Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Μικροϋπολογιστών 2021-2022

4η Εργασία (AVR)

Κωνσταντίνος Σιδέρης, Ομάδα 48

Α.Μ.: 03118134

Άσκηση 1

Ακολουθεί ο κώδικας της άσκησης 1 (Assembly) με σχόλια:

```
.include "m16def.inc"
.def blink_flag = r16   ;Flag που ελέγχει αν τα led πρέπει να αναβοσβήσουν
.def danger_flag = r17  ;Flag ένδειξης επίπεδου CO μεγαλύτερο από 70 ppm
.def leds = r18
.def code_flag = r19    ;Flag ένδειξης λήψη διψήφιου κωδικού
.def warn_on = r20      ;Flag ενεργοποίησης ειδοποίησης GAS DETECTED (εάν είναι 1 εάν η ένδειξη έχει ήδη ενεργοποιηθεί)

.DSEG
_tmp_: .byte 2

.CSEG
.macro welcome    ;Macro welcome που ανάβει το PB7 για 4sec(σωστός κωδικός)
            ldi warn_on, 0x00
            rcall lcd_init_sim
            ldi r24,'W'       ;Εμφάνιση μυνήματος WELCOME 48 στην οθόνη LCD
            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 r26, 0xA0     ;160 επαναλήψεις διάρκειας 25ms: συνολική καθυστέρηση 4sec
keep_on:
            mov r27, leds     ;Συνδυασμός των led με το PB7
            ori r27, 0x80
            out PORTB,r27     ;Εμφάνιση του επιπέδου CO και άνναμα του PB7
            rcall scan_keypad_rising_edge_sim ;Διάβασμα πληκτρολογίου κατά την διάρκεια ανάμματος των led (19ms)
            ldi r24,low(6)
            ldi r25,high(6)
            rcall wait_msec   ;Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε συνολική καθυστέρηση 25ms
            dec r26
            cpi r26, 0x00
            brne keep_on
            clr r27
```

```
            out PORTB,r27      ;Σβήνουμε τα  led της PORTB
            rcall lcd_init_sim
            ldi code_flag, 0x00 ;Μηδενισμός flag ένδειξης λήψης διψήφιου
κωδικού
.endmacro

.macro alarm      ;Macro alarm που αναβοσβήνει το PB7 για 4sec(λάθος κωδικός)
            ldi r26, 0x04      ;4 επαναλήψεις διάρκειας 1sec: συνολική
καθυστέρηση 4sec
blink:
            ldi r27, 0x14      ;20 επαναλήψεις διάρκειας 25ms: συνολική
καθυστέρηση 0.5sec
blink1:
            mov r28, leds      ;Συνδυασμός των led με το PB7
            ori r28, 0x80
            out PORTB,r28      ;Εμφάνιση του επιπέδου CO και άνναμα του PB7
            rcall scan_keypad_rising_edge_sim ;Διάβασμα πληκτρολογίου κατά την
διάρκεια που αναβοσβήνουν τα led (19ms)
            ldi r24,low(6)
            ldi r25,high(6)
            rcall wait_msec    ;Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε
συνολική καθυστέρηση 25ms
            dec r27
            cpi r27, 0x00
            brne blink1

            ldi r27, 0x14      ;20 επαναλήψεις διάρκειας 25ms: συνολική
καθυστέρηση 0.5sec
blink2:
            cpi danger_flag, 0x00   ;Ελέγχουμε το flag ένδειξης του CO (1 αν το
CO είναι μεγαλύτερο από 70 ppm)
            brne leds_blink
            mov r28, leds      ;Εάν είναι 0 κρατάμε τα  led του επιπέδου του CO
αναμμένα
            rjmp cont
leds_blink:
            clr r28      ;Εάν είναι 1 σβήνουμε όλα τα  led ώστε να αναβοσβήνει
και το επίπεδο του CO μαζί με το PB7
cont:
            out PORTB, r28     ;Διάβασμα πληκτρολογίου κατά την διάρκεια που
αναβοσβήνουν τα led (19ms)
            rcall scan_keypad_rising_edge_sim
            ldi r24,low(6)
            ldi r25,high(6)
            rcall wait_msec    ;Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε
συνολική καθυστέρηση 25ms
            dec r27
            cpi r27, 0x00
```

```
            brne blink2

            dec r26
            cpi r26, 0x00
            brne blink
            ldi code_flag, 0x00 ;Μηδενισμός flag ένδειξης λήψης διψήφιου
κωδικού
.endmacro

.macro gas_detected
            cpi warn_on, 0x01 ;Έλεγχος flag ενεργοποίησης ειδοποίησης GAS
DETECTED
            breq exit_gas     ;Αν είναι 1 η ειδοποίηση έχει ήδη ενεργοποιηθεί
οπότε δεν κάνουμε τίποτα και βγαίνουμε από το macro
            rcall lcd_init_sim
            ldi r24,'G' ;Αν είναι 0 εμφανίζουμε το μήνυμα GAS DETECTED στην
οθόνη LCD
            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
            ldi warn_on, 0x01 ;Και θέτουμε το flag ενεργοποίησης ειδοποίησης σε
1
exit_gas:
.endmacro

.org 0x00
rjmp reset
.org 0x10
```

```
rjmp ISR_TIMER1_OVF ;Ρουτίνα εξυπηρέτησης της διακοπής υπερχείλισης του timer1
.org 0x1C
rjmp ADC_ISR

reset:
            ldi   r24, LOW(RAMEND) ;Αρχικοποίηση stack pointer
            out   spl, r24
            ldi   r24, HIGH(RAMEND)
            out   sph, r24
            ldi r24, (1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
            out DDRC, r24      ;Αρχικοποίηση 4ων MSB της PORTB ως έξοδο και
4ων LSB της PORTB ως είσοδο
            ser r24
            out DDRD, r24      ;Αρχικοποίηση PORTB ως έξοδο
            out DDRB, r24      ;Αρχικοποίηση PORTD ως έξοδο
            rcall ADC_init     ;Κλήση ρουτίνας αρχικοποίησης του ADC
            ldi r24, (1<<TOIE1) ;Ενεργοποίηση διακοπής υπερχείλισης του
μετρητή TCNT1 για τον timer1
            out TIMSK, r24
            ldi r24, (1<<CS12) | (0<<CS11) | (1<<CS10) ;CK/1024
            out TCCR1B, r24
            ldi r24, 0xfc      ;Αρχικοποίηση του TCNT1 για υπερχείλιση μετά από
100 msec
            out TCNT1H, r24
            ldi r24, 0xf3
            out TCNT1L, r24

            sei   ;Ενεργοποίηση διακοπών

            ldi blink_flag, 0x01
            ldi code_flag, 0x00     ;Αρχικοποίηση των flags
            rcall lcd_init_sim

digit_1:
            cpi danger_flag, 0x01 ;Ελέγχουμε το flag ένδειξης του CO (1 αν το CO
είναι μεγαλύτερο από 70 ppm)
            brne cont_1
            gas_detected ;Εάν είναι 1 καλούμε το macro ενεργοποίησης
ειδοποίησης GAS DETECTED
cont_1:
            ldi r21, 0x00      ;Αρχικοποίηση flag (r21) ορθότητας 1ου ψηφίου
            rcall scan_keypad_rising_edge_sim ;Διάβασμα πληκτρολογίου
            rcall keypad_to_ascii_sim     ;Μετατροπή σε ψηφίο ASCII
            cpi r24, 0x00
            breq digit_1       ;Διάβασμα πληκτρολογίου μέχρι να πατηθεί το 1ο
πλήκτρο
            cpi r24, '4'       ;Έλεγχος 1ου ψηφίου
            breq digit_2
```

```
            ldi r21, 0x01      ;Εάν είναι λάθος θέτουμε το flag σε 1

digit_2:
            cpi danger_flag, 0x01 ;Ελέγχουμε το flag ένδειξης του CO (1 αν το CO
είναι μεγαλύτερο από 70 ppm)
            brne cont_2
            gas_detected ;Εάν είναι 1 καλούμε το macro ενεργοποίησης
ειδοποίησης GAS DETECTED
cont_2:
            rcall scan_keypad_rising_edge_sim ;Διάβασμα πληκτρολογίου
            rcall keypad_to_ascii_sim     ;Μετατροπή σε ψηφίο ASCII
            cpi r24, 0x00
            breq digit_2 ;Διάβασμα πληκτρολογίου μέχρι να πατηθεί το 2ο
πλήκτρο
            cpi r21, 0x01      ;Έλεγχος flag ορθότητας 1ου ψηφίου
            breq wrong_pass    ;Αν είναι 1 πηγαίνουμε στο λάθος password
            cpi r24, '8'       ;Έλεγχος 2ου ψηφίου
            brne wrong_pass    ;Αν δεν είναι σωστό πηγαίνουμε στο λάθος
password

correct_pass:                  ;Εάν φτάσουμε εδώ έχουμε σωστό password
            ldi code_flag, 0x01 ;Θέτουμε το flag ένδειξης λήψης διψήφιου
κωδικού σε 1
            rcall scan_keypad_rising_edge_sim
            welcome            ;Κλήση macro welcome
            rjmp digit_1

wrong_pass:              ;Εάν φτάσουμε εδώ έχουμε λάθος  password
            ldi code_flag, 0x01 ;Θέτουμε το flag ένδειξης λήψης διψήφιου
κωδικού σε 1
            rcall scan_keypad_rising_edge_sim
            alarm        ;Κλήση macro alarm
            rjmp digit_1

scan_row_sim:                  ;Υλοποίηση ρουτίνας scan_row_sim
            out PORTC, r25
            push r24
            push r25
            ldi r24,low(500)
            ldi r25,high(500)
            rcall wait_usec
            pop r25
            pop r24
            nop
            nop
            in r24, PINC
            andi r24 ,0x0f
            ret
```

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

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

```
            com r22
            and r24 ,r22
            and r25 ,r23
            pop r27
            pop r26
            pop r23
            pop r22
            ret

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

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

write_2_nibbles_sim:          ;Υλοποίηση ρουτίνας write_2_nibbles_sim
            push r24
            push r25
            ldi r24 ,low(6000)
            ldi r25 ,high(6000)
            rcall wait_usec
            pop r25
            pop r24
            push r24
            in r25, PIND
            andi r25, 0x0f
            andi r24, 0xf0
            add r24, r25
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
            push r24
            push r25
            ldi r24 ,low(6000)
            ldi r25 ,high(6000)
            rcall wait_usec
            pop r25
            pop r24
            pop r24
            swap r24
            andi r24 ,0xf0
            add r24, r25
            out PORTD, r24
```

```
            sbi PORTD, PD3
            cbi PORTD, PD3
            ret

lcd_data_sim:           ;Υλοποίηση ρουτίνας lcd_data_sim
            push r24
            push r25
            sbi PORTD, PD2
            rcall write_2_nibbles_sim
            ldi r24 ,43
            ldi r25 ,0
            rcall wait_usec
            pop r25
            pop r24
            ret

lcd_command_sim:        ;Υλοποίηση ρουτίνας lcd_comand_sim
            push r24
            push r25
            cbi PORTD, PD2
            rcall write_2_nibbles_sim
            ldi r24, 39
            ldi r25, 0
            rcall wait_usec
            pop r25
            pop r24
            ret

lcd_init_sim:           ;Υλοποίηση ρουτίνας lcd_init_sim
            push r24
            push r25

            ldi r24, 40
            ldi r25, 0
            rcall wait_msec
            ldi r24, 0x30
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
            ldi r24, 39
            ldi r25, 0
            rcall wait_usec
            push r24
            push r25
            ldi r24,low(1000)
            ldi r25,high(1000)
            rcall wait_usec
            pop r25
```

```
            pop r24
            ldi r24, 0x30
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
            ldi r24,39
            ldi r25,0
            rcall wait_usec
            push r24
            push r25
            ldi r24 ,low(1000)
            ldi r25 ,high(1000)
            rcall wait_usec
            pop r25
            pop r24
            ldi r24,0x20
            out PORTD, r24
            sbi PORTD, PD3
            cbi PORTD, PD3
            ldi r24,39
            ldi r25,0
            rcall wait_usec
            push r24
            push r25
            ldi r24 ,low(1000)
            ldi r25 ,high(1000)
            rcall wait_usec
            pop r25
            pop r24
            ldi r24,0x28
            rcall lcd_command_sim
            ldi r24,0x0c
            rcall lcd_command_sim
            ldi r24,0x01
            rcall lcd_command_sim
            ldi r24, low(1530)
            ldi r25, high(1530)
            rcall wait_usec
            ldi r24 ,0x06
            rcall lcd_command_sim
            pop r25
            pop r24
            ret

wait_msec:              ;Υλοποίηση ρουτίνας 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:              ;Υλοποίηση ρουτίνας wait_usec
            sbiw r24 ,1
            nop
            nop
            nop
            nop
            brne wait_usec
            ret

ADC_init:
            ldi r24,(1<<REFS0)      ;Vref = Vcc
            out ADMUX,r24     ;Επιλογή A0 για είσοδο
            ldi r24,(1<<ADEN)|(1<<ADIE)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)
;Ενεργοποίηση ADC, διακοπών και ρύθμιση prescaler CK/128=62.5Khz
            out ADCSRA,r24
            ret

ISR_TIMER1_OVF:
            push r24
            in r24, ADCSRA
            ori r24, (1<<ADSC)
            out ADCSRA, r24
            ldi r24, 0xfc     ;Επαναρύθμιση του TCNT1 για υπερχείλιση μετά από
100 msec
            out TCNT1H, r24
            ldi r24, 0xf3
            out TCNT1L, r24
            pop r24
            reti

ADC_ISR:
            push r24
            push r25

            in r24, ADCL      ;Λήψη τιμής ADC
            in r25, ADCH
            andi r25, 0x03

            cpi r25, 0x00     ;Εάν το r25 είναι μεγαλύτερο από 0 τότε το
επίπεδο CO είναι μεγαλύτερο
```

```
            brne five_leds    ;από 90 οπότε ελέγχουμε κατευθείαν από το 5ο led
και μετά

            cpi r24, 0x64     ;0 < Cx < 30
            brsh two_leds
            ldi leds, 0x01
            rjmp constant     ;Εμφάνιση επιπέδου CO με σταθερά led
two_leds:
            cpi r24, 0x99     ;30 < Cx < 50
            brsh three_leds
            ldi leds, 0x03
            rjmp constant     ;Εμφάνιση επιπέδου CO με σταθερά led
three_leds:
            cpi r24, 0xCD     ;50 < Cx < 70
            brsh four_leds
            ldi leds, 0x07
            rjmp constant     ;Εμφάνιση επιπέδου CO με σταθερά led
four_leds:
            ldi leds, 0x0F    ;70 < Cx < 90
            rjmp blinking     ;Εμφάνιση επιπέδου CO με led που αναβοσβήνουν
five_leds:
            cpi r24, 0x37     ;90 < Cx < 110
            brsh six_leds
            ldi leds, 0x1F
            rjmp blinking     ;Εμφάνιση επιπέδου CO με led που αναβοσβήνουν
six_leds:
            cpi r24, 0x72     ;110 < Cx < 140
            brsh seven_leds
            ldi leds, 0x3F
            rjmp blinking     ;Εμφάνιση επιπέδου CO με led που αναβοσβήνουν
seven_leds:
            ldi leds, 0x7F    ;Cx > 140
            rjmp blinking     ;Εμφάνιση επιπέδου CO με led που αναβοσβήνουν

constant:
            cpi code_flag, 0x01
            breq was_safe
            cpi danger_flag, 0x01 ;Έλεγχος flag ένδειξης επίπεδου CO μεγαλύτερο
από 70 ppm
            brne was_safe  ;Αν είναι 1 σημαίνει ότι πριν είμασταν σε κατάσταση
κινδύνου άρα τώρα που
            rcall lcd_init_sim ;το επίπεδο είναι κάτω από 70 πρέπει να
εμφανίσουμε το μήνυμα CLEAR
            ldi r24,'C'       ;Εμφάνιση μυνήματος CLEAR στην οθόνη LCD
            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
            ldi r24,low(70)
            ldi r25,high(70)
            rcall wait_msec
            rcall lcd_init_sim
was_safe:
            ldi warn_on, 0x00 ;Είμαστε ασφαλείς οπότε μηδενίζουμε το flag
ενεργοποίησης ειδοποίησης GAS DETECTED
            ldi danger_flag, 0x00 ;και το flag ένδειξης επίπεδου CO μεγαλύτερο
από 70 ppm
            cpi code_flag, 0x01 ;Έλεγχος του flag ένδειξης λήψης διψήφιου
κωδικού
            breq exit ;Αν είναι 1 έχουμε πάρει κωδικό οπότε την εμφάνιση του
επιπέδου αναλαμβάνουν τα macro alarm και welcome
            out PORTB, leds   ;Αν είναι 0 ανάβουμε τα led
            rjmp exit

blinking:
            ldi danger_flag, 0x01 ;Είμαστε σε κίνδυνο οπότε θέτουμ το flag
ενεργοποίησης ειδοποίησης GAS DETECTED σε 1
            cpi code_flag, 0x01 ;Έλεγχος του flag ένδειξης λήψης διψήφιου
κωδικού
            breq exit ;Αν είναι 1 έχουμε πάρει κωδικό οπότε την εμφάνιση του
επιπέδου αναλαμβάνουν τα macro alarm και welcome
            cpi blink_flag, 0x01 ;Αν είναι 0 ελέγχουμε το flag που ελέγχει αν τα
led πρέπει να αναβοσβήσουν
            breq on     ;Αν το flag που ελέγχει αν τα led πρέπει να
αναβοσβήσουν είναι 1 τα ανάβουμε
            clr r24 ;Αν το flag που ελέγχει αν τα led πρέπει να αναβοσβήσουν
είναι 1 τα σβήνουμε
            rjmp lights
on:
            mov r24, leds
lights:
            out PORTB, r24    ;Άναμμα των leds
            com blink_flag    ;Συμπλήρωμα ως προς 1 του flag που ελέγχει αν
τα led πρέπει να αναβοσβήσουν ώστε
            andi blink_flag, 0x01 ;στην επόμενη επανάληψη να ανάψουν αν ήταν
σβηστά ή να σβήσουν αν ήταν ανοιχτά

exit:
            pop r25
            pop r24
            reti
```

Άσκηση 2

Ακολουθεί ο κώδικας της άσκησης 2 (C) με σχόλια:

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

char reg[2], mem[2], fd, sd;
volatile char leds, blink_flag, code_flag; //Flag που ελέγχει αν τα led πρέπει να αναβοσβήσουν και flag ένδειξης λήψη διψήφιου κωδικού
volatile float Cx;

char swap(char x) { //Υλοποίηση εντολής swap
	return ((x & 0x0F) << 4 | (x & 0xF0) >>4);
}

char scan_row(int r) { //Υλοποίηση ρουτίνας scan_row_sim
	char o = 0x08;
	o = (o << r);	//Ολίσθηση r φορές του o = 00001000
	PORTC = o;  //Θέτουμε την γραμμή που σκανάρουμε σε 1
	_delay_us(500);   //Καθυστέρηση 500ms για απομακρυσμένη λειτουργία
	return PINC & 0x0F; //Επιστροφή τεσσάρων  LSB της PORTC (στήλες πληκτρολογίου)
}

void scan_keypad() { //Υλοποίηση ρουτίνας scan_keypad_sim
	char c;

	c = scan_row(1); //Σκανάρισμα πρώτης σειράς
	reg[1] = swap(c); //Αποθήκευση στα 4 MSB των πρώτων 8 bit του 16-bit register reg

	c = scan_row(2); //Σκανάρισμα δεύτερης σειράς
	reg[1] = reg[1] + c; //Αποθήκευση στα 4 LSB των πρώτων 8 bit του 16-bit register reg

	c = scan_row(3); //Σκανάρισμα τρίτης σειράς
	reg[0] = swap(c); //Αποθήκευση στα 4 MSB των δεύτερων 8 bit του 16-bit register reg

	c = scan_row(4); //Σκανάρισμα τέταρτης σειράς
	reg[0] = reg[0] + c; //Αποθήκευση στα 4 LSB των δεύτερων 8 bit του 16-bit register reg

	PORTC = 0x00;
}

int scan_keypad_rising_edge() {	//Υλοποίηση ρουτίνας
```

```
scan_row_rising_edge_sim
	char tmp[2];

	scan_keypad();	//Σκανάρισμα πληκτρολογίου
	tmp[0] = reg[0]; //Προσωρινή αποθήκευση του αποτελέσματος
	tmp[1] = reg[1];

	_delay_ms(15);	//Αναμονή 15ms λόγω σπινθηρισμών

	scan_keypad();	//Δεύτερο σκανάρισμα πληκτρολογίου
	reg[0] = reg[0] & tmp[0]; //Απόρριψη πλήκτρων που εμφάνισαν
σπινθηρισμό
	reg[1] = reg[1] & tmp[1];

	tmp[0] = mem[0]; //Λήψη προηγούμενης κατάστασης διακοπτών από την  RAM
	tmp[1] = mem[1];

	mem[0] = reg[0]; //Αποθήκευση τωρινής κατάστασης διακοπτών από την
RAM
	mem[1] = reg[1];

	reg[0] = reg[0] & (~tmp[0]); //Εύρεση διακοπτών που έχουν μόλις πατηθεί
	reg[1] = reg[1] & (~tmp[1]);

	return (reg[0] || reg[1]); //Επιστροφή των διακοπτών που μόλλις
πατήθηκαν (0 αν δεν έχει πατηθεί πλήκτρο)
}

char keypad_to_ascii() { //Υλοποίηση ρουτίνας keypad_to_ascii_sim
	if ((reg[0]&0x01) == 0x01)
	return '*'; //Εύρεση πατημένου πλήκτρου και επιστροφή του κωδικού ASCII
που του αντιστοιχεί

	if ((reg[0]&0x02) == 0x02)
	return '0';

	if ((reg[0]&0x04) == 0x04)
	return '#';

	if ((reg[0]&0x08) == 0x08)
	return 'D';

	if ((reg[0]&0x10) == 0x10)
	return '7';

	if ((reg[0]&0x20) == 0x20)
	return '8';
```

```
	if ((reg[0]&0x40) == 0x40)
	return '9';

	if ((reg[0]&0x80) == 0x80)
	return 'C';

	if ((reg[1]&0x01) == 0x01)
	return '4';

	if ((reg[1]&0x02) == 0x02)
	return '5';

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

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

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

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

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

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

	return 0;   //Εάν δεν έχει πατηθεί κάποιο πλήκτρο επιστρέφει 0
}

void welcome() {  //Υλοποίηση συνάρτησης welcome που ανάβει το PB7 για 4sec(σωστός κωδικός)
	char i;
	for (i = 1; i <= 160; i++) { //160 επαναλήψεις διάρκειας 25ms: συνολική καθυστέρηση 4sec
		PORTB = leds | 0x80; //Συνδυασμός των led με το PB7
		scan_keypad_rising_edge(); //Διάβασμα πληκτρολογίου κατά την διάρκεια ανάμματος των led (19ms)
		_delay_ms(6); //Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε συνολική καθυστέρηση 25ms
	}
	code_flag = 0x00; //Μηδενισμός flag ένδειξης λήψης διψήφιου κωδικού
	PORTB = 0x00;     //Σβήνουμε τα  led της PORTB
}
```

```
void alarm() {          //Υλοποίηση συνάρτησης alarm που αναβοσβήνει το PB7 για 4sec(λάθος κωδικός)
      char i, j;
      for (j = 1; j <= 4; j++) { //4 επαναλήψεις διάρκειας 1sec: συνολική καθυστέρηση 4sec
            for (i = 1; i <= 20; i++) { //20 επαναλήψεις διάρκειας 25ms: συνολική καθυστέρηση 0.5sec
                  PORTB = leds | 0x80; //Συνδυασμός των led με το PB7
                  scan_keypad_rising_edge(); //Διάβασμα πληκτρολογίου κατά την διάρκεια που αναβοσβήνουν τα led (19ms)
                  _delay_ms(6); //Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε συνολική καθυστέρηση 25ms
            }
            for (i = 1; i <= 20; i++) { //20 επαναλήψεις διάρκειας 25ms: συνολική καθυστέρηση 0.5sec
                  if (Cx <= 70)
                        PORTB = leds; //Εάν το επίπεδο του CO είναι <= 70ppm κρατάμε τα  led του επιπέδου του CO αναμμένα
                  else
                        PORTB = 0x00; //Εάν το επίπεδο του CO είναι > 70ppm σβήνουμε όλα τα  led ώστε να αναβοσβήνει και το επίπεδο του CO μαζί με το PB7
                  scan_keypad_rising_edge(); //Διάβασμα πληκτρολογίου κατά την διάρκεια που αναβοσβήνουν τα led (19ms)
                  _delay_ms(6); //Πρόσθετη καθυστέρηση 6ms ώστε να έχουμε συνολική καθυστέρηση 25ms
            }
      }
      code_flag = 0x00; //Μηδενισμός flag ένδειξης λήψης διψήφιου κωδικού
}

void ADC_init() {
      ADMUX = (1<<REFS0);     //Vref = Vcc και πιλογή A0 για είσοδο
      ADCSRA = (1 << ADEN | 1 << ADIE | 1 << ADPS2 | 1 << ADPS1 | 1 << ADPS0); //Ενεργοποίηση ADC, διακοπών και ρύθμιση prescaler CK/128=62.5Khz
}

ISR(ADC_vect) {
      Cx = (((ADC/204.8)-0.1)/0.0129); //Υπολογισμός του επίπεδου CO

      if (Cx <= 30) { //0 < Cx < 30
            leds = 0x01;
      }
      else if (Cx > 30 && Cx <= 50) { //30 < Cx < 50
            leds = 0x03;
      }
      else if (Cx > 50 && Cx <= 70) { //50 < Cx < 70
            leds = 0x07;
      }
```

```
	else if (Cx > 70 && Cx <= 90) { //70 < Cx < 90
		leds = 0x0F;
	}
	else if (Cx > 90 && Cx <= 110) { //90 < Cx < 110
		leds = 0x1F;
	}
	else if (Cx > 110 && Cx <= 140) { //110 < Cx < 140
		leds = 0x3F;
	}
	else if (Cx > 140) { //Cx > 140
		leds = 0x7F;
	}
	if (code_flag == 0x00) { //Αν το flag ένδειξης λήψης διψήφιου κωδικού είναι 1 έχουμε πάρει κωδικό οπότε την εμφάνιση του επιπέδου αναλαμβάνουν οι συναρτήσεις alarm και welcome)
		if (Cx <= 70)
			PORTB = leds; //Αν το επίπεδο CO είναι <= 70ppm ανάβουμε τα led
		else { //Αν το επίπεδο CO είναι <= 70ppm αναβοσβήνουμε τα led
			if (blink_flag == 0x00) { //Ελέγχουμε το flag που ελέγχει αν τα led πρέπει να αναβοσβήσουν
				PORTB = leds; //Αν το flag που ελέγχει αν τα led πρέπει να αναβοσβήσουν είναι 1 τα ανάβουμε
				blink_flag = (~blink_flag)&0x01; //Συμπλήρωμα ως προς 1 του flag που ελέγχει αν τα led πρέπει να αναβοσβήσουν ώστε στην επόμενη επανάληψη να ανάψουν αν ήταν σβηστά ή να σβήσουν αν ήταν ανοιχτά
			}
			else {
				PORTB = 0x00; //Αν το flag που ελέγχει αν τα led πρέπει να αναβοσβήσουν είναι 1 τα σβήνουμε
				blink_flag = (~blink_flag)&0x01; //Συμπλήρωμα ως προς 1 του flag που ελέγχει αν τα led πρέπει να αναβοσβήσουν ώστε στην επόμενη επανάληψη να ανάψουν αν ήταν σβηστά ή να σβήσουν αν ήταν ανοιχτά
			}
		}
	}

}

ISR(TIMER1_OVF_vect) {
	ADCSRA |= (1<<ADSC);
	TCNT1 = 64755;	//Επαναρύθμιση του TCNT1 για υπερχείλιση μετά από 100 msec
}

int main(void) {
	DDRB = 0xFF;	//Αρχικοποίηση PORTB ως έξοδο
	DDRC = 0xF0;	//Αρχικοποίηση 4ων MSB της PORTB ως έξοδο και 4ων LSB
```

της PORTB ως είσοδο

```
	ADC_init();		//Κλήση ρουτίνας αρχικοποίησης του ADC
	TIMSK = (1 << TOIE1); //Ενεργοποίηση διακοπής υπερχείλισης του μετρητή TCNT1 για τον timer1
	TCCR1B = (1<<CS12) | (0<<CS11) | (1<<CS10); //CK/1024
	TCNT1 = 64755;	//Αρχικοποίηση του TCNT1 για υπερχείλιση μετά από 100 msec

	sei();		//Ενεργοποίηση διακοπών

	code_flag = 0x00; //Αρχικοποίηση των flags
	blink_flag = 0x00;

	while (1) {
		mem[0] = 0x00;	//Αρχικοποίηση μεταβλητής μνήμης RAM
		mem[1] = 0x00;
		PORTB = 0x00;

		while (1) {
			if (scan_keypad_rising_edge() != 0) { //Διάβασμα πληκτρολογίου μέχρι να πατηθεί το 1ο πλήκτρο
				fd = keypad_to_ascii(); //Μετατροπή σε ψηφίο ASCII
				break;
			}
		}

		while (1) {
			if (scan_keypad_rising_edge() != 0) { //Διάβασμα πληκτρολογίου μέχρι να πατηθεί το 2ο πλήκτρο
				sd = keypad_to_ascii(); //Μετατροπή σε ψηφίο ASCII
				scan_keypad_rising_edge();
				break;
			}
		}
		if (fd != '4' || sd != '8') {
			code_flag = 0x01;
			alarm(); //Εάν δεν έχουν πατηθεί τα σωστά πλήκτρα καλούμε την alarm
		}
		else {
			code_flag = 0x01;
			welcome(); //Εάν έχουν πατηθεί τα σωστά πλήκτρα καλούμε την welcome
		}
	}
	return 0;
}
```