Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2017-2018

ΑΘΗΝΑ 6 Δεκεμβρίου 2017

8η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Εργαστήριο Μικροϋπολογιστών"
5η Εργ. Άσκ. στον Μικροελεγκτή AVR (στο easyAVR6)- **Αισθητήρας DS1820**

Εξέταση – Επίδειξη: Τετάρτη 20/12/2017

Έκθεση: Πέμπτη 28/12/2017

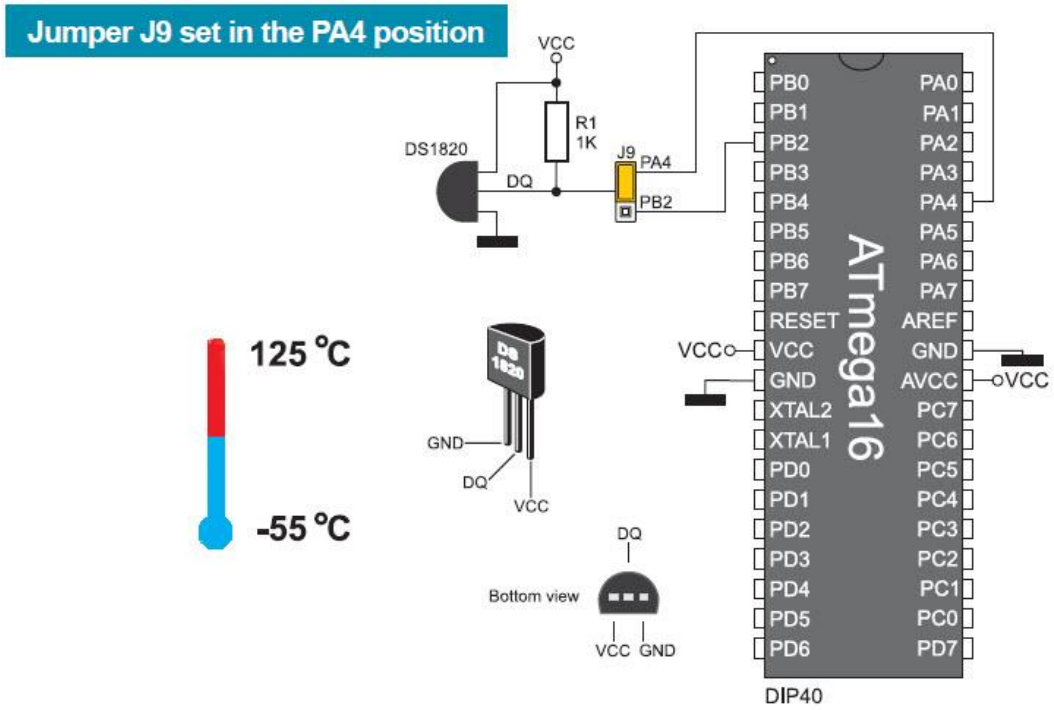
Χρήση Αισθητήρα Θερμοκρασίας DS1820 και Σειριακή Επικοινωνία 1-wire

Η σειριακή επικοινωνία ενός καλωδίου (1-wire serial communication) είναι ένα πρωτόκολλο που επιτρέπει τη μεταφορά δεδομένων από μία μόνο καλωδιακή σύνδεση. Η διαδικασία συντονίζεται από μία συσκευή Μικροελεγκτή σε ρόλο αφέντη. Το πλεονέκτημα του συγκεκριμένου πρωτοκόλλου είναι ότι απαιτεί έναν μόνο ακροδέκτη του Μικροελεγκτή. Όλες οι συσκευές που συνδέονται σε αυτόν τον ακροδέκτη έχουν έναν προκαθορισμένο κωδικό σκλάβου, που επιτρέπει στο Μικροελεγκτή να αναγνωρίζει εύκολα με ποια συσκευή επικοινωνεί κάθε φορά.

Η αναπτυξιακή πλακέτα *EasyAVR6* είναι εφοδιασμένη με τον αισθητήρα θερμοκρασίας DS1820 (σχήμα 8.1) που χρησιμοποιεί το πρωτόκολλο σειριακής επικοινωνίας ενός καλωδίου. Μετράει θερμοκρασίες από -55°C έως 125°C και παρέχει ακρίβεια ±0.5°C στην περιοχή θερμοκρασιών από -10°C έως 85°C. Για την τροφοδοσία του απαιτείται συνεχής τάση από 3V έως 5V. Ο αισθητήρας χρειάζεται μέγιστο χρόνο 750ms για να υπολογίσει τη θερμοκρασία με ακρίβεια 9 bit.

Στην αναπτυξιακή πλακέτα *EasyAVR6* μπορεί να συνδεθεί με τους ακροδέκτες του μικροελεγκτή PA4 ή PB2, ανάλογα με τη θέση του βραχυκυκλωτήρα J9, όπως φαίνεται και στα σχήματα 8.1 και 8.2. Μέσω του επιλεγμένου ακροδέκτη πραγματοποιείται η επικοινωνία μικροελεγκτή-αισθητήρα. Η τοποθέτηση του αισθητήρα στην αναπτυξιακή πλακέτα γίνεται σε ειδική βάση, **προσέχοντας το τυπωμένο ημικύκλιο στην πλακέτα να συμπίπτει με την ημικυκλική πλευρά του αισθητήρα**, όπως στο σχήμα 8.1. Το ολοκληρωμένο DS1820 δεν είναι ένας απλός αισθητήρας αλλά μια σύνθετη συσκευή (βλ. τεχνικό εγχειρίδιο του DS1820.pdf) με ενσωματωμένο τον μετατροπέα ADC και το πρωτόκολλο επικοινωνίας. Αν συνδεθεί λανθασμένα θα 'καεί'. Γι' αυτό να μην πειράξετε το ολοκληρωμένο από την υποδοχή του καθώς και την τοποθέτηση του βραχυκυκλωτήρα J9 (Σχ. 8.2).

Σχήμα 8.1 Αισθητήρας θερμοκρασίας DS1820.



Σχήμα 8.2. Κυκλωματικό διάγραμμα αισθητήρα θερμοκρασίας DS1820.

Η επικοινωνία ενός καλωδίου είναι ένα πολύ απλό πρωτόκολλο που διαβάζει και γράφει σειριακά δεδομένα τηρώντας συγκεκριμένα χρονικά παράθυρα, όπως φαίνεται στο τεχνικό εγχειρίδιο της μονάδας (DS1820.pdf). Στην αναπτυξιακή πλακέτα *EasyAVR6*, το καλώδιο αυτό μπορεί να είναι ο ακροδέκτης PA4 ή PB2, με προεπιλεγμένο τον PA4. Η σωστή λειτουργία του πρωτοκόλλου απαιτεί και την ύπαρξη μιας αντίστασης ανέλκυσης (pull-up) στο συγκεκριμένο ακροδέκτη (στο Σχ. 8.2, φαίνεται η αντίσταση αυτή για την πλακέτα *EasyAVR6*), που αναλαμβάνει, όταν δεν οδηγείται από καμμία συσκευή, είτε σε ρόλο αφέντη είτε σε ρόλο σκλάβου, να τον τοποθετεί σε κατάσταση λογικού 1. Για την επικοινωνία της συσκευής με τις εφαρμογές χρήστη είναι απαραίτητη η συγγραφή δύο οδηγών συσκευής, έναν για το πρωτόκολλο επικοινωνίας ενός καλωδίου και έναν για την ανάγνωση μετρήσεων θερμοκρασίας.

Ο οδηγός συσκευής για το πρωτόκολλο επικοινωνίας ενός καλωδίου περιλαμβάνει την υπορουτίνα αρχικοποίησης **one_wire_reset**. Η υπορουτίνα αυτή, όπως προκύπτει από την περιγραφή στη σελίδα 13 του τεχνικού εγχειριδίου του αισθητήρα θερμοκρασίας DS1820, καθορίζει τη θύρα PA ως έξοδο και τοποθετεί τον ακροδέκτη PA4 σε κατάσταση λογικού 0 για 480μsec. Στη συνέχεια, περιμένει 100μsec, διαβάζει την κατάσταση της θύρας PA, τη μεταφέρει στη στοίβα (εντολή push r24) και περιμένει άλλα 380μsec ώστε να ολοκληρωθεί ο κύκλος αρχικοποίησης. Στο τέλος, επαναφέρει στον καταχωρητή r25 την κατάσταση της θύρας PA και αν το ψηφίο PA4 είναι 1 (δηλαδή, ο ακροδέκτης δεν οδηγήθηκε από καμμία συσκευή σε λογικό 0 αλλά παρέμεινε σε λογικό 1, λόγο της ύπαρξης της αντίστασης pull-up), με τις εντολές clr r24 και sbrs r25 ,PA4, επιστρέφει μέσω του καταχωρητή r24 την τιμή 0. Σε αντίθετη περίπτωση, εκτελείται και η εντολή ldi r24 ,0x01, οπότε η τιμή επιστροφής του καταχωρητή r24 είναι 1.

```
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
```

Επίσης, ο οδηγός συσκευής του πρωτοκόλλου επικοινωνίας ενός καλωδίου περιλαμβάνει τις υπορουτίνες *one_wire_receive_bit*, *one_wire_receive_byte*, *one_wire_transmit_bit* και *one_wire_transmit_byte*, για αποστολή και λήψη δεδομένων. Οι υπορουτίνες *one_wire_receive_byte* και *one_wire_transmit_byte*, αποτελούνται από ένα βρόχο 8 επαναλήψεων, όπου με τη χρήση των *one_wire_receive_bit* και *one_wire_transmit_bit*, διαβάζουν ή γράφουν λέξεις των 8 bit, ξεκινώντας από το λιγότερο σημαντικό. Πιο συγκεκριμένα, η υπορουτίνα **one_wire_receive_byte**, όπως παρουσιάζεται παρακάτω, με τις εντολές:

```
lsr r26
sbrc r24 ,0
ldi r24 ,0x80
or r26 ,r24
```

ολισθαίνει την τιμή που επιστρέφει η *one_wire_receive_bit* **8 φορές** στον καταχωρητή r26, κάνοντας κάθε φορά λογικό or με το 0x00 (αν η *one_wire_receive_bit* επιστρέψει 0) ή το 0x80 (αν η one_wire_receive_bit επιστρέψει 1) και ολίσθηση προς τα δεξιά.

```
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
```

Η υπορουτίνα **one_wire_transmit_byte**, όπως παρουσιάζεται παρακάτω, με τις εντολές:

```
clr r24
sbrc r26 ,0
ldi r24 ,0x01
```

ελέγχει το λιγότερο σημαντικό bit του καταχωρητή r26 και ανάλογα γράφει το 0 ή το 1 στον r24. Στη συνέχεια, καλεί την *one_wire_transmit_bit* και κάνει δεξιά ολίσθηση του r26.

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
```

Τέλος, οι υπορουτίνες *one_wire_receive_bit* και *one_wire_transmit_bit*, διαβάζουν και γράφουν τα κατάλληλα bit στον ακροδέκτη PA4, σύμφωνα με όσα αναφέρονται στις σελίδες 13 και 14 του τεχνικού εγχειριδίου του αισθητήρα DS1820 (χειρισμός της γραμμής και απαραίτητες ελάχιστες χρονοκαθυστερήσεις), και παρουσιάζονται παρακάτω:

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 μsec 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
```

Και οι δύο ξεκινάνε με τις εντολές:

```
sbi DDRA ,PA4
cbi PORTA ,PA4    ; generate time slot
ldi r24 ,0x02
ldi r25 ,0x00
rcall wait_usec
```

όπου ορίζεται η θύρα PORTA ως έξοδος και για 2μsec γίνεται το bit PA4 0. Στη συνέχεια, η *one_wire_receive_bit* κάνει τη θύρα είσοδο, περιμένει 10μsec, διαβάζει τον ακροδέκτη PA4 και ανάλογα με την τιμή που διάβασε γράφει στον καταχωρητή r24 την τιμή 0 ή 1 με τις εντολές:

```
clr r24
sbic PINA ,PA4
ldi r24 ,1
```

και ολοκληρώνει τον κύκλο της με μία ακόμα χρονοκαθυστέρηση 49μsec (48μsec για ολοκλήρωση του κύκλου ανάγνωσης και 1μsec για αποκατάσταση). Η *one_wire_transmit_bit*, με τις εντολές:

```
sbrc r24 ,0
sbi PORTA ,PA4
sbrs r24 ,0
cbi PORTA ,PA4
```

θέτει το bit PA4 0 ή 1 ανάλογα με την τιμή του καταχωρητή r24 και περιμένει 58μsec μέχρι η συσκευή σκλάβου (στη συγκεκριμένη περίπτωση ο αισθητήρας θερμοκρασίας) να διαβάσει την τιμή αυτή, και τέλος, κάνει τη θύρα PORTA είσοδο και περιμένει 1μsec για αποκατάσταση.

Ο πλήρης οδηγός συσκευής του πρωτόκολλου επικοινωνίας ενός καλωδίου μαζί με τον απαραίτητο σχολιασμό, δίνεται παρακάτω:

```
; File Name: one_wire.asm
; Title: one wire protocol
; Target mcu: atmega16
; Development board: easyAVR6
; Assembler: AVRStudio assembler
; Description:
```

```
; 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 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

; 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 μsec 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 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
```

Τα ζητούμενα της 5ης εργαστηριακής άσκησης του AVR

Ζήτημα 8.1 Να υλοποιηθεί ένα πρόγραμμα για την επικοινωνία του μικροελεγκτή με τον αισθητήρα θερμοκρασίας DS1820 που θα αποτελεί τον οδηγό συσκευής για αυτόν. Ο οδηγός αυτός να περιλαμβάνει μόνο τις στοιχειώδεις λειτουργίες, όπως στην περίπτωση της αναπτυξιακής πλακέτας *EasyAVR6*, όπου γνωρίζουμε εκ των προτέρων ότι στον ακροδέκτη PA4 είναι συνδεδεμένος μόνο ένας αισθητήρας. Οι στοιχειώδεις αυτές λειτουργίες στηρίζονται στις εντολές που δέχεται ο αισθητήρας και παρουσιάζονται στις σελίδες 10 και 11 του τεχνικού εγχειριδίου. Οι εντολές αυτές μπορούν να χωριστούν σε **εντολές μνήμης**, με τις οποίες επιλέγεται η συσκευή με την οποία θα γίνει η επικοινωνία, και **εντολές λειτουργίας**, με τις οποίες υλοποιούνται λειτουργίες που έχουν σχέση με τη *μέτρηση θερμοκρασίας*. Ενδιαμέσως, ο αισθητήρας βρίσκεται σε κατάσταση *χαμηλής κατανάλωσης ισχύος* και επανέρχεται σε κατάσταση λειτουργίας με μια **εντολή αρχικοποίησης**.

Ειδικότερα να γραφεί ρουτίνα που να επιστρέφει στον διπλό καταχωρητή r25:r24 την τιμή της θερμοκρασίας και σε περίπτωση που δεν υπάρχει συνδεδεμένη συσκευή να επιστρέφει την τιμή 0x8000. Ποιο αναλυτικά η ρουτίνα να περιλαμβάνει τα εξής:

- Αρχικοποίηση της συσκευής και έλεγχος αν υπάρχει συνδεδεμένη συσκευή (*one_wire_reset*).
- Αποστέλλει την **εντολή 0xCC**, η οποία παρακάμπτει την επιλογή συσκευής από διάδρομο πολλών συσκευών, μια και γνωρίζουμε ότι υπάρχει μόνο μία.
- Στη συνέχεια, αποστέλλεται η **εντολή 0x44**, που ξεκινάει μια μέτρηση θερμοκρασίας. Το DS1820 σηματοδοτεί ότι τερματίστηκε η μετατροπή (μόλις η μέτρηση θερμοκρασίας ολοκληρωθεί) με την αποστολή-απάντηση ενός bit με τιμή '1', που ελέγχεται με την εκτέλεση της υπορουτίνας *one_wire_receive_bit* και την **εντολή sbrs r24 , 0**.
- Στο επόμενο βήμα, ο αισθητήρας αρχικοποιείται ξανά («ξυπνάει» από την κατάσταση χαμηλής κατανάλωσης ισχύος) και γίνεται ξανά ο έλεγχος αν υπάρχει συνδεδεμένη συσκευή (*one_wire_reset*).
- Αποστέλλεται η **εντολή 0xCC** και τέλος, με την **εντολή 0xBE** γίνεται ανάγνωση των 16 bit της μετρημένης θερμοκρασίας, με κλίση της υπορουτίνας *one_wire_receive_byte* δύο (2) φορές και αποθήκευση της τιμής στο ζεύγος καταχωρητών r25:r24. Η τιμή που διαβάζεται, σύμφωνα με όσα αναφέρονται στις σελίδες 3 και 4 του τεχνικού εγχειριδίου, είναι ένας αριθμός σε μορφή συμπληρώματος ως προς 2. Τα 8 λιγότερα σημαντικά bit αντιστοιχούν σε θερμοκρασία βαθμών Κελσίου, με ακρίβεια 0.5°C ανά bit, και τα 8 περισσότερο σημαντικά αποτελούν, στην προκαθορισμένη λειτουργία του αισθητήρα, επέκταση προσήμου. Για παράδειγμα, ο αριθμός 0x0032 αντιστοιχεί σε θερμοκρασία 25°C, ο 0xff92 σε -55°C και ο 0xffff σε -0.5°C.
- Η ρουτίνα συμπληρώνεται από μια εντολή που επιστρέφει την τιμή 0x8000 σε περίπτωση που δεν υπάρχει συσκευή συνδεδεμένη στον ακροδέκτη PA4.
- Τέλος η ρουτίνα να απεικονίζει την θερμοκρασία σε °C και σε μορφή συμπληρώματος ως προς 1 στην θύρα **PORTA.**

Προαιρετικά (θα δοθεί bonus) μπορείτε να φτιάξετε την υλοποίηση όλης της ζητούμενης λειτουργίας σε C (**Διευκρίνιση: θα πρέπει το πρόγραμμα να έχει όσο γίνεται παρόμοια δομή με την assembly δηλ. οι συναρτήσεις της C να αντιστοιχούν υποχρεωτικά στις υπορουτίνες που σας δίνονται**).

Ζήτημα 8.2 Γράψτε πρόγραμμα που με την χρήση της προηγούμενης ρουτίνας να απεικονίζει τη θερμοκρασίας σε °C στο LCD display σε δεκαδική τιμή τριών ψηφίων με το πρόσημο (**`-55°C`** έως **`+125°C`**).

Επίσης στην περίπτωση που δεν υπάρχει συσκευή συνδεδεμένη να εμφανίζει το μήνυμα "**`NO Device`**".

Για να μπορείτε να ελέγχετε την απεικόνιση, ανεξάρτητα από τον αισθητήρα DS1820, δώστε και μια άλλη μορφή του προγράμματος που θα λαμβάνει τα 4 Hex ψηφία των καταχωρητών r25:r24 μέσω του δεκαεξαδικού πληκτρολογίου. Για το σκοπό αυτό να γίνει μετατροπή στην ρουτίνα **keypad_to_ascii** ώστε να αναγνωρίζει το '**#**' ως '**F**' και το '*****' ως '**E**' και να μπορεί το πληκτρολόγιο να χρησιμοποιηθεί ως δεκαεξαδικό. Η νέα αυτή ρουτίνα **keypad_to_hex** να επιστρέφει την δεκαεξαδική τιμή των πλήκτρων συμπεριλαμβανομένων των '#' και '*' που θα θεωρούνται πλέον ως '*F*' και '*E*' αντίστοιχα. Χρησιμοποιήστε τη ρουτίνα αυτή για να εισάγετε τα 4 Hex ψηφία στο πρόγραμμα απεικόνισης (π.χ. FF92 => **`-55°C`**, 8000 => **`NO Device`**).