ΑΤΕΙ ΗΠΕΙΡΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΣΧΕΔΙΑΣΗ ΧΡΟΝΟΜΕΤΡΟΥ ΜΕ ΜΙΡΟΕΛΕΓΚΤΗ PIC16F887 ΣΕ ΟΘΟΝΗ ΑΦΗΣ ΚΑΙ ΣΕ LCD ΟΘΟΝΗ»

ΒΕΝΤΟΥΛΗΣ ΕΥΑΓΓΕΛΟΣ
Α.Μ. 9163

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΚΟΛΙΟΠΑΝΟΣ ΧΡΗΣΤΟΣ

ΑΡΤΑ 2013

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΠΡΟΛΟΓΟΣ

Στην παρούσα εργασία έχουν δημιουργηθεί δύο χρονόμετρα με την χρήση μικροελεγκτή PIC16F887 και γλώσσα προγραμματισμού C. Στο πρώτο χρονόμετρο η απεικόνιση του χρονομέτρου γίνετε σε LCD οθόνη ενώ η ρύθμιση του γίνετε με push buttons. Στο δεύτερο χρονόμετρο χρησιμοποιείται οθόνη αφής για την απεικόνιση και τη ρύθμιση του χρονομέτρου.

Ακόμη γίνετε γενική περιγραφή των χαρακτηριστικών των μικροελεγκτών και πιο ειδικά στο μικροελεγκτή της Microchip PIC16F887.

Τέλος γίνετε αναφορά στα χαρακτηριστικά στην Easy PIC v7 connectivity της Mikroelektronika και τον compiler της ίδιας εταιρίας που χρησιμοποιήθηκαν για την δημιουργία των δύο χρονομέτρων, όπως και στην περιγραφή της λειτουργίας των χρονομέτρων.

2. ΜΙΚΡΟΕΛΕΓΚΤΕΣ

2.1. ΕΙΣΑΓΩΓΗ ΣΤΟΥΣ ΜΙΚΡΟΕΛΕΓΚΤΕΣ (PIC)

Μικροελεγκτής είναι ένας τύπος επεξεργαστή, στην ουσία είναι μία παραλλαγή του μικροεπεξεργαστή, με τη σημαντική διαφορά πως ο μικροελεγκτής μπορεί να λειτουργήσει με ελάχιστα εξωτερικά συστήματα λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Και αυτό είναι που τον κάνει να ξεχωρίζει από ένα μικροεπεξεργαστή ότι ο μικροελεγκτής έχει ενσωματωμένες τις περιφερειακές μονάδες.

Επομένως θα μπορούσαμε να χαρακτηρίσουμε ένα μικροελεγκτή ως ένα μικρό υπολογιστικό κύκλωμα σχεδιασμένο σ' ένα και μόνο ολοκληρωμένο κύκλωμα υψηλής κλίμακας ολοκλήρωσης.

Οι μικροελεγκτές χρησιμοποιούν την αρχιτεκτονική Harvard για την επικοινωνία μεταξύ της CPU και της μνήμης.

Τους μικροελεγκτές τους χωρίζουμε σε δύο κατηγορίες με βάση τον τρόπο που χρησιμοποιούν τις εντολές για τον υπολογισμό μιας εργασίας ανεξαρτήτου μεγέθους. Αυτές οι δύο κατηγορίες είναι:

- Η αρχιτεκτονική RISC και
- Η αρχιτεκτονική CISC.

Ο προγραμματισμός των μικροελεγκτών μπορεί να γίνει είτε με γλώσσες χαμηλού επιπέδου (π.χ. Assembly), είτε με γλώσσες υψηλού επιπέδου (π.χ. C, Pascal, κ.α.).

2.2. ΓΕΝΙΚΗ ΔΟΜΗ ΕΝΟΣ ΜΙΚΡΟΕΛΕΓΚΤΗ (PIC)

Τη δομή ενός μικροελεγκτή μπορούμε να την χωρίσουμε σε δύο κατηγορίες. Αυτές είναι:

<u>Ο πυρήνας του μικροελεγκτή</u>

- Κεντρική μονάδα επεξεργασίας (CPU)
- Μνήμη
- Λειτουργίες διακοπών

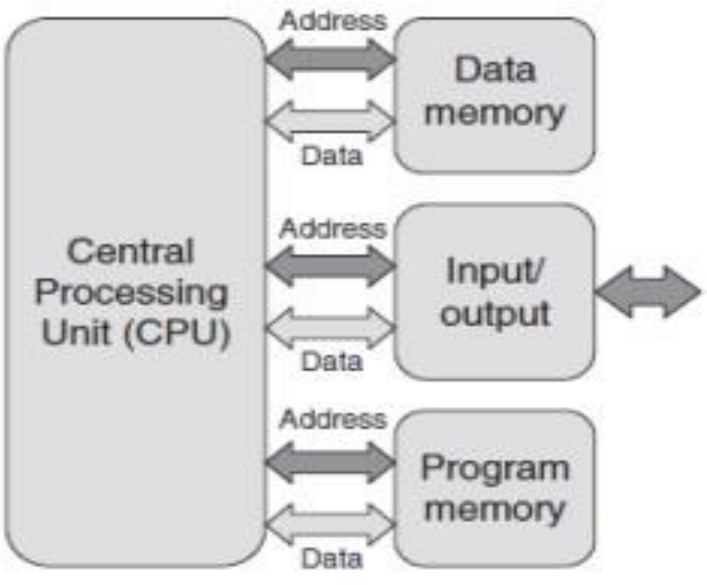
<u>Οι περιφερειακές μονάδες</u>

- Οι θύρες εισόδου/εξόδου γενικής χρήσης
- Οι θύρες σειριακής επικοινωνίας

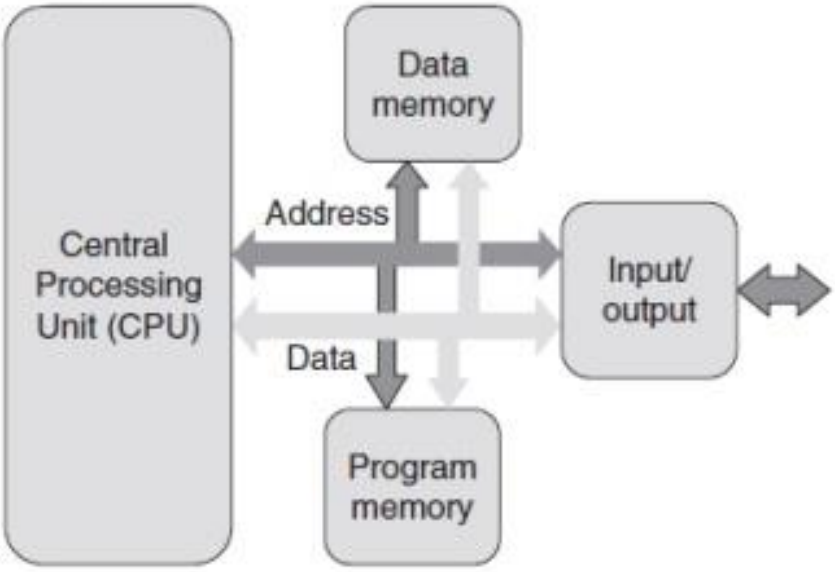
- Οι συγκριτές
- Οι μετρητές χρόνου
- Ο μετατροπέας αναλογικού σήματος σε ψηφιακό
- Η μονάδα παραγωγής τάσης αναφοράς
- Η θύρα παράλληλης επικοινωνίας
- Η μονάδα διαμόρφωσης πλάτους

2.3. ΑΡΧΙΤΕΚΤΟΝΙΚΗ Harvard ΚΑΙ Von – Neumann

Εξετάζοντας την αρχιτεκτονική των μικροελεγκτών βλέπουμε ότι χρησιμοποιούν την αρχιτεκτονική Harvard σε αντίθεση με τους μικροϋπολογιστές όπου χρησιμοποιείτε η αρχιτεκτονική Von - Neumann. Στην αρχιτεκτονική Harvard η μνήμη προγράμματος και η μνήμη δεδομένων είναι ξεχωριστές και χρησιμοποιείτε διαφορετικός δίαυλος για τις εντολές (instruction bus) και διαφορετικός δίαυλος δεδομένων (data bus) για όλα τα υπόλοιπα δεδομένα και αποτελέσματα της επεξεργασίας. Έτσι με την αρχιτεκτονική Harvard έχουμε την δυνατότητα να κινούνται την ίδια χρονική στιγμή δεδομένα και εντολές στους δύο διαύλους, ενώ μπορεί ακόμη να εκτελείτε μία εντολή ενώ προσκομίζεται η επόμενη.



Εικόνα 2.1: Αρχιτεκτονική Harvard.



Εικόνα 2.2: Αρχιτεκτονική Von-Neumann.

2.4. ΔΙΑΦΟΡΕΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ RISC ΚΑΙ CISC

Μία σημαντική διάκριση ανάμεσα στους μικροελεγκτές γίνετε με βάση τον τρόπο που χρησιμοποιούν τις εντολές για τον υπολογισμό μιας εργασίας ανεξαρτήτου μεγέθους.

- RISC

  Στους μικροελεγκτές που χρησιμοποιείται η αρχιτεκτονική RISC, για μία εργασία ανεξαρτήτου μεγέθους χρησιμοποιείτε ένα μικρό σύνολο εντολών για οποιαδήποτε πράξη και στην περίπτωση που κάποια πράξη δεν μπορεί να εκτελεστεί από μια από τις εντολές που υπάρχουν τότε γίνετε σύνθεση του συνόλου των εντολών που υπάρχουν για την εκτέλεση αυτής.

- CISC

  Στην αρχιτεκτονική CISC χρησιμοποιείται ένα μεγάλο σύνολο εντολών με αποτέλεσμα μία σύνθετη πράξη να μπορεί να εκτελεστή με μία μόνο εντολή.

Έτσι είναι πιο εύκολο να απομνημονευτούν οι εντολές τύπου RISC αλλά είναι πιο δύσκολος και πιο περίπλοκος ο προγραμματισμός τους.

2.5. ΔΙΑΚΡΗΣΗ ΤΩΝ PIC ΜΙΚΡΟΕΛΕΓΚΤΩΝ

Τους PIC μικροελεγκτές μπορούμε να τους διακρίνουμε σε τέσσερεις κατηγορίες:

- Διάκριση με βάση το μέγεθος της λέξης εντολής που μπορούν να εκτελούν
- Διάκριση με βάση τον αριθμό των ακίδων του ολοκληρωμένου
- Διάκριση με βάση την συσκευασία
- Διάκριση με βάση την ονομασία

2.5.1. Διάκριση με βάση το μέγεθος της λέξης εντολής που μπορούν να εκτελούν

Σ' αυτή την κατηγορία διάκρισης των μικροελεγκτών υπάρχουν τρεις υποκατηγορίες:

- Βασική κατηγορία με μήκος εντολής 12 ψηφίων
- Μεσαία κατηγορία με μήκος εντολής 14 ψηφίων
- Υψηλή κατηγορία με μήκος εντολής 16 ψηφίων

Έτσι θα πρέπει να έχουμε για κάθε κατηγορία ίδιο μέγεθος διαύλου και μνήμης προγράμματος έτσι ώστε κάθε εντολή που κωδικοποιείτε να διαρκεί ένα κύκλο και να καταλαμβάνουν μία θέση μνήμης. Βέβαια υπάρχει και η δυνατότητα το μήκος των εντολών να είναι μεγαλύτερο από μία λέξη όμως αυτό έχει ως αποτέλεσμα την αύξηση του χρόνου (κύκλου) εκτέλεσης.

2.5.2. Διάκριση με βάση τον αριθμό των ακίδων του ολοκληρωμένου

Μπορούμε να κάνουμε διάκριση των μικροελεγκτών με βάση τον αριθμό των ακίδων (pins) και αυτό γίνετε διότι υπάρχουν μικροελεγκτές οι οποίοι έχουν 8 pins, 14 pins, 28 pins, 40 pins ή ακόμη και με 84 pins.

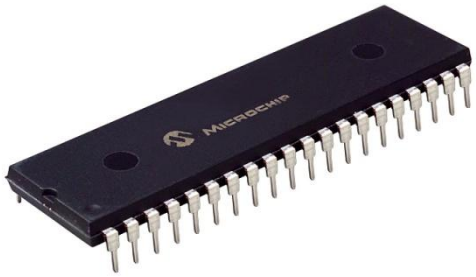
Εικόνα 2.3: Μικροελεγκτές PIC με διαφορετικό αριθμό ακίδων.

2.5.3. Διάκριση με βάση την συσκευασία

Υπάρχουν διαφόρων ειδών συσκευασίες των μικροελεγκτών PIC όπως για παράδειγμα:

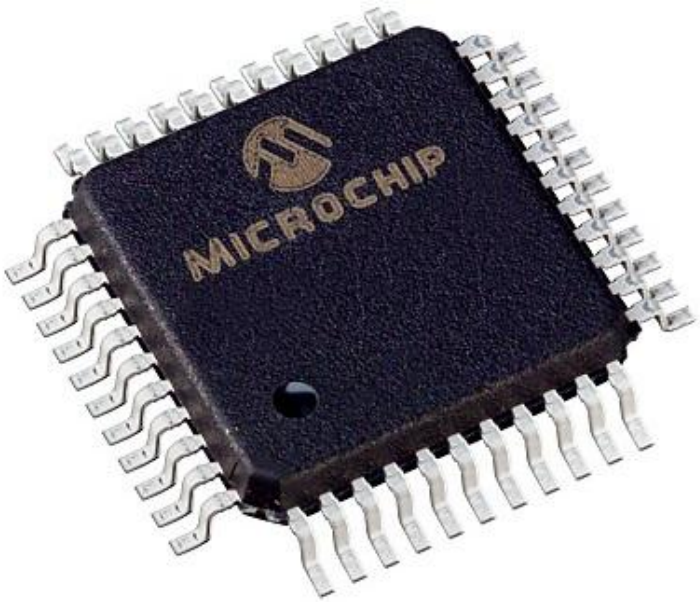
- DIP (Dual In Line)
- PLCC (Plastic Leaded Chip Carrier)
- PQFP (Plastic Quad Flat Pack)

Έτσι βάση της συσκευασίας έχουμε και διαφορετική διάταξη των ακίδων (pins) όπως φαίνεται και στις εικόνες.



Εικόνα 2.4: Συσκευασία DIP.

Εικόνα 2.5: Συσκευασία PLCC.



Εικόνα 2.6: Συσκευασία PQFP.

2.5.4. Διάκριση με βάση την ονομασία

Για την ονομασία των μικροελεγκτών PIC η εταιρία Microchip χρησιμοποιεί μία κωδικοποίηση για την περιγραφή κάποιον χαρακτηριστικών του μικροελεγκτή. Για παράδειγμα:

- Το πρώτο γράμμα (μετά τον αριθμό) που συναντάμε σε ένα όνομα μικροελεγκτή συμβολίζει τον τύπο μνήμης που

χρησιμοποιείτε στο μικροελεγκτή (π.χ. PIC16XX). Έτσι όταν βλέπουμε το γράμμα **C** σημαίνει ότι ο μικροελεγκτής έχει μνήμη προγράμματος τύπου EPROM (π.χ. PIC16CXXX). Τα γράμματα **CR** συμβολίζουν τους μικροελεγκτές που έχουν μνήμη προγράμματος τύπου ROM (π.χ. PIC16CRXXX). Συναντώντας το γράμμα **F** μπορούμε να καταλάβουμε πως ο μικροελεγκτής έχει μνήμη προγράμματος τύπου FLASH (π.χ. PIC16FXXX).

- Ο αριθμός που συναντάμε (π.χ. PIC16, PIC12) μπορεί να συμβολίζει τον αριθμό των ακίδων που έχει ο μικροελεγκτής ή το μήκος της εντολής που χρησιμοποιείται (π.χ. PIC12XXX έχει 8 ακίδες). Μικροελεγκτές που έχουν πάνω από 8 ακίδες και 12-ψήφιο ή 14-ψήφιο μήκος εντολής έχουν τον αριθμό 16 στο όνομα τους (π.χ. PIC16XXX). Ενώ αυτοί με συμβολισμό το 17 ή 18 έχουν 16-ψήφιο μήκος εντολής (π.χ. PIC17XXX ή PIC18XXX).
- Νεότερες εκδόσεις παλιών μικροελεγκτών συμβολίζονται με το γράμμα **A** στο τέλος του ονόματος τους (π.χ. PIC16F84A).

3. ΜΙΚΟΕΛΕΓΚΤΗΣ PIC16F887

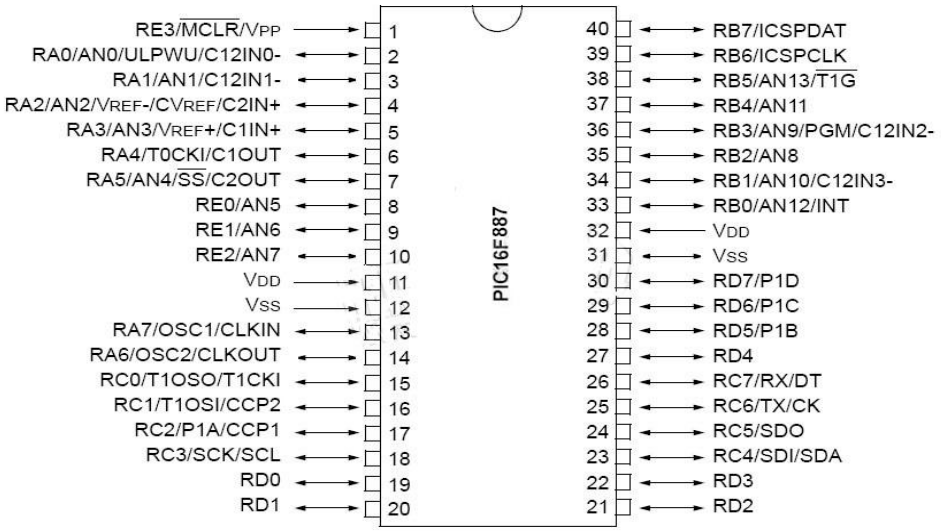
3.1. ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΜΙΚΡΟΕΛΕΓΚΤΗ PIC16F887

Στην εργασία χρησιμοποιείται ο PIC16F887 μικροελεγκτής της εταιρίας Microchip. Ο συγκεκριμένος μικροελεγκτής ανήκει στην κατηγορία DIP40 δηλαδή έχει 40 pins (ακίδες) και είναι της οικογενείας PIC16F.

Εικόνα 3.1: Μικροελεγκτής PIC16F887.

3.2. ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΟΥ PIC16F887

Τύπος μνήμης	Flash
Μέγεθος μνήμης	14 KB
Ταχύτητα CPU	5 MIPS
Μέγεθος μνήμης RAM	368 Bytes
Περιφερειακές ψηφιακές επικοινωνίες	1-UART, 1-A/E/USART, 1-SPI, 1-I^2C1-MSSP(SPI/I^2C)
Εύρος θερμοκρασίας	-40 C έως 125 C
Τάση λειτουργίας	2 V έως 5,5 V
Αριθμός ακίδων (pins)	40



Εικόνα 3.2: Μικροελεγκτής PIC16f887, έξοδοι ακίδων.

4. EASY PIC v7 connectivity

4.1. ΕΙΣΑΓΩΓΗ ΣΤΗΝ EASY PIC v7 connectivity

Η Easy PIC v7 connectivity είναι ένα προϊόν της Mikroelektronika. Έχει σχεδιαστεί για τον προγραμματισμό μικροελεγκτών PIC της Microchip και την άμεση εφαρμογή και εκτέλεση των λειτουργιών που έχουμε προγραμματίσει να εκτελέσει ο μικροελεγκτής. Αυτό μπορεί να επιτευχθεί χάρη στα ενσωματωμένα εργαλεία που έχει η Easy PIC v7 connectivity όπως για παράδειγμα οθόνη LCD, διακόπτες κτλ. Επίσης μπορούμε να προσθέσουμε επιπλέον μηχανισμούς όπως για παράδειγμα δέκτες GPS, GSM, Wi-Fi κτλ.

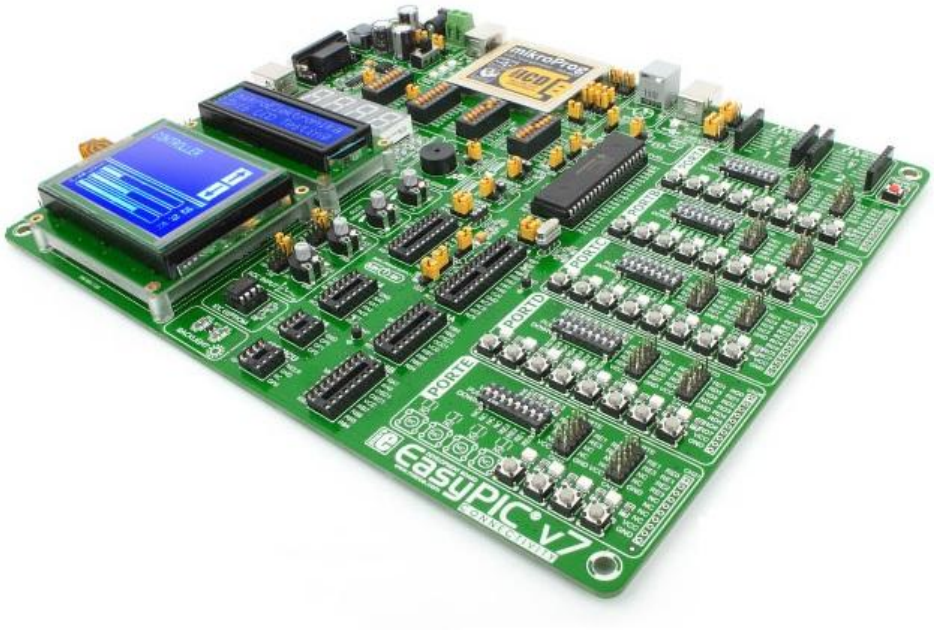
Με την Easy PIC v7 connectivity έχουμε τη δυνατότητα να προγραμματίσουμε τον μικροελεγκτή της επιλογής μας σε τρεις διαφορετικές γλώσσες προγραμματισμού:

- C
- Pascal
- Visual Basic

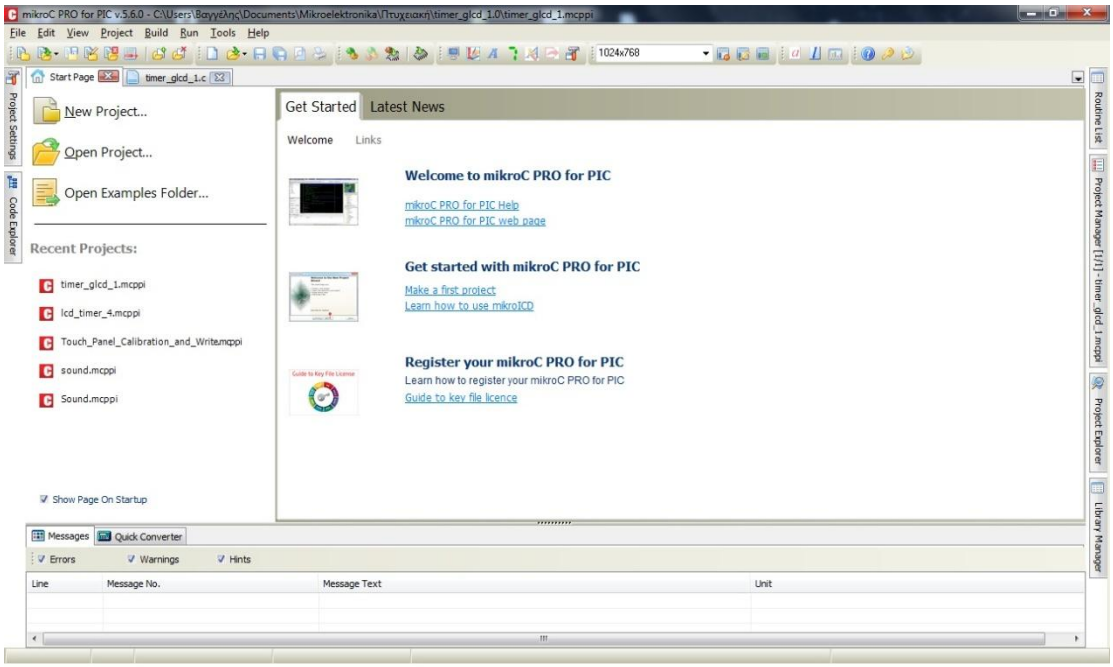
Η Microelectronica για αυτό το λόγω έχει δημιουργήσει τρεις compilers για κάθε γλώσσα προγραμματισμού:

- mikroC
- mikroPascal
- mikroBasic

Για την εργασία χρησιμοποιήθηκε ο mikroC compiler.



Εικόνα 4.1: Easy PIC v7 connectivity της Mikroelektronika.



Εικόνα 4.2: Αρχική οθόνη του compiler mikroC της Mikroelektronika.

4.2. ΜΙΚΡΟΕΛΕΓΚΤΕΣ ΣΤΗΝ EASY PIC v7 connectivity

Η Easy PIC v7 connectivity έχει την δυνατότητα να υποστηρίξει διάφορες οικογένειες μικροελεγκτών.

Αυτό γίνετε χάρη στη διπλή τάση που μπορεί να παρέχει και στις οχτώ διαφορετικές υποδοχές που υπάρχουν στην Easy PIC v7 connectivity.

Οι οικογένειες μικροελεγκτών που υποστηρίζει είναι:

- PIC10F
- PIC12F
- PIC16F
- PIC16Enh
- PIC18F
- PIC18FJ
- PIC18FK

Ενώ με βάση τον αριθμό των ακίδων έχουμε:

- DIP40
- DIP28
- DIP20
- DIP18A
- DIP18B
- DIP14
- DIP8

Επίσης υπάρχει υποδοχή για PIC 10F MCUs.

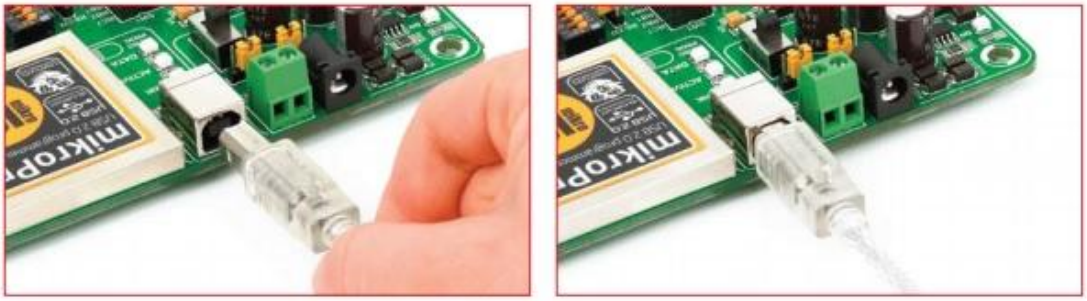
Αυτό έχει ως αποτέλεσμα η Easy PIC v7 connectivity να μπορεί να υποστηρίξει πάνω από 250 μικροελεγκτές.

4.3. ΤΡΟΦΟΔΟΣΙΑ ΣΤΗΝ EASY PIC v7 connectivity

Στην Easy PIC v7 connectivity έχουμε την δυνατότητα τροφοδοσίας στα 3.3V και στα 5V αναλόγως τον μικροελεγκτή που χρησιμοποιούμε.

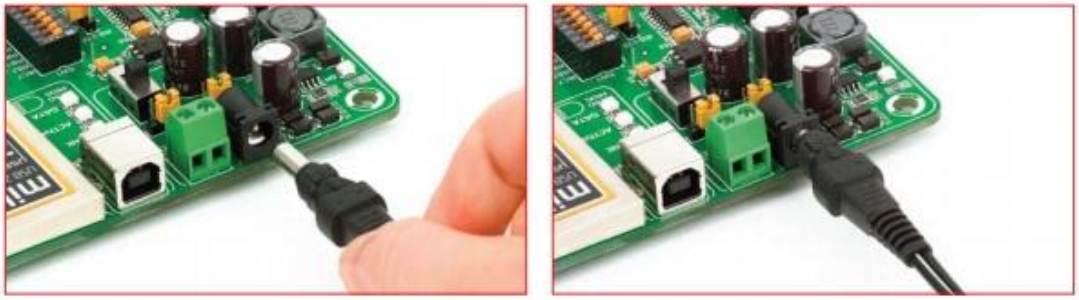
Υπάρχουν τρεις τρόποι τροφοδοσίας της Easy PIC v7 connectivity.

- Με την χρήση USB



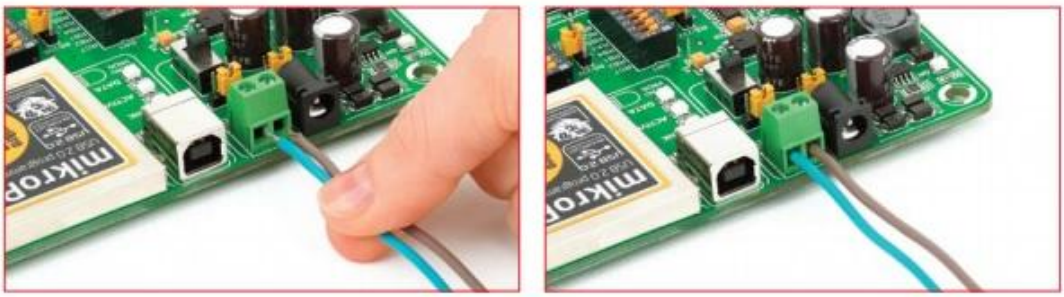
Εικόνα 4.3: Τροφοδοσία με USB.

- Με την χρήση αντάπτορα



Εικόνα 4.4: Τροφοδοσία με αντάπτορα.

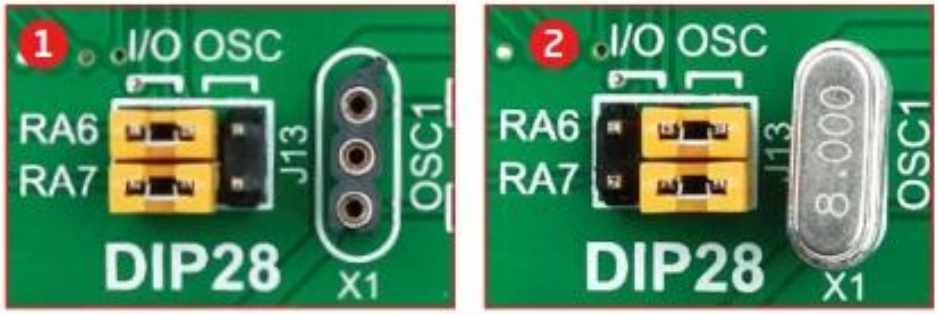
- Με την χρήση εργαστηριακής καλωδίωσης



Εικόνα 4.5: Τροφοδοσία με εργαστηριακή καλωδίωση.

4.4. ΧΡΟΝΙΣΜΟΣ ΣΤΗΝ EASY PIC v7 connectivity

Για το χρονισμό της Easy PIC v7 connectivity χρησιμοποιείτε κρύσταλλος χρονισμού στα 8MHz.

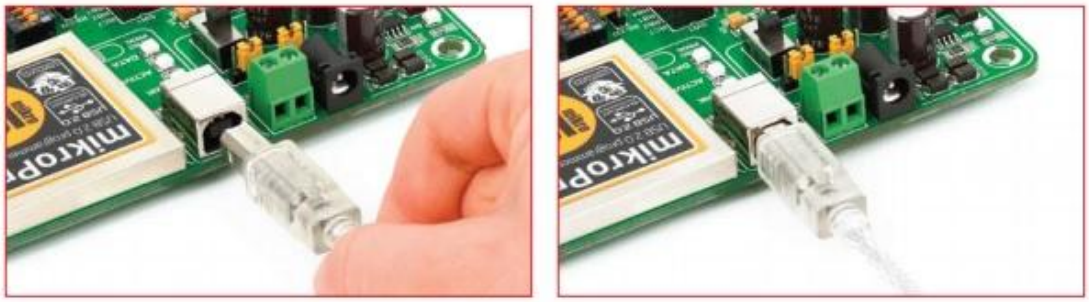


Εικόνα 4.6: 1) Είσοδος για κρύσταλλο χρονισμού, 2) Τοποθετημένος κρύσταλλος των 8MHz.

4.5. ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ ΣΤΗΝ EASY PIC v7 connectivity

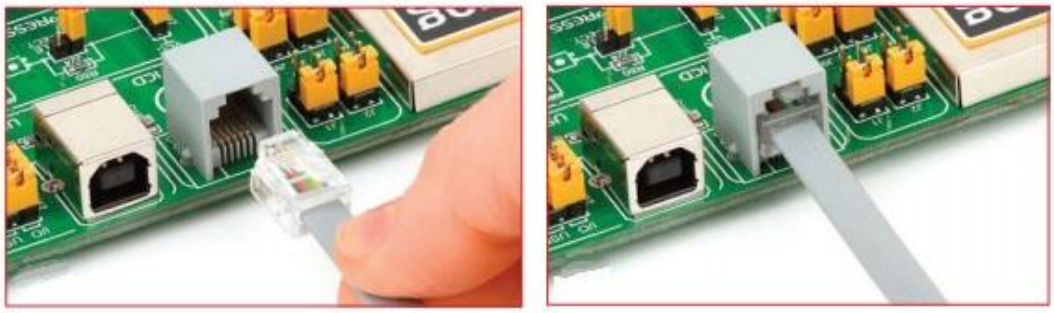
Υπάρχουν δύο τρόποι για τον προγραμματισμό του μικροελεγκτή που έχουμε τοποθετήσει στην Easy PIC v7 connectivity:

- Μέσω της θύρας USB



Εικόνα 4.7: Προγραμματισμός μέσω θύρας USB.

- Μέσω του ICD2/ICD3 με πλακέ καλώδιο και RJ12 κονέκτορα



Εικόνα 4.8: Προγραμματισμός μέσω του ICD2/ICD3 με πλακέ καλώδιο και RJ12 κονέκτορα.

Στην εργασία χρησιμοποιείτε η θύρα USB.

4.6. ΕΠΙΚΟΙΝΩΝΙΑ ΜΙΚΡΟΕΛΕΓΚΤΗ ΜΕ ΑΛΛΗ ΣΥΣΚΕΥΗ

Ο μικροελεγκτής που έχουμε συνδεμένο στην Easy PIC v7 connectivity μπορεί να έχει επικοινωνία με μία άλλη συσκευή (π.χ. PC) για την μετάδοση των δεδομένων που έχει ο μικροελεγκτής. Ο τρόπος επικοινωνίας διαφέρει ανάλογα με το μέσω που μπορεί να υποστηρίξει ο μικροελεγκτής. Για παράδειγμα μπορούμε να έχουμε επικοινωνία:

- Με θύρα RS-232

Εικόνα 4.9: Μετάδοση των δεδομένων που έχει ο μικροελεγκτής σε άλλη συσκευή μέσω θύρας RS-232.

- Με θύρα USB

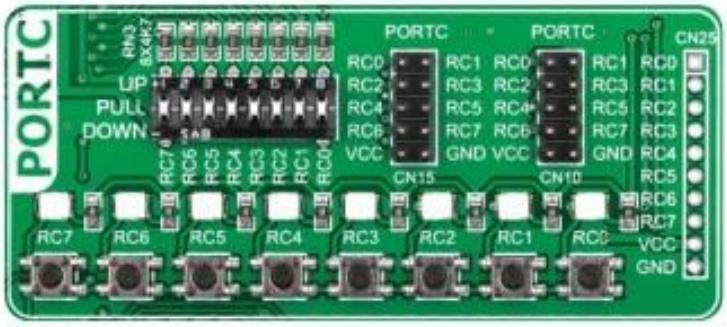
Εικόνα 4.10: Μετάδοση των δεδομένων που έχει ο μικροελεγκτής σε άλλη συσκευή μέσω θύρας USB.

4.7. ΕΙΣΟΔΟΙ / ΕΞΟΔΟΙ ΣΤΗΝ EASY PIC v7 connectivity

Στην Easy PIC v7 connectivity υπάρχουν μονάδες όπου χρησιμοποιούνται είτε σαν είσοδος για την επεξεργασία από τον μικροελεγκτή είτε σαν έξοδος μετά από την εκτέλεση κάποιας εντολής από τον μικροελεγκτή.

- Port groups

  Έχουμε πέντε port groups στην Easy PIC v7 connectivity PORT A, PORT B, PORT C, PORT D, PORT E, όπου το κάθε port έχει οχτώ LEDs, οχτώ push buttons, τριών καταστάσεων DIP switches (διακόπτες), όπου μπορούμε να ορίσουμε αν τα pins του port θα χρησιμοποιηθούν ως είσοδος ή ως έξοδος και Headers για την σύνδεση επιπλέον αξεσουάρ της Mikroelektronika.

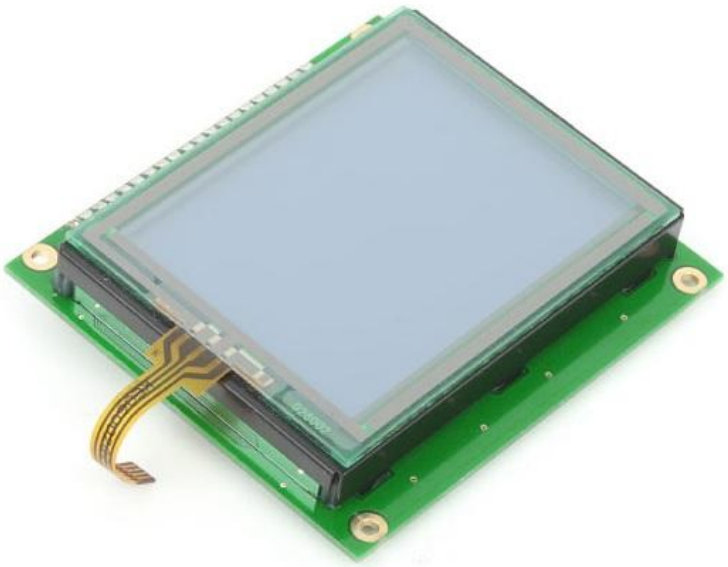


Εικόνα 4.11: Η PORT C με LEDs, push buttons, DIP switches και Headers.

- GLCD

  Η Easy PIC v7 connectivity έχει την δυνατότητα τοποθέτησης GLCD (Graphical Liquid Crystal Displays) οθόνης με ανάλυση 128x64 pixels για την απεικόνιση κάποιου κειμένου ή εικόνων κτλ. Η PORT B χρησιμοποιείται για την σύνδεση της οθόνης GLCD με τον μικροελεγκτή και η PORT D για την σύνδεση των δεδομένων της οθόνης GLCD με τον μικροελεγκτή.

  Ακόμη η GLCD οθόνη που χρησιμοποιείται σ' αυτή την εργασία έχει ήδη τοποθετημένο touch panel controller όπου μπορούμε να το χρησιμοποιήσουμε για την είσοδο δεδομένων όπως για παράδειγμα την σχεδίαση σχημάτων στην οθόνη.



Εικόνα 4.12: Οθόνη GLCD με touch panel controller.

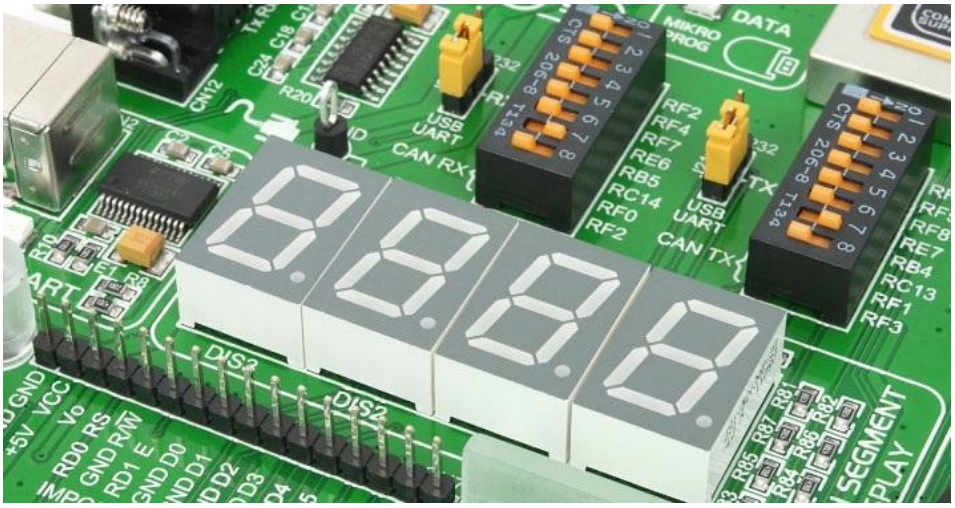
- LCD

  Επίσης στην Easy PIC v7 connectivity μπορούμε να συνδέσουμε μία LCD (Liquid Crystal Displays) οθόνη ανάλυση 2x16 pixels. Η LCD οθόνη μπορεί να χρησιμοποιηθεί για την απεικόνιση στάνταρ χαρακτήρων ή ακόμη και προσαρμοσμένων χαρακτήρων αρκεί να μην ξεπερνούν την ανάλυση της οθόνης. Ακόμη η LCD οθόνη χρησιμοποιεί και αυτή την PORT B για την σύνδεση της με τον μικροελεγκτή με αποτέλεσμα να μην μπορούμε να συνδέσουμε και τις δύο οθόνες (LCD και GLCD) ταυτόχρονα στην PIC v7 connectivity.

Εικόνα 4.13: Οθόνη LCD.

- 7-seg display

Στην Easy PIC v7 connectivity έχουμε τέσσερα 7-seg display για την αναπαράσταση τεσσάρων ψηφίων. Ένα 7-segment αποτελείται από εφτά LEDs και επιπλέον ένα ως δεκαδική τελεία (δηλαδή για να μπορεί να απεικονίσει και δεκαδικούς αριθμούς). Η σύνδεση των τεσσάρων 7-segments με τον μικροελεγκτή γίνετε μέσω της PORT D.



Εικόνα 4.14: 7-segment display τεσσάρων ψηφίων.

- Piezo Buzzer

Το Piezo Buzzer χρησιμοποιείται στην Easy PIC v7 connectivity για την παραγωγή ήχων. Το Piezo Buzzer χρησιμοποιώντας ηλεκτρικά σήματα παράγει ήχους. Για την σύνδεση του με τον μικροελεγκτή χρησιμοποιείται είτε το πρώτο pin της PORT E είτε το δεύτερο pin PORT C. Το Piezo Buzzer μπορεί να δημιουργήσει τους πιο δυνατούς ήχους στα 3,8kHz, γενικά όμως οι ήχοι που παράγει είναι στο διάστημα των 2kHz έως 4kHz.

Εικόνα 4.15: Piezo Buzzer για την παραγωγή ηχητικών σημάτων.

4.8. ΑΛΛΕΣ ΜΟΝΑΔΕΣ ΣΤΗΝ EASY PIC v7 connectivity

Στην Easy PIC v7 connectivity υπάρχουν επίσης και άλλες μονάδες που μπορούν να χρησιμοποιηθούν για την επικοινωνία δεδομένων με τον μικροελεγκτή. Αυτά είναι:

- Αναλογικός ή ψηφιακός αισθητήρας θερμοκρασίας
- ADC inputs

4.9. I^2C EEPROM (Electrically Erasable Programmable Read Only Memory)

Η EEPROM είναι συνήθως μια δευτερεύουσα μνήμη όπου τα δεδομένα μένουν αποθηκευμένα ακόμη και αν δεν τροφοδοτείται με ρεύμα. Η Easy PIC v7 connectivity υποστηρίζει συριακή EEPROM οι οποία χρησιμοποιεί I^2C communication interface και έχει μνήμη 1024 bytes.

4.9.1. Τι είναι το I^2C;

Το I^2C είναι ένας multi-master serial single-ended δίαυλος ο οποίος χρησιμοποιείται για να συνδέσει χαμηλής ταχύτητας περιφερικές συσκευές. Το I^2C χρησιμοποιεί δύο αμφίδρομους ανοιχτούς διαύλους, Serial Data Line (SDA) και Serial Clock (SCL). Οι δύο δίαυλοι επικοινωνούν με την κύρια συσκευή. Κάθε δευτερεύουσα συσκευή έχει μία μοναδική διεύθυνση, εστί έχουμε την δυνατότητα πάνω από 112 δευτερεύουσες συσκευές να μπορούν να συνδεθούν στον ίδιο δίαυλο.

4.10. mikroProg

Για το τέλος άφησα το πιο σημαντικό κομμάτι της Easy PIC v7 connectivity είναι το mikroProg. Το mikroProg είναι ένας fast USB 2.0 programmer με mikroICD hardware In-Circuit Debugger, όπου με τον έξυπνο μηχανισμό μπορεί να υποστηρίξει όλες τις οικογένειες των μικροελεγκτών που αναφέραμε παραπάνω.

Ακόμη έχουμε το software mikroProg Suite με το οποίο «φορτώνουμε» στον μικροελεγκτή το πρόγραμμα που έχουμε δημιουργήσει.

Εικόνα 4.16: mikroProg.

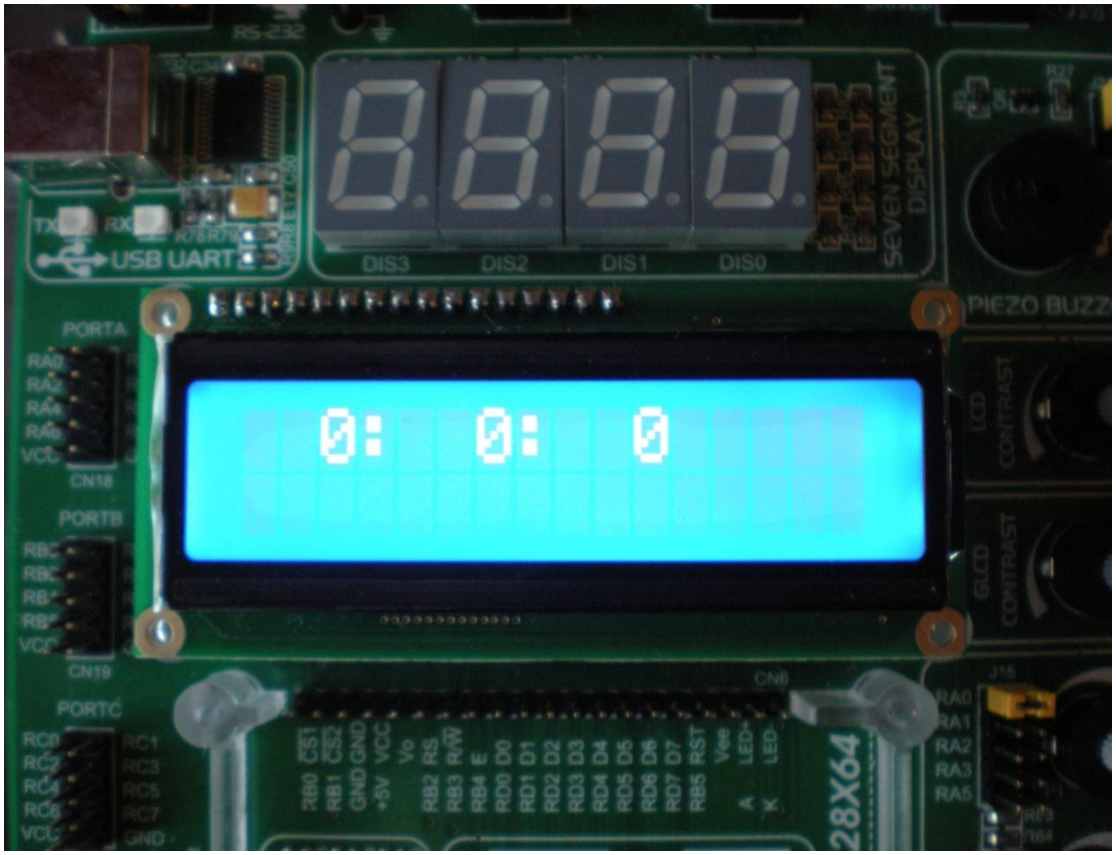
Εικόνα 4.17: mikroProg Suite.

5. ΧΡΟΝΟΜΕΤΡΟ ΜΕ PIC16F887 ΣΕ LCD ΟΘΟΝΗ

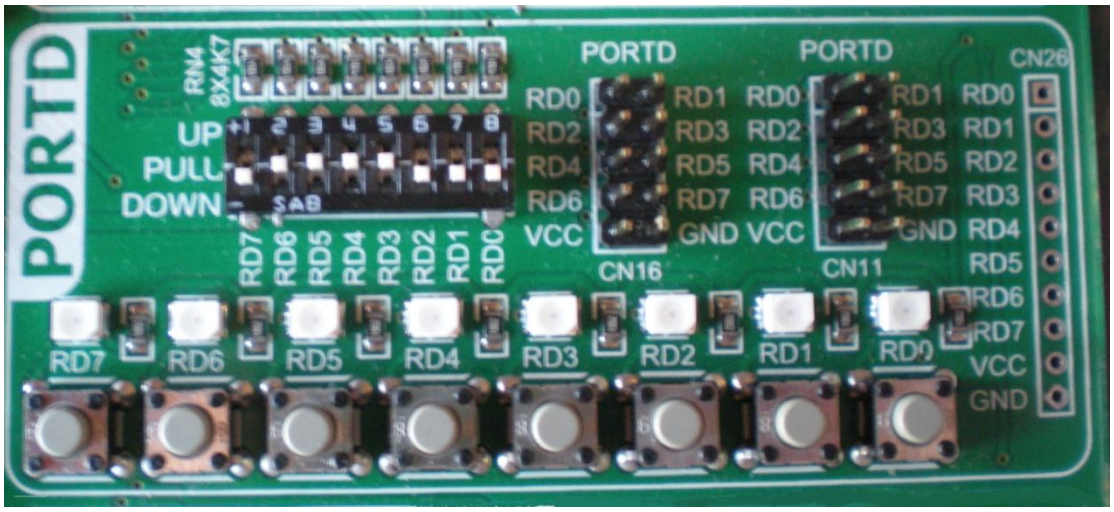
5.1. ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΧΡΟΝΟΜΕΤΡΟΥ

Σ’ αυτή την εργασία δημιουργήθηκε ένα χρονόμετρο όπου ο χρήστης εισάγει την ώρα που θέλει και το χρονόμετρο αρχίζει να μετράει έως ότου φτάσει στην ώρα που έχουμε ορίσει και μας ειδοποιήσει με χαρακτηριστικό ήχο. Για την δημιουργία του χρησιμοποιήθηκαν:

- **Μικροελεγκτής PIC16F887**. Όπου κάνει την επεξεργασία των δεδομένων, δηλαδή παίρνει ως είσοδο το χρόνο από τον χρήστη (δευτερόλεπτα, λεπτά, ώρα) και εφόσον του δοθεί η εντολή για την έναρξη του χρονομέτρου ξεκινάει την μέτρηση του χρόνου και τα εμφανίζει στην οθόνη και μόλις φτάσει στον χρόνο που του έχει ορίσει ο χρήστης «δίνει» την εντολή να χτυπήσει το buzzer.
- **Push buttons**. Ως είσοδος για την ρύθμιση του χρόνου από τον χρήστη χρησιμοποιούνται τρία push buttons από την PORT D, για την ρύθμιση των δευτερολέπτων το RC0, για τα λεπτά το RC1 και για την ώρα το RC2. Ακόμη χρησιμοποιείται το RC7 για την έναρξη του χρονομέτρου.
- **LCD οθόνη**. Στην ουσία είναι η μία έξοδος του προγράμματος όπου απεικονίζονται ο χρόνος που έχει ρυθμίσει ο χρήστης και ο χρόνος που μετράει έως ότου φτάσει στο ίδιο χρόνο που έβαλε ο χρήστης για τον τερματισμό του χρονομέτρου.
- **Piezo Buzzer**. Η άλλη έξοδός του προγράμματος είναι το piezo buzzer όπου με το που τερματίσει το χρονόμετρο παίρνει την εντολή από τον μικροελεγκτή για την παραγωγή ήχου για την διάρκεια που έχει οριστεί από το πρόγραμμα.

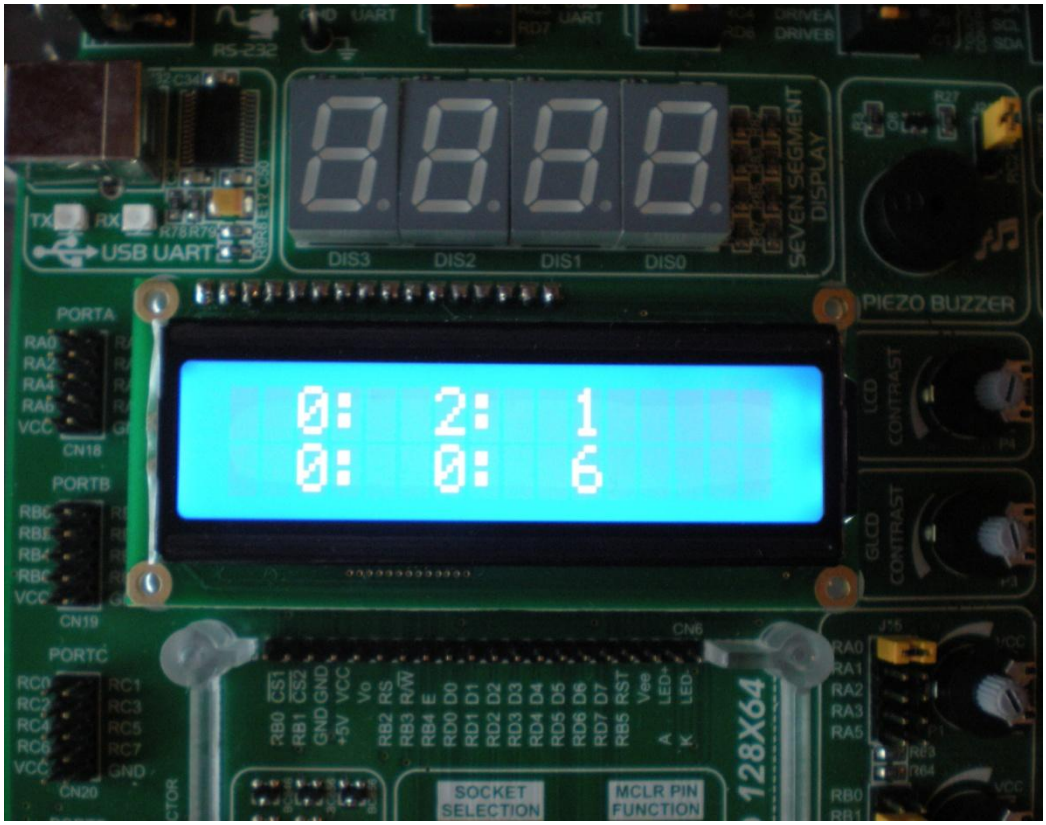


Εικόνα 5.1: Αρχική οθόνη χρονομέτρου σε οθόνη LCD.

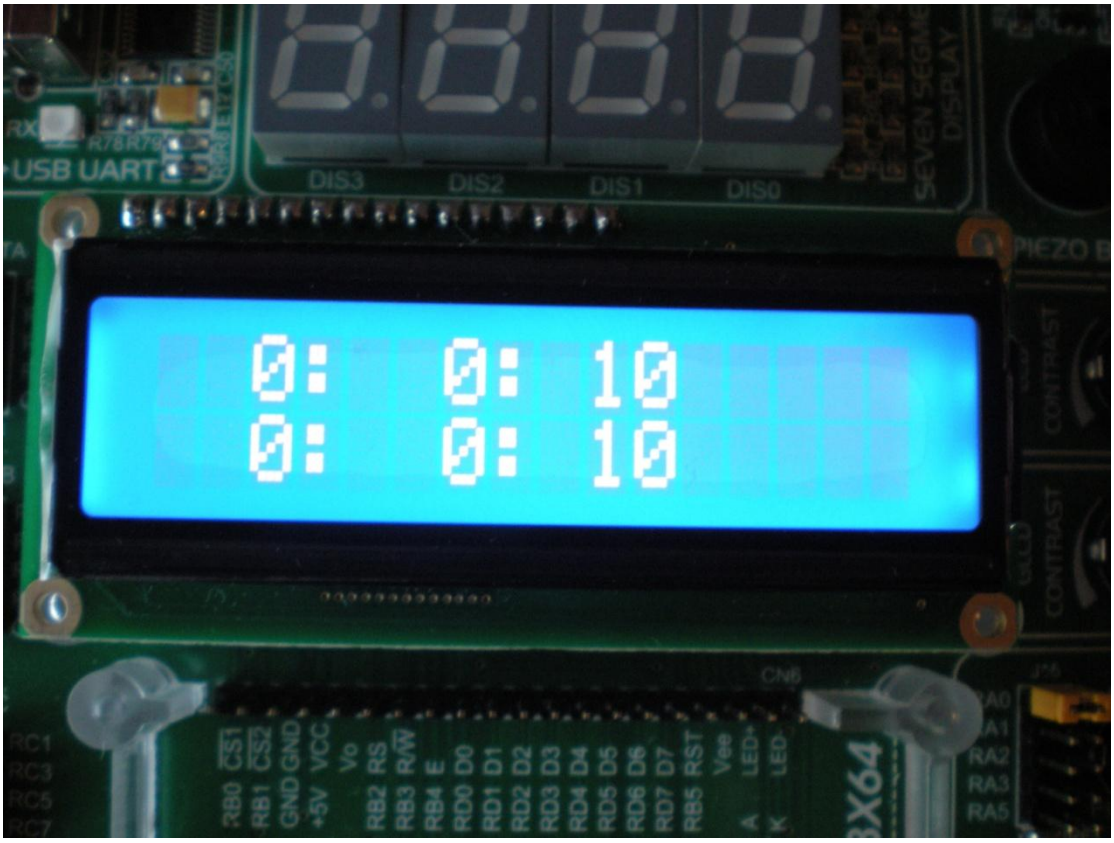


Εικόνα 5.2: Push buttons της PORT D για την ρύθμιση του χρονομέτρου.

Εικόνα 5.3: Ρυθμισμένο χρονόμετρο για να χτυπήσει σε 2 λεπτά και 1 δευτερόλεπτο.



Εικόνα 5.4: Το χρονόμετρο έχει ξεκινήσει την χρονομέτρηση.



Εικόνα 5.5: Το χρονόμετρο έχει τερματίσει στο χρόνο που ήταν ρυθμισμένο και χτυπάει το buzzer.

5.2. ΚΩΔΙΚΑΣ

```
//  δήλωση συνδέσεων της LCD οθόνης
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
// τέλος δήλωσης συνδέσεων της LCD

// δήλωση μεταβλητών
char txt1[4]; // πίνακας για τα δευτερόλεπτα
```

```
char txt2[4]; // πίνακας για τα λεπτά
char txt3[4]; // πίνακας για την ώρα

int x= 0, y= 0, a= 0;        /* x --> δευτερόλεπτα, y --> λεπτά, a --> ώρα  */
// τέλος μεταβλητών


void xronometro(int ah, int ym, int xs)    /* ah --> ώρα, ym --> λεπτά, xs --> δευτερόλεπτα */
{
 int a1= 0, y1= 0, x1= 0, i=0;          /* a1 --> ώρα, y1 --> λεπτά, x1 -->> δευτερόλεπτα */

 lcd_out(2,4,":");
 lcd_out(2,8,":");
 for (a1=0; a1<ah; a1++)          //  μετράει μέχρι την ώρα που έχουμε ορίσει
 {
  bytetostr(a1,txt3);
  lcd_out(2,1,txt3);
  for (y1=0; y1<60; y1++)        // τρέχει τα λεπτά
  {
   bytetostr(y1,txt2);
   lcd_out(2,5,txt2);
   for (x1=0; x1<60; x1++)     // τρέχει τα δευτερόλεπτα
   {
    bytetostr(x1,txt1);
    lcd_out(2,9,txt1);
    delay_ms(1000);
   }
  }
 }
 bytetostr(a1,txt3);
 lcd_out(2,1,txt3);
 y1= 0;
 x1= 0;
 bytetostr(y1,txt2);
 lcd_out(2,5,txt2);
 bytetostr(x1,txt1);
 lcd_out(2,9,txt1);

 for (y1=0; y1<ym; y1++)          // μετράει μέχρι τα λεπτά που έχουμε ορίσει
 {
  bytetostr(y1,txt2);
  lcd_out(2,5,txt2);
```

```
   for (x1=0; x1<60; x1++)          // τρέχει τα δευτερόλεπτα
   {
     bytetostr(x1,txt1);
     lcd_out(2,9,txt1);
     delay_ms(1000);
   }
  }
  bytetostr(y1,txt2);
  lcd_out(2,5,txt2);
  x1=0;
  bytetostr(x1,txt1);
  lcd_out(2,9,txt1);

  for (x1=0; x1<xs; x1++)        // μετράει μέχρι τα δευτερόλεπτα που έχουμε ορίσει
  {
   bytetostr(x1,txt1);
   lcd_out(2,9,txt1);
   delay_ms(1000);
  }
  bytetostr(x1,txt1);
  lcd_out(2,9,txt1);
  for (i=0; i<20; i++)     // όταν τελειώσει ο χρόνος χτυπάει το buzzer
  {
   sound_play(880,50);
   sound_play(1046,50);
   sound_play(1318,50);
  }

} // τέλος συνάρτησης χρονόμετρο



void main()
{
  ANSEL= 0;                    // Configure AN pins as digital I/O
  ANSELH= 0;
  C1ON_bit= 0;                  // Disable comparators
  C2ON_bit= 0;
  TRISD= 255;

  sound_init(&PORTE, 1);

  Lcd_Init();
  Lcd_Cmd(_LCD_CLEAR);
```

```
Lcd_Cmd(_LCD_CURSOR_OFF);

// αρχική εμφάνιση
bytetostr(x,txt1);
lcd_out(1,9,txt1);
bytetostr(y,txt2);
lcd_out(1,5,txt2);
lcd_out(1,8,":");
bytetostr(a,txt3);
lcd_out(1,1,txt3);
lcd_out(1,4,":");
// τέλος αρχικής εμφάνισης

// ρύθμιση χρονομέτρου
while (1)
{
  if (Button(&PORTD, 0, 100, 1)) // ρύθμιση δευτερολέπτων
  {
    x++;
    bytetostr(x,txt1);
    lcd_out(1,9,txt1);
    if (x>59)
    {
      x=0;
      bytetostr(x,txt1);
      lcd_out(1,9,txt1);
    }
  } // τέλος ρύθμισης δευτερολέπτων

  if (Button(&PORTD, 1, 100, 1)) // ρύθμιση λεπτών
  {
    y++;
    bytetostr(y,txt2);
    lcd_out(1,5,txt2);
    if (y>59)
    {
      y=0;
      bytetostr(y,txt2);
      lcd_out(1,5,txt2);
    }
  } // τέλος ρύθμισης λεπτών

  if (Button(&PORTD, 2, 100, 1)) // ρύθμιση ώρας
  {
```

```
      a++;
      bytetostr(a,txt3);
      lcd_out(1,1,txt3);
      if (a>24)
      {
        a=0;
        bytetostr(a,txt3);
        lcd_out(1,1,txt3);
      }
    } // Τέλος ρύθμισης ώρας

    if (Button(&PORTD, 7, 100, 1))
    {
      goto Error;
    }
  } // τέλος της while(1)
   // τέλος ρύθμισης χρονομέτρυ

  Error:
  xronometro(a, y, x);


} // τέλος main
```

Σημείωση: Με το σύμβολο των δύο καθέτων (π.χ. //σχόλιο) σημαίνει ότι ακολουθεί σχόλιο. Το ίδιο ισχύει και για τα σχόλια που αρχίζουν με κάθετο και αστεράκι και τελειώνουν με αστεράκι και κάθετο (π.χ. /* σχόλιο*/).

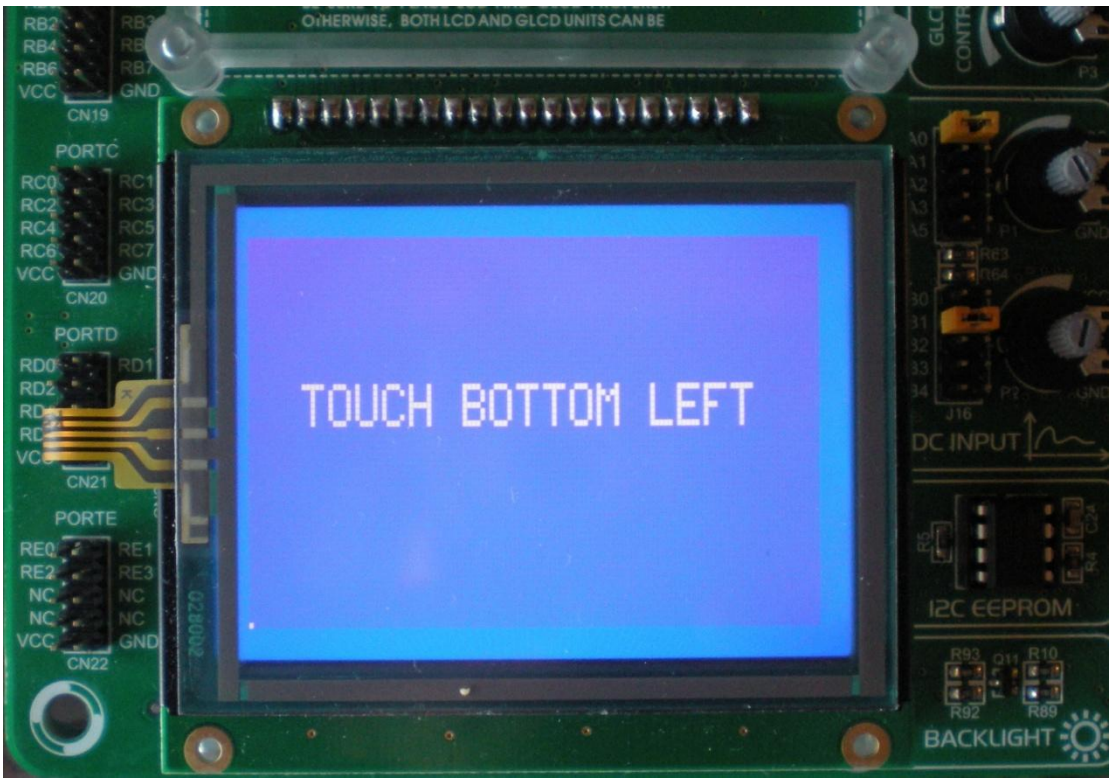
6. ΧΡΟΝΟΜΕΤΡΟ ΜΕ PIC16F887 ΣΕ ΟΘΟΝΗ ΑΦΗΣ GLCD

6.1. ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΧΡΟΝΟΜΕΤΡΟΥ

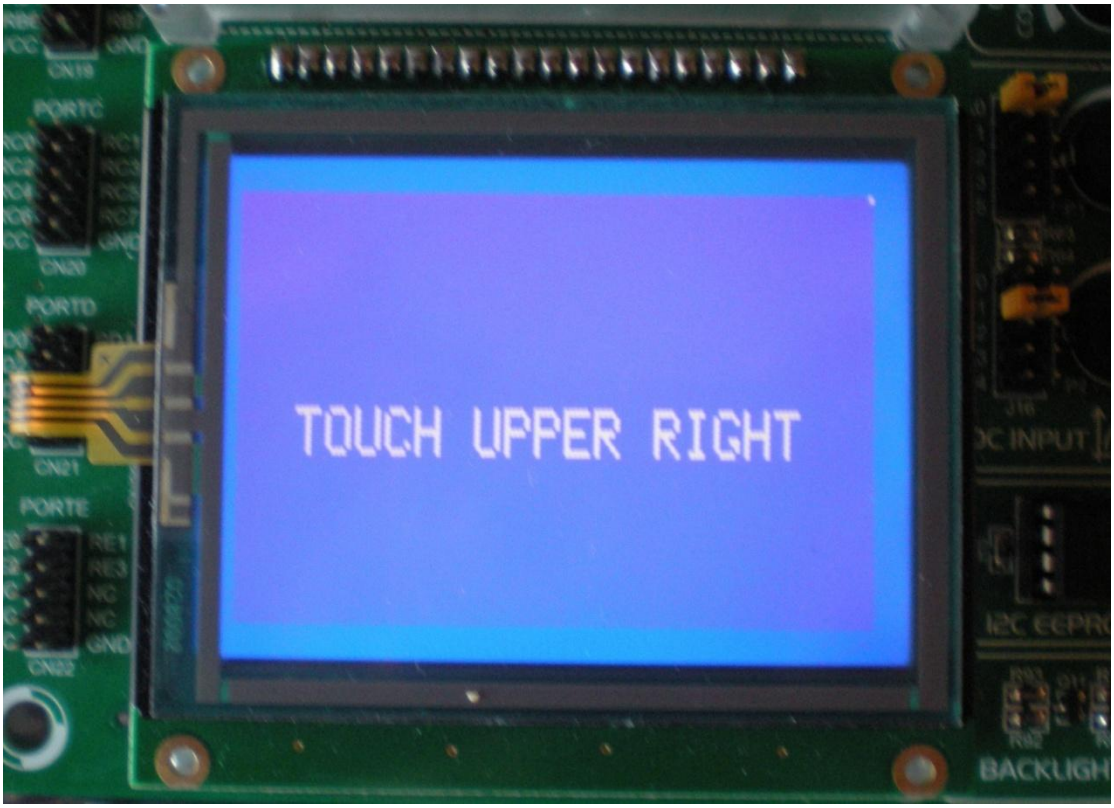
Όπως και στην προηγούμενη εργασία έτσι και σ' αυτή έχει δημιουργηθεί ένα χρονόμετρο όπου ο χρήστης εισάγει το χρόνο και επιλέγει την έναρξη του χρονομέτρου έως ότου τερματίσει και ακούσουμε το χαρακτηριστικό ήχο. Σημαντική διαφορά με την προηγούμενη εργασία είναι ότι σ' αυτή χρησιμοποιείται μια οθόνη αφής. Έτσι για την δημιουργία αυτής της εργασίας έχουμε:

- **Μικροελεγκτής PIC16F887**. Ο οποίος χρησιμοποιείται για την επεξεργασία των δεδομένων όπως και στην προηγούμενη εργασία (κεφάλαιο 5.1).
- **Οθόνη GLCD με Touch Panel**. Σ' αυτή την εργασία η είσοδος των δεδομένων γίνετε μέσω της οθόνης αφής όπου ο χρήστης χρησιμοποιώντας τα κουμπιά που έχουν σχεδιαστή στην οθόνη μπορεί να ρυθμίσει το χρόνο. Έχει σχεδιαστή ένα κουμπί για την ρύθμιση των δευτερολέπτων, ένα για τα λεπτά και ένα για την ρύθμιση της ώρας. Ακόμη έχει σχεδιαστή ένα κουμπί για την έναρξη της χρονομέτρησης και ακόμη ένα κουμπί για το μηδενισμό του χρονομέτρου που έχει ρυθμίσει ο χρήστης.
- **Piezo Buzzer**. Όπως και στην προηγούμενη εργασία (κεφάλαιο 5.1) χρησιμοποιείται για την παραγωγή του χαρακτηριστικού ήχου που μας ενημερώνει ότι το χρονόμετρο έχει τερματίσει στο χρόνο που έχει ορίσει ο χρήστης.

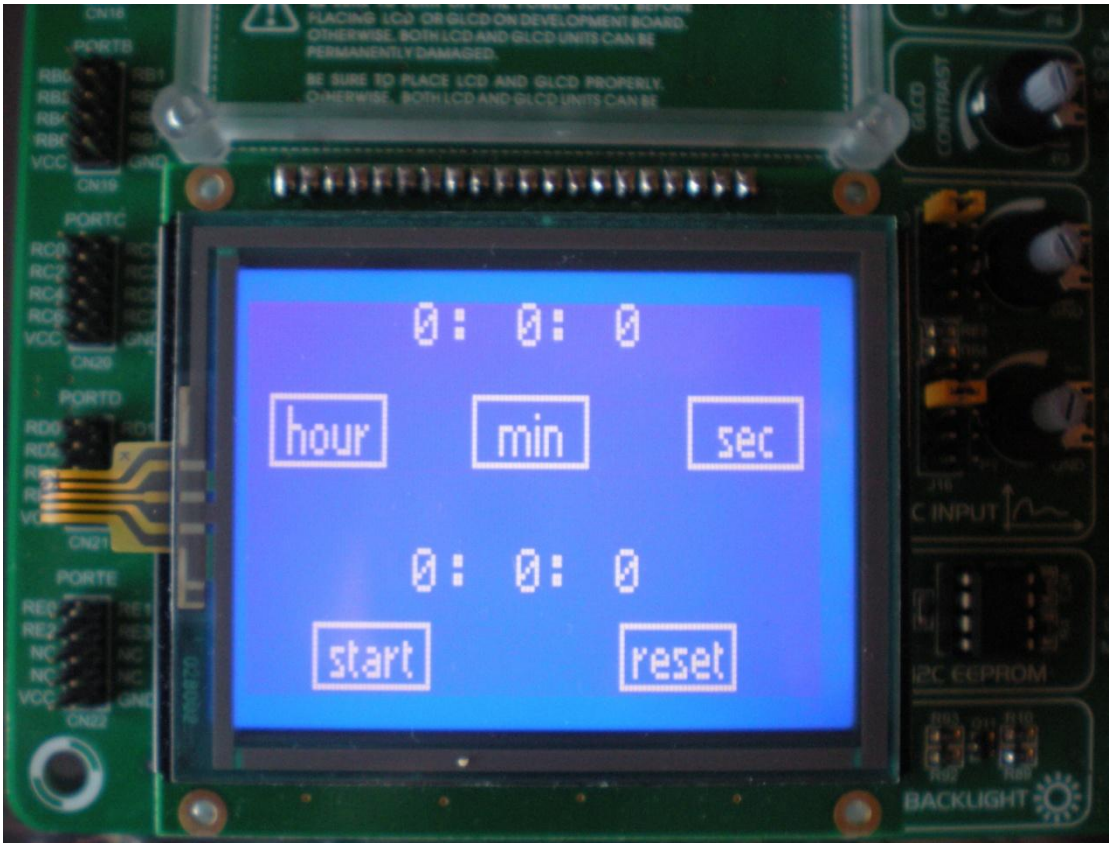
Σημείωση: Με το που ανοίξουμε την οθόνη αφής και «τρέξει» για πρώτη φορά το πρόγραμμα μας ζητείτε να κάνουμε βαθμονόμηση της οθόνης όπως φαίνετε και στις εικόνες 6.1 και 6.2.



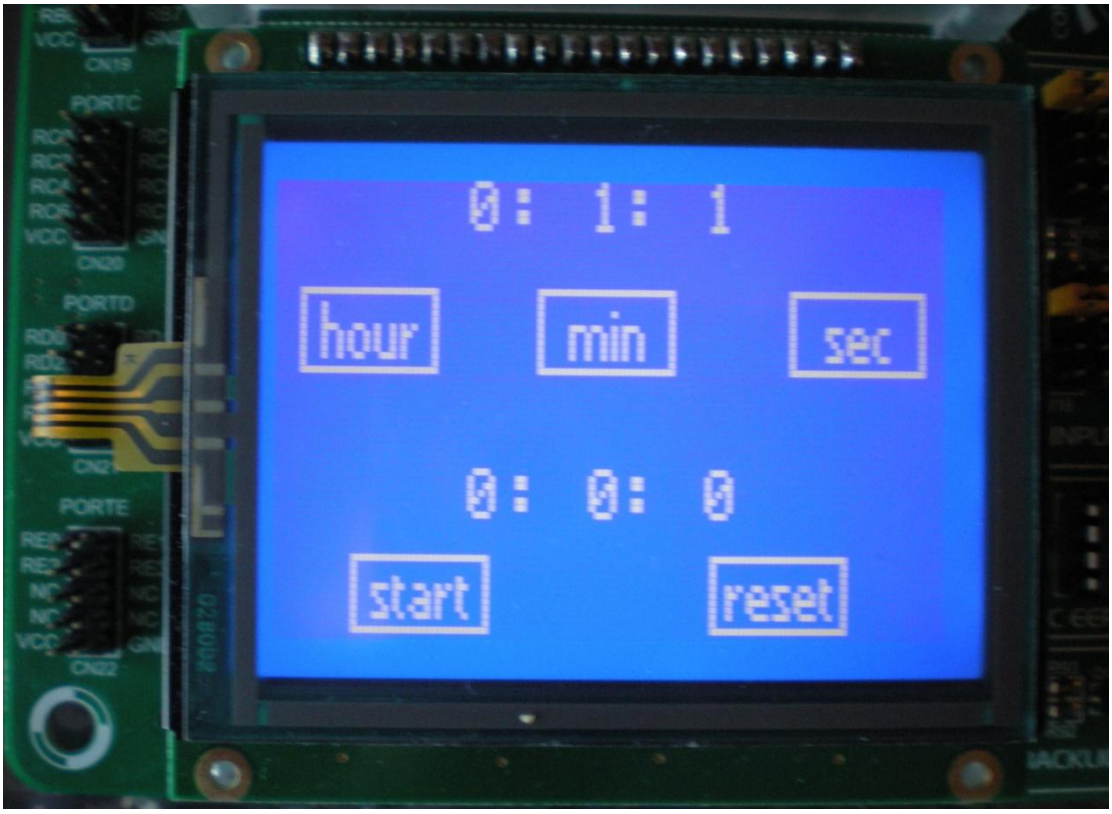
Εικόνα 6.1: Ένδειξη για την ρύθμιση της διαγωνίου στην οθόνη αφής.



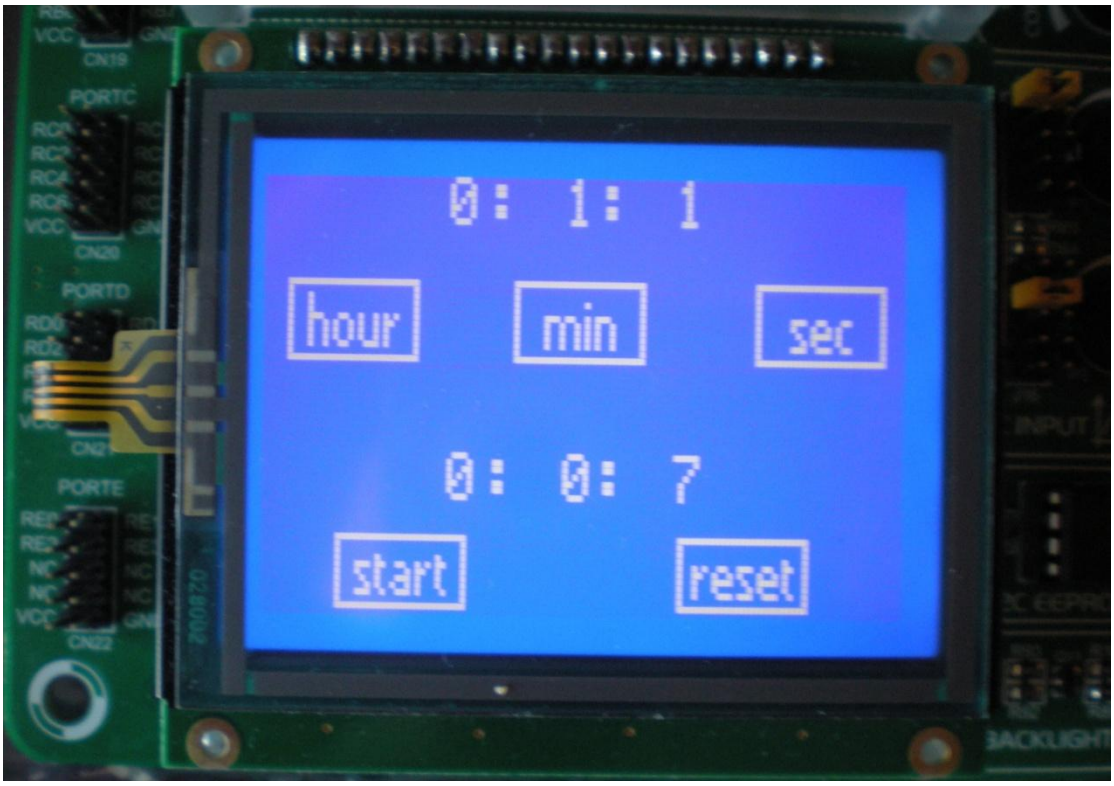
Εικόνα 6.2: Ένδειξη για την ρύθμιση της διαγωνίου στην οθόνη αφής.



Εικόνα 6.3: Αρχική οθόνη για την ρύθμιση του χρονομέτρου με οθόνη αφής.



Εικόνα 6.4: Ρυθμισμένο χρονόμετρο για να χτυπήσει σε 1 λεπτό και 1 δευτερόλεπτο.



Εικόνα 6.5: Χρονόμετρο κατά την διάρκεια χρονομέτρησης.

6.2. ΚΩΔΙΚΑΣ

```
// δήλωση συνδέσεων της GLCD οθόνης
char GLCD_DataPort at PORTD;

sbit GLCD_CS1 at RB0_bit;
sbit GLCD_CS2 at RB1_bit;
sbit GLCD_RS  at RB2_bit;
sbit GLCD_RW  at RB3_bit;
sbit GLCD_EN  at RB4_bit;
sbit GLCD_RST at RB5_bit;

sbit GLCD_CS1_Direction at TRISB0_bit;
sbit GLCD_CS2_Direction at TRISB1_bit;
sbit GLCD_RS_Direction  at TRISB2_bit;
sbit GLCD_RW_Direction  at TRISB3_bit;
sbit GLCD_EN_Direction  at TRISB4_bit;
sbit GLCD_RST_Direction at TRISB5_bit;
// τέλος δήλωσης συνδέσεων της GLCD

//  δήλωση συνδέσεων του Touch Panel
sbit DriveA at RC0_bit;
sbit DriveB at RC1_bit;
sbit DriveA_Direction at TRISC0_bit;
sbit DriveB_Direction at TRISC1_bit;
// τέλος δήλωσης συνδέσεων του Touch Panel

// δήλωση μεταβλητών
int a, y, x, a1=0, y1=0, x1=0, i;     // x --> δευτερόλεπτα, y --> λεπτά, a --> ώρα
int x_coord, y_coord;                 /* μεταβλητές για την αποθήκευση της τιμής που
πιέζουμε στον άξονα x – y */

char txt_wra[4];
char txt_lepta[4];
char txt_deuterolepta[4];
// τέλος δήλωσης μεταβλητών



void Initialize() {
  DriveA_Direction = 0;
  DriveB_Direction = 0;
  ANSEL  = 3;
```

```
  ANSELH = 0;
  TRISA  = 3;

  C1ON_bit = 0;
  C2ON_bit = 0;

  sound_init(&PORTE, 1);


  Glcd_Init();
  Glcd_Fill(0);

  ADC_Init();
  TP_Init(128, 64, 0, 1);
  TP_Set_ADC_Threshold(450);
}

void Calibrate() {

  Glcd_Dot(0,63,1);
  Glcd_Write_Text("TOUCH BOTTOM LEFT",12,3,1);
  TP_Calibrate_Bottom_Left();
  Delay_ms(1000);

  Glcd_Dot(0,63,0);
  Glcd_Dot(127,0,1);
  Glcd_Write_Text("                ",12,3,1);
  Glcd_Write_Text("TOUCH UPPER RIGHT",12,4,1);
  TP_Calibrate_Upper_Right();

  Delay_ms(1000);
}


void main()
{
  Initialize();

  Glcd_Write_Text("CALIBRATION",32,3,1);
  Delay_ms(1000);
  Glcd_Fill(0);
  Calibrate();

  Glcd_Fill(0);
```

```
Error:
a= 0;
y= 0;
x= 0;

// αρχική εμφάνιση
bytetostr (a, txt_wra);
Glcd_Write_Text(txt_wra, 25, 0, 1);
Glcd_Write_Char(':', 45, 0, 1);
bytetostr (y, txt_lepta);
Glcd_Write_Text(txt_lepta, 48, 0, 1);
Glcd_Write_Char(':', 67, 0, 1);
bytetostr (x, txt_deuterolepta);
Glcd_Write_Text(txt_deuterolepta, 70, 0, 1);
bytetostr (a, txt_wra);
Glcd_Write_Text(txt_wra, 25, 5, 1);
Glcd_Write_Char(':', 45, 5, 1);
bytetostr (y, txt_lepta);
Glcd_Write_Text(txt_lepta, 48, 5, 1);
Glcd_Write_Char(':', 67, 5, 1);
bytetostr (x, txt_deuterolepta);
Glcd_Write_Text(txt_deuterolepta, 70, 5, 1);
// τέλος αρχικής εμφάνισης

// κουμπιά hour, min και sec
Glcd_Rectangle (5, 15, 30, 26, 1);
Glcd_Write_Text_Adv("hour",9,16);
Glcd_Rectangle (50, 15, 75, 26, 1);
Glcd_Write_Text_Adv("min",56,16);
Glcd_Rectangle (98, 15, 123, 26, 1);
Glcd_Write_Text_Adv("sec",105,16);
// τέλος κουμπιά hour, min και sec

// κουμπιά start και reset
Glcd_Rectangle (15, 52, 40, 62, 1);
Glcd_Write_Text_Adv("start",19,52);
Glcd_Rectangle(83, 52, 108, 62, 1);
Glcd_Write_Text_Adv("reset",86,52);
// τέλος κουμπιά start και reset

while (1)
{
  if (TP_Press_Detect())
```

```
        {
          if (TP_Get_Coordinates(&x_coord, &y_coord) == 0)
          {
            if ((x_coord>= 98) && (x_coord<= 123) && (y_coord>= 15) &&
(y_coord<= 26))                // Αν πατήσει το κουμπί sec αυξάνει τα δευτερόλεπτα
{
              x++;
              bytetostr (x, txt_deuterolepta);
              if (x> 59)
              {
                x= 0;
                bytetostr (x,txt_deuterolepta);
              }
              Glcd_Write_Text(txt_deuterolepta, 70, 0, 1);
              delay_ms (100);
            } // τέλος κουμπί δευτερόλεπτα

            if ((x_coord>= 50) && (x_coord<= 75) && (y_coord>= 15) &&
(y_coord<= 26))          // αν πατήσει το κουμπί min αυξάνει τα λεπτά
            {
              y++;
              bytetostr (y, txt_lepta);
              if (y> 59)
              {
                y= 0;
                bytetostr (y,txt_lepta);
              }
              Glcd_Write_Text(txt_lepta, 48, 0, 1);
              delay_ms (100);
            } // τέλος κουμπί λεπτά

            if ((x_coord>= 5) && (x_coord<= 30) && (y_coord>= 15) &&
(y_coord<= 26))      // αν πατήσει το κουμπί hour αυξάνει την ώρα
            {
              a++;
              bytetostr (a,txt_wra);
              if (a> 24)
              {
                 a= 0;
                 bytetostr (a,txt_wra);
              }
              Glcd_Write_Text (txt_wra, 25, 0, 1);
              delay_ms (100);
            }   // τέλος κουμπί ώρα
```

```
        if ((x_coord>= 15) && (x_coord<= 40) && (y_coord>= 52) &&
(y_coord<= 62))      // αν πατήσει το κουμπί start αρχίζει η χρονομέτρηση
        {

          // χρονόμετρο
          for (a1=0; a1<a; a1++) /* μετράει την ώρα έως εκεί που το έχουμε
ρυθμίσει */
          {
            bytetostr(a1,txt_wra);
            Glcd_Write_Text(txt_wra, 25, 5, 1);

            for (y1=0; y1<60; y1++)                    // τρέχει τα λεπτά
            {
              bytetostr(y1,txt_lepta);
              Glcd_Write_Text(txt_lepta, 48, 5, 1);

              for (x1=0; x1<60; x1++)                  // τρέχει τα δευτερόλεπτα
              {
                bytetostr(x1,txt_deuterolepta);
                Glcd_Write_Text(txt_deuterolepta, 70, 5, 1);
                delay_ms(1000);
              }
            }
          }
          bytetostr(a1,txt_wra);
          Glcd_Write_Text(txt_wra, 25, 5, 1);
          y1= 0;
          x1= 0;
          bytetostr(y1,txt_lepta);
          Glcd_Write_Text(txt_lepta, 48, 5, 1);
          bytetostr(x1,txt_deuterolepta);
          Glcd_Write_Text(txt_deuterolepta, 70, 5, 1);

          for (y1=0; y1<y; y1++) // μετράει τα λεπτά έως εκεί που έχουμε ρυθμίση
          {
            bytetostr(y1,txt_lepta);
            Glcd_Write_Text(txt_lepta, 48, 5, 1);
            for (x1=0; x1<60; x1++)
            {
              bytetostr(x1,txt_deuterolepta);          // τρέχει τα δευτερόλεπτα
              Glcd_Write_Text(txt_deuterolepta, 70, 5, 1);
              delay_ms(1000);
            }
```

```
                }
                bytetostr(y1,txt_lepta);
                Glcd_Write_Text(txt_lepta, 48, 5, 1);
                x1=0;
                bytetostr(x1,txt_deuterolepta);
                Glcd_Write_Text(txt_deuterolepta, 70, 5, 1);
                for (x1=0; x1<x; x1++) /* τρέχει τα δευτερόλεπτα έως εκεί που έχουμε
ρυθμίσει */
                {
                  bytetostr(x1,txt_deuterolepta);
                  Glcd_Write_Text(txt_deuterolepta, 70, 5, 1);
                  delay_ms(1000);
                }
                bytetostr(x1,txt_deuterolepta);
                Glcd_Write_Text(txt_deuterolepta, 70, 5, 1);
                for (i=0; i<20; i++)          // όταν τελειώσει ο χρόνος χτυπάει το buzzer
                {
                  sound_play(880,50);
                  sound_play(1046,50);
                  sound_play(1318,50);
                }
                // τέλος χρονόμετρο

              } // τέλος κουμπί start

              if ((x_coord>= 83) && (x_coord<= 108) && (y_coord>= 52) &&
(y_coord<= 62))   // αν πατήσεις το κουμπί reset μηδενίζει το χρονόμετρο
              {
                goto Error;
              } // τέλος κουμπί reset

            }
          }
        }

    } // τέλος main
```

Σημείωση: Με το σύμβολο των δύο καθέτων (π.χ. //σχόλιο) σημαίνει ότι ακολουθεί σχόλιο. Το ίδιο ισχύει και για τα σχόλια που αρχίζουν με κάθετο και αστεράκι και τελειώνουν με αστεράκι και κάθετο (π.χ. /* σχόλιο*/).

7. ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Bert, van, Dam (2008). *Μικροελεγκτές PIC*.
2. Electronics Lab (χ.χ.). Οδηγός χρήσης PIC. Ανακτήθηκε στις 5 Αυγούστου 2013 από: http://www.electronics-lab.com/pic-in-greek/guide/Guide_to_use_the_PIC.htm
3. Microchip (χ.χ.). PIC16F887. Ανακτήθηκε στις 5 Αυγούστου 2013 από: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en026561
4. Microchip (χ.χ.). 28/40/44 – Pin Flash – Bashed, 8 – Bit CMOS Microcontrollers. Ανακτήθηκε στις 5 Αυγούστου 2013 από: http://ww1.microchip.com/downloads/en/DeviceDoc/41291G.pdf
5. WeBeing (χ.χ.). Μικροελεγκτές PIC. Ανακτήθηκε στις 6 Αυγούστου 2013 από: http://users.sch.gr/nichrist/Pages_GR/PIC_GR.htm
6. Βασιλειάδης, Β., & Κογιουμτζής, Π. (2009). *Ανάπτυξη συστήματος ελέγχου αφίξεων και αναχωρήσεων αστικών λεωφορειακών γραμμών* [πτυχιακή εργασία]. ΑΤΕΙ Θεσσαλονίκης, Τμήμα Ηλεκτρονικής, Θεσσαλονίκη.
7. Βικιπαιδεία (χ.χ.). Μικροελεγκτής. Ανακτήθηκε στις 6 Αυγούστου 2013 από: http://el.wikipedia.org/wiki/%CE%9C%CE%B9%CE%BA%CF%81%CE%BF%CE%B5%CE%BB%CE%B5%CE%B3%CE%BA%CF%84%CE%AE%CF%82
8. Βικιπαιδεία (χ.χ.). Μικροελεγκτής PIC. Ανακτήθηκε στις 7 Αυγούστου 2013 από: http://wikipedia.qwika.com/en2el/PIC_microcontroller
9. Βουρβουλάκης, Ι. (χ.χ.). Προγραμματισμός μικροελεγκτών PIC. Ανακτήθηκε στις 10 Αυγούστου 2013 από: http://www.eln.teilam.gr/sites/default/files/PIC_Tutorial.pdf
10. Καλόμοιρος, Ι. (2012). *Αρχές προγραμματισμού πραγματικού χρόνου. Εφαρμογές σε μικρά ενσωματωμένα συστήματα* [σημειώσεις σπουδαστών]. ΑΤΕΙ Σερρών, Τμήμα Πληροφορικής και Επικοινωνιών. Σέρρες.
11. Λάμπρου, Θ. (2004). *Σχεδίαση και κατασκευή συστήματος μέτρησης απόστασης με χρήση υπερήχων* [διπλωματική εργασία]. ΕΜΠ, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών. Αθήνα.
12. Λιλές, Α. & Σαμαρά, Α. (2012). *Ανίχνευση διέλευσης οχημάτων για την ρύθμιση φωτεινών σηματοδοτών* [πτυχιακή εργασία]. ΑΤΕΙ Λάρισας, Τμήμα Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών. Λάρισα.

13. Πάλλας, Β. (χ.χ.). *Προσημείωση ιατρικού θερμόμετρου με μικροελεγκτή* [πτυχιακή εργασία]. ΑΤΕΙ Λάρισας, Σχολή Τεχνολογικών Εφαρμογών. Λάρισα.
14. Πανελλήνιο Σχολικό Δίκτυο (χ.χ.). Αρχιτεκτονική και προγραμματισμός του μικροελεγκτή PIC. Ανακτήθηκε στις 6 Αυγούστου 2013 από: http://users.sch.gr/jabatzo/files/articles/PIC_theory.pdf
15. Σιόντα, Θ. (2012). *Μελέτη, ανάλυση, προγραμματισμός και εφαρμογή του μικροελεγκτή PIC στην βιομηχανία* [πτυχιακή εργασία]. ΑΤΕΙ Λάρισας, Τμήμα Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών. Λάρισα.
16. Τ.Ε.Ι. Στερεάς Ελλάδας Τμήμα Ηλεκτρονικών Μηχανικών Τ.Ε. (χ.χ.). Μικροελεγκτές. Ανακτήθηκε στις 6 Αυγούστου 2013 από: http://www.eln.teilam.gr/sites/default/files/Lesson03.pdf