**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**

**ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**



**Μάθημα: Εργαστήριο Μικροϋπολογιστών**

**Αναφορά 6ης Σειράς Εργαστηριακών Ασκήσεων**

**7ο Εξάμηνο**

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

* **Άσκηση 6.1(assembly)**

.include "m16def.inc"

.org 0x00

set\_stack:

ldi r24, low(RAMEND)

out SPL, r24

ldi r24, high(RAMEND)

out SPH, r24

init:

ser r20

out DDRA,r20 ;PortA output

step1:

rcall one\_wire\_reset

sbrs r24, 0 ;if bit is set, device attached, continue

rjmp no\_device

step2:

;entoli mnimis 0xCC prepei na einai ston r24

;mono 1 syskeui sto diadromo

ldi r24, 0xCC

rcall one\_wire\_transmit\_byte

step3:

;entoli leitourgias 0x44

;arxi metrisis thermokrasias

ldi r24, 0x44

rcall one\_wire\_transmit\_byte

;otan to r24 ginei 1 teleiwse i metrisi

still\_measuring:

rcall one\_wire\_receive\_bit

sbrs r24, 0 ;if bit is not set, check again

rjmp still\_measuring

step4:

;o aisthitiras einai se katastasi xamilis katanalosis isxyos

;opote ton arxikopoioume ksana gia na ksypnisei

rcall one\_wire\_reset

sbrs r24, 0 ;if bit is set, device attached, continue

rjmp no\_device

step5:

;entoli mnimis 0xCC prepei na einai ston r24

;mono 1 syskeui sto diadromo

ldi r24, 0xCC

rcall one\_wire\_transmit\_byte

;entoli leitourgias 0xΒΕ

;diavase 16 bits thermokrasias

ldi r24, 0xBE

rcall one\_wire\_transmit\_byte

;first read --> value of temperature

;second read --> sign of temperature

rcall one\_wire\_receive\_byte

push r24

rcall one\_wire\_receive\_byte

pop r25

;value of temperature has been stored in r25

;sign of temperature has been stored in r24

display\_celsius:

rcall show\_celsius

rjmp step6

no\_device:

ldi r25, 0x80

ldi r24, 0x00

step6:

out PORTA, r25

rjmp step1

show\_celsius:

lsr r25 ; temperature \* 0.5 (accuracy per bit)

cpi r24, 0xff ; if temperature is negative decrease by one in order ; to take 1's complement from 2's complement

brne ret\_ ; else go back and print temperature

negative:

dec r25

ret\_:

ret

one\_wire\_reset:

sbi DDRA ,PA4 ; PA4 configured for output

cbi PORTA ,PA4 ; 480 µsec reset pulse

ldi r24 ,low(480)

ldi r25 ,high(480)

rcall wait\_usec

cbi DDRA ,PA4 ; PA4 configured for input

cbi PORTA ,PA4

ldi r24 ,100 ; wait 100 µsec for devices

ldi r25 ,0 ; to transmit the presence pulse

rcall wait\_usec

in r24 ,PINA ; sample the line

push r24

ldi r24 ,low(380) ; wait for 380 µsec

ldi r25 ,high(380)

rcall wait\_usec

pop r25

; return 0 if no device was

clr r24 ; detected or 1 else

sbrs r25 ,PA4

ldi r24 ,0x01

ret

one\_wire\_receive\_byte:

ldi r27 ,8

clr r26

loop\_:

rcall one\_wire\_receive\_bit

lsr r26

sbrc r24 ,0

ldi r24 ,0x80

or r26 ,r24

dec r27

brne loop\_

mov r24 ,r26

ret

one\_wire\_transmit\_byte:

mov r26 ,r24

ldi r27 ,8

\_one\_more\_:

clr r24

sbrc r26 ,0

ldi r24 ,0x01

rcall one\_wire\_transmit\_bit

lsr r26

dec r27

brne \_one\_more\_

ret

one\_wire\_receive\_bit:

sbi DDRA ,PA4

cbi PORTA ,PA4 ; generate time slot

ldi r24 ,0x02

ldi r25 ,0x00

rcall wait\_usec

cbi DDRA ,PA4 ; release the line

cbi PORTA ,PA4

ldi r24 ,10 ; wait 10 µs

ldi r25 ,0

rcall wait\_usec

clr r24 ; sample the line

sbic PINA ,PA4

ldi r24 ,1

push r24

ldi r24 ,49 ; delay 49 µs to meet the standards

ldi r25 ,0 ; for a minimum of 60 µsec time slot

rcall wait\_usec ; and a minimum of 1 µsec recovery time

pop r24

ret

one\_wire\_transmit\_bit:

push r24 ; save r24

sbi DDRA ,PA4

cbi PORTA ,PA4 ; generate time slot

ldi r24 ,0x02

ldi r25 ,0x00

rcall wait\_usec

pop r24 ; output bit

sbrc r24 ,0

sbi PORTA ,PA4

sbrs r24 ,0

cbi PORTA ,PA4

ldi r24 ,58 ; wait 58 µsecfor the

ldi r25 ,0 ; device to sample the line

rcall wait\_usec

cbi DDRA ,PA4 ; recovery time

cbi PORTA ,PA4

ldi r24 ,0x01

ldi r25 ,0x00

rcall wait\_usec

ret

wait\_usec:

sbiw r24,1

nop

nop

nop

nop

brne wait\_usec

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

* **Άσκηση 6.1(c)**

#define *F\_CPU* 8000000UL

#include <avr/io.h>

#include <util/delay.h>

#define set\_bit(byte, bit) ( byte = byte | (1 << bit) )

#define clr\_bit(byte, bit) ( byte = byte & (~(1 << bit)) )

// Routine: one\_wire\_reset

// Description

// This routine transmits a reset pulse across the wire

// and detects any connected devices.

// parameters: None.

// return value:

// 1 if a device is detected

// 0 if a device isn't detected

char one\_wire\_reset(void)

{

char input,ret\_value;

set\_bit(DDRA, 4); //PA4 output

clr\_bit(PORTA, 4); //reset pulse

*\_delay\_us*(480);

clr\_bit (DDRA, 4); //PA4 input

clr\_bit (PORTA, 4);

*\_delay\_us*(100);

input = PINA;

*\_delay\_us*(380);

input = (input & (1<<4));

if (input) ret\_value = 0x00;

else ret\_value = 0x01;

return ret\_value;

}

// Routine: one\_wire\_receive\_bit

// Description:

// This routine generates a read time slot across the wire.

// return value:

// 0 if 0 is read or

// 1 if 1 is read.

char one\_wire\_receive\_bit(void)

{

char input,ret\_value;

set\_bit(DDRA, 4);

clr\_bit(PORTA, 4);

*\_delay\_us*(2);

clr\_bit(DDRA, 4);

clr\_bit(PORTA, 4);

*\_delay\_us*(10);

*\_delay\_us*(49);

input = (PINA & (1<<4));

if (input) ret\_value = 0x01;

else ret\_value = 0x00;

return ret\_value;

}

// Routine: one\_wire\_receive\_byte

// Description:

// This routine generates the necessary read

// time slots to receive a byte from the wire.

// return value: the byte that has been read is returned in r24.

// routines called: one\_wire\_receive\_bit

char one\_wire\_receive\_byte(void)

{

int i;

char read, temp1, temp2;

read = 0x00;

for (i = 0; i<8; i++){

temp1 = 0x00;

temp2 = one\_wire\_receive\_bit();

read = read >> 1;

if (temp2) temp1 = 0x80;

read = read | temp1;

}

return read;

}

// Routine: one\_wire\_transmit\_bit

// Description:

// This routine transmits a bit across the wire.

// if we want to transmit 1 input argument must be 1

// if we want to transmit 0 input argument must be 0

// return value: None.

void one\_wire\_transmit\_bit(char output)

{

char temp;

set\_bit(DDRA, 4);

clr\_bit(PORTA, 4);

*\_delay\_us*(2);

temp = (output & 1);

if (temp) set\_bit(PORTA, 4);

else clr\_bit(PORTA, 4);

*\_delay\_us*(58);

clr\_bit(DDRA, 4);

clr\_bit(PORTA, 4);

*\_delay\_us*(1);

}

// Routine: one\_wire\_transmit\_byte

// Description:

// This routine transmits a byte across the wire.

// return value: None.

// routines called: one\_wire\_transmit\_bit

// input arguments: a const char that implies the command that we wanna transmit

void one\_wire\_transmit\_byte(const char output)

{

int i;

char check,temp;

for(i = 0; i < 8; i++){

temp = 0x00;

check = (output & i);

if (check) temp = 0x01;

one\_wire\_transmit\_bit(temp);

}

}

int main(void)

{

char temp1,temp2,sign, value;

DDRA = 0xFF; //set portA as output

for(;;)

{

temp1 = one\_wire\_reset();

if (temp1 == 0){

PORTA = 0x80; //print 0x80 on leds if no device was found

break;

}

one\_wire\_transmit\_byte(0xCC); //memory command: choose only one device among others

one\_wire\_transmit\_byte(0x44); //operation command: start counting temperature

while(1){ //block until sensor responds with 1

temp2 = one\_wire\_receive\_bit();

if (temp2) break;

}

temp1 = one\_wire\_reset();

if (temp1 == 0){

PORTA = 0x80; //print 0x80 on leds if no device was found

break;

}

one\_wire\_transmit\_byte(0xCC); //memory command: choose only one device among others

one\_wire\_transmit\_byte(0xBE); //operation command: read the 16 bits of the temperature as measured

value = one\_wire\_receive\_byte();

sign = one\_wire\_receive\_byte();

value = value >> 1; //value \* 0.5 deg/bit

if (sign == 0xFF) value = value - 1; //if temperature is negative decrease value by 1

// in order to reach 1's complement form

//(since it's already in 2's complement form)

PORTA = value; //print temperature on leds of PORTA

}

return 0;

}

* **Άσκηση 6.2**

; TODO:

; - Implement print\_temp function

; - Test solution in lab

; - Error message routine -> uses lpm =>

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; 0x8000 => Error Code: No Device Found (Sensor not connected)

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; header file

.include "m16def.inc"

; data segment

.dseg

\_tmp\_ : .byte 2

; code segment

.cseg

; ------ RESET -------

.org 0x0

rjmp reset

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

; custom libraries

.include "include/keypad\_hex.asm" ; keypad\_to\_hex, scan\_keypad\_rising\_edge

.include "include/wait.asm" ; wait\_msec (r25:r24)

.include "include/lcd.asm" ; lcd\_data, lcd\_command, lcd\_init

.include "include/one\_wire.asm" ; one\_wire\_receive\_byte, one\_wire\_transmit\_byte, one\_wire\_reset

reset:

; initialize stack

ldi r24,low(RAMEND)

ldi r25,high(RAMEND)

out SPL,r24

out SPH,r25

; SETUP KEYPAD

ldi r18,0xf0 ; r18 = 1111 0000

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

; SETUP LCD

ser r18 ; r18 = 1111 1111

out DDRD,r18 ; PORTD: OUTPUT

out DDRA,r18 ; PORTA: OUTPUT

main:

; we can make this to change on runtime

; by checking a specific register (eg: PINAx)

; preload device select option

ldi r20,0x01

; initialize lcd screen

rcall lcd\_init

; KEYPADSELECT => READ FROM KEYPAD OR SENSOR

cpi r20,0x1

breq USE\_KEYPAD

; if r20 = 0x00 => Sensor will be used

USE\_SENSOR:

; use sensor

rcall sensor\_temp

rcall printw

rjmp SKIP\_KEYPAD

; if r20 = 0x01 => keypad will be used

USE\_KEYPAD:

; use keypad

rcall keypad\_temp

SKIP\_KEYPAD:

; wait 100ms

ldi r24,low(200)

ldi r25,high(200)

rcall wait\_msec

; while (1)

rjmp main

ret

;; [/main]

.macro NEGATIVE

ldi r24,'-'

rcall lcd\_data ; print '-'

com r19 ; One\'s complement of LO byte

.endm

.macro POSITIVE

ldi r24,'+'

rcall lcd\_data ; print '+'

.endm

; print temperature

print\_temp:

; keep backup of HO & LO byte

push r24

mov r19,r24

; POSITIVE OR NEGATIVE

sbrs r25,7

NEGATIVE ; prints '-'

; and complements r19

sbrc r25,7

POSITIVE ; prints '+'

; -55 <= TEMPERATURE <= +125

; therefore TEMPERATURE is contained only

; in 1 byte \r19\

PROCESSING\_R19:

clr r21 ; flag

clr r18 ; r18 will hold number of decades

ldi r17,'0' ; ASCII code of 0

; maximal temp = +125 = 0x7C

; therefore we mask everything with 0x7F = 0b 0111 xxx

andi r19,0x7F

; check if we have a hundred

cpi r19,0x64

brlo decades

; print 1 in the first position

ldi r24,'1'

rcall lcd\_data

; flag

ser r21

; subtract 100 = 0x64 from r19

subi r19,0x64 ; r19 holds decades

decades:

cpi r19,0x0A ; while (r19 > 10)

brlo print\_decades

inc r18 ; incr counter

subi r19,0x0A ; r19 -= 10

rjmp decades

print\_decades:

cpi r18,0x00

breq print\_units ; if there are no decades move on to units

clr r21 ; set flag off

mov r24,r18 ; decades counter

add r24,r17 ; convert to ASCII (+'0')

rcall lcd\_data ; print decades

; at this point r19 < 10

print\_units:

; if flag set then print a leading 0

; in case of we had no decades but the number

; was in range [100..109]

cpi r21,0xff

brne CONTINUE1

ldi r24,'0'

rcall lcd\_data

CONTINUE1:

add r19,r17 ; r19 holds units

mov r24,r19 ; convert value to ASCII (+ '0')

rcall lcd\_data ; print units

; print Celsius

ldi r24,0xB2

rcall lcd\_data

ldi r24,'C'

rcall lcd\_data

EXIT0:

; add delay for the temperature

; to remain on screen for 200msec

ldi r24,low(200)

ldi r25,high(200)

rcall wait\_msec

ret

;; [/print\_temp]

; Print Temperature Wrapper

printw:

cpi r25,0x80

brne no\_error

cpi r24,0x00

brne no\_error

rcall sensor\_error\_msg ; got 0x8000 => ERROR

ret

no\_error: ; r25:r24 != 0x8000 => NO ERROR

; if its 0x0000 or 0xffff

; then just print 0 without sign

; actually this is redundant as we should only check value of \r24\

NEG\_ZERO:

cpi r25,0xff

brne POS\_ZERO

cpi r24,0xff

brne POS\_ZERO

rjmp ZERO

POS\_ZERO:

cpi r25,0x00

brne CONTINUE

cpi r24,0x00

breq ZERO

CONTINUE:

; r25:r24 hold temperature

rcall print\_temp

ret

ZERO:

ldi r24,'0'

rcall lcd\_data

ret

;; [/tempw]

; read temperature from keypad

; aux function

keypad\_read:

ldi r24,low(25) ; spinthirismos

ldi r25,high(25)

rcall scan\_keypad\_rising\_edge

rcall keypad\_to\_hex

cpi r24,0x0 ; got any input?

breq keypad\_read

subi r24,'0'

cpi r24,0x0A

brlo EXIT1

subi r24,0x07

EXIT1:

ret

;; [/keypad\_read]

keypad\_temp:

; first byte (HO)

clr r21

rcall keypad\_read

mov r21,r24

swap r21 ; swap HO/LO bits

rcall keypad\_read

or r21,r24

; second byte (LO)

clr r20

rcall keypad\_read

mov r20,r24

swap r20 ; swap HO/LO bits

rcall keypad\_read

or r20,r24

; output temperature

mov r25,r21

mov r24,r20

rcall printw

; add delay of 500msec

ldi r24,low(500)

ldi r25,high(500)

rcall wait\_msec

ret

;; [/keypad\_temp]

; SENSOR NOT FOUND ROUTINE

; print error message to lcd screen

sensor\_error\_msg:

ldi r24, 0x01

rcall lcd\_command

ldi r24, low(1530)

ldi r25, high(1530)

rcall wait\_usec

ldi r24, 'N'

rcall lcd\_data

ldi r24, 'O'

rcall lcd\_data

ldi r24, ' '

rcall lcd\_data

ldi r24, 'D'

rcall lcd\_data

ldi r24, 'e'

rcall lcd\_data

ldi r24, 'v'

rcall lcd\_data

ldi r24, 'i'

rcall lcd\_data

ldi r24, 'c'

rcall lcd\_data

ldi r24, 'e'

rcall lcd\_data

ret

;; [/sensor\_error\_msg]

; read temperature from sensor

sensor\_temp:

rcall one\_wire\_reset ; returns r24=0x0 if no sensor is found

sbrs r24,0

breq sensor\_error ; goto missing sensor routine

; we only have one sensor

ldi r24,0xCC

rcall one\_wire\_transmit\_byte

; request temperature

ldi r24,0x44

rcall one\_wire\_transmit\_byte

isTxFinished:

rcall one\_wire\_receive\_bit

sbrs r24,0

rjmp isTxFinished

; reset sensor

rcall one\_wire\_reset

sbrs r24,0

rjmp sensor\_error

ldi r24,0xCC

rcall one\_wire\_transmit\_byte

ldi r24,0xBE

rcall one\_wire\_transmit\_byte

; r25:r24 = temperature

rcall one\_wire\_receive\_byte

push r24

rcall one\_wire\_receive\_byte

mov r25,r24

pop r24

; ??

sbrs r25,0

rjmp done

dec r24

done:

out PORTA,r24

ret

sensor\_error:

ldi r24,low(8000)

ldi r25,high(8000)

ret

;; [/sensor\_temp]

* **Άσκηση 6.2 – Custom libraries**
* **wait.asm**

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

* **keypad\_hex.asm**

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\_hex:

movw r26, r24

ldi r24, 'E'

sbrc r26, 0

ret

ldi r24, '0'

sbrc r26, 1

ret

ldi r24, 'F'

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

* **lcd.asm**

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, 0x0e

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

* **one\_wire.asm**

; This file includes routines implementing the one wire

; protocol over the PA4 pin of the microcontroller.

; Dependencies: wait.asm

; Routine: one\_wire\_receive\_byte

; Description:

; This routine generates the necessary read

; time slots to receives a byte from the wire.

; return value: the received byte is returned in r24.

; registers affected: r27:r26 ,r25:r24

; routines called: one\_wire\_receive\_bit

one\_wire\_receive\_byte:

ldi r27 ,8

clr r26

loop\_:

rcall one\_wire\_receive\_bit

lsr r26

sbrc r24 ,0

ldi r24 ,0x80

or r26 ,r24

dec r27

brne loop\_

mov r24 ,r26

ret

; Routine: one\_wire\_receive\_bit

; Description:

; This routine generates a read time slot across the wire.

; return value: The bit read is stored in the lsb of r24.

; if 0 is read or 1 if 1 is read.

; registers affected: r25:r24

; routines called: wait\_usec

one\_wire\_receive\_bit:

sbi DDRA ,PA4

cbi PORTA ,PA4 ; generate time slot

ldi r24 ,0x02

ldi r25 ,0x00

rcall wait\_usec

cbi DDRA ,PA4 ; release the line

cbi PORTA ,PA4

ldi r24 ,10 ; wait 10 ?s

ldi r25 ,0

rcall wait\_usec

clr r24 ; sample the line

sbic PINA ,PA4

ldi r24 ,1

push r24

ldi r24 ,49 ; delay 49 ?s to meet the standards

ldi r25 ,0 ; for a minimum of 60 ?sec time slot

rcall wait\_usec ; and a minimum of 1 ?sec recovery time

pop r24

ret

; Routine: one\_wire\_transmit\_byte

; Description:

; This routine transmits a byte across the wire.

; parameters:

; r24: the byte to be transmitted must be stored here.

; return value: None.

; registers affected: r27:r26 ,r25:r24

; routines called: one\_wire\_transmit\_bit

one\_wire\_transmit\_byte:

mov r26 ,r24

ldi r27 ,8

\_one\_more\_:

clr r24

sbrc r26 ,0

ldi r24 ,0x01

rcall one\_wire\_transmit\_bit

lsr r26

dec r27

brne \_one\_more\_

ret

; Routine: one\_wire\_transmit\_bit

; Description:

; This routine transmits a bit across the wire.

; parameters:

; r24: if we want to transmit 1

; then r24 should be 1, else r24 should

; be cleared to transmit 0.

; return value: None.

; registers affected: r25:r24

; routines called: wait\_usec

one\_wire\_transmit\_bit:

push r24 ; save r24

sbi DDRA ,PA4

cbi PORTA ,PA4 ; generate time slot

ldi r24 ,0x02

ldi r25 ,0x00

rcall wait\_usec

pop r24 ; output bit

sbrc r24 ,0

sbi PORTA ,PA4

sbrs r24 ,0

cbi PORTA ,PA4

ldi r24 ,58 ; wait 58 ?sec for the

ldi r25 ,0 ; device to sample the line

rcall wait\_usec

cbi DDRA ,PA4 ; recovery time

cbi PORTA ,PA4

ldi r24 ,0x01

ldi r25 ,0x00

rcall wait\_usec

ret

; Routine: one\_wire\_reset

; Description:

; This routine transmits a reset pulse across the wire

; and detects any connected devices.

; parameters: None.

; return value: 1 is stored in r24

; if a device is detected, or 0 else.

; registers affected r25:r24

; routines called: wait\_usec

one\_wire\_reset:

sbi DDRA ,PA4 ; PA4 configured for output

cbi PORTA ,PA4 ; 480 ?sec reset pulse

ldi r24 ,low(480)

ldi r25 ,high(480)

rcall wait\_usec

cbi DDRA ,PA4 ; PA4 configured for input

cbi PORTA ,PA4

ldi r24 ,100 ; wait 100 ?sec for devices

ldi r25 ,0 ; to transmit the presence pulse

rcall wait\_usec

in r24 ,PINA ; sample the line

push r24

ldi r24 ,low(380) ; wait for 380 ?sec

ldi r25 ,high(380)

rcall wait\_usec

pop r25 ; return 0 if no device was

clr r24 ; detected or 1 else

sbrs r25 ,PA4

ldi r24 ,0x01

ret