ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 3
Ομάδα 7 – Group 1
Χατζηθωμά Αντρέας 8026
Νήρας Δημήτρης 8057

Στην αρχή του προγράμματος κάνουμε define τους καταχωρητές R17-R25 για διευκόλυνση στη συνέχεια του προγράμματος και ακολούθως τους αρχικοποιούμε. Έπειτα αρχικοποιούμε τον stack pointer και ορίζουμε ως έξοδο το PortB και ως είσοδο το PortD. Στη συνέχεια μέσα στην main ελέγχουμε ποιος από τους 8 διακόπτες ενεργοποιείτα και αντιστοίχως καλούμε την εκάστοτε διαδικασία όπως ορίζεται στην εκφώνηση. Στην main επίσης υπάρχουν και οι έλεγχοι για το αν είναι ανοικτή η κρύα ή η ζεστή βάνα και αντιστοίχως αυξάνουμε τους μετρητές counter_cold και counter_hot κατά ένα για κάθε ένα δευτερόλεπτο λειτουργίας. Ο timer υλοποιήθηκε μέσω επαναληπτικών διαδικασιών for, όπως ακριβώς και στις προηγούμενες εργασίες. Το πρόγραμμα τερματίζει σε 2 περιπτώσεις:

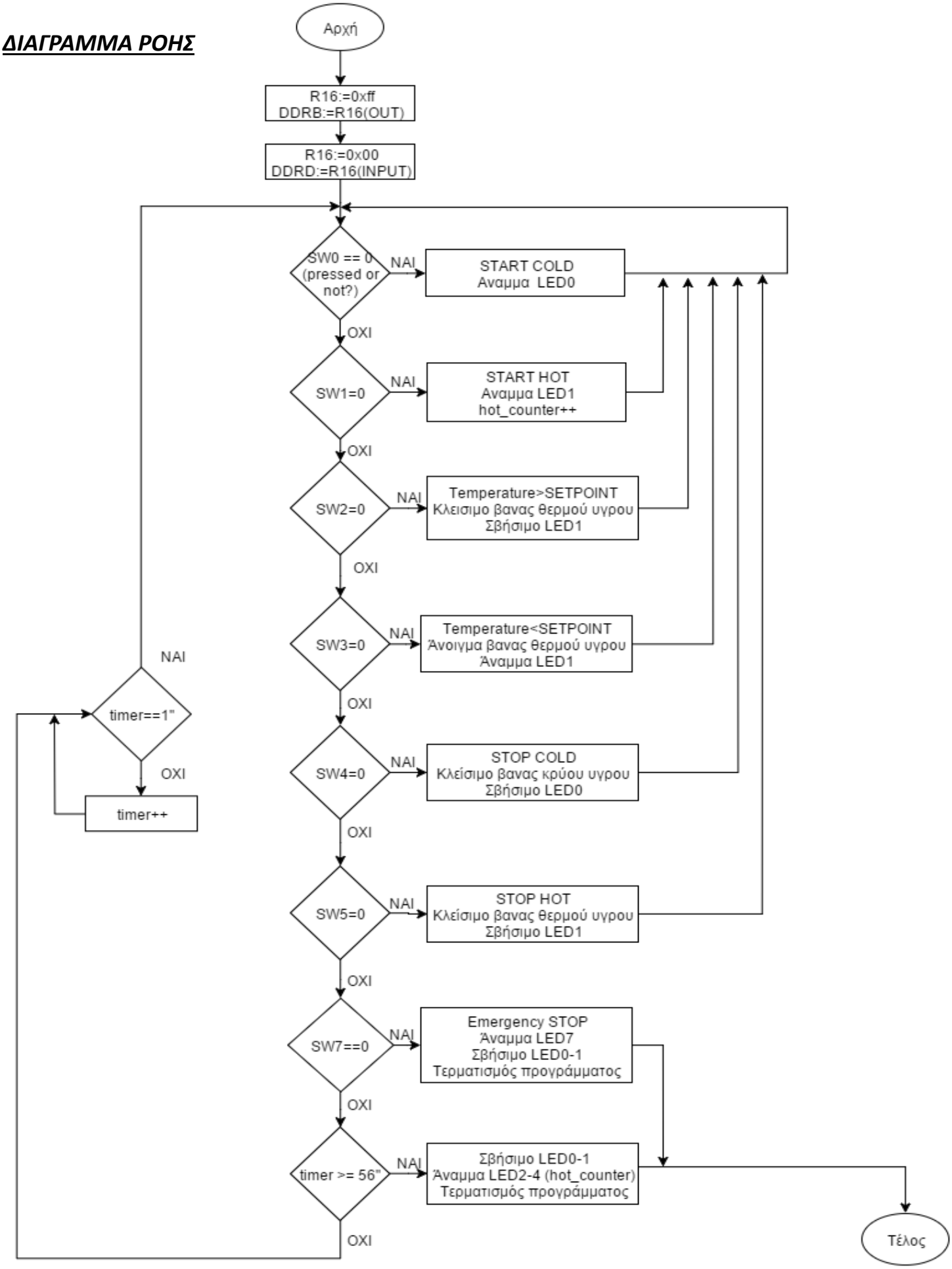
1. Όταν περάσουν 56 δευτερόλεπτα λειτουργίας και επομένως εμφανίζονται στα LEDs 2-4 ο αριθμός των ενεργοποιήσεων της ζεστής βάνας μέσω της διαδικασίας program end.
2. Όταν ενεργοποιηθεί ο διακόπτης SW7 και το πρόγραμμα τερματίζεται ακαριαία και κλείνουν και οι 2 βάνες.

Δυσκολευτήκαμε στην διαδικασία μέτρησης του χρόνου στον οποίο ήταν ανοιχτές βάνες καθώς έπρεπε να γίνει παράλληλα με τον έλεγχο ενεργοποίησης των διακοπτών.

BREAKPOINTS-DEBUGGING

Breakpoints τοποθετήθηκαν στα σημεία που καλείται κάθε διαδικασία ούτως ώστε να ελέγξουμε αν ανάβουν σωστά τα LEDs.
Επίσης, κατά την εκτέλεση του προγράμματος χρησιμοποιήθηκαν μικρότερα delays για ευκολία κατά το debugging. Στο εργαστήριο χρησιμοποιήθηκαν τα σωστά delays με τους σωστούς χρόνους, οι οποίοι χρονομετρήθηκαν για τον έλεγχο ορθότητας τους.

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ



<u>ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ</u>

```
;lab3
;Χατζηθωμά Αντρέας 8026
;Νήρας Δημήτρης 8057
;Define καταχωρητών και αρχικοποίηση stack pointer
;Έλεγχος ενεργοποίησης διακοπτών και ενεργοποίηση αντίστοιχης λειτουργίας
;Έλεγχος μετρητή και τερματισμός προγράμματος όταν περάσουν 56 δευτερόλεπτα

.include "m16def.inc"
.org 0x0000
.cseg

;define registers
.def leds=r17
.def hot_counter=r18
.def temp=r19
.def hot_flag=r20
.def cold_flag=r21
.def counter_hot=r22
.def counter_cold=r23
.def one=r24
.def fifty_six=r25

;initialize variables
clr leds
clr hot_counter
clr temp
clr hot_flag
clr cold_flag
clr counter_hot
clr counter_cold
ldi one, 1
ldi fifty_six, 56

;initialize the stack
sp_init:
        ldi r16, low(RAMEND)
        out spl, r16
        ldi r16, high(RAMEND)
        out sph, r16

;set portB as output (LEDs)
output:
        ser r16
        out DDRB, r16
        out PORTB, r16

;set portD as input (Switches)
```

```
input:
        clr r16
        out DDRD, r16
        ser r16
        out PIND, r16

;check if a switch is pressed
main:
        sbis PIND, 0x00             ;If (Port D, pin0 == 0) do
        rcall START_COLD

        sbis PIND, 0x01             ;If (Port D, pin1 == 0) do
        rcall START_HOT

        sbis PIND, 0x02             ;If (Port D, pin2 == 0) do
        rcall BGT_SETPOINT

        sbis PIND, 0x03         ;If (Port D, pin3 == 0) do
        rcall LST_SETPOINT

        sbis PIND, 0x04         ;If (Port D, pin4 == 0) do
        rcall STOP_COLD

        sbis PIND, 0x05         ;If (Port D, pin5 == 0) do
        rcall STOP_HOT

        sbis PIND, 0x07             ;If (Port D, pin7 == 0) do
        rjmp EMERGENCY_STOP
cold:
        cp cold_flag, one       ;Compare cold_flag with one
        brne hot                    ;branch if not egual (cold_flag, one)
        ;rcall timer
        inc counter_cold        ;counter_cold++

        cp hot_flag, one        ;Compare hot_flag with one
        brne loop                   ;branch if not egual (hot_flag, one)
        inc counter_hot             ;counter_hot++
        rjmp loop

hot:
        cp hot_flag, one        ;Compare hot_flag with one
        brne loop                   ;branch if not egual (hot_flag, one)
        ;rcall timer
        inc counter_hot             ;counter_hot++

loop:
        mov r26, counter_hot
        mov r27, counter_cold
        add r26, r27                ;add counter_hot+counter_cold
        cp r26, fifty_six           ;compare r26 (counter_hot+counter_cold) with fifty_six
```

```
        brge PROGRAM_END        ;branch if not equal

        rjmp main

;timer for 1 second
;255x255x21 loops
timer:
        ldi r28, 0xff
timer_3:
        ldi r29, 0xff
timer_2:
        ldi r30, 0x15
timer_1:
        dec r30
        brne timer_1
        dec r29
        brne timer_2
        dec r28
        brne timer_3
        ret

PROGRAM_END:
        ser leds
        out PORTB, leds                 ;LEDs OFF

        lsl hot_counter                 ;shift left hot_counter -> 0000XXX0
        lsl hot_counter                 ;shift left     hot_counter -> 000XXX00
        com hot_counter                 ;NOT hot counter -> 111XXX11
        out PORTB, hot_counter      ;LED2-LED4 -> ON

end_loop:
        rjmp end_loop

START_COLD:
        sbic PIND, 0x00             ;Skip next instruction if (Port D, pin0 == 0)
        rjmp start_next
        rjmp START_COLD
start_next:
        ori leds, 0b00000001 ;LED0 -> ON
        andi leds, 0b01011111       ;LED5+LED7 -> OFF
        mov temp, leds
        com temp                    ;NOT temp (LEDs Active Low Logic)
        out PORTB, temp             ;LEDs -> temp

        ldi cold_flag, 1

        ret

START_HOT:
        sbic PIND, 0x01             ;Skip next instruction if (Port D, pin1 == 0)
```

```
        rjmp next
        rjmp START_HOT
next:
        ori leds, 0b00000010 ;LED1 -> ON
        andi leds, 0b00111111       ;LED6-7 -> OFF
        mov temp, leds
        com temp                    ;NOT temp
        out PORTB, temp             ;LEDs -> temp

        inc hot_counter             ;hot_counter++
        ldi hot_flag, 1

        ret

;Temperature>SETPOINT
BGT_SETPOINT:
        sbic PIND, 0x02             ;Skip next instruction if (Port D, pin2 == 0)
        rjmp bgt_next
        rjmp BGT_SETPOINT
bgt_next:
        andi leds, 0b11111101       ;LED1 -> OFF
        mov temp, leds
        com temp                    ;NOT temp
        out PORTB, temp             ;LEDs -> temp

        ldi hot_flag, 0

        ret

;Temperature<SETPOINT
LST_SETPOINT:
        sbic PIND, 0x03             ;Skip next instruction if (Port D, pin3 == 0)
        rjmp lst_next
        rjmp LST_SETPOINT
lst_next:
        ori leds, 0b00000010 ;LED1 -> ON
        mov temp, leds
        com temp
        out PORTB, temp             ;LEDs -> temp

        inc hot_counter
        ldi hot_flag, 1

        ret

STOP_COLD:
        sbic PIND, 0x04             ;Skip next instruction if (Port D, pin4 == 0)
        rjmp cold_next
        rjmp STOP_COLD
cold_next:
```

```
        ori leds, 0b00100000 ;LED5 -> ON
        andi leds, 0b00100010       ;LED0+LED2-4+LED6-7 -> OFF
        mov temp, leds
        com temp
        out PORTB, temp             ;LEDs -> temp

        ldi cold_flag, 0

        ret

STOP_HOT:
        sbic PIND, 0x05             ;Skip next instruction if (Port D, pin5 == 0)
        rjmp hot_next
        rjmp STOP_HOT
hot_next:
        ori leds, 0b01000000 ;LED6 -> ON
        andi leds, 0b01100001       ;LED1-4+LED7 -> OFF
        mov temp, leds
        com temp
        out PORTB, temp             ;LEDs -> temp

        ldi hot_flag, 0

        ret

EMERGENCY_STOP:
        sbic PIND, 0x07             ;Skip next instruction if (Port D, pin7 == 0)
        rjmp emergency_next
        rjmp EMERGENCY_STOP
emergency_next:
        ori leds, 0b10000000 ;LED7 -> ON
        andi leds, 0b10000000       ;LED0-6 -> OFF
        mov temp, leds
        com temp
        out PORTB, temp             ;LEDs -> temp

        ldi cold_flag, 0
        ldi hot_flag, 0

;infinite loop (end loop)
endloop:
        rjmp endloop
```