που προστίθεται για τη σωστή
        push r25             ; λειτουργία του προγραμματος απομακρυσμένης
        ldi r24 ,low(1000)   ; πρόσβασης
        ldi r25 ,high(1000)
        rcall wait_usec
        pop r25
        pop r24              ; Τέλος τμήμα κώδικα
        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
        push r24              ; Τμήμα κώδικα που προστίθεται για τη σωστή
        push r25              ; λειτουργία του προγραμματος απομακρυσμένης
        ldi r24 ,low(1000)    ; πρόσβασης
        ldi r25 ,high(1000)
        rcall wait_usec
        pop r25
        pop r24               ; Τέλος τμήμα κώδικα
        ldi r24,0x28          ; Επιλογή χαρακτήρων μεγέθους 5x8 κουκίδων
        rcall lcd_command_sim ; κι εμφάνιση δύο γραμμών στην οθόνη
        ldi r24,0x0c          ; Ενεργοποίηση της οθόνης, απόκρυψη του κέρσορα
        rcall lcd_command_sim
        ldi r24,0x01          ; Καθαρισμός της οθόνης
        rcall lcd_command_sim
        ldi r24, low(1530)
        ldi r25, high(1530)
        rcall wait_usec
        ldi r24 ,0x06        ; Ενεργοποίηση αυτόματης αύξησης κατά 1 της διεύθυνσης
        rcall lcd_command_sim ; που είναι αποθηκευμένη στον μετρητή διευθύνσεων κι
        ; απενεργοποίηση της ολίσθησης ολόκληρης της οθόνης
        pop r25              ; Επανάφερε τους καταχωρητές r25:r24
        pop r24
        ret


main:

        rcall lcd_init_sim ; Αρχικοποίηση οθόνης

        rcall scan_keypad_rising_edge_sim ; Την καλούμε μία φορά για αποφυγή σφαλμάτων
    ; χρησιμοποιούμε το T-Flag για να μπορούμε να κάνουμε τα αντίστοιχα branches για
    ; τιμές αερίου κάτω από 70ppm (Flag-bit = 0) ή πάνω από 70ppm (Flag-Bit = 1)

try1: ; Αναμονή για διάβασμα του πρώτου ψηφίου από το πληκτρολόγιο

        brtc readkeys1 ; Αν η σημαία t=0 τότε έχουμε επίπεδο αερίου < 70 ppm οπότε δεν
```

```
        ; χρειάζεται να αλλάξουμε την κατάσταση των led (όσα πρέπει να είναι αναμένα,
        ; θα έχουν ήδη ανάψει σταθερά από τη ρουτίνα εξυπηρέτησης διακοπής του timer1).
        ; Αλλιώς σημαίνει ότι έχω επίπεδο αερίου 70 ppm οπότε όσο περιμένω για ανάγνωση
; του πληκτρολογίου, θα αναβοσβήνω τα αντίστοιχα led, αλλάζοντας την κατάστασή τους ;
; κάθε 500msec.
        ldi r24, 0xF4
        ldi r25, 0x01
        rcall wait_msec      ; Καθυστέρηση 500msec
        in r24, PORTB ; Ανάγνωση της κατάστασης των led
    eor r24, r20     ; Κάνουμε xor με τον r20, ο οποίος περιέχει τα led που πρέπει να
; είναι αναμμένα ανάλογα με το επίπεδο του αερίου. Το XOR θα προκαλέσει αντιστροφή της
; κατάστασης αυτών των συγκεκριμένων led
        out PORTB, r24

readkeys1:
        rcall scan_keypad_rising_edge_sim
        rcall keypad_to_ascii_sim
        cpi r24, 0x00; Αν ο r24=0, δηλαδή αν δεν έχει πατηθεί κανένα πλήκτρο, κάνε άλμα
        ; στο try1 αλλιώς συνέχισε
        breq try1
        mov r28, r24    ;Αποθήκευση του ascii του πρώτου πλήκτρου που πατήθηκε στο r28


try2:  ; Αναμονή για διάβασμα του δεύτερου ψηφίου από το πληκτρολόγιο


        brtc readkeys2       ; Ομοίως με το πρώτο ψηφίο
        ldi r24, 0xF4
        ldi r25, 0x01
        rcall wait_msec
        in r24, PORTB
eor r24, r20
        out PORTB, r24

readkeys2:
        rcall scan_keypad_rising_edge_sim
        rcall keypad_to_ascii_sim
        cpi r24, 0x00 ; Αν ο r24=0, δηλαδή αν δεν έχει πατηθεί κανένα πλήκτρο, κάνε άλμα
; στο try1 αλλιώς συνέχισε
        breq try2
        mov r29, r24  ; Αποθήκευση του ascii του πρώτου πλήκτρου που πατήθηκε στο r29

        rcall scan_keypad_rising_edge_sim ; Προστίθεται για να μπορούμε να διαβάσουμε
; κι άλλο πλήκτρο και να αγνοηθεί χωρίς να κολλήσει ο επεξεργαστής

        cpi r28, '0'  ;Σύγκρινε τον ascii του πρώτου πλήκτρου με τον ascii του χαρακτήρα
; ‘0’

        brne wrongpsw ; Αν δεν είναι ίσοι τότε άλμα στην ετικέτα wrongpsw
        cpi r29, '3'  ; Σύγκρινε τον ascii του δεύτερου πλήκτρου με τον ascii του
; χαρακτήρα ‘3’

brne wrongpsw        ; Αν δεν είναι ίσοι τότε άλμα στην ετικέτα wrongpsw
rjmp correctpsw      ; Αλλιώς, άλμα στην ετικέτα correctpsw

wrongpsw:

ldi r16, 0x80 ; Αρχικοποίηση του PB7 ως 1. Η κατάστασή του θα αλλάζει κάθε 0.5 sec.
; Η κατάσταση του PB7 κρατείται και μεταβάλλεται με τη βοήθεια του r16
ldi r17, 0x07 ; θέλουμε 8 επαναλήψεις (8*0,5 = 4 sec)
```

```
; Υπάρχουν δύο περιπτώσεις:
; Περίπτωση α): θα είναι επίπεδο αερίου > 70 ppm, άρα T = 1, το οποίο σημαίνει ότι
; πρέπει να αναβοσβήνει και το PB7 και τα αντίστοιχα led ένδειξης του επιπέδου του
; αερίου.
; Περίπτωση β): θα είναι επίπεδο αερίου < 70 ppm, άρα T = 0, το οποίο σημαίνει ότι
; πρέπει να
; αναβοσβήνει μόνο το PB7, αλλά ταυτόχρονα να είναι σταθερά αναμένα τα αντίστοιχα leds
; ένδειξης του επιπέδου
; του αερίου.

brts blinking_pb7_high_gas ; Αν T=1 έχουμε την περίπτωση α, δηλαδή την περίπτωση
; υψηλού επιπέδου του αερίου
; οπότε άλμα στην ετικέτα blinking_pb7_high_gas

blinking_pb7_low_gas: ; Αλλιώς συνεχίζουμε εδώ, δηλαδή στην περίπτωση χαμηλού επιπέδου
; του αερίου.

        add r16, r20
        out PORTB, r16       ; Αρχικά ανάβουμε το PB7 και τα bit ένδειξης επιπέδου
        ldi r24, 0xF4
        ldi r25, 0x01
        rcall wait_msec      ; Καθυστέρηση 500msec
        com r16
        andi r16, 0x80       ; Αντιστροφή της κατάστασης του MSB του r16
        dec r17                  ; Μείωση του μετρητή επαναληψεων κατά 1
        brne blinking_pb7_low_gas ; όσο ο μετρητής r17 δεν είναι 0, ξανά άλμα στην
        ; blinking_pb7_low_gas
        out PORTB, r20       ; Μετά το πέρας των 4 δευτερολέπτων, σβήσιμο του PB7 ενώ
        ; παραμένουν αναμένα τα led ένδειξης
        ; επιπέδου
        rcall scan_keypad_rising_edge_sim ; προστίθεται για να μπορούμε να διαβάσουμε
        ; και άλλο πλήκτρο και
        ; να αγνοηθεί χωρίς να κολλήσει ο επεξεργαστής
        rjmp main            ; Επανάληψη της λειτουργίας του προγράμματος

blinking_pb7_high_gas:       ; Περίπτωση υψηλού επιπέδου αερίου

        add r16, r20  ; ο r16 κρατάει συνολικά όλα τα led που θα πρέπει να αναβοσβήνουν,
        ; δηλαδή το pb7
        ; και τα led ένδειξης του επιπέδου του αερίου
        mov r15, r16  ; ο r15 θα καθορίζει την κατάσταση των led κάθε στιγμή

loopa:
        out PORTB, r15       ; Εμφάνιση τη θύρα εξόδου B του περιεχομένου του r15.
        ldi r24, 0xF4
        ldi r25, 0x01
        rcall wait_msec
        eor r15, r16  ; Η xor αυτή θα δώσει αντιστροφή της κατάστασης των επιθυμητών
; led, δηλαδή του pb7
; και των led ένδειξης επιπέδου του αερίου. Τα υπόλοιπα δεν επηρεάζονται (παραμένουν
; σβηστά)
        dec r17
        brne loopa
        out PORTB, r20 ; Μετά το πέρας των 4 δευτερολέπτων, σβήσιμο του PB7 ενώ
        ; παραμένουν αναμένα τα
        ; led ένδειξης επιπέδου
        rcall scan_keypad_rising_edge_sim
        rjmp main
```

```
correctpsw: ;Αν λάβουμε σωστό κωδικό, θελουμε η λειτουργία του συναγερμού να διακοπεί
; για 4 sec, να ανάβει μόνο το pb7
	; σταθερά, και η οθόνη να εμφανίσει το μήνυμα WELCOME
	Cli ; εφόσον έχουμε αναστολή της λειτουργίας του συναγερμού,
	;κλείνουμε τις διακοπές αγνοώντας τον
	; timer και τον adc για αυτά τα 4sec.
	ldi r24,0x01  ; Καθαρισμός της οθόνης
	rcall lcd_command_sim
	clr r24
	ldi r24, 'W'  ; Αποστολή του μηνύματος WELCOME στην οθόνη, στέλνοντας κάθε
	; γράμμα ως δεδομένο
	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 r16, 0x80 ; Άναμα του pb7 σταθερά για 0AF0 = 4000 msec = 4sec
	out PORTB, r16
	ldi r24, 0xA0
	ldi r25, 0x0F
	rcall wait_msec
	clr r16
	out PORTB, r16; Σβήσιμο το pb7
	rcall scan_keypad_rising_edge_sim
	ldi r24,0x01  ; Καθαρισμός της οθόνης
	rcall lcd_command_sim
	sei ; Ενεργοποίηση των διακοπών ώστε οσυναγερμός να συνεχίσει κανονικά τη
	; λειτουργία του
	rjmp main

timer1_isr: ; Ρουτίνα εξυπηρέτησης διακοπής του timer1. Εκτελείται κάθε φορά που ο
; timer υπερχειλίζει, δηλαδή κάθε φορά που περνούν 100 msec από την τελευταία
; αρχικοποίηση του timer1

	clr r30 ; ο r30 θα χρησιμοποιηθεί ως σημαία ετοιμότητας του ADC,  προκειμένου η
	; ρουτίνα εξυπηρέτησης
	; του ADC να είναι μικρή σε έκταση
	sei                  ; Ενεργοποιώ τις διακοπές, ώστε να μπορέσουμε να λάβουμε
	; διακοπή από τον ADC ενώ τρέχει
	; η ρουτίνα εξυπηρέτησης διακοπής του timer

	; Αρχικοποίηση και εκκίνηση του ADC
	ldi r24,(1<<ADEN)|(1<<ADSC)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)
	out ADCSRA,r24

wait_for_it:
	cpi r30, 0x00 ; Περιμένουμε ο ADC να τελειώσει τη μετατροπή. Αυτό γίνεται
	; μένοντας σε ένα
	; βρόχο όσο r30 = 0. Μόλις η μετατροπή ολοκληρωθεί, θα προκληθεί διακοπή από
	; τον ADC, η οποία θα στείλει
	; το πρόγραμμα στην ρουτίνα εξυπηρέτησης διακοπής του ADC. Η ρουτίνα αυτή απλώς
	; θέτει r30 = 0x01 και επιστρέφει.
```

```
        ; Θα επιστρέψουμε λοιπόν στη ρουτίνα εξυπηρέτησης διακοπής του timer1. Εδώ η
        ; σύγκριση στο  βρόχο θα δει πλέον ότι
        ; η συνθήκη r30 = 0 δεν ικανοποιείται, ο βρόχος θα σπάσει και θα συνεχίσουμε
        ; στις παρακάτω εντολές.
        breq wait_for_it

; Τα led επιλέχθηκε να ανάβουν με τον εξής τρόπο:
; Συγκέντρωση αερίου -> ψηφιακή τιμή από τον ADC -> led που πρέπει να ανάψουν
; CCO = 0-35    ppm   -> ADC = 21 - 113          -> *
; CCO = 35-50   ppm   -> ADC = 113 - 153         -> **
; CCO = 50-60   ppm   -> ADC = 153 - 179         -> ***
; CCO = 60-70   ppm   -> ADC = 179 - 205         -> ****
; CCO = 70-85   ppm   -> ADC = 205 - 245         -> *****
; CCO = 85-105  ppm   -> ADC = 245 - 298         -> ******
; CCO = 105-150 ppm   -> ADC = 298 - 417         -> *******
; Χρήση του τύπου από το Data_Sheet του αισθητήρα
; Cx = 1/129*10^4 * (Vgas-Vgas0) = 129*Cx*10^(-4) + Vgas0 = Vgas
        clt                 ; μηδενίζουμε την τιμή της σημαίας T
        ldi r20, 0x00       ; θα τον χρησιμοποιήσω για τα Led

; Η τιμή του ADC αποθηκεύεται στους καταχωρητές ADCH και ADCL. Συγκεκριμένα, τα 2 MSB
; της τιμής αποθηκεύονται
; στα 2 LSB του ADCH, ενώ τα υπόλοιπα 8 bit της τιμής αποθηκεύονται στον ADCL. Έτσι
; για οποιαδήποτε τιμή του
; ADC <= 255, θα είναι ADCH = 0, άρα για ADCH = 0 γνωρίζουμε σίγουρα ότι δεν είμαστε
; στο 7ο επίπεδο, και αρκεί
; να ελέγξουμε την τιμή του ADCL για να δούμε σε ποιο επίπεδο είμαστε. Αντίθετα, αν
; ADCH > 0, μας ενδιαφέρουν
; τιμές του ADC μέχρι 298, για να δούμε αν 245<=ADC<=298 ή αν ADC>298, αφού αν ADC>298
; τότε είμαστε σίγουρα
; στο έβδομο επίπεδο. Επειδή 298 = 0x12A, ο ADCH το πολύ να έχει τιμή 0x01, επομένως ο
; έλεγχος αν ADCH = 0 αρκεί.

        in r18, ADCL
        in r19, ADCH
        cpi r19, 0x00       ; Εάν ο ADCH είναι μηδέν τότε είμαστε σε κάποιο από τα
; επίπεδα 1-6. Σε αντίθετη περίπτωση είμαστε είναι στο 6ο είτε στο 7ο επίπεδο.
        brne sixth_level
    cpi r18, 0x15       ; Αν λοιπόν ADCH = 0, ελέγχω τον ADCL και τον συγκρίνω με 21
; για να δω αν είμαι στο μηδενικό επίπεδο.
        brsh first_level      ; αν ADCL >= 21 τότε άλμα στο πρώτο επίπεδο
        out PORTB, r20        ; αλλιώς, στέλνω στην θύρω B το περιεχόμενο του r20, το
; οποίο ακόμα είναι 0 (βρίσκομαι στο μηδενικό επίπεδο)
    rjmp done_low_gas       ; Άλμα στην ετικέτα done_low_gas

first_level:
        cpi r18, 0x71       ; Σύγκριση του ADCL με το 113
        brsh second_level   ; Αν είναι >=113 τότε άλμα στο 2ο επίπεδο
        ldi r20, 0x01       ; Αλλιώς είμαι στο πρώτο επίπεδο, οπότε ανάβω μόνο το pb0
        out PORTB, r20
        rjmp done_low_gas

second_level:
        cpi r18, 0x99       ; Σύγκριση του ADCL με το 153
        brsh third_level    ; Αν είναι >=153 τότε άλμα στο 3ο επίπεδο
        ldi r20, 0x03       ; Αλλιώς είμαι στο δεύτερο επίπεδο, οπότε ανάβω μόνο τα
                            ; pb1pb0
        out PORTB, r20
        rjmp done_low_gas

third_level:
```

```
        cpi r18, 0xB3        ; Σύγκριση του ADCL με το 179
        brsh fourth_level
        ldi r20, 0x07
        out PORTB, r20
        rjmp done_low_gas

fourth_level:
        cpi r18, 0xCD        ; Σύγκριση του ADCL με το 205

        brsh fifth_level
        ldi r20, 0x0F
        out PORTB, r20
        rjmp done_low_gas

; Πλέον η συγκέντρωση αερίου ξεπερνάει τα 70ppm και τα Led πρέπει να αναβοσβήνουν,
; άρα από αυτό το επίπεδο και στα επόμενα, θα πρέπει να θέτω την σημαία T = 1.
fifth_level:
        cpi r18, 0xF5        ; Σύγκριση του ADCL με το 245
        brsh sixth_level     ; Αν είναι >=243 τότε άλμα στο 6ο επίπεδο
        set                  ; Αλλιώς θέτω T=1 και είμαι στο 5ο επίπεδο
        ldi r20, 0x1F        ; οπότε ανάβω τα 5 LSB (1F = 00011111)
        out PORTB, r20
    rjmp done_high_gas       ; Άλμα στην ετικέτα done_high_gas


sixth_level:
        cpi r19, 0x00        ; Αν ADCH = 0 τότε έχω 245 <= ADC <= 255, άρα είμαι σίγουρα
        ; στο 6ο επίπεδο
        breq aux_label
        cpi r19, 0x01        ; Αλλιώς, αν ADCH ≠ 0 και ADCH ≠ 1, τότε είμαι σίγουρα στο
        brne seventh_level   ; Έβδομο επίπεδο, γιατί έχω ADC >= 512
        cpi r18, 0x2A        ; σλλιώς, αν δηλαδή ADCH = 1, τότε επειδή 256 = 100hex και
        ; 298 = 12Ahex,
        ; αρκεί να συγκρίνω το low μέρος, δηλαδή τον ADCL ΜΕ ΤΟ 2A. Αν ADCL>=2A, επειδή
        ; ADCH = 1 σε αυτή την περίπτωση,
        ; τότε έχω αριθμό ADC >= 12Ahex = 298, άρα πρέπει να κάνω άλμα στο έβδομο
        ; επίπεδο.
        ; Αλλιώς, συνεχίζω στο έκτο επίπεδο
    brsh seventh_level

aux_label:
        set            ; θέτω T=1 κι είμαι στο 6ο επίπεδο
        ldi r20, 0x3F ; Οπότε ανάβω όλα τα led εκτός από τα pb7 και pb6
        out PORTB, r20
        rjmp done_high_gas

seventh_level:
        set                  ; θέτω T=1 και είμαι στο 7ο επίπεδο
        ldi r20, 0x7F  ; οπότε ανάβω όλα τα led εκτός από το pb7
        out PORTB, r20

done_high_gas:          ;Τυπώνουμε στην LCD το μήνυμα 'GAS DETECTED', όταν έχουμε τιμές
        ; συγκέντρωσης του αερίου

        push r24
        rcall lcd_init_sim
        clr r24
        ldi r24, 'G'
        rcall lcd_data_sim
        ldi r24, 'A'
        rcall lcd_data_sim
```

```
        ldi r24, 'S'
        rcall lcd_data_sim
        ldi r24, ' '
        rcall lcd_data_sim
        ldi r24, 'D'
        rcall lcd_data_sim
        ldi r24, 'E'
        rcall lcd_data_sim
        ldi r24, 'T'
        rcall lcd_data_sim
        ldi r24, 'E'
        rcall lcd_data_sim
        ldi r24, 'C'
        rcall lcd_data_sim
        ldi r24, 'T'
        rcall lcd_data_sim
        ldi r24, 'E'
        rcall lcd_data_sim
        ldi r24, 'D'
        rcall lcd_data_sim

; Στο τέλος της ρουτίνας εξυπηρέτησης διακοπής του timer1, αρχικοποιούμε ξανά τον
; Timer1 με τις ίδιες τιμές,
; προκειμένου να μετρήσει το επόμενο διάστημα 100msec.
    ldi r24, hightime
        out TCNT1H, r24
        ldi r24, lowtime
        out TCNT1L, r24
        pop r24
        reti

done_low_gas: ;Τυπώνουμε στην LCD το μήνυμα 'CLEAR', όταν έχουμε τιμές συγκέντρωσης
; του αερίου
        push r24
        rcall lcd_init_sim
        clr r24
        ldi r24, 'C'
        rcall lcd_data_sim
        ldi r24, 'L'
        rcall lcd_data_sim
        ldi r24, 'E'
        rcall lcd_data_sim
        ldi r24, 'A'
        rcall lcd_data_sim
        ldi r24, 'R'
        rcall lcd_data_sim
; Στο τέλος της ρουτίνας εξυπηρέτησης διακοπής του timer1, αρχικοποιούμε ξανά τον
; Timer1 με τις ίδιες τιμές,
; προκειμένου να μετρήσει το επόμενο διάστημα 100msec
        ldi r24, hightime
        out TCNT1H, r24
        ldi r24, lowtime
        out TCNT1L, r24
        pop r24
        reti
adc_r:
        ldi r30, 0x01
        reti
```

Ζήτημα 4.2

Στην C επιλέξαμε να χρησιμοποιήσουμε τη μέθοδο της διακοπής προκειμένου να διαβάσουμε τον ADC. Αν επιλέγαμε τη μέθοδο polling ο κώδικας θα ήταν παρόμοιος με τη διαφορά ότι δεν θα ενεργοποιούσαμε την ADIE, δεν θα υπήρχε ρουτίνα εξυπηρέτησης διακοπής για τον ADC, κι η ρουτίνα εξυπηρέτησης διακοπής για τον Timer, αντί να περιμένει να υψωθεί η adc_ready σημαία, θα περίμενε το bit ADSC του καατχωρητή ADSCRA να γίνει 0. Τέλος, να αναφέρουμε ότι δεν υλοποιήσαμε τη λειτουργία της οθόνης LCD στη C.

```
#include <avr/io.h>
#include <avr/interrupt.h>
unsigned char r25, r24, w, x, y, z, r25_new, r24_new, temp_high, temp_low, output;
char adc_ready, tflag;      // Αντιπροσωπεύουν τον ADC

void wait_usec (int x){
	for(int i=0; i<x;  i++) {
		asm("nop");
		asm("nop");
		asm("nop");
		asm("nop");
	}
}

void wait_msec (int y) {
	for(int i=0; i<y; i++) {
		wait_usec(990);
	}
}

void scan_keypad_sim(void)
{
	unsigned char row=0x10;

	w = x = y = z = 0x00;
	// Έλεγχος για πλήκτρα 123A
	PORTC = row;
	wait_usec(500);
	asm("nop");
	asm("nop");
	w = 0x0F & PINC;
	w = w << 4;

	// Έλεγχος για πλήκτρα 456B
	row = row << 1;
	PORTC = row;
	wait_usec(500);
	asm("nop");
	asm("nop");
	x = 0x0F & PINC;

	// Έλεγχος για πλήκτρα 789C
	row = row << 1;
	PORTC = row;
	wait_usec(500);
	asm("nop");
	asm("nop");
	y = 0x0F & PINC;
	y = y << 4;
```

```
	// Έλεγχος για πλήκτρα *0#D
	row = row << 1;
	PORTC = row;
	wait_usec(500);
	asm("nop");
	asm("nop");
	z = 0x0F & PINC;

	PORTC = 0x00;
}

void scan_keypad_rising_edge_sim(void) {
	scan_keypad_sim();
	r25 = w + x; // r25 = A321B654
	r24 = y + z; // r24 = C987D#0*
	wait_msec(15);
	scan_keypad_sim();
	r25_new = w + x;
	r24_new = y + z;
	// Αποφυγή σπινθηρισμού
	r24_new = r24_new & r24;
	r25_new = r25_new & r25;
	r25 = temp_low;
	r24 = temp_high;
	temp_high = r24_new;
	temp_low = r25_new;
	r25 = r25^0xFF;
	r24 = r24^0xFF;
	r24_new = r24_new & r24;
	r25_new = r25_new & r25;
}

unsigned char keypad_to_ascii_sim(void) {
	unsigned char key_pointer = 0x01, which_key[16] = {'*', '0', '#', 'D', '7',
'8', '9', 'C', '4', '5', '6', 'B', '1', '2', '3', 'A'}, pos = 0;
	while(key_pointer != 0) {
		if ((r24_new & key_pointer) == key_pointer) {
			return which_key[pos];
		}
		else{
			key_pointer = key_pointer << 1;
			pos++;
		}
	}
	key_pointer = 0x01;
	while(key_pointer != 0) {
		if ((r25_new & key_pointer) == key_pointer) {
			return which_key[pos];
		}
		else{
			key_pointer = key_pointer << 1;
			pos++;
		}
	}
}

ISR(ADC_vect) {			// Διακοπή του ADC
	adc_ready = 1;
}

ISR(TIMER1_OVF_vect) {		// Διακοπή υπερχείλισης του Timer1
```

```
	int adc_value;		// Ορίζουμε ως ακέραιο την τιμή που θα επιστρέφει ο ADC
	unsigned char low_adc;
	adc_ready = 0;
	asm("sei");
	ADCSRA = 0xCF;
	while(!adc_ready) {
	}
	tflag = false;
	low_adc = ADCL;		// Αποθηκεύω τις τιμές του ADC
	adc_value = ADCH;
	adc_value = (adc_value << 8) + low_adc;
	// Μηδενικό επίπεδο
	if(adc_value < 21) {
		PORTB = 0x00;
		output = 0x00;
	}
	// Πρώτο επίπεδο
	else if(adc_value < 113) {
		PORTB = 0x01;
		output = 0x01;
	}
	// Δεύτερο επίπεδο
	else if(adc_value < 153) {
		PORTB = 0x03;
		output = 0x03;
	}
	// Τρίτο επίπεδο
	else if(adc_value < 179) {
		PORTB = 0x07;
		output = 0x07;
	}
	// Τέταρτο επίπεδο
	else if(adc_value < 205) {
		PORTB = 0x0F;
		output = 0x0F;
	}
	// Πέμπτο επίπεδο. Τα Led αναβοσβήνουν και το tflag παίρνει τιμή 1
	else if(adc_value < 245) {
		tflag = true;
		PORTB = 0x1F;
		output = 0x1F;
	}
	// Έκτο επίπεδο
	else if(adc_value < 298) {
		tflag = true;
		PORTB = 0x3F;
		output = 0x3F;
	}
	// Έβδομο επίπεδο
	else {
		tflag = true;
		PORTB = 0x7F;
		output = 0x7F;
	}

	// Επαναρχικοποίηση της διακοπής υπερχείλισης του Timer1
	TCNT1H = 0xFC;
	TCNT1L = 0xF3;
}

int main(void){
```

```
unsigned char prwto, deytero;
DDRC = 0xF0;
DDRB = 0xFF;
ADMUX = 0x40;
TIMSK = 0x04;
TCCR1B = 0x05;
TCNT1H = 0xFC;
TCNT1L = 0xF3;
asm("sei");
while(1) {
	scan_keypad_rising_edge_sim();
	// Διαβάζουμε το πρώτο πλήκτρο.
	// Ωστόσο συνεχίζουμε και ελέγχουμε τα επίπεδα του αερίου.
	// Εάν είναι υψηλά τότε τα Led αναβοσβήνουν

	//Εάν ο tflag είναι 1 τότε είμαι σε τιμές αερίου πάνω από 70ppm και
	// αναβοσβήνουν τα led
	while(1) {
		if(tflag) {
			unsigned char aux;
			wait_msec(500);
			aux = PORTB;
			PORTB = aux ^ output;
		}
	// Αναμονή για πάτημα πρώτου πλήκτρου και παράλληλα εάν έχω υψηλές τιμές
	// αερίου τα Led αναβοσβήνουν

		scan_keypad_rising_edge_sim();
		if(r24_new != 0 || r25_new != 0) {
			break;
		}
	}
	// Παίρνω τον ASCII του πλήκτρου που πατήθηκε
	prwto = keypad_to_ascii_sim();
	wait_msec(15);
	while(1) {
		if(tflag) {
			unsigned char aux;
			wait_msec(500);
			aux = PORTB;
			PORTB = aux ^ output;
		}
	// Εφόσον έχει δοθεί πρώτο πλήκτρο, αναμένω για δεύτερο
		scan_keypad_rising_edge_sim();
		if(r24_new != 0 || r25_new != 0) {
			break;
		}
	}
	// Αντίστοιχα παίρνω τον ASCII του δεύτερου πλήκτρου που πατήθηκε
	deytero = keypad_to_ascii_sim();
	scan_keypad_rising_edge_sim();

	// Εάν δοθεί σωστός αριθμός Ομάδας, εν προκειμένη 03, τότε ανάβει το PB7
	// για 4sec
	if(prwto == '0' && deytero == '3') {
		asm("cli");
		PORTB = 0x80;
		wait_msec(4000);
		PORTB = 0x00;
		asm("sei");
	}
```

```
                // Εάν δοθεί λάθος αριθμός Ομάδας ελέγχουμε σε τι επίπεδο αερίου
                // βρισκόμαστε
                // Αν είμαστε σε χαμηλά επίπεδα, τότε αναβοσβήνει μόνο το PB7 ενώ το
                // output είναι σταθερά αναμμένο
                else {
                        unsigned char pb7 = 0x80;
                        if(!tflag) {
                                for(int i=0; i<8; i++) {
                                        PORTB = pb7 + output;
                                        wait_msec(500);
                                        pb7 = pb7^0x80;
                                }
                                PORTB = output;
                        }

                // Αλλιώς εάν είμαστε σε υψηλά επίπεδα αερίου αναβοσβήνει και το PB7 και
                // τα υπόλοιπα Led του output
                        else {
                                pb7 = pb7 + output;
                                for(int i=0; i<8; i++) {
                                        PORTB = pb7;
                                        wait_msec(500);
                                        pb7 = pb7 ^ (0x80 + output);
                                }
                                PORTB = output; // Εμφάνιση των Led (αναμμένα ή σβηστά)
                        }
                }
        }
}
```