Εργαστήριο Μικροϋπολογιστών
7η Άσκηση

Ομάδα: Δ12
Βακαλόπουλος Θεόδωρος, ΑΜ: 03114013
Μαυρομμάτης Ιάσων, ΑΜ: 03114771
Νικητοπούλου Δήμητρα, ΑΜ: 03114954

Άσκηση 1

Ο κώδικας της άσκησης φαίνεται παρακάτω:

```
.include "m16def.inc"

 .def reg = r18
 .def flag = r21
 .def temp = r17
 .org 0x00
 rjmp main
 .org 0x10
 rjmp ISR_TIMER1_OVF

 .DSEG
 _tmp_:.byte 2

 .CSEG

 main:
    ldi reg, low(RAMEND)
    out SPL, reg
    ldi reg, high(RAMEND)
    out SPH, reg
    ser reg
    out DDRA, reg               //PortA as output
    clr reg
    out DDRB, reg               //PortB as input
    ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
    out DDRC ,r24
    ldi r24 ,0xfc ;11111100, PD7-2 output
    out DDRD ,r24
    ldi r26,low(_tmp_) ;r26-r27 -> X
    ldi r27,high(_tmp_)
    clr reg
    st X+,reg
    st X,reg
    sei

 initialize:
   rcall lcd_init
```

```
read_sensors:
   in reg,PINB
   cpi reg,0x00     ;reading sensors
   breq read_sensors

   ldi reg, (1<<TOIE1)
   out TIMSK, reg         ;timer1
   ldi reg, (1<<CS12)|(0<<CS11)|(1<<CS10)
   out TCCR1B, reg
         ldi reg, 0x67   ;Initialise start point of timer1 at 67(hex)
   out TCNT1H, reg ;so as to cause an interrupt after 5s
   ldi reg, 0x69
   out TCNT1L, reg
   ldi r24, 0x0E    ;show cursor
         rcall lcd_command

read_shift:
   clr flag       ;0,1,2->wrong combination,3->right
   clr temp       ;indicates number of keys pressed
   ldi r24, 10   ;Initialise for 0.01s delay
   rcall scan_keypad_rising_edge
   rcall keypad_to_ascii
   cpi r24,0x00
   breq read_shift
         push r24
   rcall lcd_data
         pop r24
   inc temp
   cpi r24,'4'
   brne read1st
   inc flag

read1st:
   ldi r24, 10   ;Initialise for 0.01s delay
   rcall scan_keypad_rising_edge
   rcall keypad_to_ascii
   cpi r24,0x00
   breq read1st
         push r24
   rcall lcd_data
   pop r24
         inc temp
   cpi r24,'1'
   brne read2nd
         inc flag

read2nd:
   ldi r24, 10   ;Initialise registers for 0.01s delay
   rcall scan_keypad_rising_edge
   rcall keypad_to_ascii
```

```
    cpi r24,0x00
    breq read2nd
        push r24
    rcall lcd_data
        ldi r24, 0x0C
        rcall lcd_command
    pop r24
        inc temp
    cpi r24,'2'
    brne wrong
    inc flag

    cpi flag,0x03
    brne wrong

right:
        ldi r25, HIGH(500)
    ldi r24, LOW(500)  ;Initialise registers for 0.5s delay
    rcall wait_msec    ;Wait for the last digit to show
        rcall lcd_init
    rcall ALARM_O
    ldi r24,'F'
    rcall lcd_data
    ldi r24,'F'
    rcall lcd_data
    ldi r25, HIGH(3000)
    ldi r24, LOW(3000) ;Initialise registers for 3s delay
    rcall wait_msec

loop_forever:
        rjmp loop_forever

wrong:
        ldi r25, HIGH(500)
    ldi r24, LOW(500) ;Initialise registers for 0.5s delay
    rcall wait_msec    ;Wait for the last digit to show
    rcall lcd_init
        rcall ALARM_O
    ldi r24,'N'
    rcall lcd_data
again:              ;on-off Leds forever
    ser reg
    out PORTA,reg
    ldi r25, HIGH(400)
    ldi r24, LOW(400) ;Initialise registers for 0.4s delay
    rcall wait_msec
    ldi r25, HIGH(100) ;Initialise registers for 0.1s delay
    ldi r24, LOW(100)
    clr reg
    out PORTA,reg
    rcall wait_msec
```

```
    jmp again


ISR_TIMER1_OVF:
   cpi flag,0x03
   breq leave
   cpi temp,0x03
   breq leave
         rcall lcd_init
   rcall ALARM_O
   ldi r24,'N'
   rcall lcd_data
on_off:
   ser reg
   out PORTA,reg
   ldi r25, HIGH(400)
   ldi r24, LOW(400)  ;Initialise registers for 0.4s delay
   rcall wait_msec
   ldi r25, HIGH(100)  ;Initialise registers for 0.1s delay
   ldi r24, LOW(100)
   clr reg
   out PORTA,reg
   rcall wait_msec
   jmp on_off
leave:
   ret

ALARM_O:
   ldi r24,'A'
   rcall lcd_data
   ldi r24,'L'
   rcall lcd_data
   ldi r24,'A'
   rcall lcd_data
   ldi r24,'R'
   rcall lcd_data
   ldi r24,'M'
   rcall lcd_data
   ldi r24,' '
   rcall lcd_data
   ldi r24,'O'
   rcall lcd_data
   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

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

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

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

scan_row:
     ldi r25,0x08
     back:
     lsl r25
     dec r24
     brne back
     out PORTC, r25
     nop
     nop
     in r24, PINC
     andi r24, 0x0f
     ret

 scan_keypad:

     ldi r24,0x01
```

```
    rcall scan_row
    swap r24
    mov r27,r24
    ldi r24,0x02
    rcall scan_row
    add r27,r24
    ldi r24,0x03
    rcall scan_row
    swap r24
    mov r26,r24
    ldi r24,0x04
    rcall scan_row
    add r26,r24
    movw r24,r26
    ret

scan_keypad_rising_edge:
    mov r22, r24
    rcall scan_keypad
    push r24
    push r25
    mov r24, r22
    clr r25
    rcall wait_msec
    rcall scan_keypad
    pop r23
    pop r22
    and r24,r22
    and r25,r23
    ldi r26,low(_tmp_) ;r26-r27 -> X
    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


keypad_to_ascii:
   movw r26 ,r24
   ldi r24 ,'*'
   sbrc r26 ,0
   ret
   ldi r24 ,'0'
   sbrc r26 ,1
   ret
   ldi r24 ,'#'
```

```
sbrc r26 ,2
ret
ldi r24 ,'D'
sbrc r26 ,3
ret
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'
sbrc r27 ,0
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
```

Σχόλια:

- Οι ρουτίνες που χρησιμοποιήθηκαν για το διάβασμα του πληκτρολογίου, την αρχικοποίηση της οθόνης και τη μεταφορά εντολών και δεδομένων προς την οθόνη έχουν παρθεί αυτούσιες από την εκφώνηση.

- Αρχικά το πρόγραμμα αναμένει διαρκώς το πάτημα κάποιου push button μεταξύ PB0 και PB7 που αντιστοιχούν στους αισθητήρες για την ενεργοποίηση του συναγερμού.
- Μόλις ενεργοποιηθεί κάποιο push button αρχικοποιείται ο χρονιστής ώστε να προκαλέσει διακοπή ύστερα από 5sec. Στη συνέχεια διαβάζονται από τα πληκτρολόγιο 3 χαρακτήρες. Εφόσον η εισαγωγή των χαρακτήρων γίνει εντός των 5sec (δηλαδή δεν έχει προκαλέσει διακοπή ο χρονιστής) και αντιστοιχούν στο σωστό συνδυασμό απενεργοποιείται ο συναγερμός και εμφανίζεται το μήνυμα ALARM OFF. Αν ο συνδυασμός είναι λανθασμένος ή δεν ολοκληρωθεί η εισαγωγή 3 χαρακτήρων εντός του προβλεπόμενου χρόνου ενεργοποιείται ο συναγερμός και εμφανίζεται το μήνυμα ALARM ON.
- Αν η ρουτίνα keypad_to_ascii επιστρέψει μέσω του καταχωρητή r24 την τιμή 0 συνεπάγεται ότι δεν έχει διαβαστεί κανένας αριθμός οπότε συνεχίζουμε να διαβάζουμε μέχρι να δοθεί κάποιος αριθμός.
- Στη ρουτίνα εξυπηρέτησης της διακοπής του χρονιστή ελέγχουμε αν το flag έχει την τιμή 3 που σημαίνει ότι δόθηκε σωστός συνδυασμός ή αν ο καταχωρητής temp έχει την τιμή 3 που σημαίνει ότι δόθηκαν 3 χαρακτήρες οπότε η διαχείριση της ενεργοποίησης ή μη του συναγερμού γίνεται από το κύριο πρόγραμμα. Σε κάθε άλλη περίπτωση (δηλαδή αν δεν έχουν δοθεί 3 χαρακτήρες εντός των 5sec) η ρουτίνα εξυπηρέτησης ενεργοποιεί το συναγερμό.
- Το αναβοσβήσιμο των leds στην περίπτωση ενεργοποίησης του συναγερμού επιτυγχάνεται με μία συνεχόμενη επαναληπτική διαδικασία όπου κάθε φορά ανάβουμε τα leds για 400ms και στη συνέχεια τα σβήνουμε για 100ms.
- Ανεξάρτητα από την ενεργοποίηση του συναγερμού ή μη το σύστημα επανέρχεται μόνο με reset.

Άσκηση 2

Ο κώδικας της άσκησης φαίνεται παρακάτω:

```
.include "m16def.inc"

.def reg = r18
.def hund = r19
.def deca = r17
.def units = r16
.def cnt = r20
.def flag = r21
.def temp = r22
.def mask = r23

reset:
   ldi reg, low(RAMEND)
   out SPL, reg
   ldi reg, high(RAMEND)
   out SPH, reg
   clr reg
   out DDRA, reg              ;PortA as input
```

```
        ser reg
        out DDRD,reg
main:
    rcall lcd_init
    clr flag        ;0->positive
        ldi mask,0x80
        ldi cnt,0x08
    in reg,PINA
        mov temp,reg
        and temp,mask
        cpi temp,0x00
    breq loopA
    ldi flag,0x01
loopA:

        cpi temp, 0x00
        breq pr0
        ldi r24, '1'
        rcall lcd_data
        jmp step
pr0:
        ldi r24, '0'
        rcall lcd_data
step:
        CLC
    ror mask
        mov temp,reg
        and temp,mask
    dec cnt
    brne loopA
    ldi r24,'='
    rcall lcd_data
    cpi reg,0x00
    breq print0
    cpi reg,0xff
    breq print0
    cpi flag,0x00
    breq positive
    ldi r24,'-'
    rcall lcd_data
    com reg
    jmp BCD
positive:
    ldi r24,'+'
    rcall lcd_data
BCD:
    clr hund
    clr deca
    clr units
loopB:          ;Convert hex number to decimal
    subi reg,0x64  ;Subtract 1 hundred
```

```
    brcs case1
    inc hund       ;hundreds++
    jmp loopB
case1:
    ldi temp,0x64
    add reg,temp   ;Restore one extra subtraction
    loopC:
    subi reg,0x0a  ;Subtract 1 ten
    brcs case2
    inc deca       ;tens++
    jmp loopC
case2:
    ldi temp,0x0a
    add reg,temp   ;Restore one extra subtraction
    mov units,reg  ;what is left is the units
    ldi flag,0x00  ;hund=0
    cpi hund,0x00
    breq dig2
    ldi flag,0x01
    mov r24,hund
          ldi reg,0x30
          add r24,reg
    rcall lcd_data
dig2:
    cpi flag,0x00
    brne show
    cpi deca,0x00
    breq dig1
show:
    mov r24,deca
          ldi reg,0x30
          add r24,reg
    rcall lcd_data
dig1:
    mov r24,units
          ldi reg,0x30
          add r24,reg
    rcall lcd_data
    ldi r25, HIGH(1000)
    ldi r24, LOW(1000)     ;Initialise registers for 1s delay
    rcall wait_msec
    jmp main

print0:
    ldi r24,'0'
    rcall lcd_data
    ldi r25, HIGH(1000)
    ldi r24, LOW(1000)              ;Initialise registers for 1s delay
    rcall wait_msec
          jmp main
```

```
 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

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

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

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
```

Σχόλια:

- Στον καταχωρητή flag αποθηκεύουμε το πρόσημο του αριθμού που δίνεται ως είσοδος. Συγκεκριμένα απομονώνουμε το MSB εφόσον ο αριθμός δίνεται σε μορφή συμπληρώματος ως προς 1 και αν είναι 0 πρόκειται για θετικό οπότε θέτουμε flag = 0 αλλιώς flag = 1.
- Στη συνέχεια απομονώνουμε ένα-ένα bit του αριθμού εισόδου και το εκτυπώνουμε. Ανάλογα με την τιμή του flag εκτυπώνουμε το σωστό πρόσημο

και αν πρόκειται για αρνητικό θεωρούμε το συμπλήρωμά του ως προς 1 (δηλαδή παίρνουμε την απόλυτη τιμή του). Στη συνέχεια μετατρέπουμε το δυαδικό αριθμό σε BCD με τη μέθοδο των επαναλαμβανόμενων αφαιρέσεων και εκτυπώνουμε τα δεκαδικά ψηφία αποφεύγοντας την εκτύπωση περιττών μηδενικών με κατάλληλους ελέγχους.

- Για να περιλαμβάνει η υλοποίηση και τη διπλή αναπαράσταση του μηδενός εκτελούμε στην αρχή δύο επιπρόσθετους ελέγχους για να διαπιστώσουμε αν ο αριθμός εισόδου είναι 0x00 ή 0xff οπότε και εκτυπώνουμε απευθείας 0 ως αποτέλεσμα.

Άσκηση 3

Ο κώδικας της άσκησης φαίνεται παρακάτω:

```
.include "m16def.inc"
.def reg = r20
.def temp = r21
.def Min = r16
.def Second = r17
.def tens = r18
.def units = r19

.org 0x00
rjmp reset

reset:
   ldi reg, low(RAMEND)
   out SPL, reg
   ldi reg, high(RAMEND)
   out SPH, reg
   clr reg
   out DDRB, reg       ;PortB as input
   ldi r24 ,0xfc   ;11111100, PD7-2 output
   out DDRD ,r24

main:
   rcall lcd_init
   clr Min
   clr Second
   rcall display_init_clk

start:
   in reg, PINB
   mov temp, reg
   andi temp, 0x80
   cpi temp, 0x80
```

```
    breq main
next:
    mov temp, reg
    andi temp, 0x01
    cpi temp, 0x01      ;Check for PB0 press
    brne start

    rcall inc_clock_and_display
    rjmp start

;This routine displays the following message to LCD display
;                   "00 MIN:00 SEC"
display_init_clk:
    ldi r24, '0'
    rcall lcd_display
        ldi r24, '0'
    rcall lcd_display
    ldi r24, ' '
    rcall lcd_display
    ldi r24, 'M'
    rcall lcd_display
    ldi r24, 'I'
    rcall lcd_display
    ldi r24, 'N'
    rcall lcd_display
    ldi r24, ':'
    rcall lcd_display
    ldi r24, '0'
    rcall lcd_display
        ldi r24, '0'
    rcall lcd_display
    ldi r24, ' '
    rcall lcd_display
    ldi r24, 'S'
    rcall lcd_display
    ldi r24, 'E'
    rcall lcd_display
    ldi r24, 'C'
    rcall lcd_display
ret


inc_clock_and_display:
    push temp
```

```
    push reg
    inc Second
    cpi Second, 0x3C
    brne display
    clr Second              ;If Sec=60(10) Sec=0 and Min=Min+1
    inc Min
    cpi Min, 0x3C
    brne display
    clr Min             ;If Min=60(10) Min=0
display:
    ldi r24, LOW(1000)
    ldi r25, HIGH(1000)
    rcall wait_msec      ;Wait for 1s
          rcall lcd_init
    mov reg, Min
    rcall hex_to_bcd     ;Convert minutes to BCD
    mov temp, tens
    ldi reg, 0x30
    add temp, reg        ;Convert to ASCII code
    mov r24, temp
    rcall lcd_display      ;Display
    mov temp, units
    add temp, reg        ;Convert to ASCII code
    mov r24, temp
    rcall lcd_display       ;Display minutes

    ;Display " MIN:"
    ldi r24, ' '
    rcall lcd_display
    ldi r24, 'M'
    rcall lcd_display
    ldi r24, 'I'
    rcall lcd_display
    ldi r24, 'N'
    rcall lcd_display
    ldi r24, ':'
    rcall lcd_display

    mov reg, Second
    rcall hex_to_bcd     ;Convert seconds to BCD
    mov temp, tens
    ldi reg, 0x30
    add temp, reg        ;Convert to ASCII code
    mov r24, temp
```

```
    rcall lcd_display       ;Display
    mov temp, units
          out PORTA, units
          ldi reg, 0x30
    add temp, reg        ;Convert to ASCII code
    mov r24, temp
    rcall lcd_display     ;Display seconds

    ;Display " SEC"
    ldi r24, ' '
    rcall lcd_display
    ldi r24, 'S'
    rcall lcd_display
    ldi r24, 'E'
    rcall lcd_display
    ldi r24, 'C'
    rcall lcd_display

    pop reg
    pop temp
ret

;This routine takes a hex number as input stored in reg register
;and returns the BCD representation of this number
;The units are stored in units register and the tens in tens register
hex_to_bcd:
     push temp
     clr tens
     clr units
loopA:
     subi reg, 0x0A  ;Subtract 1 ten
     brcs case1
     inc tens        ;tens++
     jmp loopA
 case1:
     ldi temp,0x0A
     add reg,temp    ;Restore one extra subtraction
     mov units, reg
           pop temp
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

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

lcd_display:
   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

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
```

Σχόλια:

- Ο κώδικας ελέγχει πρώτα αν είναι πατημένο το push button PB7 που αντιστοιχεί στο πλήκτρο reset του χρονομέτρου. Αν είναι πατημένο αρχικοποιείται ξανά η οθόνη και εμφανίζεται η αρχική ένδειξη “00 MIN:00 SEC”.
- Στη συνέχεια ελέγχεται αν είναι ενεργοποιημένο το πλήκτρο για τη λειτουργία του χρονομέτρου που αντιστοιχεί στο push button PB0. Αν δεν είναι ενεργοποιημένο οι ενδείξεις παραμένουν σταθερές στην οθόνη και επαναλαμβάνονται διαρκώς οι παραπάνω έλεγχοι.
- Εφόσον το χρόνομετρο βρίσκεται σε λειτουργία καλείται η ρουτίνα inc_clock_and_display η οποία αρχικά αυξάνει τα δευτερόλεπτα κατά ένα εφόσον δεν υπερβαίνουν την τιμή 59. Διαφορετικά μηδενίζει τα δευτερόλεπτα και αυξάνει τα λεπτά κατά ένα, ενώ αν το χρονόμετρο βρίσκεται στο σημείο 59:59 επανέρχεται στο 00:00. Στη συνέχεια προκαλείται καθυστέρηση ενός δευτερολέπτου με τη βοήθεια της ρουτίνας wait_msec.
- Επιπλέον η ρουτίνα inc_clock_and_display μετατρέπει τα λεπτά και τα δευτερόλεπτα σε BCD με κλήση της ρουτίνας hex_to_bcd και εκτυπώνει το κατάλληλο μήνυμα στην οθόνη “”XX MIN:XX SEC”.