ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ
ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΤΕ

Κατασκευή αριθμομηχανής με τη χρήση του Μικροελεγκτή PIC18F4550

Construction of calculator with the use of the PIC18F4550 microcontroller

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΊΑ

ΤΗΣ

Τεμενούγκα Ελένης

ΚΑΣ: 508803

ΚΩΔΙΚΟΣ ΠΤΥΧΙΑΚΗΣ : 14161Μ

Επιβλέπων: Καζακόπουλος Αριστοτέλης, Καθηγητής Α.Τ.Ε.Ι.

Ημερομηνία ανάληψης: 11/10/2014

Ημερομηνία περάτωσης: 10/02/2017

ΘΕΣΣΑΛΟΝΙΚΗ, ΦΕΒΡΟΥΑΡΙΟΣ 2017

ΠΕΡΙΛΗΨΗ

Η εργασία αυτή έχει ως στόχο τη δημιουργία κώδικα ώστε με την χρήση μέρους της πλακέτας του εργαστηρίου να κατασκευαστεί αριθμομηχανή που θα μπορεί να εκτελέσει τις τέσσερις βασικές πράξεις μεταξύ ακεραίων και δεκαδικών αριθμών χρησιμοποιώντας τον μικροελεγκτή PIC18F4550.

Ο κώδικας είναι γραμμένος σε γλώσσα C στο προγραμματιστικό περιβάλλον του προγράμματος MPLAB. Για την επίτευξη της εργασίας έγινε βοηθητική χρήση των προγραμμάτων PIC C Compiler και PDFSUSB. Για τον αρχικό προγραμματισμό του μικροελεγκτή χρησιμοποιήθηκε ο προγραμματιστής PIC IT 3.

Στην αναπτυξιακή πλακέτα υπάρχουν τμήματα της τα οποία δεν είναι απαραίτητα για την δημιουργία της αριθμομηχανής. Τα τμήματα που ήταν χρήσιμα για την περάτωση της εργασίας είναι για την είσοδο στοιχείων στη πλακέτα το πληκτρολόγιο 4x4,ο μικροελεγκτής για την επεξεργασία των στοιχείων και η οθόνη LCD ως συσκευή εξόδου του αποτελέσματος.

SUMMARY

This project aims creating a code to construct a calculator which can perform the four basic arithmetic operations between integers and decimal numbers using the PIC18F4550 microcontroller.

The code is written in C language firstly at the programming environment of proteus for simulation end then at the programming environment of MPLAB program. To achieve the project was auxiliary use PIC C Compiler programs and PDFSUSB. For the initial programming of microcontroller we use programmer PIC IT 3.

In development circuit board are parts of which are not necessary for the creation of the calculator. The sections were useful for the completion of the work is to input data on circuit board the 4x4 keypad, the microcontroller for processing the data and the LCD display as the output device of the result.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΑ

1. ΕΙΣΑΓΩΓΗ……..……..13

1.1- Τι είναι οι αριθμομηχανές……...…13
1.2- Στόχος και βήματα εργασίας…….….14

2. ΜΙΚΡΟΕΛΕΓΚΤΕΣ PIC

2.1- Γενικά για τους μικροελεγκτές……...……15
2.2- Διαφορετικοί τύποι μικροελεγκτών……16
2.3- Οικογένεια μικροελεγκτών PIC……..16
2.4- Μικροελεγκτής PIC18F455……...…17
2.4.1- Χαρακτηριστικά…….……...….18
2.4.2- Αρχιτεκτονική του PIC18F4550……..……..…19
2.4.3- Κεντρική Μονάδα Επεξεργασίας CPU…….……20
2.4.3.1- Μνήμη δεδομένων RAM……..…21
2.4.3.2- Μνήμη FLASH……22
2.4.3.3- Μνήμη EEPROM……..…23
2.4.4- Αριθμητική λογική μονάδα (ALU)……..….23
2.4.5- Διακόπτες (Interrupts)……..….23
2.4.6- Παράλληλες είσοδοι/έξοδοι (I/O)……..…24
2.4.7- Χρονιστές (Timers)……..….24

3. ΠΕΡΙΦΕΡΙΑΚΑ ΣΤΟΙΧΕΙΑ ΤΗΣ ΑΡΙΘΜΟΜΗΧΑΝΗΣ

3.1- Πληκτρολόγιο 4 x 4…….…...25
3.2- Οθόνη χαρακτήρων LCD……26
3.2.1- Χαρακτηριστικά …….……..….27
3.2.2- Γενικά για την οθόνη LCD……..……28
3.2.3- Ακροδέκτες οθόνης LCD…….….28
3.2.4- Είδη εντολών οθόνης και χρόνοι μεταφοράς δεδομένων……..…33
3.3- Άλλα στοιχεία του μικροελεγκτή……35
3.3.1- USB Port……...……35
3.3.2- Σειριακή πόρτα RS232……..36
3.3.3- Ολοκληρωμένο κύκλωμα MAX232……..…36
3.3.4- Ολοκληρωμένο κύκλωμα DS1302……37
3.3.5- Άλλα στοιχεία του κυκλώματος…….……..37

4. ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΠΕΡΙΦΕΡΕΙΑΚΩΝ ΣΤΟΙΧΕΙΩΝ ΠΛΑΚΕΤΑΣ

4.1- Συνδεσμολογία πληκτρολογίου 4x4……...……...38
4.2- Συνδεσμολογία οθόνης LCD……..……39
4.3- Συνδεσμολογία τροφοδοσίας ……...…..40
4.4- Συνδεσμολογία Reset……...…41

5. ΚΩΔΙΚΑΣ

5.1- Βιβλιοθήκες κώδικα……..…..42
5.1.1- Keypad.h……42
5.1.2- Flex_lcd…….….42
5.1.3- Main.h……...….42
5.2- Συναρτήσεις κώδικα……...……42
5.2.1- Συνάρτηση init()…….....42
5.2.2- Συνάρτηση lcd_init()…….…43
5.2.3- Συνάρτηση kbd_init()……..…..43
5.2.4- Συνάρτηση clearall…….……43

6. ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΑΡΑΤΗΡΗΣΕΙΣ

6.1- Προβλήματα …….…….……44

6.2- Βελτιώσεις…….……44

6.3- Συμπεράσματα…….……45

ΠΑΡΑΡΤΗΜΑ 1: Κυρίως πρόγραμμα……46

ΠΑΡΑΡΤΗΜΑ 2: ……...…….62

2.1- Περιγραφή του προγραμματιστικού περιβάλλοντος MPLAB…….62

2.2- Αναπτυξιακή πλακέτα PICkit 3……...……66

ΠΑΡΑΡΤΗΜΑ 3: Πλακέτα εργαστηρίου με τον PIC18F4550……..…….67

3.α- Στοιχεία του της πλακέτας…….……67
3.β- Τυπωμένο κύκλωμα πλακέτας……...……68
3.γ- Τελικό κύκλωμα πλακέτας…….……68

ΒΙΒΛΙΟΓΡΑΦΙΑ…… ……..…..69

ΕΙΚΟΝΕΣ

1.1 Άβακας

2.1 Μικροελεγκτές

2.2 Μικροελεγκτής PIC18F4550

2.3 Πίνακας Χαρακτηριστικών PIC18F4550

2.4 Block διάγραμμα του PIC18F4550

2.5 Αρχιτεκτονική Harvard

2.6 Μνήμη Δεδομένων RAM

2.7 Εγγραφή μνήμης FLASH

3.1 Keyboard 3x4 7pin

3.2 Keyboard 4x4 8pin

3.3 Διάγραμμα του πληκτρολογίου

3.4 Οθόνη LCD

3.5 Διάγραμμα οθόνης LCD

3.6 Ακροδέκτες οθόνης LCD

3.7 Πίνακας θέσεων, διευθυνσιοδότησης, αριστερής και δεξιάς ολίσθησης

3.8 Ενσωματωμένοι γραφικοί χαρακτήρες της οθόνης LCD

3.9 Λειτουργία 4 bit

3.10 Λειτουργία 8 bit

3.11 Πίνακας εντολών οθόνης

3.12 Χρόνοι μεταφοράς δεδομένων από τη μνήμη στην οθόνη

3.13 Χρόνοι μεταφοράς δεδομένων από την οθόνη στη μνήμη

3.14 Καλώδιο ακροδεκτών τύπου Α και Β

3.15 Θύρα τύπου Β

3.16 RS232

3.17 Συνδεσμολογία MAX232 με RS232

3.18 DS1302

3.19 Αντιστάσεις

3.20 LED

3.21 Πυκνωτές

3.22 Καλωδιοταινίες

3.23 Μπαταρία λιθίου

3.24 Διακόπτης 16pin

3.25 Κρύσταλλος

4.1 Keypad 4x4 input-output

4.2 Συνδεσμολογία Μικροελεγκτή με το πληκτρολόγιο και την οθόνη LCD

4.3 LCD pins

4.4 Συνδεσμολογία DS1302

4.5 Συνδεσμολογία Reset

ΣΥΝΤΟΜΟΓΡΑΦΊΕΣ

PIC – Programmable Intelligent Computer

LCD – Liquid Crystal Display

CPU – Central Processing Unit

ROM – Read Only Memory

RAM - Read Access Memory

CMOS – Complementary Metal Oxide Semiconductor

MIPS – Million Instructions per Second

EEPROM – Electrically Erasable Programmable read Only Memory

ALU – Arithmetical Logical Unit

DDRAM – Data Display Read Access Memory

DR – Data Register

IR – Instruction Register

RS – Register Select

R/W – Read / Write

BF – Busy Flag

AC – Address Counter

ΚΕΦΑΛΑΙΟ 1

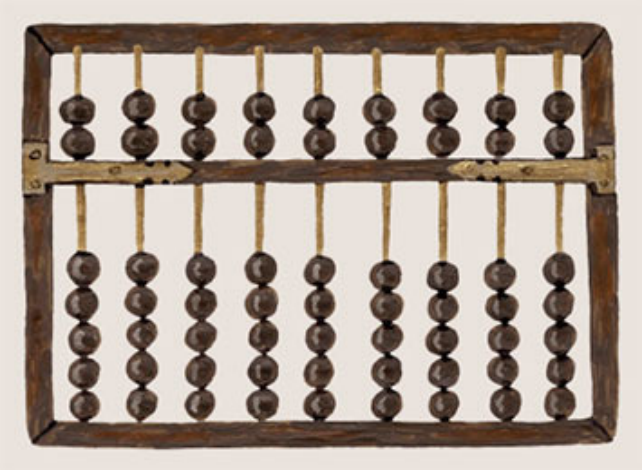
ΕΙΣΑΓΩΓΗ

6.3- Τι είναι οι αριθμομηχανές

Πριν ακόμα μπει τόσο βαθιά η τεχνολογία στη ζωή μας, πριν την ύπαρξη οποιασδήποτε μορφής ηλεκτρονικής συσκευής ο άνθρωπος έδειξε την ανάγκη του να υπολογίζει αριθμητικά μεγέθη μέσα από διάφορα ευρήματα που ηρθαν στο φως χρόνια ή ακόμα κ αιώνες αργότερα.

Ένα από τα πρώτα δείγματα αριθμομηχανής είναι ο γνωστός άβακας, το αρχαιότερο εύρημα που χρονολογείται γύρω στο 300 π.Χ. και χρησιμοποιείται ακόμα και σήμερα.

Άβακας



Εικόνα 1.1

Με τον ερχομό της τεχνολογίας στη ζωή μας και την εξέλιξη της σε βάθος χρόνου έκαναν την εμφάνιση τους πολλών ειδών αριθμομηχανές.

Η αριθμομηχανή γενικά είναι ένα εργαλείο το οποίο πραγματοποιεί αριθμητικές πράξεις. Έχει επικρατήσει περισσότερο στις μέρες μας να ονομάζεται κομπιουτεράκι αφού αποτελεί ένα κινητό, τσέπης υπολογιστή (computer), ιξού και η επικρατούμενη ονομασία του.

Υπάρχουν πολλών ειδών αριθμομηχανές που μπορούν να υπολογίσουν από τις απλές βασικές πράξεις (πρόσθεση, αφαίρεση, πολλαπλασιασμό, διαίρεση), μέχρι και σύνθετες μαθηματικές πράξεις.

Η πιο κοινή και ευρείας χρήσης είναι αυτή των τεσσάρων βασικών πράξεων της οποίας των κώδικα θα καταστρώσουμε και θα προγραμματίσουμε με την βοήθεια της πλακέτας του εργαστηρίου και του μικροελεγκτή PIC18F4550.

1.2 Στόχος και βήματα εργασίας

Στόχος της εργασίας είναι η κατανόηση και η αναλυτική περιγραφή του κώδικα που δημιουργήθηκε με σκοπό τη λειτουργία αριθμομηχανής η οποία πραγματοποιεί τις τέσσερις βασικές πράξεις με την βοήθεια της εργαστηριακής πλακέτας.

Για την επίτευξη του στόχου εκτός από τις μονάδες εισόδου και εξόδου σημαντική είναι η επιλογή της εργαστηριακής πλακέτας και κατά επέκταση του μικροελεγκτή λόγω του εύκολου προγραμματισμού του και του χαμηλού του κόστους σε σχέση με άλλες εφαρμογές, καθώς και η ήδη εξοικείωση με την πλακέτα. I

Στην εργασία δίνεται αναλυτικά ο η μέθοδος προγραμματισμού και λειτουργίας των περιφερειακών στοιχείων τις πλακέτας, δηλαδή του πληκτρολογίου και της οθόνης LCD, ο τρόπος λειτουργίας του μικροελεγκτή καθώς και μια γενική αναφορά στις χρήσεις της εργαστηριακής πλακέτας.

Στο κεφάλαιο 2 γίνεται γνωριμία γενικά με τους μικροελεγκτές, τους τύπους και τις ιδιότητες τους και δίνεται μια πιο αναλυτική περιγραφή στον μικροελεγκτή PIC18F4550 με τον οποίο πραγματοποιείται η εργασία.

Στο κεφάλαιο 3 γίνεται αναλυτική περιγραφή των περιφερειακών συστημάτων που απαρτίζουν συγκεκριμένα την αριθμομηχανή, αλλά και μια γενική αναφορά στα υπόλοιπα στοιχεία από τα οποία αποτελείται η πλακέτα.

Στο κεφάλαιο 4 γίνεται ανάλυση του κώδικα που γράφτηκε για να προγραμματιστεί η αριθμομηχανή.

Στο κεφάλαιο 5 βρίσκονται τα συμπεράσματα και οι παρατηρήσεις που προέκυψαν κατά την διάρκεια αλλά και στο τέλος της εργασίας.

ΚΕΦΑΛΑΙΟ 2

ΜΙΚΡΟΕΛΕΓΚΤΕΣ PIC

2.1 Γενικά για τους μικροελεγκτές

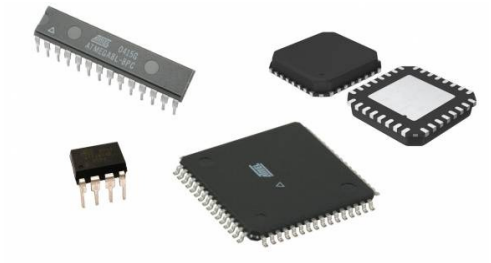
Παρατηρώντας γύρω μας μπορούμε να διαπιστώσουμε ότι η ζωή μας κινείται και εξαρτάται από κάθε είδους ηλεκτρονική συσκευή, από την πιο μικρή και απλή, μέχρι την πιο μεγάλη και σύνθετη. Όλες αυτές κρύβουν στο εσωτερικό τους μικροελεγκτές.

Ο μικροελεγκτής δεν είναι τίποτα άλλο από ένα εύκολα προγραμματιζόμενο chip που βελτιστοποιείται όσο εξελίσσεται η τεχνολογία. Είναι η βάση πολλών ηλεκτρονικών συστημάτων. Περίπου το 50% αυτών είναι απλοί ελεγκτές, ενώ το 20% περίπου είναι εξειδικευμένοι επεξεργαστές ψηφιακών σημάτων.

Μελετώντας εγχειρίδια τεχνικών δεδομένων που διατίθενται από διάφορες κατασκευαστικές εταιρίες, εύκολα μπορεί κανείς να διαπιστώσει ότι υπάρχουν αμέτρητοι τύποι μικροελεγκτών με πάρα πολλές διαφορετικές ιδιότητες και χαρακτηριστικά ανάλογα με την οικογένεια που ανήκει.

Αν συγκρίνουμε παρόλα αυτά τις δυνατότητες που προσφέρει κάθε σειρά μικροελεγκτών ανάλογα με τον κατασκευαστή του μπορούμε να διαπιστώσουμε ότι παρουσιάζουν πολλές παρόμοιες ή ακόμα και κοινές εφαρμογές και ιδιότητες. Η επιλογή του κατάλληλου μικροελεγκτή για την εφαρμογή που θέλουμε να πραγματοποιήσουμε γίνεται σύμφωνα με κάποια βασικά κριτήρια όπως το κόστος, την διαθεσιμότητα του στο εμπόριο, η υποστήριξη που προσφέρει και κυρίως την εξοικείωση μας ως προς αυτών και τις λειτουργίες που επιτελεί.

Μικροελεγκτές



Εικόνα 2.1

Όσον αφορά την αρχιτεκτονική τους οι περισσότεροι είναι βασισμένοι στην αρχιτεκτονική Von Neumann, που αποτελείται από τέσσερα βασικά δομικά στοιχεία τα οποία είναι μια κεντρική μονάδα επεξεργασίας (CPU), μια μνήμη για την αποθήκευση του προγράμματος (FLASH ή ROM), μια μνήμη για τις μεταβλητές

(RAM), χρονοδιακόπτες και μονάδες εισόδου/εξόδου για την επικοινωνία των περιφερειακών συστημάτων.

2.2 Διαφορετικοί τύποι μικροελεγκτών

Όπως αναφέραμε και παραπάνω οι τύποι των μικροελεγκτών ποικίλουν ανάλογα με τις δυνατότητες τους. Μερικά από τα χαρακτηριστικά τους που παίζουν ρόλο στην επιλογή του κατάλληλου για την εκάστοτε εφαρμογή που θέλουμε να δημιουργήσουμε είναι οι προγραμματιζόμενες ψηφιακές είσοδοι/έξοδοι παράλληλου τύπου ή αναλογικές,

2.3 Οικογένεια μικροελεγκτών PIC

Το όνομα τους το οφείλουν σε ένα ολοκληρωμένο κύκλωμα που παρήγαγε η General Instruments που ονομάστηκε PIC1650 και αναφέρθηκε ως Προγραμματιζόμενος Έξυπνος Υπολογιστής (**P**rogrammable **I**ntelligent **C**omputer).

Οι μικροελεγκτές PIC είναι από τους πιο διαδεδομένους αφού όταν καλείται ένας χρήστης να επιλέξει τον μικροελεγκτή που καλύπτει τις προδιαγραφές που απαιτεί η εφαρμογή που θέλει να πραγματοποιήσει αυτοί τις οικογένειας PIC καλύπτουν βασικές παραμέτρους όπως το ότι χρειάζεται απλές απαιτήσεις τροφοδοσίας, χρονισμού και συστήματος επανατοποθέτησης (reset), αναπτυξιακά εργαλεία λογισμικού χαμηλού κόστους, ικανοποιητικά μεγάλη βάση πρόσβασης από την πλευρά του χρήστη, ετοιμοπαράδοτα υλικά και ευκολία προγραμματισμού και χρήση ερασιτεχνικού εξοπλισμού από μη εξειδικευμένους χρήστες.

Οι οικογένειες μικροελεγκτών PIC μπορούν να διακριθούν σε πέντε κύριες οικογένειες :

1. PIC12CXXX / PIC12FXXX
2. PIC16C5X
3. PIC16CXXX / PIC16FXXX
4. PIC17CXXX
5. PIC18CXXX / PIC18FXXX

Στην πρώτη οικογένεια των PIC12CXX/PIC12FXXX ανήκουν μικροελεγκτές 8 ακίδων χαμηλού κόστους, οι εντολές τους απαρτίζονται από 12 ή 14 ψηφία, λειτουργούν με τροφοδοσία 2,5V, χρησιμοποιούν μνήμες FLASH, OTR ή ROM, μνήμες δεδομένων RAM 64 bytes και EEPROM 128 bytes.

Οι PIC16C5X έχουν εντολές μήκους 12 ψηφίων με αριθμό ακίδων 14, 18, 20 ή 28, λειτουργούν με 2V και χρησιμοποιούν μνήμη OTP.

Στην οικογένεια των PIC16CXXX/PIC16FXXX βρίσκουμε εντολές μήκους 14 ψηφίων με αριθμό ακίδων από 18 έως 68 και διαθέτουν δεκαψήφιους ή δωδεκαψήφιους μετατροπείς αναλογικού σε ψηφιακό.

Η τέταρτη οικογένεια των PIC17CXXX έχει μήκος εντολής 16 ψηφίων και αριθμό ακίδων από 40 έως 84.

Τέλος η οικογένεια των μικροελεγκτών PIC18CXXX/PIC18FXXX, όπου ανήκει και ο μικροελεγκτής που θα χρησιμοποιηθεί σε αυτή την εργασία, αποτελεί μια οικογένεια υψηλής απόδοσης CMOS με ενσωματωμένο μετατροπέα αναλογικού σε ψηφιακό με μήκος εντολής των 16 ψηφίων. Οι μικροελεγκτές αυτής της οικογένειας διαθέτουν σωρό 32 σορών, πολλών περισσότερων σε σχέση με άλλους μικροελεγκτές. Διαθέτουν πολλαπλές πηγές εσωτερικών και εξωτερικών διακοπτών και αριθμό εντολών που αγγίζουν τις 77. Η ταχύτητα εκτέλεσης των εντολών τους μπορεί να φτάσει τα 10 MIPS (million instructions per second) λόγο των επιπρόσθετων καταχωρητών τους.

2.4- Μικροελεγκτής PIC18F4550

Στην πλακέτα του εργαστηρίου και κατά επέκταση και στην εργασία γίνεται η χρήση του μικροεπεξεργαστή PIC18F4550 όπως αναφέρθηκε και παραπάνω. Παρόλο που ανήκει σε μια μεγάλη οικογένεια μικροελεγκτών και αυτός όπως και ο καθένας παρουσιάζει μεν ομοιότητες με άλλους αλλά έχει και δικά του μοναδικά χαρακτηριστικά.

Μικροελεγκτής PIC18F4550

Εικόνα 2.2

2.4.1 Χαρακτηριστικά

- Τάση τροφοδοσίας 2 – 5,5 V
- Ταχύτητα έως 48 MHz
- 75 εντολές προγραμματισμού που απαιτούν ένα κύκλο ρολογιού για την εκτέλεση τους
- Μνήμη
- Αυτοπρογραμματιζόμενη μνήμη FLASH 32 Kbytes x 16 words
- Data memory (RAM) έως 2048 x 8 bytes
- EEPROM Data memory έως 256 x 8 bytes
- Πλήθος περιφερειακών
- 5 πόρτες εισόδου / εξόδου
- 4 Χρονιστές (timers) 8/16–bit
- 13 κανάλια 10- bit μετατροπέα από αναλογικό σε ψηφιακό (analog to digital converter)
- Λειτουργία SPI και I²C Master
- Ειδικά χαρακτηριστικά
- Επιτρέπει τον ασφαλή τερματισμό λειτουργίας εάν σταματήσει ο παλμός
- Σειριακή θύρα με δυνατότητες σύγχρονης και ασύγχρονης λειτουργίας (USART)
- 20 πηγές διακοπτών
- Αυτόματο τερματισμό και επανεκκίνηση
- Συνδεσιμότητα με θήρα USB 2.0
- 40 ακροδέκτες

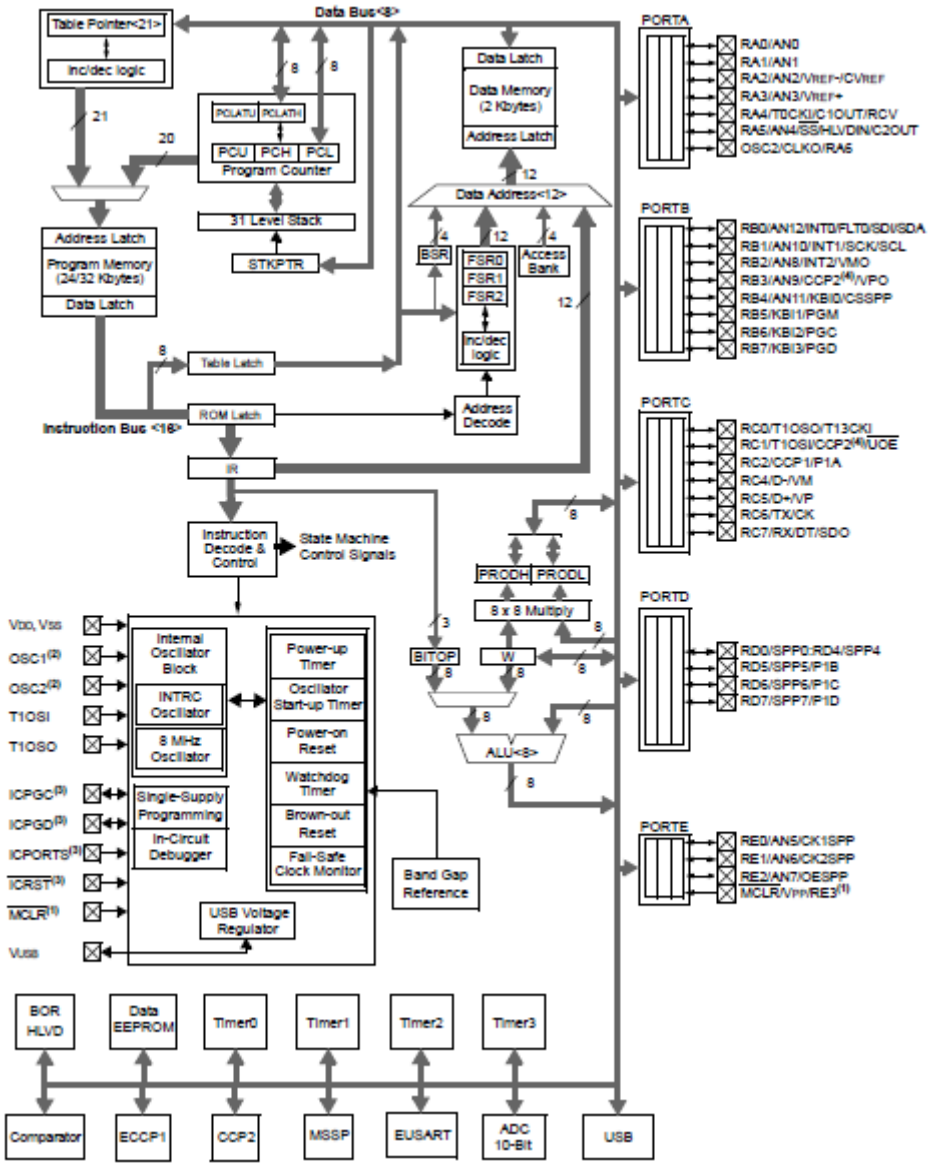
Πίνακας Χαρακτηριστικών PIC18F4550

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-bit
	FLASH (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI	Master I²C			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

Εικόνα 2.3

Όπως φαίνεται και από τα χαρακτηριστικά του ο μικροελεγκτής PIC18F4550 δίνει πολλές αναπτυξιακές δυνατότητες και σε συνδυασμό με το χαμηλό κόστος και την αποτελεσματικότητα του μπορεί να ανταποκριθεί άκρως ικανοποιητικά σε πολλές εφαρμογές.

Block διάγραμμα του PIC18F4550



Εικόνα 2.4

2.4.2 Αρχιτεκτονική του PIC18F4550

Ο μικροελεγκτής PIC χρησιμοποιεί αρχιτεκτονική Harvard. Σύμφωνα με την αρχιτεκτονική Harvard η μνήμη προγράμματος και η μνήμη δεδομένων είναι

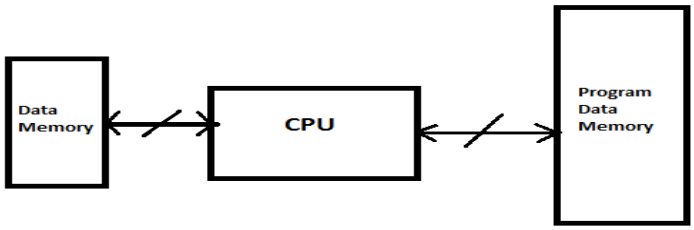
χωρισμένες και προσπελαύνονται από διαφορετικούς διαύλους. Ουσιαστικά διαχωρίζεται το τμήμα των μεταβλητών και των καταχωρητών εισόδων / εξόδων ή αλλιώς της μνήμης από την μνήμη προγραμματισμού.

Για την εκτέλεση μιας εντολής σύμφωνα με την αρχιτεκτονική Harvard ο κύκλος εντολής πραγματοποιείται σε έναν κύκλο εντολής με 12, 14 ή 16 ψηφία ανάλογα με την οικογένεια του μικροελεγκτή που περατώνει την εφαρμογή.

Έτσι δίνεται η δυνατότητα να μπορούν να γράφονται και να διαβάζονται εντολές ταυτόχρονα αφού είναι ουσιαστικά σαν να λειτουργούν δυο ξεχωριστές μνήμες με μεγάλη ταχύτητα και χαμηλό κόστος. Η ταυτόχρονη προσπέλαση εντολών είναι αυτή που κάνει τον επεξεργαστή να εκτελεί τις εντολές με μεγάλη ταχύτητα φτάνοντας ακόμα σε απόδοση κοντά στο 1MIPS ανά MHz ρολογιού.

Μπορούμε να θεωρήσουμε ότι η κεντρική μονάδα επεξεργασίας του μικροελεγκτή λειτουργεί ως μια αριθμητική λογική μονάδα (ALU).

Αρχιτεκτονική Harvard



Εικόνα 2.5

2.4.3 Κεντρική Μονάδα Επεξεργασίας CPU

Η CPU είναι το κεντρικό εξάρτημα που επεξεργάζεται δεδομένα σε έναν ηλεκτρονικό σύστημα, ελέγχει τη λειτουργία του και εκτελεί βασικές λειτουργίες διασύνδεσης και μεταβίβασης εντολών.

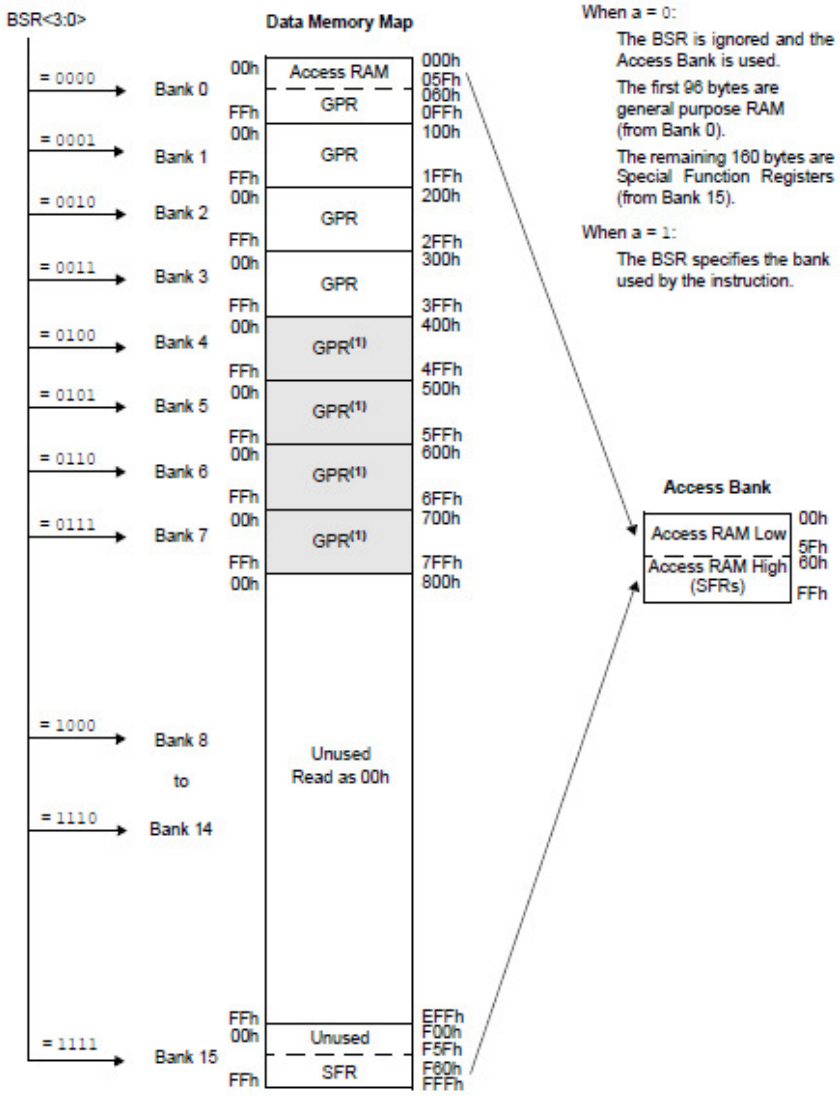
Όπως αναφέραμε παραπάνω σύμφωνα με την αρχιτεκτονική Harvard τα σημαντικά τμήματα μνήμης είναι χωρισμένα οπότε η CPU είναι απευθείας συνδεδεμένη με αυτά. Οι μνήμες με τις οποίες είναι ξεχωριστά συνδεδεμένη η CPU είναι η μνήμη δεδομένων RAM, η μνήμη προγράμματος FLASH και η μνήμη EEPROM. Θα γίνει παρακάτω αναφορά σε καθεμία από αυτές σχετικά με τις λειτουργίες και την χρήση τους.

2.4.3.1 Μνήμη δεδομένων RAM

Η μνήμη τυχαίας προσπέλασης (**R**andom **A**ccess **M**emory) δεν έχει πάρει τυχαία το όνομα της αλλά από τον τρόπο με τον οποίο γίνεται η προσπέλαση των εντολών. Η RAM χρησιμοποιείται για την προσωρινή αποθήκευση εντολών και προγραμμάτων. Γενικά η μνήμη RAM χωρίζεται σε τράπεζες όπου υπάρχουν δύο ειδών καταχωρητές, οι καταχωρητές γενικής χρήσεως και οι ειδικοί καταχωρητές.

Στο χαμηλότερο τμήμα διευθύνσεων κάθε τράπεζας βρίσκονται οι καταχωρητές οι ειδικοί καταχωρητές οι οποίοι χρησιμοποιούνται για την επικοινωνία συσκευών εισόδου / εξόδου ή άλλων περιφερειακών. Στα επόμενα τμήματα υπάρχουν καταχωρητές γενικής χρήσεως οι οποίοι χρησιμοποιούνται για την αποθήκευση προσωρινών μεταβλητών και δεδομένων όσο εκτελείται το πρόγραμμα.

Μνήμη Δεδομένων RAM



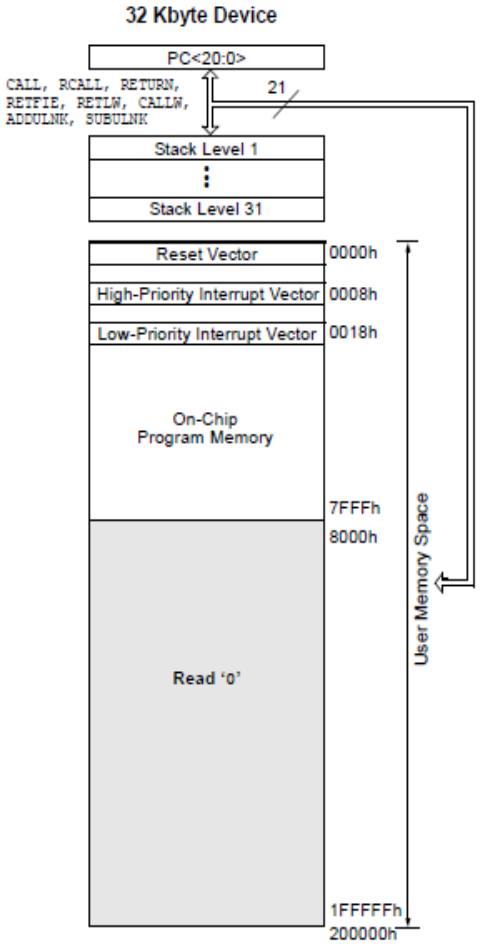
Εικόνα 2.6

2.4.3.2 Μνήμη FLASH

Η μνήμη FLASH είναι μια ακόμα μνήμη που είναι απαραίτητη για την λειτουργία του μικροεπεξεργαστή. Έχει ταχύτητα εγγραφής/διαγραφής 32 Kbytes, διατηρεί τα δεδομένα της και μετά την παύση της τροφοδοσίας και έχει διάρκεια ζωής έως 10.000 κύκλους εγγραφής/διαγραφείς. Διαβάζεται/εγγράφεται από το πρόγραμμα του χρήστη. Η μνήμη προγράμματος FLASH διαβάζει μία λέξη κάθε φορά και γράφεται με μπλοκ των τεσσάρων λέξεων κάθε φορά, δηλαδή το ελάχιστο μπλοκ που μπορεί να γραφτεί είναι τέσσερις λέξεις διαδοχικών διευθύνσεων.

Κατά το σβήσιμο το ελάχιστο μπλοκ αποτελείται από 32 λέξεις ή 64 bytes εκτός αν χρησιμοποιηθεί εξωτερικός προγραμματιστής που μπορεί να σβήσει με μιας ολόκληρη τη μνήμη.

Εγγραφή μνήμης FLASH



Εικόνα 2.7

2.4.3.3 Μνήμη EEPROM

Στους περισσότερους μικροελεγκτές, όπως και στον PIC18F4550, υπάρχει μια ενσωματωμένη μνήμη EEPROM η οποία γράφεται και διαβάζεται χρησιμοποιώντας καταχωρητές. Η EEPROM γράφεται και διαβάζεται με ένα byte κάθε φορά. Η περιοχή διευθύνσεων της είναι 0h – FFh. Το μέγεθος της είναι μικρό, 256 Kbytes, αλλά έχει διάρκεια ζωής έως και 1.000.000 κύκλους εγγραφής/διαγραφής.

Τα δεδομένα διατηρούνται όπως και στην μνήμη FLASH σε περίπτωση διακοπής της τροφοδοσίας που είναι και η κύρια χρήση αυτής της μνήμης.

Μέγεθος [byte]	**Περιοχή διευθύνσεων**
64	0h – 3Fh
128	0h – 7Fh
256	0h – FFh

2.4.4 Αριθμητική λογική μονάδα (ALU)

Η αριθμητική λογική μονάδα επεξεργασίας (**A**rithmetical **L**ogical **U**nit), είναι υπεύθυνη για όλες τις αριθμητικές, λογικές και πράξεις σύγκρισης που εκτελούνται με τις εντολές του μικροεπεξεργαστή. Η ALU είναι απευθείας συνδεδεμένη με τους καταχωρητές γενικής χρήσης. Μέσα σε ένα κύκλο ρολογιού εκτελούνται οι αριθμητικές πράξεις μεταξύ δύο καταχωρητών ή ενός καταχωρητή και ενός ορίσματος.

Στους πιο απλούς μικροεπεξεργαστές εκτελούνται μόνο πράξεις πρόσθεσης και αφαίρεσης, άλλοι πιο σύνθετοι μπορούν να εκτελέσουν πολλαπλασιασμό και διαίρεση μέσα από μια ακολουθία πολλαπλών προσθέσεων και αφαιρέσεων.

Οι διευθύνσεις προσδιορίζονται με σαφή τρόπο μέσα στις εντολές στον κώδικα εντολών όπου δεσμεύεται χώρος μήκους 7 bits για τις διευθύνσεις, πράγμα που σημαίνει ότι ο PIC μπορεί να πραγματοποιήσει προσπέλαση μέχρι και 128 διευθύνσεων.

2.4.5 Διακόπτες (Interrupts)

Οι διακόπτες είναι μία από τις λειτουργίες των PIC που χρησιμοποιούνται λιγότερο λόγω του ότι θεωρούνται δύσκολοι ως προς την κατανόηση και την εφαρμογή τους. Αυτό όμως είναι μύθος αφού είναι ένα εργαλείο που με την σωστή χρήση του μπορεί να διευκολύνει κατά πολύ την ανάπτυξη των εφαρμογών, καθώς καθιστά και πιο εύκολη την αποτελμάτωση του προγράμματος.

Είναι ουσιαστικά κλήσεις συναρτήσεων από το hardware οι οποίες διακόπτουν την ροή του κυρίως προγράμματος και το βάζουν να συνεχίσει σε μια ρουτίνα διακοπής.

Ένα κοινό όλων των μικροελεγκτών PIC είναι ότι ο καταχωρητής INTCON είναι η κύρια πηγή ελέγχου των διακοπτών όπου το σημαντικότερο bit αυτού του καταχωρητή χρησιμοποιείται για να επιτρέπει την οποιαδήποτε διακοπή. Τα επόμενα bits του καταχωρητή είναι σημαίες διακοπτών του συστήματος.

2.4.6 Παράλληλες είσοδοι – έξοδοι (I/O)

Οι παράλληλες θύρες εισόδου / εξόδου που χρησιμοποιούνται στο PIC είναι πολύ απλοί ως προς την κατανόηση και την χρήση τους. Αποτελούνται από μια θύρα ανάγνωσης / εγγραφής και δυο καταχωρητές, τον καταχωρητή ελέγχου TRIS και τον καταχωρητή δεδομένων της θύρας PORT.

Τόσο ο καταχωρητής TRIS όσο και ο PORT είναι τυπικοί καταχωρητές μέσα στο χώρο διευθύνσεων των καταχωρητών. Ο καταχωρητής PORT χρησιμοποιείται για την διευθυνσιοδότηση της μνήμης και ο καταχωρητής TRIS δηλώνει ποια bit της πόρτας είναι είσοδοι και ποια έξοδοι.

Η εγγραφή δεδομένων στη θύρα εξόδου μπορεί να γίνει οποιαδήποτε στιγμή, αλλά κανένα bit δεν εμφανίζεται στην έξοδο αν δεν έχει μηδενιστεί το αντίστοιχο bit ελέγχου του καταχωρητή TRIS.

2.4.7 Χρονιστές (Timers)

Ένα από τα πιο σημαντικά και χρήσιμα χαρακτηριστικά των μικροελεγκτών PIC είναι οι χρονιστές. Στον PIC18F4550 γίνεται χρήση τεσσάρων χρονιστών (TIMER0, TIMER1, TIMER2, TIMER3).

Ο λόγος που κάνει τους χρονιστές τόσο σημαντικούς είναι ότι μπορούν να χρησιμοποιηθούν για πολλών ειδών σκοπούς. Με την βοήθεια τους μπορούν να καθορίσουν το εύρος των παλμών, τις χρονικές περιόδους, καθώς και την ταχύτητα και την συχνότητα διαφόρων σημάτων.

ΚΕΦΑΛΑΙΟ 3

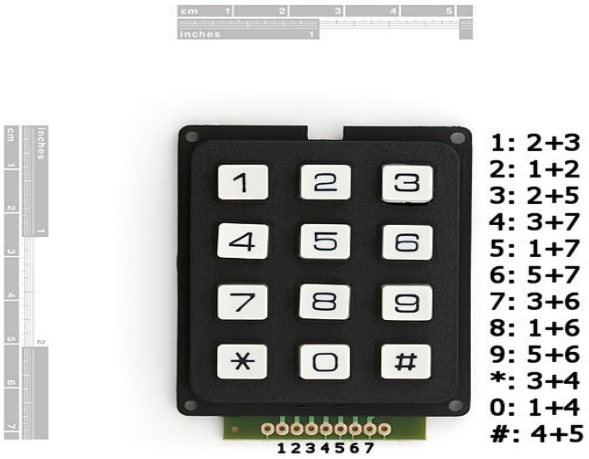
ΠΕΡΙΦΕΡΕΙΑΚΑ ΣΤΟΙΧΕΙΑ ΤΗΣ ΑΡΙΘΜΟΜΗΧΑΝΗΣ

3.1 Πληκτρολόγιο 4 x 4

Στην αγορά υπάρχουν πολλών ειδών και τύπων πληκτρολόγια η επιλογή των οποίων γίνεται ανάλογα με την εφαρμογή και την χρήση που το θέλουμε και που στην προκειμένη εργασία είναι πολύ εύκολο μέσω αυτού να περαστούν δεδομένα στον μικροελεγκτή. Γενικά κατασκευάζονται με όσο τον δυνατόν λιγότερους ακροδέκτες εισόδου/εξόδου και τα περισσότερα αποτελούνται από σειρές και στήλες.

Τα δύο πιο ευρέως χρησιμοποιήσιμα πληκτρολόγια είναι αυτά με 12 πλήκτρα και διαμόρφωση στηλών και γραμμών 3x4 με 7 ακροδέκτες και αυτά με 16 πλήκτρα και διαμόρφωση στηλών και γραμμών 4x4 με 8 ακροδέκτες.

Keyboard 3x4 7pin



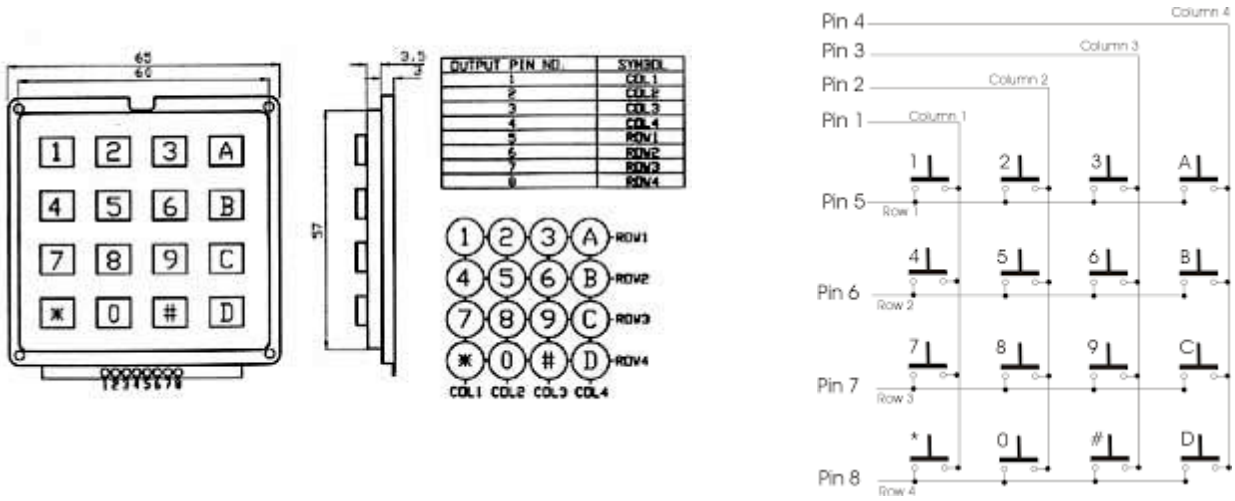
Εικόνα 3.1

Keyboard 4x4 8pin

Εικόνα 3.2

Για την εργασία αυτή χρησιμοποιήθηκε το πληκτρολόγιο keyboard 4x4 16keys matrix output της εταιρίας velleman το οποίο με μια πρώτη ματιά φαίνεται ότι ανήκει στη δεύτερη κατηγορία. Ο τρόπος με τον οποίο λειτουργεί το πληκτρολόγιο θυμίζει σκακιέρα καθώς πατώντας ένα πλήκτρο συνδέονται 2 καλώδια που αντιστοιχούν σε μια γραμμή και μια στήλη δίνοντας ένα μοναδικό συνδυασμό στηλών και γραμμών κάθε φορά.

Διαγράμματα του πληκτρολογίου



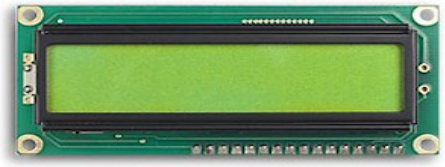
Εικόνα 3.3

Η τροφοδοσία που απαιτείται είναι στα 5V μέσω των ακροδεκτών του. Για τον εντοπισμό του πλήκτρου που πατήθηκε γίνεται σάρωση κάθε γραμμής και στήλης. Αρχικά μετατρέπεται η Σειρά 1 σε λογικό 0 και ελέγχει αν κάποια από τις στήλες έγινε 0, δηλαδή αν πατήθηκε κάποιο κουμπί από την Σειρά 1. Με τον ίδιο ακριβώς τρόπο κάνει σάρωση και στις επόμενες Σειρές 2,3 και 4. Σε περίπτωση που έχει πατηθεί κάποιο άλλο πλήκτρο επιστρέφεται το ανάλογο αποτέλεσμα.

3.2 Οθόνη χαρακτήρων LCD

Στο εμπόριο γενικά μπορούμε να βρούμε πολλών ειδών οθόνες. Για την κατασκευή της αριθμομηχανής στη συγκεκριμένη περίπτωση έγινε χρήση της οθόνης LCD 1602. Όπως φαίνεται και κωδικοποιημένα από το όνομα της είναι οθόνη 2x16, δηλαδή 2 σειρών και 16 χαρακτήρων.

Οθόνη LCD

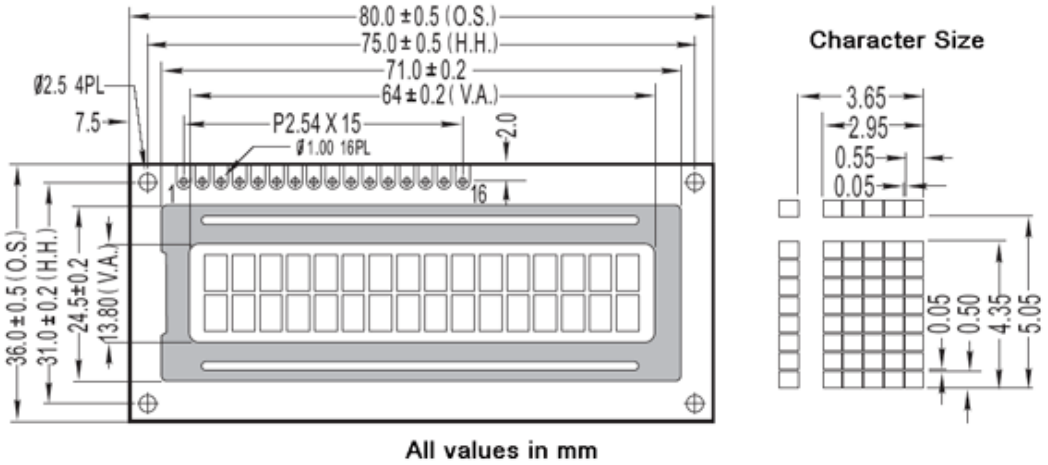


Εικόνα 3.4

3.2.1 Χαρακτηριστικά

- Τροφοδοσία 5V
- V_{LCD} ρυθμιζόμενη για καλύτερη αντίθεση 5V (Vop)
- Θερμοκρασία λειτουργίας -10°C με +60°C
- Θερμοκρασία αποθήκευσης -20°C με +70°C
- Κατασκευή COB (Chip On Board)
- Μορφή εμφάνισης 16x2 χαρακτήρες
- Τύπος οθόνης STN Y-G , ανακλαστική
- Ελεγκτής SPLC780D1 ή ισοδύναμος ελεγκτής
- Διασύνδεση 8 bit παράλληλου τύπου
- Blacklight κίτρινο-πράσινο bottom lights
- Οπτική κατεύθυνση 6 O'clock
- Πρόγραμμα οδήγησης 1/16 duty cycle, 1/5 Bias
- Εσωτερική DDRAM με 80 αποθηκευμένους χαρακτήρες
- Μνήμη ROM για την δημιουργία χαρακτήρων στην οθόνη
- Επιπρόσθετη μνήμη RAM για την δημιουργία έως 8 bit διαφορετικών χαρακτήρων 5x7 που καθορίζονται από τον χρήστη
- Απεικόνιση των χαρακτήρων 5x7 με 160 διαφορετικά γραφικά
- Δυνατότητα λειτουργίας 4 bit ή 8 bit
- Εσωτερικό ρολόι
- Εσωτερικό κύκλωμα αρχικοποίησης μετά από κάθε τροφοδότηση
- Δυνατότητα εκτέλεσης πολλών εντολών (π.χ. καθαρισμός οθόνης)

Διάγραμμα οθόνης LCD



Εικόνα 3.5

3.2.2 Γενικά για την οθόνη LCD

Ο μικροελεγκτής στέλνει στην οθόνη 8 bit κωδικούς και από αυτήν αποστέλλεται ο κωδικός στη μνήμη απεικόνισης δεδομένων DDRAM (Data Display Ram). Στη συνέχεια η DDRAM πραγματοποιεί μετατροπή του κώδικα στον αντίστοιχο χαρακτήρα που αναγνωρίζει η οθόνη ώστε να τον εμφανίσει στον πίνακα κουκίδων 5x7. Εκτός από τους ήδη υπάρχοντες χαρακτήρες υπάρχει η δυνατότητα δημιουργίας 8 νέων χαρακτήρων από τον χρήστη οι οποίοι αποθηκεύονται στη μνήμη RAM της οθόνης με χωρητικότητα 64 bytes. Η οθόνη διαθέτει επίσης επιπλέον μνήμη ROM 160 bytes στην οποία περιέχονται γραφικές αναπαραστάσεις των χαρακτήρων.

Με την σωστή σειριακή αποστολή των έγκυρων κωδικών από τον μικροελεγκτή στην οθόνη μπορεί να επιτευχθεί η σωστή απεικόνιση ενός χαρακτήρα σε συγκεκριμένη θέση. Η οθόνη μπορεί να εμφανίζει μηνύματα από αριστερά προς τα δεξιά ή και το αντίστροφο, καθώς και να αυξάνει ή να μειώνει, ανάλογα με τον κώδικα που δέχεται, τη θέση του κέρσορα. Ο μικροελεγκτής θέτει την επόμενη θέση με την κατάλληλη εντολή προς την οθόνη και στη συνέχεια στέλνει τον χαρακτήρα ο οποίος απεικονίζεται στην καθορισμένη θέση.

Όπως αναφέραμε ο μικροελεγκτής βρίσκεται σε άμεση επαφή με την οθόνη. Για την μεταξύ τους επικοινωνία ο μικροελεγκτής χρησιμοποιεί έναν καταχωρητή δεδομένων DR (Data Register) και έναν καταχωρητή εντολών IR (Instruction Register) μεγέθους 8 bit.

Στον καταχωρητή δεδομένων γίνεται προσωρινή καταχώρηση δεδομένων μεταξύ του μικροελεγκτή και της οθόνης όπου στη συνέχεια μεταφέρονται σε άλλη μνήμη για να γίνει η αποκωδικοποίηση τους και να μεταφερθούν στην οθόνη. Ο καταχωρητής εντολών από την άλλη χρησιμοποιείται μόνο για την εγγραφή εντολών από τον μικροελεγκτή και είναι αυτός στον οποίο περιέχονται οι κωδικοί των εντολών και των πληροφοριών που αφορούν τις μνήμες της οθόνης.

3.2.3 Ακροδέκτες οθόνης LCD

Οι οθόνες χαρακτήρων LCD του εμπορίου είναι τυποποιημένες και ακολουθούν την αρχιτεκτονική του μικροϋπολογιστή HDD44780U. Σύμφωνα με την τυποποίηση αυτή υπάρχουν τρεις γραμμές ελέγχου E, RS, R/W και οκτώ γραμμές δεδομένων που είναι DB0, DB1, DB2, DB3, DB4, DB5, DB6, DB7.

Ακροδέκτες οθόνης LCD

Pin No.	Symbol	Level	Description
1	VSS	0V	Ground.
2	VDD	+5.0V	Power supply for logic operating.
3	V0	--	Adjusting supply voltage for LCD driving.
4	RS	H/L	A signal for selecting registers: 1: Data Register (for read and write) 0: Instruction Register (for write), Busy flag-Address Counter (for read).
5	R/W	H/L	R/W = "H": Read mode. R/W = "L": Write mode.
6	E	H/L	An enable signal for writing or reading data.
7	DB0	H/L	This is an 8-bit bi-directional data bus.
8	DB1	H/L	
9	DB2	H/L	
10	DB3	H/L	
11	DB4	H/L	
12	DB5	H/L	
13	DB6	H/L	
14	DB7	H/L	
15	LED+	+5.0V	Power supply for backlight.
16	LED-	0V	The backlight ground.

Εικόνα 3.6

Η γραμμή E (Enable) για την ενεργοποίηση της οθόνης LCD ώστε να της σταλούν δεδομένα. Θα πρέπει το πρόγραμμα του χρήστη πρώτα να τοποθετήσει αυτή τη γραμμή, μετά τις άλλες δύο και μετά τα δεδομένα στο διάδρομο δεδομένων DB0 – DB7.

Με την γραμμή RS () σε κατάσταση 1 (high), τα δεδομένα αποτελούν κείμενο προς εμφάνιση, ενώ όταν είναι σε κατάσταση 0 (low), ειδοποιείται η LCD να χειριστεί τα δεδομένα σαν ειδική εντολή (π.χ. καθαρισμός οθόνης, τοποθέτηση του κέρσορα).

Όταν η γραμμή R/W είναι σε κατάσταση 1 (high), διαβάζεται η οθόνη LCD. Υπάρχει μόνο μια εντολή ανάγνωσης ενώ οι υπόλοιπες είναι εντολές εγγραφείς για αυτό και σχεδόν πάντα βρίσκεται σε κατάσταση 0 (low). Όταν βρίσκεται σε κατάσταση 0 (low), γράφονται τα δεδομένα του διαδρόμου δεδομένων DB0 – DB7 στην οθόνη LCD. Ουσιαστικά η οθόνη LCD καλείται να λάβει δεδομένα όταν υπάρχει αλλαγή κατάστασης από 1 σε 0.

Αφού δοθεί η εντολή για την σωστή τοποθέτηση των γραμμών ελέγχου RS, R/W αλλά και των γραμμών δεδομένων DB0…DB7, πρέπει η γραμμή E να αλλάξει σε κατάσταση 0. Εάν δεν γίνει αυτό δεν εκτελείται η εντολή στην οθόνη. Ο χρόνος εκτέλεσης της εντολής καθορίζεται από τον κρύσταλλο. Η εντολή έχει περατωθεί όταν το πιο σημαντικό ψηφίο DB7 της οθόνης είναι σε κατάσταση 1, το οποίο ονομάζεται και σημαία ένδειξης απασχολημένου ψηφίου BF (Busy Flag). Όταν αυτή μηδενιστεί τότε είναι έτοιμη να δεχτεί νέα εντολή.

Πίνακας θέσεων, διευθυνσιοδότησης, αριστερής και δεξιάς ολίσθησης

Display	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Position DDRAM Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
For Shift Left	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50
For Shift Right	27	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E
	67	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E

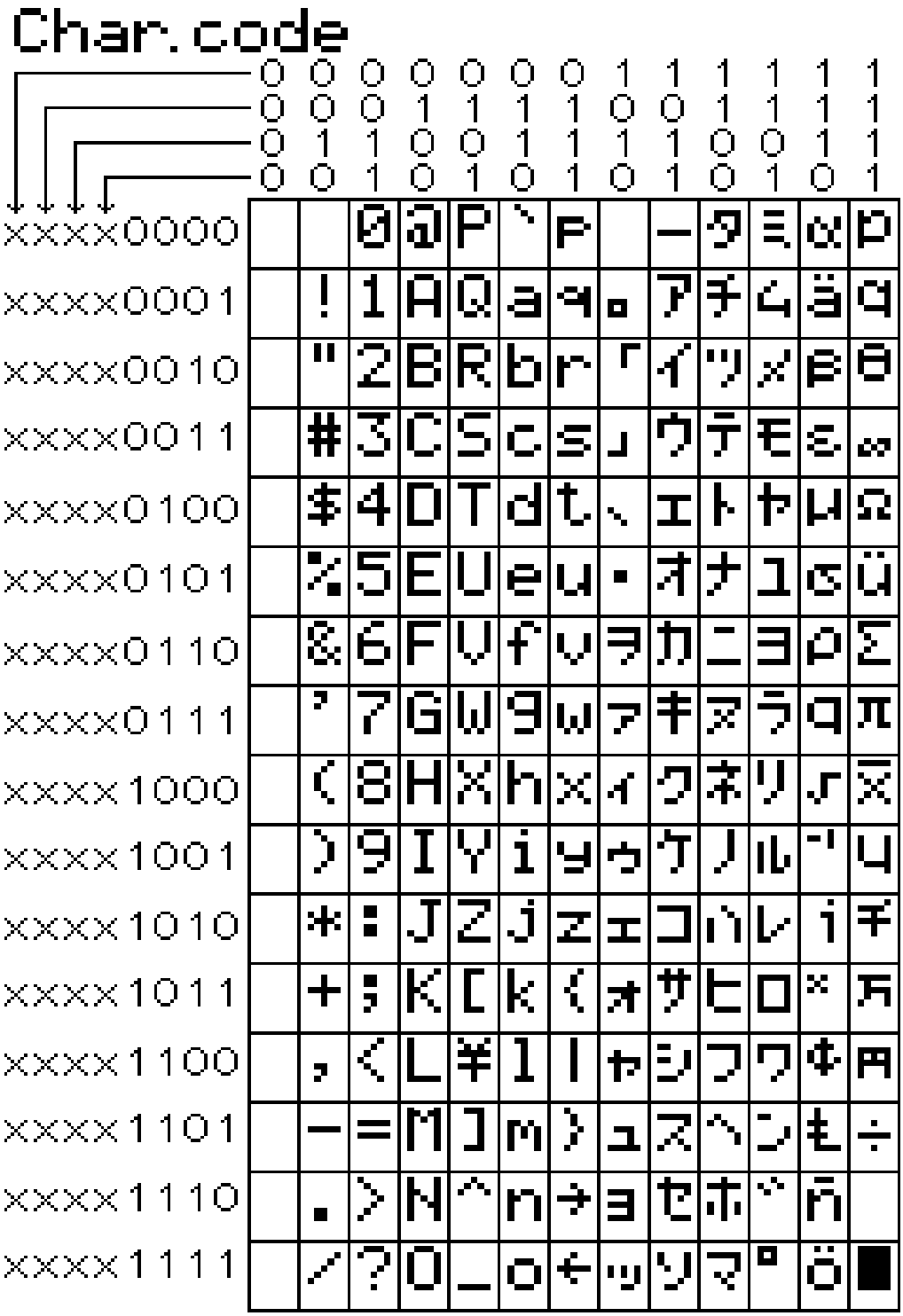
Εικόνα 3.7

Όπως αναφέραμε και παραπάνω η οθόνη επικοινωνεί και αποθηκεύει διευθύνσεις σε δύο μνήμες την DDRAM και την RAM. Η οθόνη με την βοήθεια ενός μετρητή διευθύνσεων AC (Address Counter) δημιουργεί τις διευθύνσεις που περιέχουν τους χαρακτήρες στις δύο μνήμες. Σε κάθε διεύθυνση της μνήμης αντιστοιχεί και μια θέση στην οθόνη. Κάθε χαρακτήρας γράφεται αυτόματα σε διαδοχικές θέσεις. Αυτόματα προσαυξάνεται ο εσωτερικός απαριθμητής κατά 1 έτσι ώστε να περιέχει την επόμενη θέση εγγραφής στην οθόνη όπως φαίνεται και στην εικόνα 3.5.

Όπως είδαμε και από τα χαρακτηριστικά της οθόνης η DDRAM έχει 80 αποθηκευμένους χαρακτήρες και η επιπρόσθετη μνήμη RAM μπορεί να δημιουργήσει 8 bit διαφορετικών χαρακτήρων. Βλέποντας τις διευθύνσεις στην παραπάνω εικόνα από την 1 θέση μέχρι και την 8 εμφανίζονται οι χαρακτήρες της μνήμης της οθόνης ενώ από την 9 μέχρι την 16 εμφανίζονται οι χαρακτήρες της επιπρόσθετης μνήμης. Στην εικόνα 3.5 μπορούμε να διακρίνουμε επίσης την αριστερή και δεξιά ολίσθηση που συμβαίνει όταν σε κάθε γραμμή υπάρχουν λιγότερες από 40 θέσεις.

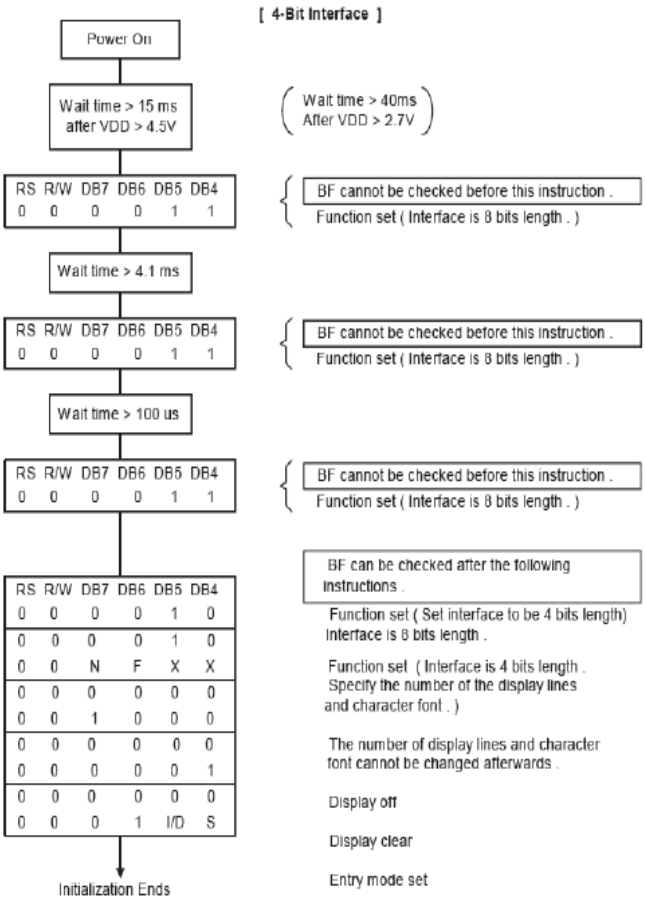
Τέλος οθόνη μπορεί να τεθεί είτε σε λειτουργία 4 bit, είτε σε λειτουργία 8 bit αφού η αρχιτεκτονική της ακολουθεί αυτή του μικροελεγκτή HD44780. Με την λειτουργία των 8 bit μεταφέρονται πιο γρήγορα τα δεδομένα αφού μεταφέρονται ταυτόχρονα και στις 8 γραμμές δεδομένων (DB0…DB7) , ενώ στην λειτουργία των 4 bit η μεταφορά των δεδομένων γίνεται σε δύο χρόνους με τις μισές γραμμές δεδομένων (DB4...DB7),, ενώ οι υπόλοιπες οδηγούνται απευθείας στη γείωση. Η αρχικοποίηση μπορεί να γίνει είτε μετά την τροφοδοσία, είτε με την βοήθεια εντολών του εσωτερικού κυκλώματος.

Ενσωματωμένοι γραφικοί χαρακτήρες της οθόνης LCD



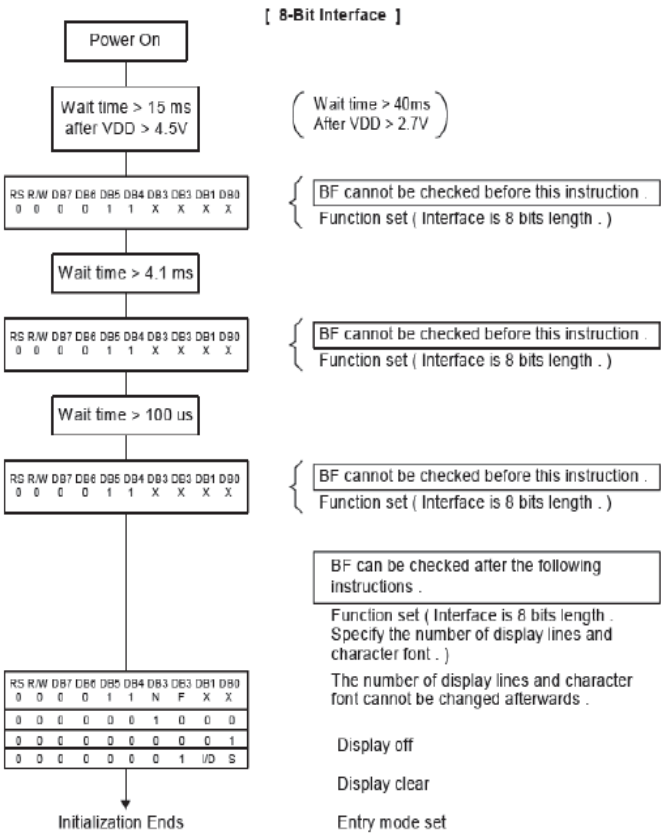
Εικόνα 3.8

Λειτουργία 4 bit



Εικονα 3.9

Λειτουργία 8 bit



Εικόνα 3.10

3.2.4 Είδη εντολών οθόνης και χρόνοι μεταφοράς δεδομένων

Αν και οι εντολές που οδηγούν την οθόνη ποικίλουν μπορούμε να τις τμηματοποιήσουμε χωρίζοντας τες σε 3 διαφορετικά λειτουργικά είδη. Είναι οι εντολές οι οποίες καθορίζουν την συμπεριφορά της οθόνης, οι εντολές που αφορούν την διευθυνσιοδότηση της μνήμης και οι εντολές που αναλαμβάνουν την μεταφορά των δεδομένων στις διευθύνσεις της RAM. Επίσης υπάρχει και μια άλλη κατηγορία η οποία περιλαμβάνει τις εντολές γενικού τύπου.

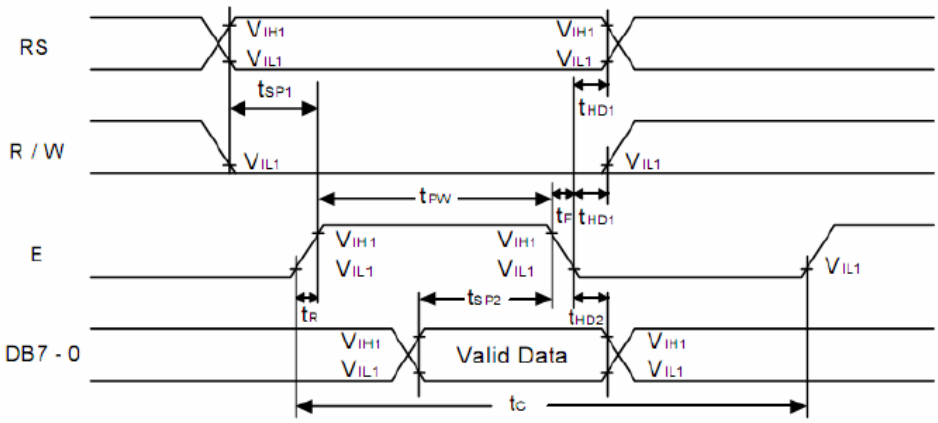
Πίνακας εντολών οθόνης

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms
Return Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (address 0). Also returns a shifted display to the home position. DD RAM contents remain unchanged.	40μs~1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and enables/disables the display.	40μs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Turns the display ON/OFF (D), or the cursor ON/OFF (C), and blink of the character at the cursor position (B).	40μs
Cursor & Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing the DD RAM contents.	40μs
Function Set	0	0	0	0	1	DL	N$	F	*	#	Sets the data width (DL), the number of lines in the display (L), and the character font (F).	40μs
Set CG RAM Address	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data can be read or altered after making this setting.	40μs
Set DD RAM Address	0	0	1	A_{DD}							Sets the DD RAM address. Data may be written or read after making this setting.	40μs
Read Busy Flag & Address	0	1	BF	AC							Reads the BUSY flag (BF) indicating that an internal operation is being performed and reads the address counter contents.	1μs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	46μs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	46μs
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift. S/C= 1: Display shift S/C = 0: cursor move R/L= 1: Shift to the right. R/L= 0: Shift to the left. DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line F = 1: 5x10 dots F = 0: 5 x 7 dots BF = 1: Busy BF = 0: Can accept data *# Set to 1 on 24x4 modules* *$ With KS0072 is Address Mode.*										DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM Address A_{DD}: DD RAM Address Corresponds to cursor address. AC: Address counter Used for both DD and CG RAM address.	Execution times are typical. If transfers are timed by software and the busy flag is not used, add 10% to the above times.

Εικόνα 3.11

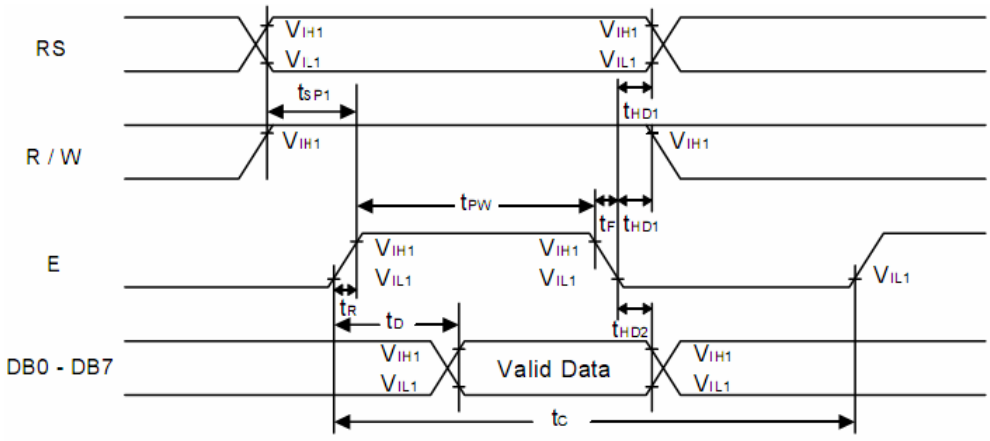
Όπως μπορούμε εύκολα να αντιληφθούμε όλη αυτή η μάζα δεδομένων που μεταφέρεται από και προς την οθόνη στον μικροεπεξεργαστή απαιτεί όσο τον δυνατόν μεγαλύτερη ταχύτητα μεταφοράς δεδομένων. Η δειγματοληψία των δεδομένων αυτών γίνεται κατά την αλλαγή κατάστασης της γραμμής δεδομένων E από την κατάσταση 0 στη κατάσταση 1. Στις παρακάτω δύο εικόνες δίνεται ένα δείγμα μεταφοράς των δεδομένων μέσω των τριών γραμμών μεταφοράς E, RS, R/W και των γραμμών δεδομένων DB0 – DB7, στην πρώτη από την μνήμη προς την οθόνη και στη δεύτερη από την οθόνη προς την μνήμη.

Χρόνοι μεταφοράς δεδομένων από τη μνήμη στην οθόνη



Εικόνα 3.12

Χρόνοι μεταφοράς δεδομένων από την οθόνη στη μνήμη



Εικόνα 3.13

3.3 Άλλα στοιχεία του μικροελεγκτή

Η πλακέτα έχει σχεδιαστεί για πολλών ειδών εφαρμογές για τις ανάγκες του εργαστηρίου Μικροελεγκτές Ι και ΙΙ. Κάποια από τα στοιχεία της πλακέτας δεν χρησιμοποιούνται για την εργασία αυτή για αυτό και θα γίνει απλή αναφορά σε αυτά ενώ σε όσα είναι απαραίτητα για την εφαρμογή θα δοθεί πιο αναλυτική περιγραφή της χρήσης και της συνδεσμολογίας τους.

3.3.1 USB Port

Η σύνδεση του μικροελεγκτή με τον υπολογιστή για τον προγραμματισμό του επιτυγχάνεται μέσω ενός καλωδίου που στο ένα άκρο του έχει θύρα USB τύπου Α για την σύνδεση στον υπολογιστή, ενώ στο άλλο άκρο του έχει θύρα USB τύπου Β για την σύνδεση του στην πλακέτα.

Το θηλυκό βύσμα που βρίσκεται στην πλακέτα επικοινωνεί μέσω αντιστάσεων με τον μικροελεγκτή και τον πομποδέκτη MAX232 ο οποίος είναι συνδεδεμένος και με τη σειριακή πόρτα RS232.

Καλώδιο ακροδεκτών τύπου Α και Β

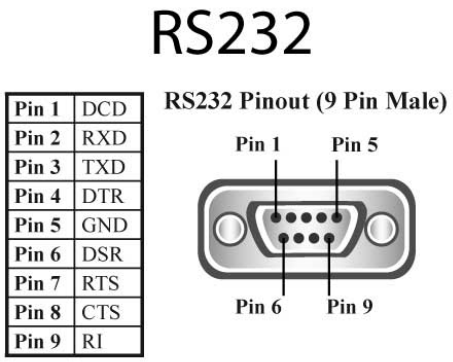
Εικόνα 3.14

Θύρα τύπου Β

Εικόνα 3.15

Σειριακή πόρτα RS232

Η σειριακή πόρτα RS232 είναι ένας ακόμα τρόπος σύνδεσης της πλακέτας με τον υπολογιστή ή αν μιλήσουμε για την γενική του χρήση είναι ένα πρότυπο για σειριακή μετάδοση δυαδικών σημάτων δεδομένων μεταξύ ενός DTE (Data Terminal Equipment) και ενός DCE (Data Circuit terminal Equipment). Χρησιμοποιείται συχνά στις σειριακές θύρες των προσωπικών υπολογιστών. Λόγω της διάδοσής του, το πρότυπο RS-232 συχνά θεωρείται ταυτόσημο με τη σειριακή θύρα.

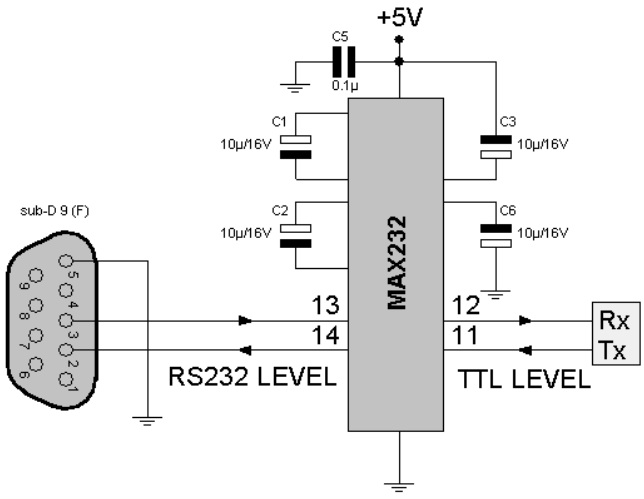


Εικόνα 3.16

Ολοκληρωμένο κύκλωμα MAX232

Το MAX232 είναι ένα ολοκληρωμένο κύκλωμα δημιουργήθηκε για πρώτη φορά το 1987 από την εταιρία Maxim Integrated Products που μετατρέπει τα σήματα από μια σειριακή θύρα RS-232 σε σήματα κατάλληλα για χρήση σε ψηφιακά λογικά κυκλώματα. Στην εικόνα παρακάτω φαίνεται η συνδεσμολογία του ολοκληρωμένου MAX232 με την σειριακή θύρα RS232.

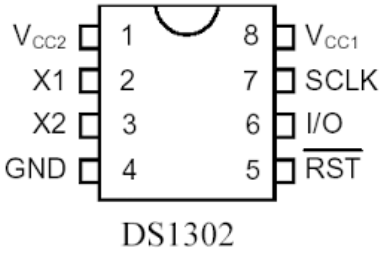
Συνδεσμολογία MAX232 με RS232



3.3.4 Ολοκληρωμένο κύκλωμα DS1302

Το ολοκληρωμένο κύκλωμα DS1302 μαζί με τον κρύσταλλο 32768 Hz χρησιμοποιούνται ως σταθεροποιητές τάσης λαμβάνοντας τροφοδοσία από την μπαταρία λιθίου 3V.

DS1302



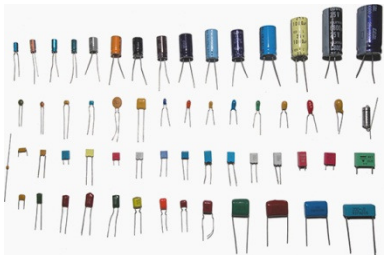
Εικόνα 3.18

3.3.5 Άλλα στοιχεία του κυκλώματος

Εικόνα 3.19 Αντιστάσεις

Εικόνα 3.20 LED

Εικόνα 3.21 Πυκνωτές



Εικόνα 3.22 Καλωδιοταινίες

Εικόνα 3.23 Μπαταρία λιθίου

Εικόνα 3.24 Διακόπτης 16pin

Εικόνα 3.25 Κρύσταλλος

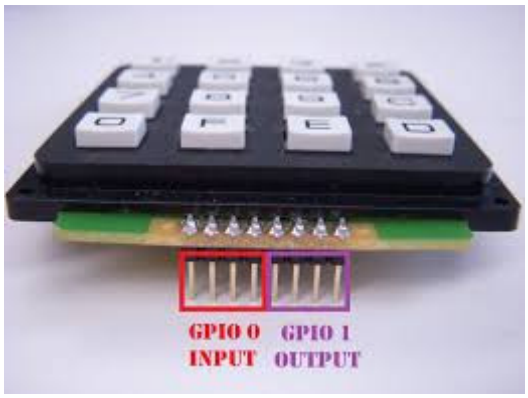
ΚΕΦΑΛΑΙΟ 4

ΣΥΝΔΕΣΜΟΛΟΓΙΑ ΠΕΡΙΦΕΡΕΙΑΚΩΝ ΣΤΟΙΧΕΙΩΝ ΠΛΑΚΕΤΑΣ

4.1- Συνδεσμολογία πληκτρολογίου 4x4

Όπως μπορούμε να δούμε στην εικόνα τα 8 pin του πληκτρολογίου χωρίζονται στα πρώτα 4 pin στήλης 1-4pin τα οποία είναι τα pin εισόδου και στα υπόλοιπα pin 5-8pin εξόδου.

Keypad 4x4 input-output



Εικόνα 4.1

Τα pin εισόδου συνδέονται κατά αντιστοιχία μέσω αντιστάσεων των 10k με την τροφοδοσία και τα pin του μικροελεγκτή

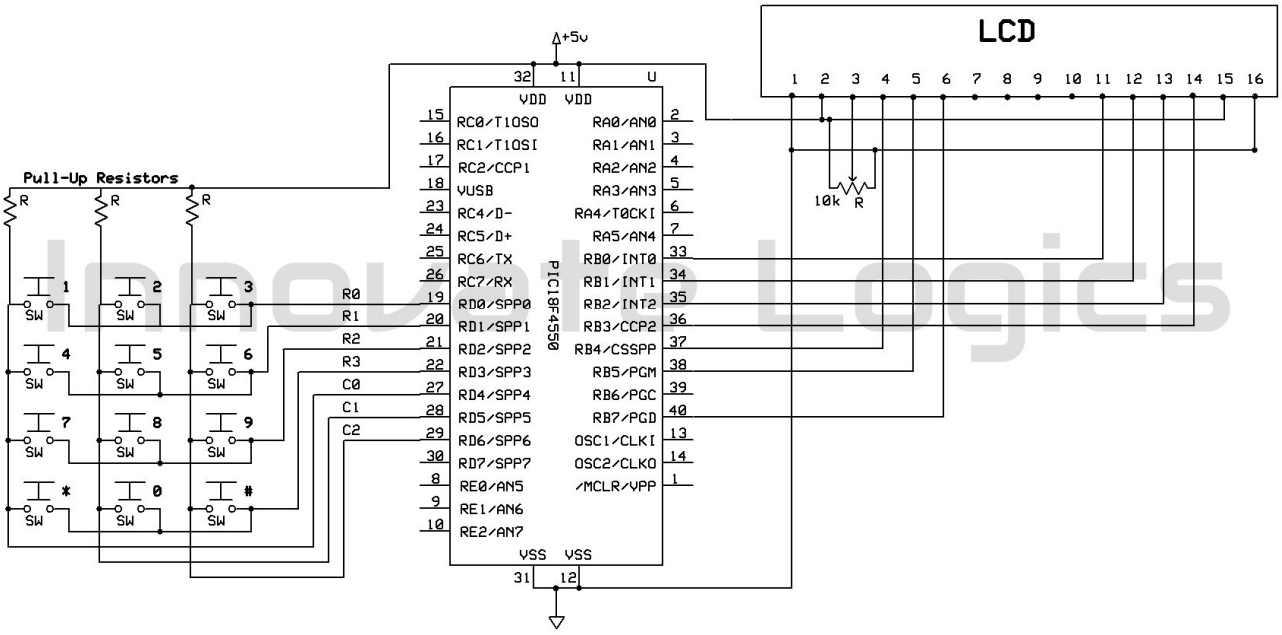
- COL1 – RB0
- COL2 – RB1
- COL3 – RB2
- COL4 – RB3

ενώ τα pin εξόδου είναι συνδεδεμένα

- ROW1 – RB4
- ROW2 – RB5
- ROW3 – RB6
- ROW4 – RB7

Στην εικόνα 4.2 φαίνεται η συνδεσμολογία του μικροελεγκτή PIC18F4550 με το πληκτρολόγιο και την οθόνη LCD. Το πληκτρολόγιο που εμφανίζεται είναι 3x4 ενώ στην πλακέτα την εργασίας είναι 4x4, οπότε υπάρχει ακόμα στήλη η COL4 που συνδέεται με τις γραμμές όπως και οι άλλες στήλες και στη συνέχεια στο RD7 pin του μικροελεγκτή.

Συνδεσμολογία Μικροελεγκτή με το πληκτρολόγιο και την οθόνη LCD

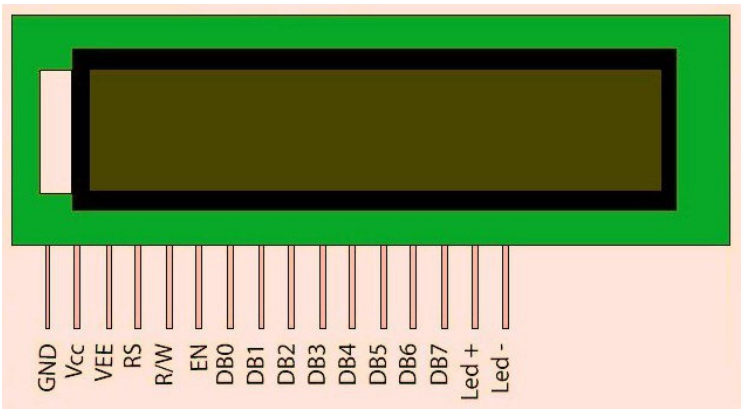


Εικόνα 4.2

4.2- Συνδεσμολογία οθόνης LCD

Η συνδεσμολογία της οθόνης φαίνεται στην παραπάνω εικόνα και με την βοήθεια της εικόνας 4.3 μπορούμε να την κατανοήσουμε καλύτερα.

LCD pins



Εικόνα 4.3

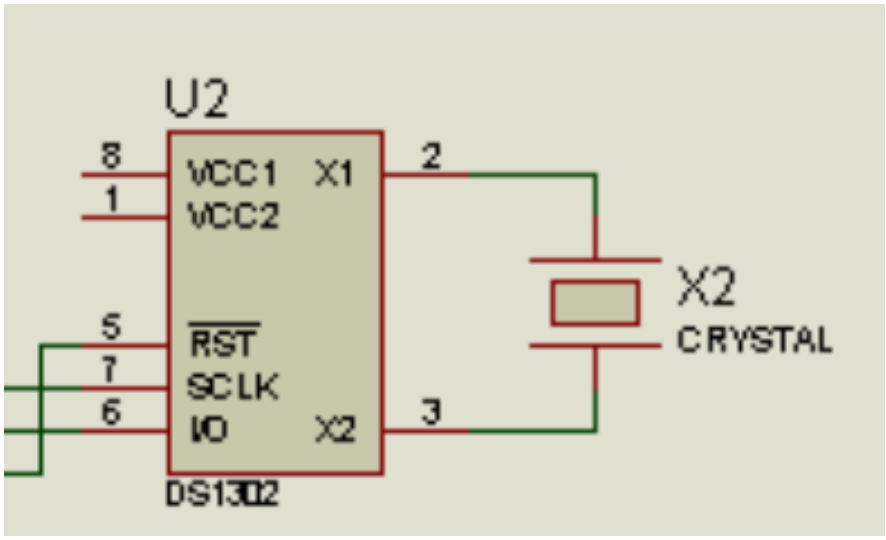
Η συνδεσμολογία της οθόνης που προκύπτει είναι

- pin1 – Γείωση
- pin2 – Τροφοδοσία
- pin3 – Ρύθμιση αντίθεσης με την μεταβλητή αντίσταση 1k
- pin4 – RB4 για την γραμμή εντολών RS
- pin5 – RB5 για την γραμμή εντολών R/W
- pin6 – RB7 για την γραμμή εντολών E
- pin7
- pin8
- pin9
- pin10
- pin11 – RB0
- pin12 – RB1
- pin13 – RB2
- pin14 – RB3
- pin15 – LED +
- pin16 – LED –

4.3- Συνδεσμολογία τροφοδοσίας

Η συνδεσμολογία τροφοδοσίας του μικροελεγκτή αποτελείται από την μπαταρία λιθίου, τον κρύσταλλο και το ολοκληρωμένο κύκλωμα DS1302. Ουσιαστικά δημιουργείται ένας σταθεροποιητής τάσης για τις ανάγκες του κυκλώματος.

Συνδεσμολογία DS1302

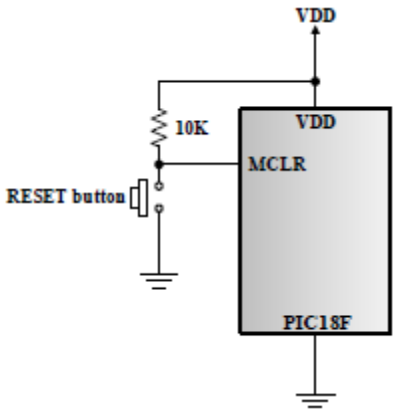


Εικόνα 4.4

4.4- Συνδεσμολογία Reset

Η συνδεσμολογία Reset ρυθμίζεται από έναν διακόπτη push button που επαναφέρει το σύστημα στην αρχική του κατάσταση. Στην αριθμομηχανή συγκεκριμένα μηδενίζει ώστε να μπορέσει να εισάγει ο χρήστης τον νέο αριθμό.

Συνδεσμολογία Reset



Εικόνα 4.5

Ο διακόπτης έχει το ένα άκρο στο του στο pin1 του μικροελεγκτή το οποίο είναι συνδεδεμένο στην τροφοδοσία διαμέσου μιας αντίστασης 10k. Με το πάτημα του button γειώνεται το pin1 και πραγματοποιείται το reset ώστε να δεχθεί το σύστημα τον επόμενο αριθμό.

ΚΕΦΑΛΑΙΟ 5

ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

5.1- Βιβλιοθήκες προγράμματος

Οι τρείς βιβλιοθήκες που χρησιμοποιήθηκαν για την δημιουργία του κώδικα αναφέρονται παρακάτω.

5.1.1- Keypad.h

Η βιβλιοθήκη <keypad.h> είναι υπεύθυνη για την επικοινωνία και την σύνδεση του κώδικα με το πληκτρολόγιο μέσω του μικροεπεξεργαστή PIC18F4550 και στη συνέχεια την απεικόνιση στην οθόνη.

5.1.2- Flex_lcd.h

Η βιβλιοθήκη <flex_lcd.h> είναι αυτή που λειτουργεί ως συνδετικός κρίκος με τις εντολές της μεταξύ της οθόνης και του πληκτρολογίου ώστε να εμφανίζονται τα πλήκτρα που έχουν πατηθεί και να απεικονίζεται το αποτέλεσμα

5.1.3- Main.h

Η βιβλιοθήκη <main.h> είναι μια απαραίτητη βιβλιοθήκη για να μπορέσει να τρέξει το πρόγραμμα και να προγραμματιστεί η πλακέτα. Είναι ένα αρχείο σε γλώσσα μηχανής η οποία περιέχει κώδικα, δηλώσεις δεδομένων, πίνακας συμβόλων και μεταβλητών.

5.2- Συναρτήσεις κώδικα

Οι συναρτήσεις που χρησιμοποιούνται στον κώδικα ουσιαστικά είναι ρουτίνες αρχικοποίησης για το κάθε επιμέρους τμήμα που απαρτίζει την αριθμομηχανή.

5.2.1- Συνάρτηση init()

Η συνάρτηση init είναι η ρουτίνα αρχικοποίησης η οποία θέτει την πόρτα D ως έξοδο.

5.2.2- Συνάρτηση lcd_init()

Η συνάρτηση lcd_init είναι η ρουτίνα αρχικοποίησης εμφάνισης των χαρακτήρων που έχουν πατηθεί από το πληκτρολόγιο στην οθόνη

5.2.3- Συνάρτηση kbd_init()

Η συνάρτηση kbd_init είναι η ρουτίνα αρχικοποίησης της συνάρτησης ανάγνωσης χαρακτήρων από το πληκτρολόγιο της αριθμομηχανής.

5.2.4-Συνάρτηση clearall()

Η συνάρτηση clearall έχει ως στόχο τον καθαρισμό της οθόνης και των μεταβλητών που χρησιμοποιήθηκαν στον κώδικα μετά το τέλος της κάθε πράξεις και με το πάτημα του πλήκτρου C.

ΚΕΦΑΛΑΙΟ 6

6.1- Προβλήματα και βελτιώσεις

Κατά την δημιουργία του κώδικα προέκυψαν διάφορα προβλήματα τα οποία είτε μπόρεσαν να επιλυθούν είτε απλά εντοπίστηκαν και θα μπορούσαν να αποτελέσουν «τροφή» για την επέκταση της πτυχιακής και του κώδικα και των δυνατοτήτων τους.

Το πρώτο πρόβλημα που εντοπίστηκε ήταν ότι ενώ σχεδιάστηκε στο προγραμματιστικό περιβάλλον του proteus και δεν παρουσίασε κανένα πρόβλημα στο περιβάλλον προσομοίωσης, όταν έγινε ο προγραμματισμός της πλακέτας και πατήθηκε το erase εμφανίστηκε στην οθόνη ο αριθμός 7. Πατώντας τους επόμενους αριθμούς για να εκτελέσουμε την πράξη που θέλουμε δεν σβηνόταν, αλλά αντιθέτως το πρόγραμμα το καταλάβαινε ως μέρος του αριθμού της πράξης και το λάμβανε υπόψη για τον υπολογισμό του αποτελέσματος.

Το δεύτερο πρόβλημα που αντιμετωπίσαμε ήταν ότι μπορούσε να δεχτεί την πληκτρολόγηση μέχρι 4 σημαντικών ψηφίων και μετά έκανε στρογγυλοποίηση σε όποιον αριθμό πληκτρολογούσαμε από αυτό και μετά.

Το τρίτο πρόβλημα που προέκυψε ήταν το ότι επειδή έγινε χρήση float μεταβλητών από το $8^{ο}$ σημαντικό ψηφίο και μετά το αποτέλεσμα παρουσιάζει σφάλματα. Αυτό συμβαίνει γιατί οι float μεταβλητές είναι χωρητικότητας 23bit, που σημαίνει ότι στο δυαδικό σύστημα το σφάλμα εμφανίζεται μετά το $23^{ο}$ σημαντικό ψηφίο, ενώ στο δεκαδικό στο $8^{ο}$ (2^{8}).

6.2- Βελτιώσεις

Για το πρώτο πρόβλημα που παρουσιάστηκε εγγράφτηκε μια συνθήκη if(k<0x30 || k>0x39) η οποία απαγορεύει την εμφάνιση οποιουδήποτε συμβόλου από τον ASCII κώδικα καθώς το 7 που εμφανιζόταν ήταν λόγω του ότι σύμβολο του ASCII κώδικα περνούσε μέσω του προγράμματος και εμφανιζόταν στην οθόνη.

Για το δεύτερο πρόβλημα που προέκυψε διορθώθηκε στον κώδικα στην εντολή εμφάνισης του αποτελέσματος ο επιθυμητός αριθμός σημαντικών ψηφίων που θέλαμε να εμφανίζεται.

Από printf(lcd_putc,"Res=%.4f",result) σε printf(lcd_putc,"%10.5f",result)

Για το τρίτο πρόβλημα δεν βρέθηκε λύση διότι που θα μπορούσε να εφαρμοστεί στο δικό μας κύκλωμα διότι θα μπορούσε να λυθεί με μεταβλητές double των 64bit που όμως δεν υποστηρίζονται από την αριθμομηχανή που κατασκευάστηκε.

6.3- Συμπεράσματα

Η εργασία πραγματοποιήθηκε για ακαδημαϊκούς λόγους εφαρμόζοντας τον κώδικα στην πλακέτα του εργαστηρίου ώστε να υπάρχει η δυνατότητα περάτωσης και άλλων εφαρμογών στη πορεία. Μέσα από την εργασία έγινε κατανοητός ο τρόπος συνδέσεις των περιφερειακών συστημάτων μεταξύ τους καθώς και η χρήση μεταβλητών και γλώσσας C για την δημιουργία του κώδικα.

Η κατασκευή της πλακέτας ολόκληρης κόστισε περίπου 50€ ενώ αν κατασκευαζόταν για να λειτουργεί ως αριθμομηχανή θα κόστιζε λιγότερο, κοντά στα 25€.

Γνωρίζοντας τον ανταγωνισμό και την εισαγωγή φθηνών αριθμομηχανών από Κίνα μόλις στα 2-5€ για απλή αριθμομηχανή των τεσσάρων βασικών πράξεων όπως αυτή που κατασκευάστηκε, αλλά και όταν η επιστημονική αριθμομηχανή που δίνει δυνατότητα πολλών και περίπλοκων πράξεων κοστίζει μόλις στα 25€ δεν θα μπορούσε να βγει στην ανταγωνιστική αγορά ως προϊόν.

ΠΑΡΑΡΤΗΜΑ 1

ΚΥΡΙΩΣ ΠΡΟΓΡΑΜΜΑ

```
#include <main.h>
#include <keypad.h>                    // Ρουτίνα ανάγνωσης από το πληκτρολόγιο
#include <flex_lcd.h>                  //Ρουτίνα εμφάνισης χαρακτήρων στην οθόνη.

#byte PORTB=0xf81          //η θέση μνήμης του καταχωρητή δεδομένων της πόρτας B
#byte PORTD=0xf83          //η θέση μνήμης του καταχωρητή δεδομένων της πόρτας D

//------------------- Ορισμός μεταβλητών-----------------------------------------------------------------

char state=1               //μεταβλητή ένδειξης εισαγωγής του πρώτου ή του δεύτερου αριθμού

char N1=0;                 //μεταβλητή για αποθήκευση του ASCII πλήκτρου πρώτου αριθμού
char N2=0;                 //μεταβλητή για αποθήκευση του ASCII πλήκτρου δεύτερου αριθμού
char praxi='+';                    //μεταβλητή για αποθήκευση της πράξης που θα εκτελεστεί

float32 number1;                   //Μεταβλητή  float όπου θα αποθηκευτεί ο 1ος αριθμός
float32 number2;                   //Μεταβλητή float όπου θα αποθηκευτεί ο 2ος αριθμός
float32 result;            //Μεταβλητή float όπου θα αποθηκευτεί το αποτέλεσμα της πράξης

float number1decimalvalue=1;
                                   //Μεταβλητή υπολογισμού της αξίας
                                   // των δεκαδικών ψηφίων του 1ο αριθμού
float number2decimalvalue=1;
                                   //Μεταβλητή υπολογισμού της αξίας
                                   // των δεκαδικών ψηφίων του 2ο αριθμού
```

```
//------------------Ορισμός συναρτήσεων---------------------------------------

void init (void);                    // Συνάρτηση αρχικοποίησης(Η πόρτα D γίνεται έξοδος)
void clearall (void);                //Συνάρτηση καθαρισμού της οθόνης και μεταβλητών
//---------------------Κυρίως πρόγραμμα-----------------------------------------------------------
void main()                                                    // κυρίως  πρόγραμμα
{
char k;                  //μεταβλητή αποθήκευσης ASCII κώδικα του πλήκτρου που πατήθηκε

// --------------------Δηλώσεις συναρτήσεων-------------------------------------------------------

init();                                  //ρουτίνα αρχικοποίησης(Η πόρτα D γίνεται έξοδος)
lcd_init();                              //αρχικοποίησης ρουτίνας εμφάνισης στην οθόνη
kbd_init();              // αρχικοποίηση της συνάρτησης ανάγνωσης από το πληκτρολόγιο
clearall();                              // συνάρτηση καθαρισμού οθόνης και μεταβλητών

//--------------------Δηλώσεις συναρτήσεων-------------------------------------------------------
  while(TRUE)
   {
    switch(state){                  //ελέγχει την τιμή της μεταβλητής state που μας δείχνει
                          //αν θα είμαστε σε κατάσταση εισαγωγής του 1ου  αριθμού (state=1)
                                    // ή σε κατάσταση εισαγωγής 2ου αριθμού (state=2)

  case 1:                                   //κατάσταση εισαγωγής του πρώτου αριθμού

       k=kbd_getc();                                  //διάβασε την τιμή που επιστρέφει
                                          //η συνάρτηση ανάγνωσης από το πληκτρολόγιο
      while (k!=0)
                          // αν πατήθηκε ένα πλήκτρο εκτελείται το περιεχόμενο των αγκυλών
                    // του while γιατί τότε είναι το  κ ≠ 0.  Εκτελούνται οι εντολές μέσα στις
                    // αγκύλες μόνο αν πατηθεί ένα πλήκτρο. Τότε εμφανίζεται η τιμή του
```

```
                    // πλήκτρου που πατήθηκε, το σύμβολο x (πολλοπλασισμός) και γίνεται
                    // μετάβαση στην state 2 για να εισαχθεί ο δεύτερος αριθμός.

        {
if(k=='=') {clearall();}                    //Πρώτα ελέγχεται αν πατήθηκε το'='.
                                            //Αν ναι γίνεται καθαρισμός της οθόνης
                                            //καθαρισμός της οθόνης αν πατηθεί το =

    if((k=='+') || (k=='-') || (k=='*') || (k== '/'))
                                            //Ελέγχεται αν πατήθηκε πλήκτρο πράξης
      { state = 3; praxi=k; break;}         //Κατάσταση εισαγωγής του δεύτερου
ψηφίου
        if(k=='.')                          //Ελέγχεται αν πατήθηκε η υποδιαστολή
         { state = 2; printf(lcd_putc,"%c",'.');break; }    //πάει στην κατάσταση2
                                            //εκτύπωση της υποδιαστολής και
                                            //κατάσταση εισαγωγής του δεκαδικού μέρους του 1ου αριθμού
                                            //Αν πατήθηκε πλήκτρο ψηφίου υπολογίζεται ο αριθμό

        if( k<0x30||k>0x39){state=1;break; }     //έλεγχος ότι πατήθηκε πλήκτρο αριθμού
                                            //και περιορισμός εισόδου οποιουδήποτε  άλλου συμβόλου
                                            //ή χαρακτήρα που μπορεί να προκύψει από τον
                                            //ASCII κώδικα
        N1= k & 0b00001111;                 //από τον ASCII κώδικα λαμβάνουμε την τιμή
                                            // του ψηφίου που πατήθηκε στην μεταβλητή N1
       number1=(float)N1+number1*10;        //Υπολογίζουμε την νέα τιμή του number1
                                            //μετά την εισαγωγή νέου ψηφίου
       lcd_gotoxy(1,1);                     //πήγαινε στην πρώτη θέση της πρώτης γραμμής

       printf(lcd_putc,"%c",k);
                                            //Εκτυπώνεται ο χαρακτήρας του πλήκτρου που πατήθηκε
```

```
    break;                                          // έξοδος από το while(k!=0)
     }                                              //αγκύλη που κλείνει το while(k!=0)
   break;                                           // πάει στο select case

case 2:                                 // Εισαγωγή δεκαδικού μέρους του πρώτου αριθμού
    k=kbd_getc();
                                                    //διάβασε την τιμή που επιστρέφει η συνάρτηση
                                                    //ανάγνωσης από το πληκτρολόγιο
    while (k!=0)                                    // αν πατήθηκε ένα πλήκτρο εκτελείται το
                                                    //περιεχόμενο των αγκυλών
                                                    // του while γιατί τότε είναι κ διάφορο του 0.
                                                    //Εκτελούνται οι εντολές μέσα στις
                                                    // αγκύλες μόνο αν πατηθεί ένα πλήκτρο.
        {
    if(k=='=') {clearall(); state=1; break; }
                                                    //Πρώτον ελέγχεται αν πατήθηκε το'='
                                                    //Αν πατηθεί το = καθαρισμός της οθόνης
                                                    //και επιστροφή στην κατάσταση 1 για εισαγωγή
                                                    //νέου 1ου αριθμού
    if((k=='+') || (k=='-') || (k=='*') || (k== '/'))
                                                    //Δεύτερον ελέγχεται αν πατήθηκε πλήκτρο πράξης
     {state = 3; praxi=k; break;}
                                                    //μετάβαση στην κατάσταση 3 για την
                                                    //εμφάνιση του συμβόλου της πράξης
                                                    //Αν πατήθηκε πλήκτρο ψηφίου συμπληρώνεται
                                                    // το δεκαδικό μέρος του πρώτου αριθμού

    if(k<0x30||k>0x39){state=2;break;               //έλεγχος ότι πατήθηκε πλήκτρο αριθμού
     N1= k & 0b00001111;
                                                    //από τον ASCII κώδικα λαμβάνουμε την τιμή
                                                    // του ψηφίου που πατήθηκε στην μεταβλητή N1
```

```
                                        // Π.χ. ο ASCII κώδικας του 7 είναι (37)h.
                                        //Με το AND κρατιέται το7το οποίο
                                    // θα χρησιμοποιηθεί για να γίνει ο πολλαπλασιασμός
                                        // και η εμφάνιση του ψηφίου στην οθόνη.

      number1decimalvalue=number1decimalvalue*0.1;
                                                //με την εισαγωγή κάθε νέου ψηφίου
                                               //η αξία του πολλαπλασιάζεται με 0.1
      number1=number1+((float)N1)*number1decimalvalue;
                                             //υπολογισμός της νέας τιμής του πρώτου
                                               //αριθμού μετά την εισαγωγή ψηφίου
      printf(lcd_putc,"%c",k);              //τυπώνεται ο χαρακτήρας του πλήκτρου
                                                //που πατήθηκε με την αντίστοιχη
                                                //τιμή του στο δεκαδικό σύστημα
       break;                                       // έξοδος από το while(k!=0)
        }                                          //αγκύλη που κλείνει το while(k!=0)
     break;                                        // Έξοδος και πάει στο select case

case 3:                                  // Κατάσταση εισαγωγής συμβόλου επιλογής πράξης
       lcd_gotoxy(1,2);                  //πήγαινε στην πρώτη θέση της δεύτερης γραμμής
      printf(lcd_putc,"%c",praxi);                //εκτύπωσε το σύμβολο της πράξης
      state=4;                                          // πάει στο select case 4
      break;                                                  //Εξοδος

case 4:                                        //Από το switch ερχόμαστε εδώ όταν είμαστε
                                              //σε κατάσταση εισαγωγής του δεύτερου ψηφίου
     k=kbd_getc();              //εκτέλεση της συνάρτησης ανάγνωσης από το πληκτρολόγιο
     while (k!=0)              //αν πατηθεί κάποιο πλήκτρο εκτελούνται οι εντολές μέσα στις
                                                //αγκύλες της while(k!=0)
     {
     if(k=='=')                                          //Άμα πατηθεί το "="
```

```
 {state = 6; break;                          //εμφανίζεται το αποτέλεσμα της πράξης
 }
if(k=='.')                                   //Αν πατηθεί υποδιαστολή
 {state = 5; printf(lcd_putc,"%c",'.');break;}          //πάμε στην case 5
                                             //εκτύπωση της υποδιαστολής και
                                             //κατάσταση εισαγωγής του δεκαδικού
                                             //μέρους του 2ου αριθμού
 if(k<0x30||k>0x39){state=2;break;}          //έλεγχος ότι πατήθηκε πλήκτρο αριθμού
 N2 = k & 0b00001111;
                                  //υπολογίζουμε τον δεύτερο αριθμό από τον ASCII κώδικά του
 number2=(float)N2+number2*10;
                                  //Υπολογίζουμε την νέα τιμή του number2 μετά την
                                  //εισαγωγή νέου ψηφίου
 printf(lcd_putc,"%c",k);      //τυπώνεται ο χαρακτήρας του πλήκτρου που πατήθηκε
 break;                                      // βγαίνει από το while
}                                            //αγκύλη που κλείνει το while(k!=0)
break;                                       // πάει στο select case
case 5:                                      // Εισαγωγή δεκαδικού μέρους του  δεύτερου αριθμού
                                             // Εισαγωγή δεκαδικού μέρους του πρώτου αριθμού
 k=kbd_getc();                               //διάβασε την τιμή που επιστρέφει η συνάρτηση
                                             // ανάγνωσης από το πληκτρολόγιο
while (k!=0)                                 // αν πατήθηκε ένα πλήκτρο εκτελείται το περιεχόμενο
                                             // των αγκυλών του while γιατί τότε είναι κ διάφορο του 0.
                                             //Εκτελούνται οι εντολές μέσα στις
                                             // αγκύλες μόνο αν πατηθεί ένα πλήκτρο
  {                                          //αρχική αγκύλη του while
 if(k=='=')                                  //Πρώτον ελέγχεται αν πατήθηκε το'='
  {state = 6; break;                         //εμφάνιση του αποτελέσματος της πράξης
  }
if(k<0x30||k>0x39){state=2;break;}           //έλεγχος ότι πατήθηκε πλήκτρο αριθμού
 N2= k & 0b00001111;      //από τον ASCII κώδικα λαμβάνουμε την τιμή του ψηφίου
```

```
                                                //που πατήθηκε στην μεταβλητή N2
                                    // Π.χ. ο ASCII κώδικας του 7 είναι (37)h. Με το AND κρατιέται το
                                    // 7 το οποίο θα χρησιμοποιηθεί για να γίνει ο πολλαπλασιασμός
                                                // και η εμφάνιση του ψηφίου στην οθόνη.
      number2decimalvalue=number2decimalvalue*0.1;
                                                //με την εισαγωγή κάθε νέου ψηφίου
                                                //η αξία του πολλαπλασιάζεται με 0.1
      number2=number2+((float)N2)*number2decimalvalue;
                                                //υπολογισμός της νέας τιμής του 2ου
                                                //αριθμού μετά την εισαγωγή ψηφίου
      printf(lcd_putc,"%c",k);                  //τυπώνεται ο χαρακτήρας του πλήκτρου
                                    //που πατήθηκε και αντιστοιχεί στο δεκαδικό σύστημα
       break;                                   // έξοδος από το while(k!=0)
        }                                       //αγκύλη που κλείνει το while(k!=0)
     break;                                     // πάει στο select case

case 6:                             //Εκτέλεση της πράξης και εμφάνιση του αποτελέσματος

   if (praxi=='+'){result=number1+number2;}          //Αν πατήθηκε + γίνεται πρόσθεση
   if (praxi=='-'){result=number1-number2;}          //Αν πατήθηκε - γίνεται αφαίρεση
   if (praxi=='*'){result=number1*number2;}  //Αν πατήθηκε * γίνεται πολλαπλασιασμός
   if (praxi=='/'){result=number1/number2;}          //Αν πατήθηκε / γίνεται διαίρεση

      lcd_gotoxy(7,2);                      //πήγαινε στην 7η θέση της δεύτερης γραμμής
      result=result+0.00000;   //Γίνεται στρογγυλοποίηση στα τελευταία 5 δεκαδικά ψηφία
      printf(lcd_putc,"%10.5f",result);             //εκτύπωση του αποτελέσματος
                                          //g(rounded decimal),g(truncated decimal)
   state=1;                       // επιστροφή στην κατάσταση 1 για εισαγωγή νέου αριθμού
                                  // Για να καθαριστούν οι μεταβλητές και η οθόνη πρέπει να
                                 // πατήσουμε =. Το = ελέγχεται στην κατάσταση 1 και γίνεται
                                          // ο καθαρισμός των μεταβλητών και της οθόνης
```

```
        break;                                          // Έξοδος και πάει στο select case
    }                                                   //αγκύλη που κλείνει το switch(state)

    }                                                   //αγκύλη που κλείνει το while(TRUE)
}                                                       //αγκύλη που κλείνει το main

//--------------------Συναρτήσεις-------------------------------

void init (void){                                       //Συνάρτηση αρχικοποίησης
set_tris_d(0x00);                                       // Η θύρα D γίνεται έξοδος
        }                                               //κλείσιμο συνάρτησης αρχικοποίησης
void clearall (void){                                   // Συνάρτηση καθαρισμού οθόνης και μεταβλητών
        number1=0;                                      //ο πρώτος αριθμός γίνεται 0
        number2=0;                                      //ο δεύτερος αριθμός γίνεται 0
        result=0;                                       // μηδενισμός του αποτελέσματος
        number1decimalvalue=1;
                                //επαναφορά της μεταβλητής στην αρχική της κατάστασης
        number2decimalvalue=1;
                                //επαναφορά της μεταβλητής στην αρχική της κατάστασης
        printf(lcd_putc,"\f");                          //καθαρισμός οθόνης
        }                                               // κλείσιμο συνάρτησης καθαρισμού
//----------------------------Τέλος κώδικα------------------------------
```

ΣΥΝΑΡΤΗΣΗ FLEX_LCD

```
#define LCD_DB4   PIN_B4
#define LCD_DB5   PIN_B5
#define LCD_DB6   PIN_B6
#define LCD_DB7   PIN_B7

#define LCD_E     PIN_B3
#define LCD_RS    PIN_B2
#define LCD_RW    PIN_B1
#define lcd_type 2        // 0=5x7, 1=5x10, 2=2 lines
#define lcd_line_two 0x40 // LCD RAM address for the 2nd line
int8 const LCD_INIT_STRING[4] =
{
 0x20 | (lcd_type << 2), // Func set: 4-bit, 2 lines, 5x8 dots
 0xc,                   // Display on
 1,                     // Clear display
 6                      // Increment cursor
 };
//=================================================================
void lcd_send_nibble(int8 nibble);
void lcd_send_byte(int8 address, int8 n);
void lcd_init(void);
void lcd_gotoxy(int8 x, int8 y);
void lcd_putc(char c);
//=================================================================
//-----------------------------------------------------------
void lcd_send_nibble(int8 nibble)
{
// Note:  !! converts an integer expression
// to a boolean (1 or 0).
```

```
 output_bit(LCD_DB4, !!(nibble & 1));
 output_bit(LCD_DB5, !!(nibble & 2));
 output_bit(LCD_DB6, !!(nibble & 4));
 output_bit(LCD_DB7, !!(nibble & 8));

 delay_cycles(20);
 output_high(LCD_E);
 delay_us(50);
 output_low(LCD_E);
}
//-----------------------------------
// This sub-routine is only called by lcd_read_byte().
// It's not a stand-alone routine.  For example, the
// R/W signal is set high by lcd_read_byte() before
// this routine is called.
//----------------------------------------
// Send a byte to the LCD.
void lcd_send_byte(int8 address, int8 n)
{
output_low(LCD_RS);

if(address)
   output_high(LCD_RS);
else
   output_low(LCD_RS);

 delay_cycles(10);
output_low(LCD_E);
lcd_send_nibble(n >> 4);
lcd_send_nibble(n & 0xf);
}
```

```
//-----------------------------
void lcd_init(void)
{
int8 i;
output_low(LCD_RS);
output_low(LCD_E);
delay_ms(200);
for(i=0 ;i < 3; i++)
  {
   lcd_send_nibble(0x03);
   delay_ms(10);
  }
lcd_send_nibble(0x02);

for(i=0; i < sizeof(LCD_INIT_STRING); i++)
  {
   lcd_send_byte(0, LCD_INIT_STRING[i]);

   // If the R/W signal is not used, then
   // the busy bit can't be polled.  One of
   // the init commands takes longer than
   // the hard-coded delay of 60 us, so in
   // that case, lets just do a 5 ms delay
   // after all four of them.

   delay_ms(10);
   }
}
//-----------------------------
void lcd_gotoxy(int8 x, int8 y)
{
```

```
int8 address;

if(y != 1)
   address = lcd_line_two;
else
   address=0;
address += x-1;
lcd_send_byte(0, 0x80 | address);
}
//------------------------------
void lcd_putc(char c)
{
 switch(c)
   {
    case '\f':
      lcd_send_byte(0,1);
      delay_ms(4);
      break;

    case '\n':
       lcd_gotoxy(1,2);
       break;
    case '\b':
       lcd_send_byte(0,0x10);
     break;
    default:
       lcd_send_byte(1,c);
       break;
   }
}
//------------------------------
```

ΣΥΝΑΡΤΗΣΗ KEYPAD

```
//Keypad connection:
#define col0 PIN_D0
#define col1 PIN_D1
#define col2 PIN_D2
#define col3 PIN_D3
#define row0 PIN_D4
#define row1 PIN_D5
#define row2 PIN_D6
#define row3 PIN_D7
//------------------- Variable definition---------------------------
char const KEYS[4][4] =
{{'1','2','3','/'},
 {'4','5','6','*'},
 {'7','8','9','-'},
 {'C','0','=','+'}};
#define KBD_DEBOUNCE_FACTOR 33 // Set this number to apx n/333 where
// n is the number of times you expect
// to call kbd_getc each second
//------------------Function Definition-----------------------------
short int ALL_ROWS (void);
void kbd_init(void);
char kbd_getc(void);
//-----------------------------------------------------------------
void kbd_init(void)
{
set_tris_d(0xF0);
output_d(0x0F);
}
//-----------------------------------------------------------
short int ALL_ROWS (void)
```

```
{
if(input (row0) & input (row1) & input (row2) & input (row3))
return (0);
else
return (1);
}
//-------------------------------------------------------
char kbd_getc(void)
{
static byte kbd_call_count;
static short int kbd_down;
static char last_key;
static byte col;

byte kchar;
byte row;

kchar='\0';

if(++kbd_call_count>KBD_DEBOUNCE_FACTOR)
  {
     switch (col)
            {
             case 0:
              output_low(col0);
            output_high(col1);
            output_high(col2);
            output_high(col3);
            break;

            case 1:
```

```
    output_high(col0);
    output_low(col1);
    output_high(col2);
    output_high(col3);
    break;

     case 2:
    output_high(col0);
      output_high(col1);
    output_low(col2);
    output_high(col3);
    break;

     case 3:
    output_high(col0);
    output_high(col1);
    output_high(col2);
    output_low(col3);
    break;
     }

  if(kbd_down)
      {
       if(!ALL_ROWS())
           {
           kbd_down=false;
           kchar=last_key;
            last_key='\0';
             }
      }
else
```

```
            {
             if(ALL_ROWS())
               {
                if(!input (row0))
                     row=0;
                else if(!input (row1))
                     row=1;
                else if(!input (row2))
                    row=2;
                else if(!input (row3))
                     row=3;

                last_key =KEYS[row][col];
                kbd_down = true;
               }
           else
                 {
              ++col;
              if(col==4)
                   col=0;
                  }
            }
           kbd_call_count=0;
    }
       return(kchar);
}
//-------------------------------------------------------------
```

ΠΑΡΑΡΤΗΜΑ 2

2.1 Περιγραφή του προγραμματιστικού περιβάλλοντος MPLAB

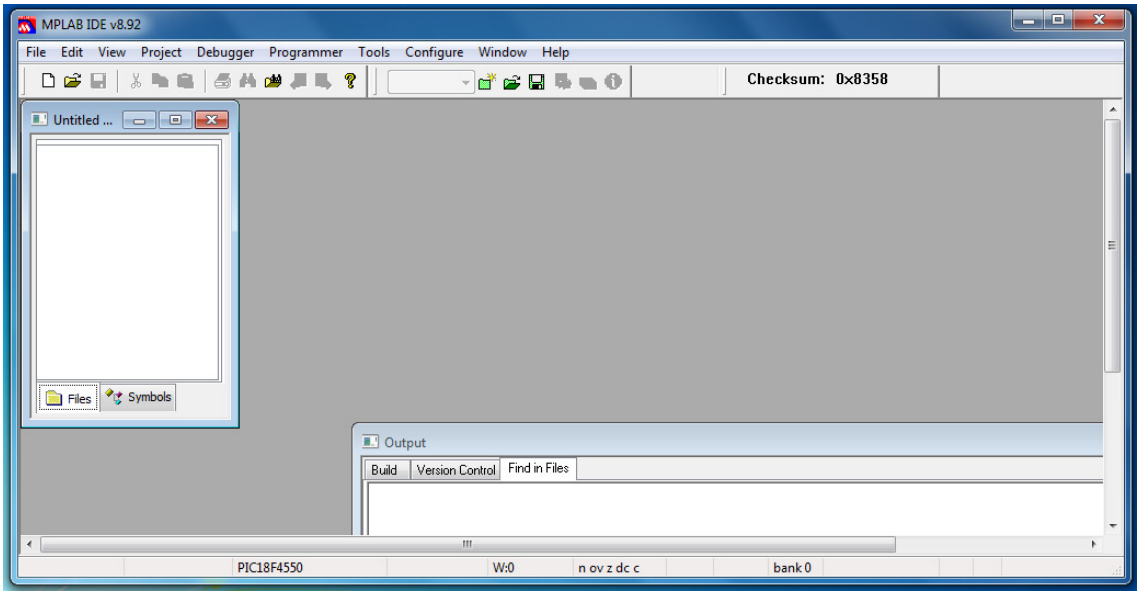
Εισαγωγή

Σε αυτό το παράρτημα αναφέρουμε εν συντομία τις ρυθμίσεις μετά την εγκατάσταση του προγράμματος MPLAB IDE v8.92 και στη συνέχεια το πώς δημιουργούμε ένα project στο προγραμματιστικό του περιβάλλον.

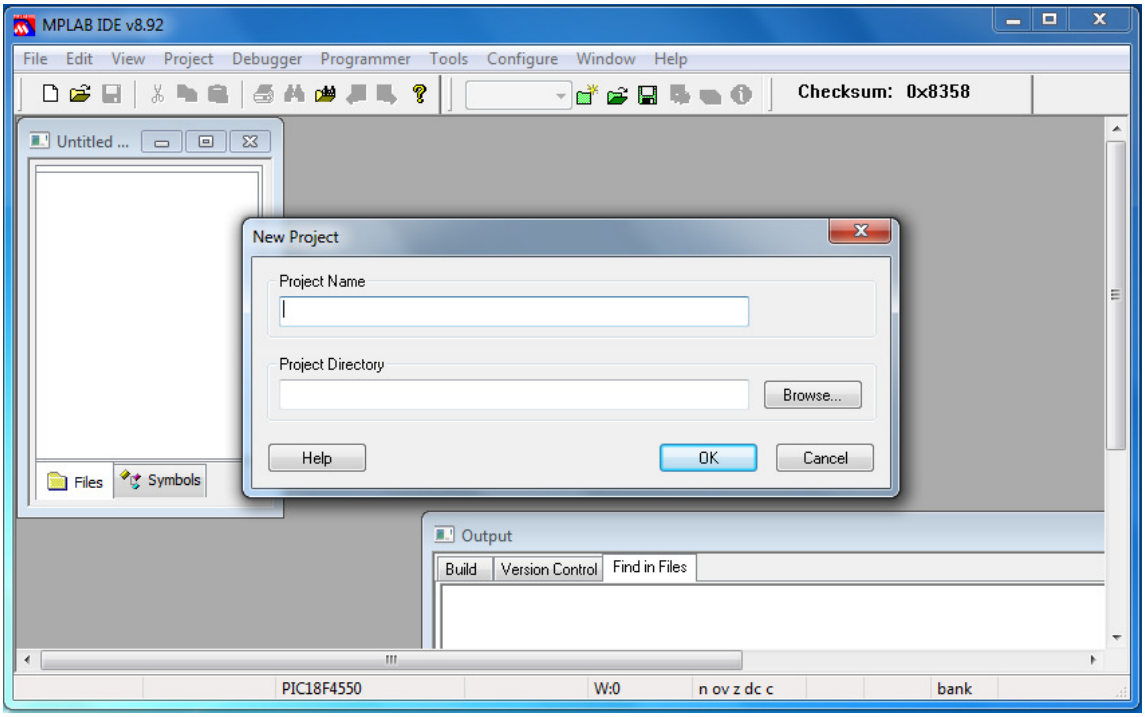
Στόχος είναι η κατανόηση του ώστε να καθιστά εύκολη τη δημιουργία ενός project βήμα βήμα με την βοήθεια και του προγράμματος PIC C Compiler αλλά και της αναπτυξιακής πλακέτας PICkit 3.

- Ανοίγουμε το MPLAB

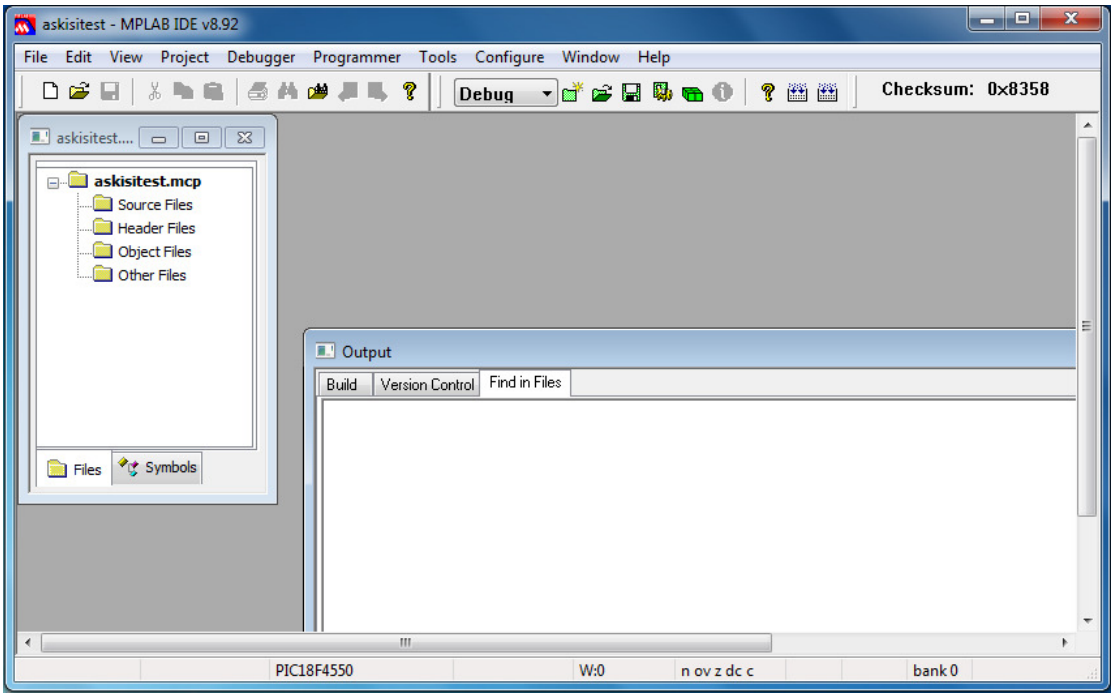
- Μας εμφανίζει το αρχικό παράθυρο παρακάτω



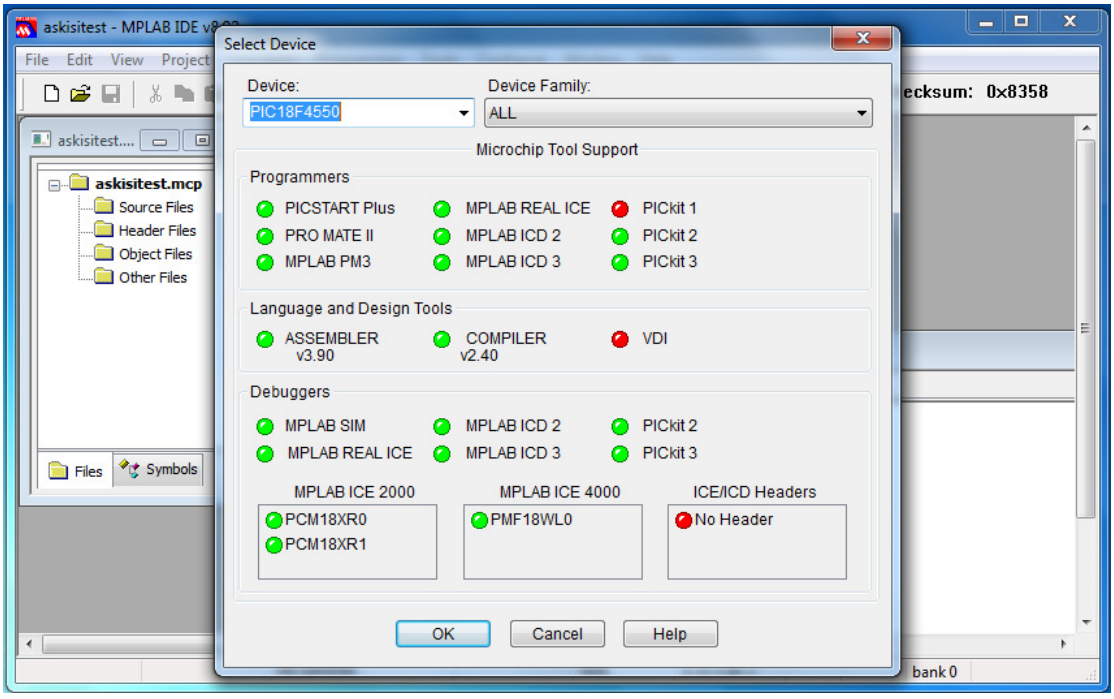
- Πηγαίνουμε στο κεντρικό μενού και επιλέγουμε **project → new** όπου εμφανίζεται το παρακάτω πλαίσιο



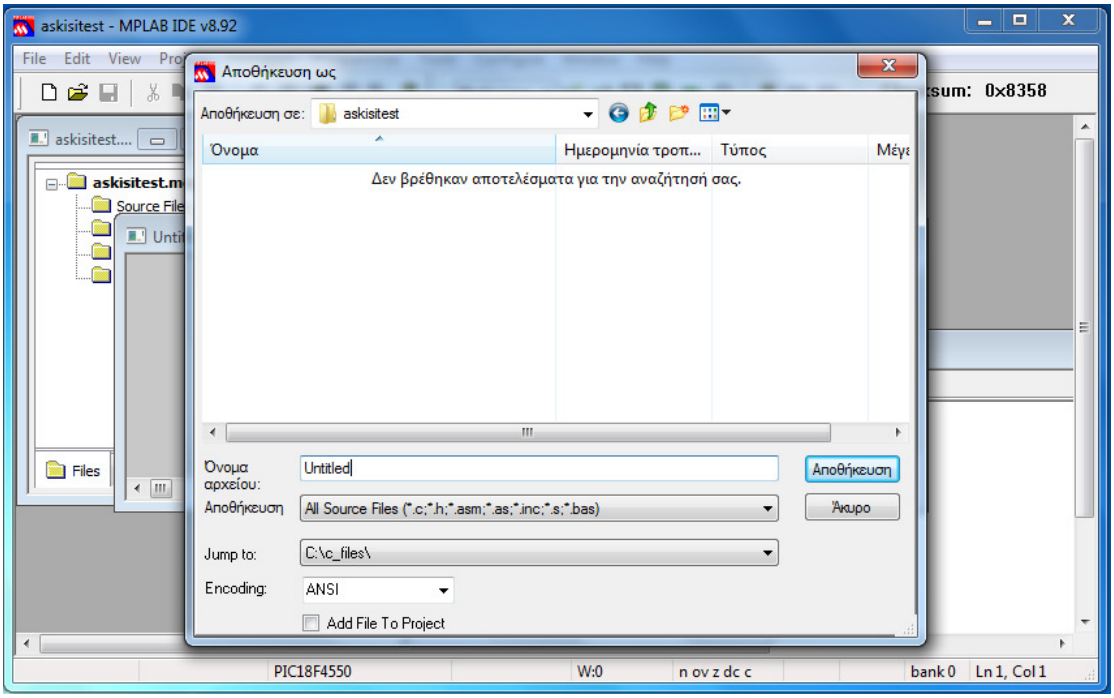
- Επιλέγουμε να δώσουμε όνομα με λατινικούς χαρακτήρες και χωρίς σύμβολα στο πεδίο **project name** για να μην δημιουργήσουμε πρόβλημα κατά την εκτέλεση και στο πεδίο **project directory** τον φάκελο στον οποίο θέλουμε να αποθηκευτεί μαζί με όλα τα αρχεία που συνδέονται με αυτό και πατάμε οκ.
- Στη συνέχεια μας εμφανίζει την παρακάτω οθόνη στο περιβάλλον όπου θα γραφτεί ο κώδικας του προγράμματος. Σε περίπτωση που δεν εμφανιστεί πατάμε από το κυρίως μενού **view → project → output** .



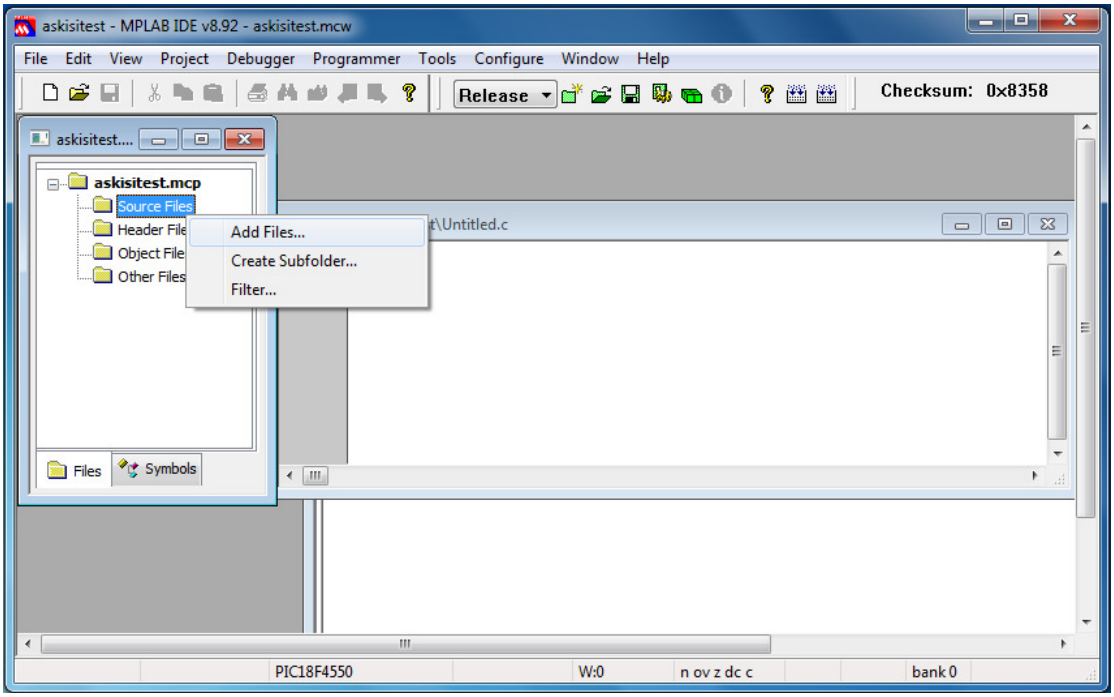
- Ρυθμίζουμε από το κεντρικό μενού την επιλογή **debug** από debug σε **release** και επιλέγουμε **configure → select device** από όπου επιλέγουμε από το μενού που εμφανίζεται τον μικροεπεξεργαστή που θέλουμε να χρησιμοποιήσουμε, στη δική μας περίπτωση τον **PIC18F4550** και οκ.



- Για την δημιουργία του πηγαίου κώδικα επιλέγουμε **file → new**. Η γλώσσα C θα γραφτεί στο νέο παράθυρο που ανοίγει με τίτλο untitled. Για να ονομάσουμε το αρχείο και να το αποθηκεύσουμε στον ίδιο φάκελο που αποθηκεύσαμε το project επιλέγουμε **file → save as** και επιλέγουμε το ίδιο μονοπάτι. Το όνομα του αρχείου θα πρέπει να έχει κατάληξη .c.



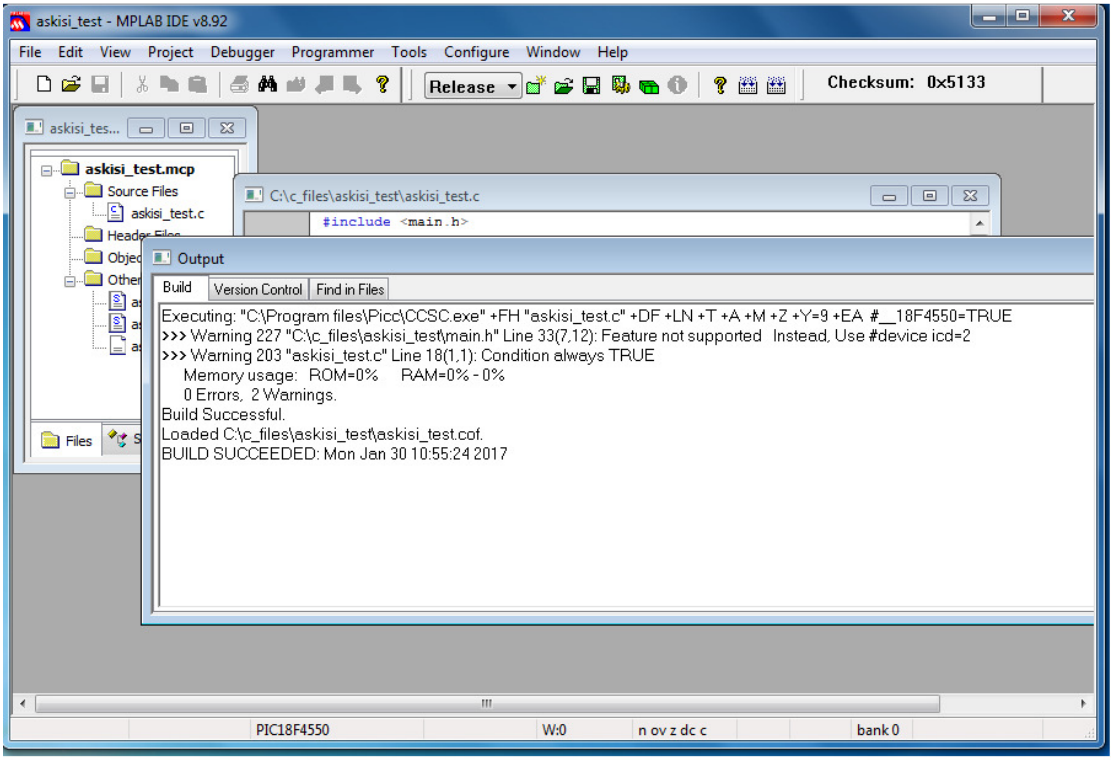
- Από το κυρίως μενού επιλέγουμε **project → select language toolsuite** και από το παράθυρο που ανοίγει επιλέγουμε **CCS C Compiler for pic10/12/14/16/18/24dsPIC30/dsPIC33.**
- Παρατηρούμε ότι στο όνομα του project έχει αστεράκι που σημαίνει ότι δεν έχει αποθηκευτεί οπότε από το κυρίως μενού επιλέγουμε **project → save project.**
- Για την προσθήκη του πηγαίου κώδικα στο project πατάμε δεξί κλικ πάνω στο **source files** και αντίστοιχα με δεξί κλικ επιλέγουμε **add files**



και εισάγουμε το αρχείο με κατάληξη .c.

- Στη συνέχεια γράφουμε τον κώδικα της εφαρμογής μας και ανά διαστήματα κάνουμε save καθώς και στο τέλος σε περίπτωση που δημιουργηθεί κάποιο πρόβλημα κατά την πορεία να υπάρχει το αρχείο αποθηκευμένο.
- Αφού ολοκληρωθεί ο κώδικας επιλέγουμε από το κυρίως μενού **project → compile**. Αυτό έχει ως αποτέλεσμα την μετατροπή του κώδικα από γλώσσα C σε γλώσσα μηχανής. Τα τρία προγράμματα που πραγματοποιούν την διαδικασία μετατροπής είναι ο **compiler**, ο **assembler** και ο **linker**. Ο compiler μετατρέπει τον κώδικα σε γλώσσα C για να είναι αναγνωρίσιμη από τον assembler, ο οποίος τον μεταφράζει σε γλώσσα μηχανής.
- Μετά την ολοκλήρωση της διαδικασίας εμφανίζονται διάφορα αρχεία. Ο linker λειτουργεί ως συνδετικός κρίκος για εξωτερικές βιβλιοθήκες και υποπρογράμματα που εμφανίζονται ως hex αρχείο για να περαστεί από το boot-loaders για να περαστεί στο chip. Το αρχείο ist είναι για την διόρθωση συντακτικών λαθών.

- Με το πέρας της διαδικασίας υπάρχει πιθανότητα να βγάλει κάποια error στο παράθυρο output και θα εμφανίσει **Built Failed**. Με διπλό κλικ πάνω στα error μας πετάει στην αντίστοιχη γραμμή που υπάρχει το λάθος.
- Αφού διορθωθούν τα λάθη ξανά κάνουμε compile και περιμένουμε να εμφανίσει **Built Succeeded**. Εάν δεν εμφανιστεί και βρεθούν κι άλλα error επαναλαμβάνουμε την διαδικασία.



2.2 Αναπτυξιακή πλακέτα PICkit 3

Η αναπτυξιακή πλακέτα PICkit3 είναι μια παραλλαγή της PICkit2 της Microchip με την ίδια μορφή αλλά σε διαφανή έκδοση. Ο επεξεργαστής της PIC24F είναι πιο γρήγορος των 16-bit, διαθέτει μεγαλύτερο εύρος ρύθμισης τάσης και έχει εσωτερικό διακόπτη τάσης λειτουργίας ώστε να λειτουργεί 2,5V – 5,5V περίπου στα 100mA. Για κάποιους μικροεπεξεργαστές PIC η τάση προγραμματισμού που μπορεί να παραχθεί είναι 13-14V για τον επαναπρογραμματισμό της μνήμης flash.

ΠΑΡΑΣΤΗΜΑ 3

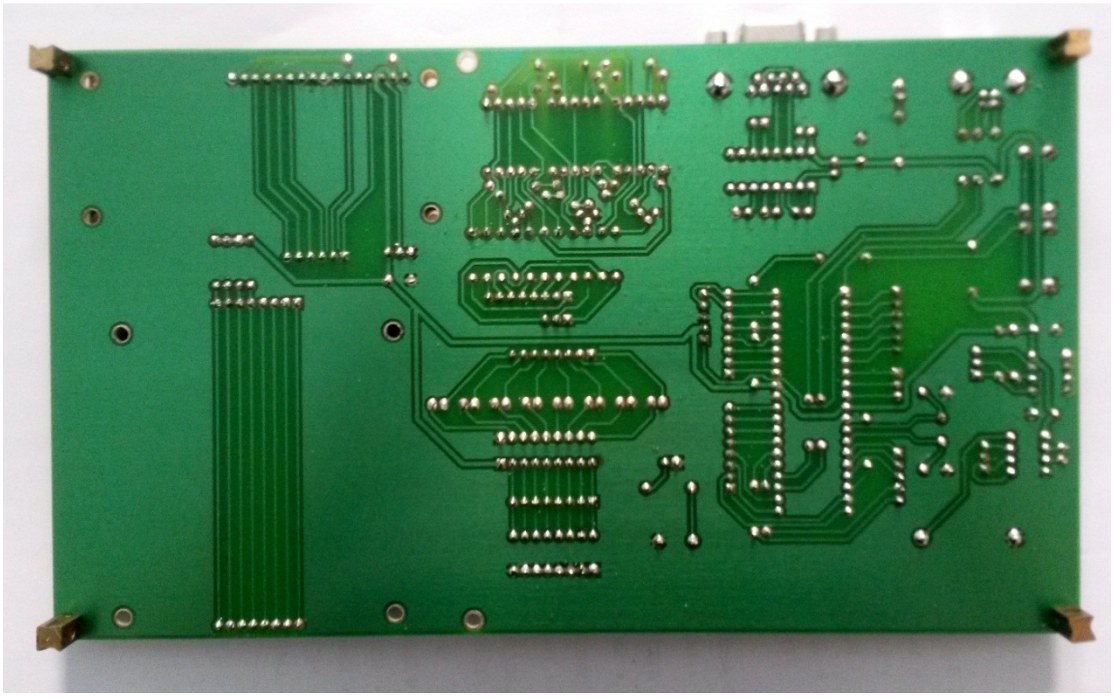
Πλακέτα του εργαστηρίου με τον PIC18F4550

3.α Στοιχεία της πλακέτας

Ο παρακάτω πίνακας αναφέρει ενδεικτικά τα πιο βασικά στοιχεία της πλακέτας για την πραγματοποίηση της πτυχιακής.

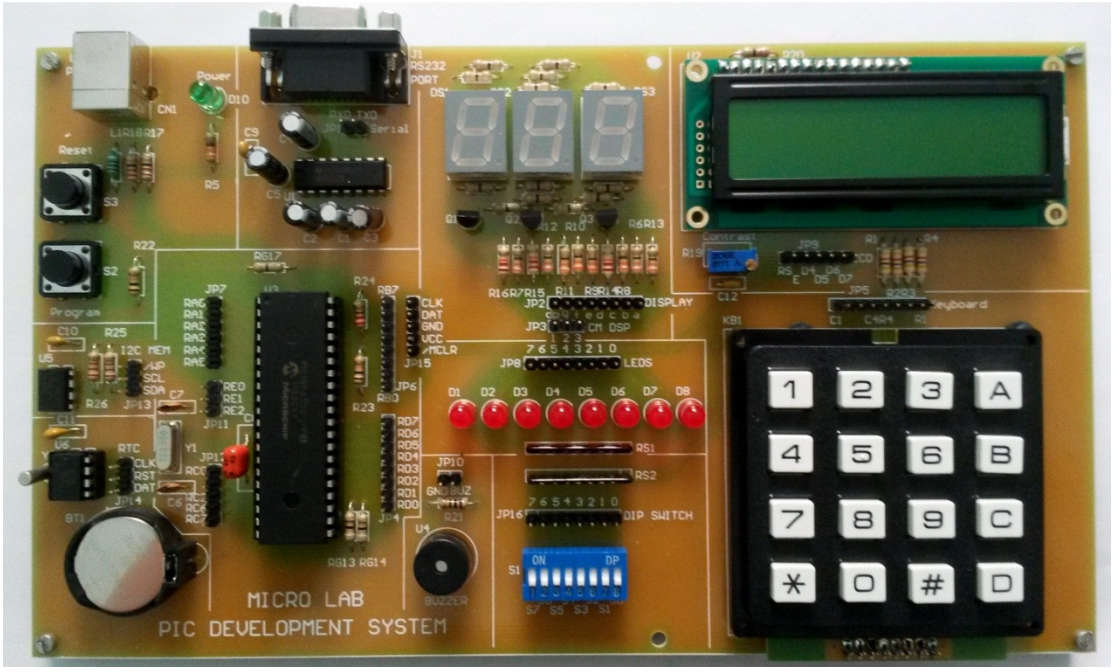
ΥΛΙΚΟ	ΟΝΟΜΑ	ΤΙΜΗ/ΤΥΠΟΣ	ΚΟΣΤΟΣ
Μπαταρία(lithium2032)	BT1	3V	0,80
Πυκνωτές (x5)	C1,C2,C3,C4,C5	10 uF/16V	0,45
Πυκνωτές (x2)	C6,C7	15pF	0,04
Πυκνωτής	C8	470nF	0,24
Πυκνωτής	C9,C10,C11,C12	100nF	0,56
Θύρα USB	CN1	CN-USB	0,40
LED (x8)	D1,D2,D3,D4,D4,D5,D6,D7,D8	-	0,63
LED (power)	D10	-	0,07
Σειριακή πόρτα	J1	RS232	0,45
Πληκτρολόγιο	KB1	Keypad 4x4 8pins	11,30
Τρανζίστορ (x3)	Q1,Q2,Q3	NPN - BC547	0,15
Αντιστάσεις (x4)	R1,R2,R3,R4,	4.7k	0,04
Ποτενσιόμετρο	R19	10 turns 1kOhm	0,40
Διακόπτες (x2)	S2,S3	Switch1.2mmx1.2mm	0,81
Πομποδέκτης	U1	RS-232 Transceivers	0,80
Οθόνη LCD	U2	LCD 2x16 ledbacklight	6,90
Μικροελεγκτής	U3	PIC18F4550	9,62
Βομβητής	U4	Buzzer with voltage 5V	1,00
Ολοκληρωμένο	U6	DS1302	4,50
Κρύσταλλος	Y1	20MHz	0,35
Κρύσταλλος	Y2	Crystal 32768Hz	0,30
Αντιστάσεις (x19)	RG1,…, RG19	0Ohm 1/4W	0,52

3.β Τυπωμένο κύκλωμα πλακέτας



Εικόνα Π3.2

3.γ Τελική κατασκευή πλακέτας



Εικόνα Π3.3

ΒΙΒΛΙΟΓΡΑΦΙΑ

- **Φύλλα δεδομένων**

  - PIC18F4550
  - KEYPAD 4x4
  - LCD 2x16

- **Βιβλία**

  - **Προγραμματίζοντας τον Μικροελεγκτή PIC**, Myke Predko, Εκδόσεις: Τζιόλα

  - **Εισαγωγή στους Μικροελεγκτές PICmicro®,** Δρ. Σταμάτης Αλατσαθιανός, Εκδόσεις: Σταμάτης Αλατσαθιανός

  - **Μικροϋπολογιστές – Μικροελεγκτές,** Δ. Πογαρίδης, Εκδόσεις Ίων

- **Διαδίκτυο**

  - http://www.microchip.com/

  - http://html.alldatasheet.com/html-pdf/93911/MICROCHIP/PIC18F4550/409/1/PIC18F4550.html

  - Σημειώσεις του μαθήματος Μικροελεγκτές ΙΙ από την διαδικτυακή φόρμα σημειώσεων e-μάθηση της Σχολής Ηλεκτρονικών Μηχανικών του ΑΤΕΙ Σίνδου.