**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**

**ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**



**Μάθημα: Εργαστήριο Μικροϋπολογιστών**

**Αναφορά 5ς Σειράς Εργαστηριακών Ασκήσεων**

**7ο Εξάμηνο**

|  |  |
| --- | --- |
| **Ομάδα Δ15** | |
| **Ονοματεπώνυμο** | **Αριθμός Μυτρώου** |
| **Ιωάννης Μύτης** | **031 13133** |
| **Νικόλαος Μαρίτσας** | **031 14934** |
| **Μιχάλης Παπαδόπουλλος** | **031 14702** |

* **Άσκηση 5.1**

; DESCRIPTION:

; Read INPUT from KEYPAD (4x4) PORTC

; if INPUT == 15 then BLINK\_YES (LEDS PA[7:0] ON FOR 4 SECONDS)

; else BLINK\_NO (LEDS PA[7:0] ON FOR 8 ROUNDS OF ON/OFF = 0.25 + 0.25 SECONDS)

.include "m16def.inc"

; data segment

.dseg

\_tmp\_ : .byte 2

; code segment

.cseg

.def cnt=r16

; ------------------------------------------------

; =============== [MACROS] ===============

; ------------------------------------------------

.macro SET\_LEDS\_ON

; MACRO: SET ALL LEDS OF PORTA TO ON

; AFFECTED REGISTER:

ser r18

out PORTA,r18

.endm

; ------------------------------------------------

.macro SET\_LEDS\_OFF

; MACRO: SET ALL LEDS OF PORTA TO ON

; AFFECTED REGISTER: r20

clr r20

out PORTA,r20

.endm

; ------------------------------------------------

.org 0x0

rjmp INIT ; ON RESET JUMP TO INIT

INIT:

; INIT STACK

ldi r24,low(RAMEND)

ldi r25,high(RAMEND)

out SPL,r24

out SPH,r25

; SET PORTA AS OUTPUT

ser r18 ; r18 = 1111 1111

out DDRA,r18

; === [DEBUGGING] ===

out DDRD,r18

; === [DEBUGGING] ===

; SETUP KEYPAD

andi r18,0xf0 ; r18 = 1111 0000

out DDRC,r18 ; PORTC[7:4] OUTPUT, PORTC[3:0] INPUT

MAIN:

; START WITH LEDS OFF

clr r18

out PORTA,r18

; =========== [SCAN KEYPAD] ===========

SCAN\_FST:

; r24 xronos spinthirismou (\*tha doume stin pra3i poio value volevei\*)

ldi r24,0xBC

; r25: [A|3|2|1]|[B|6|5|4]

; r24: [C|9|8|7]|[D|#|0|\*]

; r25:r25 holds input

rcall scan\_keypad\_rising\_edge

adiw r24,0x0000 ; if r24 == 0 then

breq SCAN\_FST ; goto SCAN\_FST

; =============================================

; EPISTREFEI r24=0 AN DEN PATITHIKE KNAS DIAKOPTIS

rcall keypad\_to\_ascii

; === [DEBUGGING] ===

out PORTD,r24

; === [\DEBUGGING] ===

cpi r24,0x31 ; check if '1' pressed

brne B\_NO ; if YES then continue else goto BLINK\_NO (jumps to MAIN)

; =============================================

SCAN\_SND:

; delay value (spinthirismos)

ldi r24,0xBC

; r25: [A|3|2|1][B|6|5|4]

; r24: [C|9|8|7][D|#|0|\*]

; r25:r25 holds input

rcall scan\_keypad\_rising\_edge

adiw r24,0x0000

breq SCAN\_SND

; =============================================

; CONVERT TO ASCII

rcall keypad\_to\_ascii ; EPISTREFEI r24=0 AN DEN PATITHIKE KNAS DIAKOPTIS

; === [DEBUGGING] ===

out PORTD,r24

; === [\DEBUGGING] ===

cpi r24,0x35 ; if r24 == 5 then

brne B\_NO ; goto SCAN\_SND

; =============================================

B\_YES:

rcall BLINK\_YES ; BLINK BLINK\_YES (jumps to MAIN)

rjmp MAIN

B\_NO:

rcall BLINK\_NO ; BLINK BLINK\_NO (jumps to MAIN

rjmp MAIN ; PROGRAMMA SYNEXOUS LEITOURGIAS

; ------------------------------------------------

; =============== [PROCEDURES] ===============

; ------------------------------------------------

BLINK\_YES:

; LEDS PA[7:0] OPEN FOR 4 SECONDS

SET\_LEDS\_ON ; ALL LEDS ON (MACRO)

ldi r24,low(4000)

ldi r25,high(4000)

rcall wait\_msec ; DELAY 4 SECONDS (MACRO)

ret

; ------------------------------------------------

BLINK\_NO:

; LEDS PA[7:0] BLINK ON/OF FOR 4 SECONDS

ldi cnt,0x08 ; iterate 8 times

L1:

SET\_LEDS\_ON ; set leds on (MACRO)

ldi r24,low(250)

ldi r25,high(250)

rcall wait\_msec ; delay 0.25msec (MACRO)

SET\_LEDS\_OFF ; set leds off (MACRO)

ldi r24,low(250)

ldi r25,high(250)

rcall wait\_msec

dec cnt ; cnt--

cpi cnt, 0x0 ; (cnt == 0) ?

brne L1 ; if cnt != 0 goto L1

ret

; ------------------------------------------------

wait\_usec:

sbiw r24,1 ;2 cycles (0,250 usec)

nop ;1 (0,125 usec)

nop

nop

nop

brne wait\_usec ;1 or 2 cycles

ret ;4 cycles (0,5 usec)

; ------------------------------------------------

wait\_msec:

push r24 ;2

push r25 ;2

ldi r24,low(998) ;fortwse ton r25:r24 me 998 ( 1 cc - 0,125 usec)

ldi r25,high(998) ;1

rcall wait\_usec ;3 cycles (0,375usec), kathisterhsh 998,375

pop r25 ;2

pop r24 ;2

sbiw r24,1 ;2

brne wait\_msec ;1 or 2

ret ;4

; ------------------------------------------------

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

ldi r25,0

rcall wait\_msec

rcall scan\_keypad

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

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

* **Άσκηση 5.1c**

#include <avr/io.h>

#include <util/delay.h>

/\*

;------------------------------------------------

;=============== [MACROS] ===============

;------------------------------------------------

\*/

#define SET\_LEDS\_ON (PORTA=0xff)

#define SET\_LEDS\_OFF (PORTA=0x00)

void BLINK\_NO (void) {

PORTA=0xff;

*\_delay\_ms*(250);\

PORTA=0x00;

*\_delay\_ms*(250);\

PORTA=0xff;

*\_delay\_ms*(250);\

PORTA=0x00;

*\_delay\_ms*(250);\

PORTA=0xff;

*\_delay\_ms*(250);\

PORTA=0x00;

*\_delay\_ms*(250);\

PORTA=0xff;

*\_delay\_ms*(250);\

PORTA=0x00;

*\_delay\_ms*(250);\

PORTA=0xff;

*\_delay\_ms*(250);\

PORTA=0x00;

*\_delay\_ms*(250);\

PORTA=0xff;

*\_delay\_ms*(250);\

PORTA=0x00;

*\_delay\_ms*(250);\

PORTA=0xff;

*\_delay\_ms*(250);\

PORTA=0x00;

*\_delay\_ms*(250);\

PORTA=0xff;

*\_delay\_ms*(250);\

PORTA=0x00;

*\_delay\_ms*(250);\

}

void BLINK\_YES (void) {

PORTA=0xff;

*\_delay\_ms*(4000);

}

/\*

; ------------------------------------------------

; =============== [PROCEDURES] ===============

; ------------------------------------------------

\*/

/\* GLOBAL VARIABLES \*/

unsigned char column,row,input;

unsigned char keypad[4][4] = {

{'0','1','2','3'},

{'4','5','6','7'},

{'8','9','A','B'},

{'C','D','E','F'}

};

unsigned int GND\_PORTS [] = { 0xEF, 0xDF, 0xBF, 0x7F };

/\* FUNCTION PROTOTYPES \*/

// void scan\_row(void);

void read\_keypad(void);

int main(void)

{

/\*

PORTA => OUTPUT (LEDS)

KEYPAD =>

PORTC[3:0] => INPUT=0

PORTC[4:7] => OUTPUT=1

\*/

while(1){

// INIT:

DDRA = 0xff;

DDRC = 0xf0;

MAIN\_L:

PORTA = 0x0;

// SPINTHIRISMOS

//SCAN\_FST:

read\_keypad();

if (input != '1') {

BLINK\_NO();

goto MAIN\_L;

}

//SCAN\_SND:

if (input != '5') {

BLINK\_NO();

goto MAIN\_L;

}

BLINK\_YES();

}

}

/\*

void scan\_row(void)

{

do {

PORTC &= 0x0f;

column = (PINC & 0x0f);

} while (column != 0x0f);

}

\*/

void read\_keypad(void) {

int i = 0;

do {

do {

*\_delay\_ms*(20);

column = PINC & 0x0f;

} while (column != 0x0f);

*\_delay\_ms*(20);

column = PINC & 0x0f;

} while (column != 0x0f);

while (1) {

for (i = 0; i < 4; i++) {

PORTC &= GND\_PORTS[i];

column = PINC & 0x0f;

if (column != 0x0f) {

row = i;

goto exit\_w;

}

}

}

exit\_w:

for (i = 0; i < 4; i++) {

if (column == GND\_PORTS[i]) {

input = keypad[row][i];

}

}

}

* **Άσκηση 5.2**

Εδώ δεχόμαστε από το πληκτρολόγιο έναν διψήφιο δεκαεξαδικό αριθμό μέσω της ρουτίνας scan\_keypad\_rising\_edge και δίνουμε τα δύο ψηφία στον r19 και στον r24. Στην συνέχεια υπολογίζουμε τον αριθμό και τον βάζουμε στον temp σε δυαδική μορφή και εμφανίζουμε τον αριθμό σε hex στην οθόνη μέσω της ρουτίνας lcd\_data . Έπειτα υπολογίζουμε τις εκατοντάδες, δεκάδες και μονάδες με επαναληπτικές αφαιρέσεις και τις εμφανίζουμε στην οθόνη αφού τα μετατρέψουμε σε ascii.  
Τέλος γυριζουμε στην αρχή του προγραμματος όπου περιμένουμε για input απο το πληκτρολογιο

; Functionality:

; - Read a 2 digit and signed hex number from keyboard

; - Convert number to 3 digit Dec number

; - Output number to the LED screen

; - Repeat

.DSEG

\_tmp\_: .byte 2

.CSEG

.include "m16def.inc"

.def temp=r20

.org 0x00

rjmp init

.macro Bit\_to\_Ascii

cpi r24,10

brge let ;if it's 0-9 add 30h else add 37h

mov r25,0x30

add r24,r25

rjmp exit

let:

mov r25,0x37

add r24,r25

exit:

.endm

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_MAIN\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

init:

clr temp

ldi r24, low(RAMEND)

out SPL, r24

ldi r24, high(RAMEND)

out SPH, r24

ser r24

out DDRD, r24

out DDRA, r24 ; r25: [A|3|2|1]|[B|6|5|4]

; r24: [C|9|8|7]|[D|#|0|\*]

ldi temp,0xf0 ; r18 = 1111 0000

out DDRC,temp ; PORTC[7:4] OUTPUT, PORTC[3:0] INPUT

rcall lcd\_init ;initiallize lcd values

clr r25

start:

ldi r24,20

rcall scan\_keypad\_rising\_edge ;r17-18 has most significant hex number bit, ;r24-25 has least significant number bit

sbiw r24,0

breq start

movw r16,r24

ldi r24 ,0x01 ;reset screen for new numbers

rcall lcd\_command

ldi r24 ,low(1530)

ldi r25 ,high(1530)

rcall wait\_usec

digit2:

ldi r24,20

rcall scan\_keypad\_rising\_edge

sbiw r24,0

breq digit2

rcall keypad\_to\_num ;lsn (least significant number) in hex number in ;r19 and msn in r24

mov r19,r24

movw r24,r16

rcall keypad\_to\_num

mov temp,r24 ;temp has 2 digit full number

swap temp

add temp,r19

out PORTA,temp

bit\_to\_ascii ;gets r24 and converts it to ascii code

rcall lcd\_data ;print msn

mov r24,r19

bit\_to\_ascii

rcall lcd\_data ;print lsn

ldi r24,'='

rcall lcd\_data

sbrc temp,7 ;if last bit is 1 then number is negative

ldi r24,'-'

sbrs temp,7

ldi r24,'+'

rcall lcd\_data ;operation print

sbrc temp,7 ;get temp to its correct form

neg temp

rcall print\_dec

rjmp start

print\_dec:

ser r16

subi temp,100 ; if num >100 print 1 and then continue with printing next digits

brcs below ; else add 100 back and continue with next digits

ldi r24,'1'

rcall lcd\_data ;print ekatodades

rjmp noadd

below:

ldi r24,100

add temp,r24

noadd:

inc r16

subi temp,10

brcc noadd

ldi r24,10

add temp,r24 ;reset last num

mov r24,r16 ;print decades

bit\_to\_ascii

rcall lcd\_data

mov r24,temp ;print monades

bit\_to\_ascii

rcall lcd\_data

ret

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_LCD\_routines\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lcd\_data:

sbi PORTD ,PD2 ;epilogh tou data register (PD2=1)

rcall write\_2\_nibbles ;apostolh tou byte

ldi r24 ,43 ;anamonh 43µsec mexri na oloklhrwthei h lhpsh

ldi r25 ,0 ;twn dedomenwn apo ton elegkth ths lcd

rcall wait\_usec

ret

lcd\_command:

cbi PORTD ,PD2 ;epilogh tou command register (PD2=0)

rcall write\_2\_nibbles ;apostolh ths entolhs kai anamonh 39usec

ldi r24 ,39 ;gia thn oloklhrwsh ths ekteleshs ths apo ton elegkth ths lcd.

ldi r25 ,0 ;uparxoun duo entoles, oi clear display kai return home,

rcall wait\_usec ;pou apaitoun shmantika megalytero xroniko diasthma

ret

write\_2\_nibbles:

push r24 ;stelnei ta 4 MSB

in r25 ,PIND ;diavazontai ta 4 LSB kai ta ksanastelnoume

andi r25 ,0x0f ;gia na mhn xalasoume thn opoia prohgoumenh katastasj

andi r24 ,0xf0 ;apomonwnontai ta 4 MSB kai

add r24 ,r25 ;sunduazontai me ta prouparxonta 4 LSB

out PORTD ,r24 ;kai dinontai sthn eksodo

sbi PORTD ,PD3 ;dhmiourgeitai palmos enable ston akrodekth PD3

cbi PORTD ,PD3 ;PD3=1 kai µeta PD3=0

pop r24 ;stelnei ta 4 LSB. anaktatai to byte

swap r24 ;enallassontai ta 4 MSB me ta 4 LSB

andi r24 ,0xf0 ;pou me thn siera tous apostellontai

add r24 ,r25

out PORTD ,r24

sbi PORTD ,PD3 ;neos palmos enable

cbi PORTD ,PD3

ret

lcd\_init:

ldi r24 ,40 ;otan o elekths ths lcd trofodoteitai me

ldi r25 ,0 ;reuma prwth fora ektelei thn dikh tou arxikopoihsh.

rcall wait\_msec ;anamonh 40 msec mexri auth na oloklhrwthei

ldi r24 ,0x30 ;entolh metavashs se 8 bit mode

out PORTD ,r24 ;epeidh den mporoume na eimaste vevaioi

sbi PORTD ,PD3 ;gia th diamorfwsh eisodou tou elegkth

cbi PORTD ,PD3 ;ths othonhs, h entolh apostelletai dyo fores

ldi r24 ,39

ldi r25 ,0 ;ean o elegkths ths othonhs vrisketai se 8-bit mode

rcall wait\_usec ;de tha sumvei tipota, alla an o elegkths exei diamorfwsh

;eisodu 4 bit tha metavei se diamorfwsh 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 ;allagh se 4-bit mode

out PORTD ,r24

sbi PORTD ,PD3

cbi PORTD ,PD3

ldi r24 ,39

ldi r25 ,0

rcall wait\_usec

ldi r24 ,0x28 ;epilogh xarakthrwn megethous 5x8 koukidwn

rcall lcd\_command ;kai emfanish 2 grammwn sthn othonh

ldi r24 ,0x0c ;energopoihsh ths othonhs, apokrypsh tou kersora

rcall lcd\_command

ldi r24 ,0x01 ;katharismos ths othonhs

rcall lcd\_command

ldi r24 ,low(1530)

ldi r25 ,high(1530)

rcall wait\_usec

ldi r24 ,0x06 ;energopoihsh automaths aukshshs kata 1 ths dieuthynshs

rcall lcd\_command ;pou einai apothhkaumenh sto metrhth dieuthynsewn kai

;apenergopoihsh ths olisthishs oloklhrhs ths othonhs

ret

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Keypad\_Routines\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

ldi r25 ,0

rcall wait\_msec

rcall scan\_keypad

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

ret

keypad\_to\_num:

movw r26 ,r24

ldi r24 ,0x0E

sbrc r26 ,0

ret

ldi r24 ,0

sbrc r26 ,1

ret

ldi r24 ,0x0F

sbrc r26 ,2

ret

ldi r24 ,0x0D

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 ,0x0C

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 ,0x0B

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 ,0x0A

sbrc r27 ,7

ret

clr r24

ret

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Delay\_routines\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

wait\_usec:

sbiw r24 ,1 ;2 cycles (0.250 micro sec)

nop ;1 cycles (0.125 micro sec)

nop ;1 cycles (0.125 micro sec)

nop ;1 cycles (0.125 micro sec)

nop ;1 cycles (0.125 micro sec)

brne wait\_usec ;1 or 2 cycles (0.125 or 0.250 micro sec)

ret ;4 cycles (0.500 micro sec)

wait\_msec:

push r24 ;2 cycles (0.250 micro sec)

push r25 ;2 cycles

ldi r24 , 0xe6 ;load register r25:r24 with 998 (1 cycles - 0.125 micro sec)

ldi r25 , 0x03 ;1 cycles (0.125 micro sec)

rcall wait\_usec ;3 cycles (0.375 micro sec), total delay 998.375 micro sec

pop r25 ;2 cycles (0.250 micro sec)

pop r24 ;2 cycles

sbiw r24 , 1 ;2 cycles

brne wait\_msec ;1 or 2 cycles (0.125 or 0.250 micro sec)

ret

* **Άσκηση 5.3**

.include "m16def.inc"

.org 0x0

rjmp reset

.def minutes=r20

.def seconds=r21

reset:

ldi r26,low(RAMEND) ;stack initialization

out SPL,r26

ldi r26,high(RAMEND)

out SPH,r26

clr r26 ;portA eisodos

out DDRA,r26

ser r26 ;portD eksodos

out DDRD,r26

rcall lcd\_init

rcall arxikopoihsh

loop1:

in r22,PINA

ror r22 ;koitaw to bit0

brcs loop2

rjmp sunexeia

loop2:

rcall arxikopoihsh

rjmp loop1

sunexeia:

rol r22

rol r22 ;gia na parw sto kratoymeno to bit 7

brcs sunexeia2

rjmp loop1

sunexeia2:

` ldi r24,low(1000)

ldi r25,high(1000)

rcall wait\_msec

cpi seconds,0x3b ;an eftasa sta 59 sec

breq lepta ;phgaine sta lepta

inc seconds

ldi r24,0x88 ;paw sthn thesi gia seconds,sugkekrimena gia tis monades

rcall lcd\_command

mov r24,seconds

rcall lcd\_number ;emfanizw ta deuterolepta

rjmp loop1

lepta:

cpi minutes,0x3b

breq loop2

ldi seconds,0

ldi r24,0x88 ;paw sthn thesi gia seconds,sugkekrimena gia tis monades

rcall lcd\_command

mov r24,seconds

rcall lcd\_number

inc minutes

ldi r24,0x81 ;paw sthn thesi gia minutes,sugkekrimena gia tis monades

rcall lcd\_command

mov r24,minutes

rcall lcd\_number

rjmp loop1

arxikopoihsh:

ldi minutes,0

ldi seconds,0

ldi r24,0x80 ;paw sthn prwth thesi ths othonis

rcall lcd\_command

ldi r24,0x30 ;fortwnw to 0

rcall lcd\_data ;to emfanizw

ldi r24,0x81 ;paw sthn deuteri thesi ths othonis

rcall lcd\_command

ldi r24,0x30 ;fortwnw to 0

rcall lcd\_data

ldi r24,0x82 ;paw sthn triti thesi ths othonis

rcall lcd\_command

ldi r24,' ' ;fortwnw to space

rcall lcd\_data

ldi r24,0x83 ;paw sthn tetarth thesi ths othonis

rcall lcd\_command

ldi r24,'M' ;fortwnw to 'M'

rcall lcd\_data

ldi r24,0x84 ;paw sthn pempth thesi ths othonis

rcall lcd\_command

ldi r24,'I' ;fortwnw to 'I'

rcall lcd\_data

ldi r24,0x85 ;paw sthn ekth thesi ths othonis

rcall lcd\_command

ldi r24,'N' ;fortwnw to 'N'

rcall lcd\_data

ldi r24,0x86 ;paw sthn evdsomh thesi ths othonis

rcall lcd\_command

ldi r24,':' ;fortwnw to ':'

rcall lcd\_data

ldi r24,0x87 ;paw sthn ogdoh thesi ths othonis

rcall lcd\_command

ldi r24,'0' ;fortwnw to 0

rcall lcd\_data

ldi r24,0x88 ;paw sthn enath thesi ths othonis

rcall lcd\_command

ldi r24,'0' ;fortwnw to 0

rcall lcd\_data

ldi r24,0x89 ;paw sthn dekath thesi ths othonis

rcall lcd\_command

ldi r24,' ' ;fortwnw to space

rcall lcd\_data

ldi r24,0x8A ;paw sthn entekath thesi ths othonis

rcall lcd\_command

ldi r24,'S' ;fortwnw to 'S'

rcall lcd\_data

ldi r24,0x8B ;paw sthn dwdekath thesi ths othonis

rcall lcd\_command

ldi r24,'E' ;fortwnw to 'E'

rcall lcd\_data

ldi r24,0x8C ;paw sthn dekath trith thesi ths othonis

rcall lcd\_command

ldi r24,'C' ;fortwnw to 'C'

rcall lcd\_data

ret

lcd\_data:

sbi PORTD ,PD2 ;epilogh tou data register (PD2=1)

rcall write\_2\_nibbles ;apostolh tou byte

ldi r24 ,43 ;anamonh 43µsec mexri na oloklhrwthei h lhpsh

ldi r25 ,0 ;twn dedomenwn apo ton elegkth ths lcd

rcall wait\_usec

ret

lcd\_command:

cbi PORTD ,PD2 ;epilogh tou command register (PD2=0)

rcall write\_2\_nibbles ;apostolh ths entolhs kai anamonh 39usec

ldi r24 ,39 ;gia thn oloklhrwsh ths ekteleshs ths apo ton elegkth ths lcd.

ldi r25 ,0 ;uparxoun duo entoles, oi clear display kai return home,

rcall wait\_usec ;pou apaitoun shmantika megalytero xroniko diasthma

ret

write\_2\_nibbles:

push r24 ;stelnei ta 4 MSB

in r25 ,PIND ;diavazontai ta 4 LSB kai ta ksanastelnoume

andi r25 ,0x0f ;gia na mhn xalasoume thn opoia prohgoumenh katastasj

andi r24 ,0xf0 ;apomonwnontai ta 4 MSB kai

add r24 ,r25 ;sunduazontai me ta prouparxonta 4 LSB

out PORTD ,r24 ;kai dinontai sthn eksodo

sbi PORTD ,PD3 ;dhmiourgeitai palmos enable ston akrodekth PD3

cbi PORTD ,PD3 ;PD3=1 kai µeta PD3=0

pop r24 ;stelnei ta 4 LSB. anaktatai to byte

swap r24 ;enallassontai ta 4 MSB me ta 4 LSB

andi r24 ,0xf0 ;pou me thn siera tous apostellontai

add r24 ,r25

out PORTD ,r24

sbi PORTD ,PD3 ;neos palmos enable

cbi PORTD ,PD3

ret

lcd\_init:

ldi r24 ,40 ;otan o elekths ths lcd trofodoteitai me

ldi r25 ,0 ;reuma prwth fora ektelei thn dikh tou arxikopoihsh.

rcall wait\_msec ;anamonh 40 msec mexri auth na oloklhrwthei

ldi r24 ,0x30 ;entolh metavashs se 8 bit mode

out PORTD ,r24 ;epeidh den mporoume na eimaste vevaioi

sbi PORTD ,PD3 ;gia th diamorfwsh eisodou tou elegkth

cbi PORTD ,PD3 ;ths othonhs, h entolh apostelletai dyo fores

ldi r24 ,39

ldi r25 ,0 ;ean o elegkths ths othonhs vrisketai se 8-bit mode

rcall wait\_usec ;de tha sumvei tipota, alla an o elegkths exei diamorfwsh

;eisodu 4 bit tha metavei se diamorfwsh 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 ;allagh se 4-bit mode

out PORTD ,r24

sbi PORTD ,PD3

cbi PORTD ,PD3

ldi r24 ,39

ldi r25 ,0

rcall wait\_usec

ldi r24 ,0x28 ;epilogh xarakthrwn megethous 5x8 koukidwn

rcall lcd\_command ;kai emfanish 2 grammwn sthn othonh

ldi r24 ,0x0c ;energopoihsh ths othonhs, apokrypsh tou kersora

rcall lcd\_command

ldi r24 ,0x01 ;katharismos ths othonhs

rcall lcd\_command

ldi r24 ,low(1530)

ldi r25 ,high(1530)

rcall wait\_usec

ldi r24 ,0x06 ;energopoihsh automaths aukshshs kata 1 ths dieuthynshs

rcall lcd\_command ;pou einai apothhkaumenh sto metrhth dieuthynsewn kai

;apenergopoihsh ths olisthishs oloklhrhs ths othonhs

ret

lcd\_number:

push r24

ldi r24,0x04 ;auto decrement mode(giati exw na emfanisw dyo psyfia kai to ;shmantikotero tha emfanistei sthn prohgoumenh thesh mnhmhs)

rcall lcd\_command

pop r24

clr r30

vres\_dekades:

cpi r24,10

brlo continue

inc r30

subi r24,10

rjmp vres\_dekades

continue: ;emfanizw tis monades eite twn leptwn eite twn ;monadwn(analogws apo pou kaleitai h routina)

ldi r23,0x30 ;prosthetw to 0x30 wste na paei ston antistoixo ;xarakthra-arithmo ascii

add r24,r23

rcall lcd\_data

add r30,r23 ;emfanizw tis dekades eite twn leptwn eite twn ;monadwn(analogws apo pou kaleitai h routina)

mov r24,r30

rcall lcd\_data

ldi r24,0x06 ;back to auto increment mode

rcall lcd\_command

ret

wait\_usec:

sbiw r24 ,1 ;2 cycles (0.250 micro sec)

nop ;1 cycles (0.125 micro sec)

nop ;1 cycles (0.125 micro sec)

nop ;1 cycles (0.125 micro sec)

nop ;1 cycles (0.125 micro sec)

brne wait\_usec ;1 or 2 cycles (0.125 or 0.250 micro sec)

ret ;4 cycles (0.500 micro sec)

wait\_msec:

push r24 ;2 cycles (0.250 micro sec)

push r25 ;2 cycles

ldi r24 , 0xe6 ;load register r25:r24 with 998 (1 cycles - 0.125 micro sec)

ldi r25 , 0x03 ;1 cycles (0.125 micro sec)

rcall wait\_usec ;3 cycles (0.375 micro sec), total delay 998.375 micro sec

pop r25 ;2 cycles (0.250 micro sec)

pop r24 ;2 cycles

sbiw r24 , 1 ;2 cycles

brne wait\_msec ;1 or 2 cycles (0.125 or 0.250 micro sec)

ret