ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Έλεγχος και απεικόνιση κατάστασης βιομηχανικού καυστήρα μέσω προγραμματιζόμενου λογικού ελεγκτή (PIC)

Του φοιτητή

Βενιώτη Δημήτριου

Αρ. Μητρώου : 052881

Επιβλέπων Καθηγητής

Διαμαντάρας Κωνσταντίνος

Θεσσαλονίκη 2012

ΠΡΟΛΟΓΟΣ

Η πτυχιακή εργασία πραγματοποιήθηκε στο Τμήμα Πληροφορικής του Αλεξάνδρειου Τεχνολογικού Εκπαιδευτικού Ιδρύματος Θεσσαλονίκης και αφορά την υλοποίηση ενός προγράμματος που θα βρίσκει εφαρμογή σε βιομηχανικούς καυστήρες λεβήτων βιομάζας. Στόχος της είναι η ανάγνωση των τιμών του ελεγκτή και η απεικόνιση της κατάστασης του καυστήρα σε γραφικό περιβάλλον στον υπολογιστή που θα βοηθάει τους χειριστές στο να εξοικονομούν χρόνο και χρήματα διότι δεν θα απαιτείται η μεταφορά του ελεγκτή σε κάποιο εργοστάσιο παραγωγής λεβήτων βιομάζας για να γίνουν οι κατάλληλες δοκιμές για την σωστή του λειτουργία επάνω σε αληθινούς λέβητες. Ένα ακόμη σημαντικό στοιχείο της είναι τα οικολογικά οφέλη διότι όπως αναφέρθηκε παραπάνω δεν θα γίνονται δοκιμές σε αληθινούς λέβητες για κάθε αλλαγή ή για κάθε σφάλμα που θα συμβαίνει στον ελεγκτή οπότε δεν θα μπαίνει σε λειτουργία ο λέβητας με αποτέλεσμα την μείωση των ρύπων.

Για να ολοκληρωθεί η εργασία αυτή απαιτήθηκε η εφαρμογή ειδικευμένων γνώσεων στον προγραμματισμό με χρήση της αντικειμενοστραφούς γλώσσας προγραμματισμού JAVA αλλά και η απόκτηση νέων γνώσεων απάνω στους μικροελεγκτές PIC και την γενικότερη φιλοσοφία που τους διέπει.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία εστιάζει στην απλούστευση της τεχνολογίας της διασύνδεσης ενός προγραμματιζόμενου μικροελεγκτή (PIC) με τον ηλεκτρονικό υπολογιστή. Η τεχνολογία αυτή για να γίνει όσο το δυνατόν καλύτερα αντιληπτή υλοποιήθηκε επάνω σε βιομηχανικούς καυστήρες βήμα προς βήμα.

Στην εισαγωγή δίνεται ο ορισμός του τι είναι ένας μικροελεγκτής PIC ενώ επίσης γίνεται και μια ιστορική αναδρομή για τον πυρήνα γύρω από τον οποίο δομείται ο επεξεργαστής PIC .

Το που μπορείς να συναντήσεις έναν μικροελεγκτή καθώς και το που μπορούν να βρουν αυτοί εφαρμογή παρουσιάζονται στο πρώτο κεφάλαιο. Στο ίδιο κεφάλαιο γίνεται μια αναφορά στις εταιρείες οι οποίες παράγουν μικροελεγκτές καθώς επίσης αναφέρονται και τα πλεονεκτήματα τους .

Στο δεύτερο κεφάλαιο γίνεται μια εκτεταμένη ανάλυση της δομής των μικροελεγκτών PIC με σκοπό την σταδιακή εμβάθυνση στον κόσμο τους και συγκεκριμένα αναλύεται η κεντρική μονάδα επεξεργασίας, η μνήμη, οι εντολές, η χρησιμότητα της λειτουργίας των διακοπών καθώς και οι περιφερειακές μονάδες. Απαραίτητη είναι όμως η αναφορά και η ανάλυση των διαφόρων γλωσσών προγραμματισμού των μικροελεγκτών PIC οι οποίες αναλύονται σε αυτό το κεφάλαιο.

Στο τρίτο κεφάλαιο παραθέτονται κάποια παραδείγματα εφαρμογών των μικροελεγκτών PIC όπως το άνοιγμα μιας πόρτας μέσω τηλεφώνου και γενικά ο τρόπος με τον οποίο χρησιμοποιείται ο πίνακας λεβήτων βιομάζας που χρησιμοποίησα και στην εργασία μου και οι διάφορες λειτουργίες που μπορεί να εκτελέσει με την βοήθεια του μικροελεγκτή PIC.

Τέλος στο τέταρτο κεφάλαιο, χρησιμοποιώντας τις γνώσεις που αποκτήθηκαν από τα προηγούμενα, παρουσιάζεται και αναλύεται το πρόγραμμα που παράχθηκε για την εφαρμογή του προγραμματιζόμενου λογικού ελεγκτή στους βιομηχανικούς καυστήρες.

Μια επισήμανση των σημαντικότερων σημείων της εργασίας αυτής γίνεται στο τμήμα των συμπερασμάτων, ως επίλογος του έργου.

ABSTRACT

The current thesis focuses in an attempt that is being made for the simplification of the style of the interface of a Peripheral Interface Controller (PIC) with a PC. This style was implemented on industrial boilers step by step so it could become as understandable as possible.

In the introduction there is the definition of the Peripheral Interface Controller and there is also given the historical background of the core that the PIC is based.

The first chapter states the uses and the devices where someone can find such a controller. In the same chapter there is a reference to the companies that produce them and there are mentioned some of the advantages.

In the second chapter there is a wide analysis of the structure of PICs in a purpose of gradually deepening into their world and the main components are being analyzed; in particular the CPU, the memory, the commands, the utility of the use of the interruptions and the peripheral units. The reference and the analysis of the different programming languages that are used to handle a PIC was crucial so it is a part of the same chapter.

The next chapter lists some examples of applications of PICs like the opening of a door through the telephone and spefically how the biomass boiler table that I used in my thesis is used and all of the different functions that it can perform with the help of this controller.

In conclusion, at the forth and last chapter, based on the knowledge that has been acquired from the previous chapters, is presented and analyzed the program that was written for the application of the PIC on the industrial boilers.

ΕΥΧΑΡΙΣΤΙΕΣ

Στα πλαίσια της εργασίας αυτής θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Κωνσταντίνο Διαμαντάρα για την καθοδήγηση του τόσο στο προγραμματιστικό στάδιο όσο και στο συγγραφικό μέρος της εργασίας αλλά και για την γενικότερη καθοδήγηση και υποστήριξη του καθ' όλη την διάρκεια διεκπεραίωσης της παρούσας πτυχιακής . Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου και όλους εκείνους με την συνεχή συμπαράσταση, την αγάπη και την κατανόηση που έδειξαν και με υποστήριξαν όλο αυτόν τον καιρό.

Πίνακας Περιεχομένων

ΠΙΝΑΚΑΣ ΕΥΡΕΣΗΣ ΕΙΚΟΝΩΝ

ΠΙΝΑΚΑΣ ΕΥΡΕΣΗΣ ΣΧΗΜΑΤΩΝ

ΠΙΝΑΚΑΣ ΕΥΡΕΣΗΣ ΠΙΝΑΚΩΝ

ΕΙΣΑΓΩΓΗ

Αν παρατηρήσει κανείς το φυλλάδιο τεχνικών οδηγιών ενός IBM προσωπικού υπολογιστή, και συγκεκριμένα το σχηματικό διάγραμμα της μονάδας πληκτρολογίου του συστήματος θα δει ότι αποτελείται από διακόπτες σε συστοιχία τύπου Μήτρας(MATRIX), οι οποίοι συνδέονται σε μια μονάδα σχεδιασμένη ως ένα απλό τετράγωνο που φέρνει την ετικέτα «I 8048», η οποία με την σειρά της συνδέεται στον υπολογιστή.

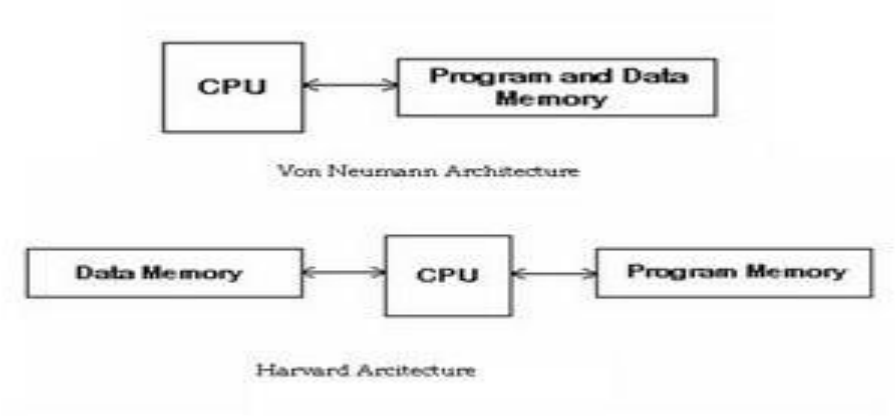
Το 8048 είναι ένα ολοκληρωμένο κύκλωμα των μικροελεγκτών της Intel στο οποίο περιέχεται ένας επεξεργαστής, μνήμη, έλεγχος διακοπών και σύστημα εισόδου-εξόδου, δηλαδή ένα ανάλογο μικροϋπολογιστικό σύστημα με αυτό του κυρίως τμήματος του υπολογιστή, το οποίο κατέχει είκοσι και πλέον σελίδες με σχηματικά διαγράμματα. Αυτά τα μικροσκοπικά αντικείμενα όσο παράξενο και να φαίνεται εκτελούν πάρα πολλές λειτουργίες.

Ο πυρήνας γύρω από τον οποίο δομείται ο επεξεργαστής των PIC, ανήκει σε μια αρχιτεκτονική γνωστή ως« αρχιτεκτονική Harvard». Στα μέσα περίπου της δεκαετίας του 1970 το υπουργείο εθνικής άμυνας των ΗΠΑ ζήτησε από τα πανεπιστήμια του Harvard και του Princeton να σχεδιάσουν έναν τύπο υπολογιστή με σκοπό τη σχεδίαση τροχιάς βλημάτων πυροβολικού.

Το Harvard παρουσίασε μια αρχιτεκτονική στα πλαίσια της οποίας το τμήμα μνήμης χωρίζεται σε δύο μέρη: το ένα χρησιμοποιείται για την αποθήκευση του προγράμματος, ενώ το δεύτερο χρησιμοποιείται για προσωρινή αποθήκευση μεταβλητών(RAM). Το πλεονέκτημα αυτής της μεθόδου ήταν ότι η ανάκληση των εντολών του προγράμματος από την μνήμη ελέγχου μπορεί να πραγματοποιηθεί ταυτόχρονα με την διαδικασία πρόσβασης στην RAM, επιτυγχάνοντας με τον τρόπο αυτό την εκτέλεση των εντολών.

Η αρχιτεκτονική που παρουσίασε το Princeton και που ανταγωνιζόταν την προηγούμενη, χρησιμοποιούσε την ίδια μνήμη ως μνήμη ελέγχου και ως μνήμη RAM και είναι γνωστή ως « αρχιτεκτονική Von Neumann».

Η ιδέα του διαχωρισμού του συστήματος μνήμης σε μνήμη ελέγχου και μνήμη RAM, της αρχιτεκτονικής του Harvard, έδινε ένα πολύ πιο γρήγορο σύστημα από αυτό του Von Neumann αλλά με ένα μειονέκτημα που ήταν το κόστος του. Έτσι η αρχιτεκτονική που προτάθηκε από τον Von Neumann του Princeton κέρδισε στον παραπάνω διαγωνισμό, ενώ κέρδισε και σημαντικό έδαφος αφού αποτέλεσε την βασική αρχιτεκτονική για τους περισσότερους τύπους υπολογιστών(όπως τον προσωπικό υπολογιστή του καθενός).



ΣΧΗΜΑ I: ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ HARVARD ΚΑΙ VON NEUMMAN

Πολλά χρόνια αργότερα μετά από αυτό τον διαγωνισμό, κάποιοι σχεδιαστές μηχανικοί της εταιρείας General Instruments, θέλοντας να σχεδιάσουν ένα δικό τους τύπο μικροελεγκτή, έφεραν ξανά στο φως την αρχιτεκτονική Harvard. Κρατώντας την μνήμη του συστήματος χωρισμένη σε δύο μέρη, το ένα για μνήμη ελέγχου και το άλλο για μνήμη RAM, ο σχεδιασμός του μικροελεγκτή πλεονεκτεί με την έννοια της *απλότητας* στην υλοποίηση αλλά και την *εξαιρετικά γρήγορη λειτουργία* που αποτελούν πολύ σημαντικούς παράγοντες για τους μικροελεγκτές.

Στην πραγματικότητα, οι διατάξεις στις οποίες χρησιμοποιείται η αρχιτεκτονική Harvard είναι ελαφρά φθηνότερες από τις αντίστοιχες της κλασικής αρχιτεκτονικής του Von Neumann επειδή στις πρώτες δεν είναι απαραίτητο να συνδυάζονται μαζί η μνήμη ROM με την μνήμη RAM.

Χρησιμοποιώντας την αρχιτεκτονική Harvard η σειρά των μικροελεγκτών PIC χαρακτηρίζεται από τη μεγάλη ταχύτητα λειτουργίας και από χαμηλό κόστος σε σχέση με άλλα ανταγωνιστικά προϊόντα ανάλογου τύπου. Λαμβάνοντας τα παραπάνω υπόψη οι μικροελεγκτές PIC(Peripheral Interface Controller) μπορούν να οριστούν ως ένα αυτόνομο υπολογιστικό σύστημα με πολύ μικρό μέγεθος, σε ένα και μοναδικό ολοκληρωμένο κύκλωμα (computer on chip).

ΚΕΦΑΛΑΙΟ 1

ΓΕΝΙΚΑ ΣΤΟΙΧΕΙΑ ΓΙΑ ΤΟΥΣ ΜΙΚΡΟΕΛΕΓΚΤΕΣ PIC

1.1 ΕΦΑΡΜΟΓΕΣ ΜΙΚΡΟΕΛΕΓΚΤΩΝ

Τους μικροελεγκτές μπορεί κανείς να τους συναντήσει παντού. Στο αυτοκίνητο, για παράδειγμα, υπάρχει ένας αριθμός από αυτούς που χρησιμοποιούνται για τον έλεγχο της μηχανής, των φρένων, του ραδιοφώνου, ακόμη και των καθισμάτων.

Ο εξοπλισμός της κουζίνας ενός σπιτιού σαν ένα άλλο παράδειγμα, περιέχει τόσα πολλά «έξυπνα» συστήματα που ακόμη και η χιουμοριστική εικόνα μιας τοστιέρας που «μιλάει» δεν απέχει πλέον και πολύ από την πραγματικότητα. Άλλα παραδείγματα καθημερινών συσκευών που ελέγχονται εν μέρει ή και πλήρως ακόμη από κάποιον μικροελεγκτή, είναι τα ηλεκτρονικά ρολόγια, οι φούρνοι μικροκυμάτων καθώς και τα ηλεκτρικά πλυντήρια.

Ένα άλλο παράδειγμα στο οποίο φαίνεται η εφαρμογή των μικροελεγκτών σε επιστημονικούς τομείς σε πειραματικό και ερευνητικό στάδιο είναι αυτό που συνέβη σε ένα Boeing 747 των Καναδικών αερογραμμών. Το αεροσκάφος προσγειώθηκε με τη βοήθεια τεχνητών εμποδίων διαδρόμου για αναγκαστική προσγείωση λόγω έλλειψης καυσίμων. Το σύστημα ελέγχου του αποτελούνταν από εξήντα εννιά περίπου μικροελεγκτές οι οποίοι σταμάτησαν να λειτουργούν ταυτόχρονα όταν σταμάτησαν οι γεννήτριες οδήγησης των μηχανών του.

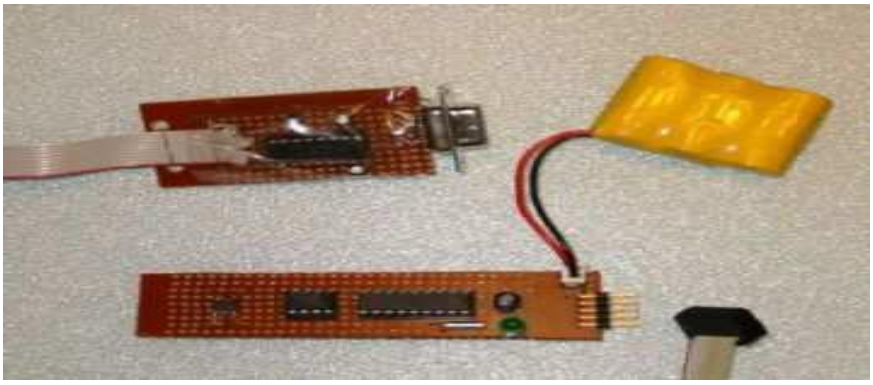
Βέβαια, μετά από αυτό, πιθανότατα να νομίζει κανείς ότι το σύστημα εκείνης της τοστιέρας να διαφέρει ριζικά από εκείνο ενός αεροσκάφους. Στην πραγματικότητα αυτό δεν ισχύει.

Παρακάτω ακολουθούν μερικά παραδείγματα εφαρμογών με PIC

Διπλό θερμόμετρο με σειριακή έξοδο, PIC16F876:
Χρησιμοποιήθηκε ένας PIC 16F876, ένας αισθητήρας θερμοκρασίας MCP1047A, ένας MCP1541για τάση αναφοράς, ένας τελεστικός ενισχυτής MCP6022A και ένα LCD display δυο γραμμών HD74780.

ΕΙΚΟΝΑ I: ΔΙΠΛΟ ΘΕΡΜΟΜΕΤΡΟ ΜΕ ΣΕΙΡΙΑΚΗ ΕΞΟΔΟ PIC16F876

Καταγραφέας θερμοκρασίας με PIC16F628A:
Χρησιμοποιεί PIC16F628A για επεξεργαστή TC-77 αισθητήρα θερμοκρασίας και 24LC64 για την αποθήκευση (μνήμη). Έχει χαμηλή κατανάλωση 400uA και τάση τροφοδοσίας 3.6V.



ΕΙΚΟΝΑ II: ΚΑΤΑΓΡΑΦΕΑΣ ΘΕΡΜΟΚΡΑΣΙΑΣ ΜΕ PIC16F628A

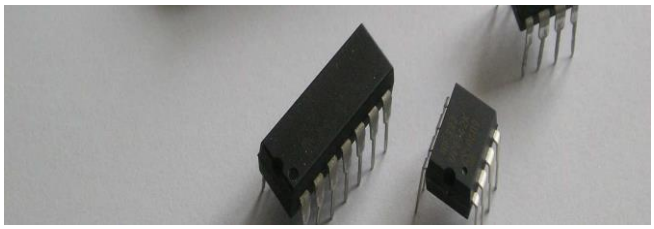
1.2 ΚΡΙΤΗΡΙΑ ΕΠΙΛΟΓΗΣ ΕΝΟΣ ΜΙΚΡΟΕΛΕΓΚΤΗ

Τα κριτήρια επιλογής ενός μικροελεγκτή για κάποια εφαρμογή, εξαρτώνται από τις απαιτήσεις και τις ιδιαιτερότητες της συγκεκριμένης εφαρμογής. Έτσι θα πρέπει πριν από την τελική εκλογή ενός μικροελεγκτή να συμπληρώνεται μια λίστα με τις απαιτήσεις του συστήματος από αυτόν. Με βάση αυτή την λίστα όπως επίσης και με την μελέτη των περιγραφών διαφόρων τύπων μικροελεγκτών, είναι σχετικά εύκολο να οδηγηθεί κανείς στην πιο εφικτή λύση.

Ωστόσο πριν από αυτό θα πρέπει να γίνει κατανοητό αρκετά καλά τι μπορούν να προσφέρουν οι μικροελεγκτές. Αυτό μπορεί να επιτευχθεί με την λεπτομερή διερεύνηση των εγχειριδίων τεχνικών δεδομένων που διατίθενται από τις διάφορες κατασκευαστικές εταιρίες, όπου θα βρούμε αμέτρητους τύπους μικροελεγκτών με πάρα πολλές και διαφορετικές ιδιότητες. Συγκρίνοντας αυτά που προσφέρει ο κάθε κατασκευαστής και μελετώντας προσεκτικά τις διάφορες σειρές μικροελεγκτών, διαπιστώνει κανείς ότι προκύπτουν αρκετοί με παρόμοιες ή ακόμη και με κοινές ιδιότητες.

Η εκλογή του τύπου που θα χρησιμοποιηθεί εξαρτάται από παράγοντες όπως το κόστος, η διάθεση του στο εμπόριο, η υποστήριξη που προσφέρει καθώς, βέβαια, και το επίπεδο εξοικείωσης του χρήστη με το συγκεκριμένο αντικείμενο.

Μια συγκεκριμένη οικογένεια μικροελεγκτών με αρκετά κοινά χαρακτηριστικά, σε γενικές γραμμές και σε σχέση με το σύνολο των άλλων μικροελεγκτών, η οποία επιπρόσθετα έχει και το πλεονέκτημα που επιτρέπει στους χρήστες να πειραματιστούν με τα διάφορα μέλη της σειράς της, εύκολα και με σχετικά χαμηλό κόστος είναι η οικογένεια των μικροελεγκτών της σειράς PIC.



ΕΙΚΟΝΑ ΙΙΙ: ΔΙΑΦΟΡΑ ΜΟΝΤΕΛΑ PIC

Οι ελεγκτές αυτοί περιέχουν έναν επεξεργαστή τύπου RISC (Reduced Instruction Set Computer η Υπολογιστής Μειωμένου Ρεπερτορίου Εντολών),μνήμη προγράμματος από 256 bytes έως και 16K bytes, δεκάδες έως και εκατοντάδες bytes μνήμης τύπου RAM και κάποιο αριθμό από bits (δυαδικά ψηφία) εισόδου-εξόδου, μερικά από τα οποία έχουν και εναλλακτικό τρόπο λειτουργίας και γενικά

υψηλές δυνατότητες. Όπως συμβαίνει με αρκετούς μικροελεγκτές έτσι και με τους συγκεκριμένους, ο λόγος που τους καθιστά ελκυστικούς για την χρήση τους, σε μια μεγάλη ποικιλία εφαρμογών, βρίσκεται στην ισχυρή σχεδίαση τους σε σχέση με κατανάλωση ισχύος, χρονισμό και συστήματος επανατοποθέτησης (RESET).

Κάθε κατασκευαστής ολοκληρωμένων επεξεργαστών διαθέτει και προσφέρει ανάλογες, επίσης, σειρές μικροελεγκτών. Στις διατάξεις αυτές των μικροελεγκτών, ενδέχεται να χρησιμοποιείται η ίδια αρχιτεκτονική με αυτή των μικροεπεξεργαστών ή και όχι. Για να μπορεί να επιλέξει κάποιος ένα συγκεκριμένο τύπο, κάθε ένας τύπος τέτοιων διατάξεων περιλαμβάνει διάφορους κωδικούς αριθμούς εξαρτημάτων, κάθε ένα από τα οποία έχει ελαφρώς διαφορετικά χαρακτηριστικά σε σύγκριση με τα υπόλοιπα.

Στα χαρακτηριστικά αυτά συμπεριλαμβάνονται:

- Προγραμματιζόμενες ψηφιακές είσοδοι-έξοδοι, παράλληλου τύπου.
- Προγραμματιζόμενες αναλογικές είσοδοι-έξοδοι.
- Σειριακές είσοδοι-έξοδοι (ασύγχρονες, σύγχρονες, και περιφερειακού ελέγχου).
- Έξοδοι παλμικών κυματομορφών για τον έλεγχο σερβοκινητήρων.
- Διακοπές που ενεργοποιούνται σε δεδομένη κατάσταση εισόδου.
- Χρονιστές καθώς και διακοπές, που προκαλούνται από συμβάντα των χρονιστών.
- Διατάξεις διασύνδεσης με εξωτερική μνήμη.
- Διασύνδεση με εξωτερικούς διαύλους(όπως PC ISA).
- Διάφοροι τύποι εσωτερικής μνήμης προγράμματος(ROM, EPROM, PROM και EEPROM).
- Διάφοροι τύποι εσωτερικής μνήμης τυχαίας προσπέλασης(RAM).
- Υπολογισμοί κινητής υποδιαστολής.

Η λίστα αυτή γίνεται μεγαλύτερη και πιο πολύπλοκη αν αναλογιστεί κανείς ότι υπάρχουν και άλλοι τύποι διατάξεων που μπορεί να θεωρηθούν ως μικροελεγκτές. Τέτοιες διατάξεις περιλαμβάνουν μικροϋπολογιστικά συστήματα στο ίδιο ολοκληρωμένο κύκλωμα καθώς και επεξεργαστές ψηφιακού σήματος.

Όλα αυτά σημαίνουν ότι υπάρχει μια καταιγιστική πληθώρα διαφόρων τύπων και διατάξεων μικροελεγκτών μέσα από την οποία επιλέγεται η κατάλληλη ανάλογα με την εφαρμογή του καθενός.

1.2.1 ΓΙΑΤΙ ΕΠΙΛΕΓΟΥΜΕ PIC

Όπως αναφέρθηκε στην προηγούμενη ενότητα είναι χρήσιμο να συμπληρώνει κάποιος μια λίστα με προδιαγραφές όταν πρόκειται να επιλέξει τον κατάλληλο τύπο μικροελεγκτή για μια δεδομένη εφαρμογή. Όταν επιλέγει λοιπόν έναν τύπο μικροελεγκτή για μια εφαρμογή , πρέπει να είναι βέβαιος ότι πληροί τις προδιαγραφές που έβαλε συμπεριλαμβανομένου και του παράγοντα του κόστους. Μπορεί για παράδειγμα να αντικαταστήσει τον τύπο που επέλεξε αρχικά με έναν άλλο τύπο που να περιέχει λιγότερο υλικό μέρος(Hardware) , με την προϋπόθεση να είναι διαθέσιμη μια φθηνότερη έκδοση του αρχικού τύπου και να υπάρχει η δυνατότητα προσομοίωσης του σχετικού υλικού που παραλείπεται, από αντίστοιχο λογισμικό(Software).

Οι παράμετροι που πρέπει να εξεταστούν και είναι αυτοί που μας οδηγούν στην επιλογή των μικροελεγκτών PIC είναι οι ακόλουθοι:

- Απλές απαιτήσεις τροφοδοσίας, χρονισμού και συστήματος επανατοποθέτησης(Reset).
- Αναπτυξιακά εργαλεία λογισμικού, χαμηλού κόστους.
- Ικανοποιητικά μεγάλη βάση πρόσβασης από τη πλευρά του χρήστη.
- Παρακαταθήκη ετοιμοπαράδοτων υλικών.
- Ευκολία προγραμματισμού και χρήση ερασιτεχνικού εξοπλισμού.

Όλες αυτές οι παράμετροι είναι πολύ σημαντικές επειδή κανείς δεν θέλει να αποκτήσει κάτι που θα κοστίζει μια περιουσία μόνο για τους σκοπούς της έρευνας του, ούτε να στραφεί σε εξαρτήματα τα οποία θα ήταν πολύ δύσκολο να τα προμηθευτεί κάποιος.

Γι’ αυτό η επιλογή των μικροελεγκτών PIC είναι πολύ καλή γιατί προκαλεί εντύπωση η ποικιλία διατάξεων που διαθέτει με τις διάφορες χαρακτηριστικές ιδιότητες τους, η ικανοποιητικά μεγάλη περιοχή πρόσβασης από την πλευρά του χρήστη, καθώς και το επίπεδο υποστήριξης που παρέχει η εκάστοτε εταιρεία το οποίο κυμαίνεται από συστήματα προγραμματισμού και προσομοίωσης έως διάθεση δωρεάν αναπτυξιακών εργαλείων λογισμικού και μια ευρεία συλλογή από σημειώσεις και παρατηρήσεις πάνω σε διάφορες εφαρμογές.

Όλοι οι ελεγκτές PIC έχουν ένα κοινό πυρήνα. Αυτό σημαίνει ότι μεταξύ όλων των μελών της σειράς υπάρχει μεγάλος βαθμός συμβατότητας από την άποψη του λογισμικού και της εσωτερικής κυκλωματικής διασύνδεσης.

1.3 ΔΥΝΑΤΟΤΗΤΕΣ ΚΑΙ ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΜΙΚΡΟΕΛΕΓΚΤΗ

Σύμφωνα με όσα αναφέρθηκαν προηγουμένως λίγο πολύ έγινε κατανοητό πια είναι τα πλεονεκτήματα των μικροελεγκτών αλλά δεν είναι τα μοναδικά. Κάποια εξίσου σπουδαία πλεονεκτήματά τους είναι τα παρακάτω:

- Οι περισσότεροι μικροελεγκτές έχουν δυνατότητες πολλαπλών αναλογικό/ψηφιακών μετατροπών (ADC είσοδοι) για λήψη μετρήσεων από πολλών ειδών αισθητήρων που υπάρχουν στην αγορά, έλεγχο κινητήρων (συνεχούς και εναλλασσόμενου ρεύματος, βηματικούς και άλλα) με χρήση διαμόρφωσης εύρους παλμού(Pulse-Width Modulation PWM) και δυνατότητα προγραμματισμού κατευθυντών PID.

- Επιπλέον, οι συνηθισμένοι μικροελεγκτές μπορούν να συνδεθούν με όλους τους αισθητήρες και υπολογιστές οι οποίοι είναι συμβατοί και όχι μόνο, με χρήση διαδεδομένων πρωτοκόλλων επικοινωνίας όπως τα I2C, CAN, SPI, RF, BLUETOOTH και USB, ακόμα και σύνδεση με το διαδίκτυο μέσω ETHERNET.

- Για ένα μέσο μικροελεγκτή, η ταχύτητά τους ξεκινά από το 1 MIPS και μπορεί να φτάσει τα 100 MIPS, ταχύτητα αρκετά ικανοποιητική για απαιτητικές εφαρμογές όπως μετάδοση βίντεο συνεχούς ροής(video streaming),επεξεργασία εικόνας(image processing) και ψηφιακή επεξεργασία σήματος(DSP).

- Ένας μέσος μικροελεγκτής έχει σχεδίαση 8-bit, ενώ υπάρχουν και μικροελεγκτές με αρχιτεκτονική 32-bit, γεγονός που τους δίνει την ικανότητα να εκτελούν γρήγορα πράξεις σε πραγματικό χρόνο.

1.3.1 ΤΙ ΕΙΝΑΙ ΤΑ PLC ΚΑΙ ΠΟΙΕΣ ΟΙ ΔΙΑΦΟΡΕΣ ΑΠΟ ΤΟΥΣ PIC

Τους προγραμματιζόμενους λογικούς ελεγκτές PLC (Programmable Logic Controllers) μπορούμε να τους ορίσουμε ως ψηφιακές ηλεκτρονικές συσκευές οι οποίες χρησιμοποιούν μια προγραμματιζόμενη μνήμη για την αποθήκευση οδηγιών και ειδικές λειτουργίες όπως είναι η λογική , η ακολουθία, ο χρόνος και η αρίθμηση, για να ελέγξουν τις μηχανές και την διαδικασία (Εριφάκη Μαρία, Καραμούτα Πηνελόπη, 2005).

ΕΙΚΟΝΑ IV : ΓΕΝΙΚΗ ΑΠΟΨΗ PLC

Το σημαντικότερο πλεονέκτημα τους είναι η χρήση τους στην επιστήμη του αυτοματισμού, με την ευελιξία στις μετατροπές και οι πολλές τους δυνατότητες. Αυτό όμως δεν είναι το μόνο πλεονέκτημα των PLC και είναι ουσιώδης έστω η αναφορά των πιο σπουδαίων από αυτά. Έτσι γνωρίζουμε πως όσον αφορά τα PLC :

- Η ταχύτητα της διαδικασίας παραγωγής αγγίζει το μέγιστο, και κατά συνέπεια μειώνεται στο ελάχιστο ο χρόνος απόσβεσης της εγκατάστασης.
- Η τοποθέτηση τους είναι ουσιαστικά ακίνδυνη ακόμη και μέσα σε πεδία ισχύος.
- Ο αυτοματισμός μπορεί να αλλάξει πολύ εύκολα είτε βρίσκεται στο στάδιο της μελέτης και της κατασκευής, είτε ακόμη και αφού έχει τεθεί σε λειτουργία.
- Είναι δυνατή αλλά και επιθυμητή η σύνδεση του με περιφερειακές μονάδες για τον έλεγχο του υλικού. Επίσης, μπορεί να συνδεθεί με ηλεκτρονικό υπολογιστή για ανταλλαγή στοιχείων.
- Εξοικονομούν χώρο, προσπάθεια όσον αφορά τη συντήρηση (από τη στιγμή που δεν υπάρχουν μηχανικές επαφές) και ενέργεια.
- Τα PLC είναι συμβατά με υψηλά ηλεκτρομαγνητικά πεδία, έχουν καλή αντοχή σε κραδασμούς, και λειτουργούν σε ακραίες θερμοκρασίες 0^{o} έως $60^{o}C$ χωρίς την ανάγκη προσθήκης επιπλέον ανεμιστήρων.

Παρόλα αυτά οι σύγχρονοι μικροελεγκτές μπορούν επάξια να αντικαταστήσουν σε ένα μεγάλο ποσοστό εργασιών τους υπολογιστές, τις ακριβείς κάρτες οδήγησης- συλλογής δεδομένων και τα PLC για τους εξής λόγους:

- Είναι σαφώς φθηνότεροι, επομένως εάν καταστραφούν δεν δημιουργούν πρόβλημα.
- Καταλαμβάνουν μικρότερο όγκο.
- Δεν απαιτούν εξειδικευμένες γνώσεις όπως προγραμματισμό σε διαγράμματα ladder στα PLC, αλλά μόνο γενικές γνώσεις προγραμματισμού.
- Δεν έχουμε κανένα περιορισμό όσον αφορά το είδος του προγράμματος, σε αντίθεση με τα PLC ή με ακριβές κάρτες οδήγησης που έχουν συγκεκριμένες προ-πληρωμένες δυνατότητες όπως μορφές ελέγχου από τις οποίες ο χρήστης επιλέγει.
- Για την επίτευξη ελέγχου ή μετρήσεων σε πραγματικό χρόνο, οι ηλεκτρονικοί υπολογιστές πρέπει να τρέχουν λειτουργικά συστήματα πραγματικού χρόνου (όπως RTLinux, QNX κ.ά.), ενώ οι μικροελεγκτές δεν απαιτούν λογισμικό.
- Πολλή μικρή κατανάλωση ισχύος.

Τέλος οι υπολογιστές όσο και τα PLC έχουν μικρή ενεργειακή αυτονομία σε σχέση με τους μικροελεγκτές. Έτσι για παράδειγμα ένα laptop μπορεί να εργαστεί για 2 ώρες χωρίς ρεύμα από το δίκτυο, ενώ ένας μικροελεγκτής μπορεί να εργάζεται για μέρες. Υπάρχουν και μικροελεγκτές χαμηλής κατανάλωσης οι οποίοι λειτουργούν με τάση κάτω των 2v.

1.4 ΕΤΑΙΡΙΕΣ ΠΑΡΑΓΩΓΗΣ ΜΙΚΡΟΕΛΕΓΚΤΩΝ

Μία από τις πρώτες εταιρείες, η οποία σχεδίασε μικροελεγκτές με πολλά περιφερειακά ενσωματωμένα σε ένα μόνο chip, ήταν η Intel με τη σειρά 8051 που χρησιμοποιείται ακόμα και σήμερα στο σχεδιασμό νέων προϊόντων.

Άλλες εταιρείες που ακολούθησαν και τροφοδοτούν σήμερα την παγκόσμια αγορά με μικροελεγκτές είναι κυρίως οι: Atmel, Microchip, Motorola, Hitachi, Toshiba, AMD, Zilog, National Semiconductor, Philips/Signetics. Από αυτές το μεγαλύτερο μερίδιο της αγοράς έχουν οι τρεις πρώτες μαζί με την Intel.

Από αυτές η Atmel φαίνεται να έχει επικρατήσει στη βιομηχανία λόγω της ύπαρξης δωρεάν εργαλείων και της γρήγορης υιοθέτησης μιας ειδικού τύπου μνήμης που ονομάζεται Flash.

Η Microchip έχει επικρατήσει ανάμεσα στους ≪χομπίστες≫ με τους φθηνούς μικροελεγκτές PIC και μετά το 2004, με τη διάθεσή της στην αγορά των μικροελεγκτών 16-bit dspic, έχει κερδίσει ένα μεγάλο κομμάτι της αγοράς.

Αρκετά γνωστοί και αξιόπιστοι είναι και οι μικροελεγκτές της εταιρείας Motorola, όπως τα μοντέλα MC68HC11 και M68HC12.

Παρόλο που οι μικροελεγκτές άλλων εταιρειών δεν υστερούν σε τίποτα, δεν είναι ευρέως γνωστοί καθώς χρησιμοποιούνται σε πιο εξειδικευμένες εφαρμογές.

Ακολουθεί μια αναλυτικότερη περιγραφή των σημαντικότερων και κορυφαίων εταιριών παραγωγής μικροελεγκτων.

1.4.1 Η ΕΤΑΙΡΕΙΑ ATMEL

Οι μικροελεγκτές AVR αναπτύχθηκαν από την Atmel για πρώτη φορά το 1996 και χρησιμοποιούν τροποποιημένη Αρχιτεκτονική Χάρβαρντ 8-bit RISC. Η AVR ήταν μια από τις οικογένειες μικροελεγκτών που έκαναν χρήση της on-chip μνήμης flash για την αποθήκευση του προγράμματος, σε αντίθεση με τα Programmable ROM, EPROM ή EEPROM που χρησιμοποιούνται από άλλους μικροελεγκτές.

Η βασική αρχιτεκτονική των AVR επινοήθηκε από δύο μαθητές στο Νορβηγικό Ινστιτούτο Τεχνολογίας τους Alf-Bogen EGIL και Vegard Wollan. Αργότερα η πατέντα για τους AVR μικροελεγκτές αγοράστηκε από την εταιρία ATMEL και η εσωτερική αρχιτεκτονική τους αναπτύχθηκε περαιτέρω. Η θυγατρική της ATMEL στην Νορβηγία ιδρύθηκε από τους δύο φοιτητές.

Το όνομα AVR δεν αποτελεί κάτι ιδιαίτερο όσον αφορά την ερμηνεία του. Απλά ονομάστηκε έτσι και ορίζει όλη την οικογένεια των μικροελεγκτών τύπου 8-bit RISC, ενώ σε αυτό το σημείο αξίζει να αναφερθεί ότι οι μικροελεγκτές διακρίνονται σε συστήματα των 8-, 16- ή 32-bit(ή 65-bit). Η εταιρία Atmel ισχυρίζεται ότι πρόκειται για μια απλή ονομασία. Ωστόσο ενδέχεται το ακρώνυμο AVR να αποτελείται από τα αρχικά των μηχανικών που σχεδίασαν τον συγκεκριμένο ελεγκτή.

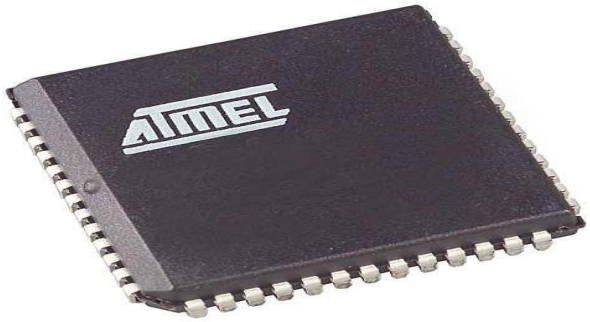
Από τις διάφορες κατηγορίες μικροελεγκτών, τη μεγαλύτερη γενικά απήχηση στη διεθνή αγορά παρουσιάζει εκείνη των 8-bit. Το έτος 1999 η αγορά της εν λόγω κατηγορίας ανήλθε στο συνολικό ποσό των 4.8 δισεκατομμυρίων δολαρίων, ενώ οι κατηγορίες των 16-bit και 32-bit έφθαναν την ίδια χρονιά μόλις το ποσό των 452.000.000 δολαρίων. Ωστόσο εκτός των παραπάνω διατάξεων υπάρχουν και μικροελεγκτές των 4-bit, για τους οποίους το ενδιαφέρον μάλλον βρίσκεται σε χαμηλά επίπεδα.

Οι μικροελεγκτές των 8-bit έχουν γίνει ιδιαίτερα δημοφιλείς σε σχέση με τους υπόλοιπους, όχι μόνο εξαιτίας του σχετικά χαμηλού κόστους, αλλά και εξαιτίας των αναπτυξιακών συστημάτων χαμηλού κόστους. Επίσης σημαντικό ρόλο παίζει και η διαθεσιμότητα των διατάξεων αυτών σε μια μεγάλη ποικιλία, από την άποψη των επιδόσεων και των διαθέσιμων ολοκληρωμένων περιφερειακών μονάδων.

Οι κυριότερες ιδιότητες του AVR της Atmel είναι οι ακόλουθες :

- Συνδυάζουν την αρχιτεκτονική Risc με ως επί το πλείστον σταθερού μήκους εντολές, διαδικασίες αποθήκευσης-φόρτωσης στη μνήμη και 32 καταχωρητές γενικής χρήσης.
- Μηχανισμό συνεχούς διοχέτευσης εντολών (instruction pipeline) σε δύο στάδια, που επιταχύνει σημαντικά τη διαδικασία εκτέλεσης.
- Οι περισσότερες από τις εντολές που περιλαμβάνει το ρεπερτόριό τους εκτελούνται στην διάρκεια μιας περιόδου του κεντρικού ρολογιού.
- Λειτουργούν σε συχνότητες χρονισμού μέχρι τα 10MHz.
- Διαθέτουν μεγάλη ποικιλία σε ότι αφορά ενσωματωμένες περιφερειακές μονάδες όπως, ψηφιακές εισόδους-εξόδους (I/O), μετατροπείς αναλογικού σήματος σε ψηφιακό ή ADC, μνήμη τύπου EEPROM, χρονιστές, μονάδες ασύγχρονης σειριακής επικοινωνίας ή UART (Universal Asynchronous Receiver Transmitter), ρολόγια πραγματικού χρόνου (RTC ή Real Time Clock), μονάδες διαμόρφωσης εύρους παλμών (PWM ή Pulse Width Modulation),και άλλα.
- Ενσωματωμένες μνήμες προγράμματος και δεδομένων.

- Δυνατότητα προγραμματισμού εντός του συστήματος (ISP ή In-System Programmable).
- Διατίθενται σε συσκευασίες των 8 έως 64 ακροδεκτών οπότε κρίνονται κατάλληλοι για έναν μεγάλο αριθμό διαφορετικών εφαρμογών.
- Είναι περίπου 12 φορές ταχύτεροι και αποδοτικότεροι σε σχέση με τους ελεγκτές κλασσικής αρχιτεκτονικής CISC (Complex Instruction Set Computer).
- Ευρεία περιοχή τάσεων λειτουργίας από 2.7 V έως 6.0 V.
- Η σχετικά απλή αρχιτεκτονική τους δίνει το πλεονέκτημα του μικρού σε απαίτηση κύκλου εκμάθησης στους αρχάριους.



ΕΙΚΟΝΑ IV : ΜΙΚΡΟΕΛΕΓΚΤΗΣ 8081 ΤΗΣ ATMEL

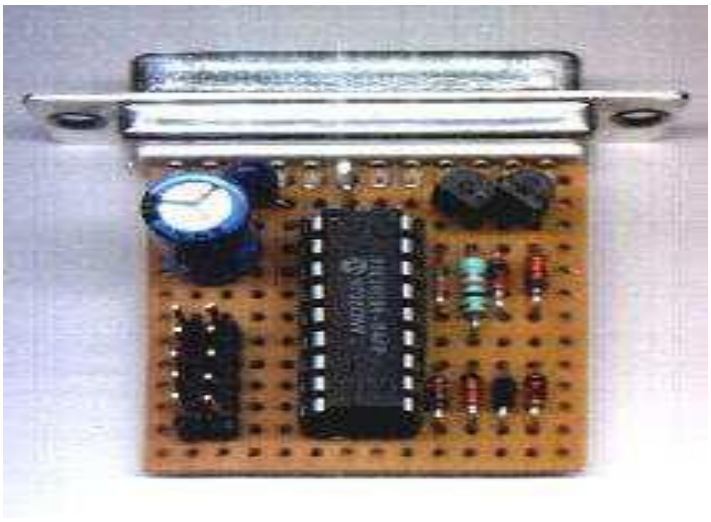
1.4.2 Η ΕΤΑΙΡΕΙΑ MICROCHIP

Οι PIC είναι ολοκληρωμένα κυκλώματα που παράγονται από την Microchip Technology Inc., και ανήκουν στην κατηγορία των μικροεπεξεργαστών. Μέσα τους περικλείουν όλα τα απαραίτητα στοιχεία για την κατασκευή ενός ψηφιακώς προγραμματιζόμενου συστήματος. Εξωτερικά μοιάζουν με ψηφιακά ολοκληρωμένα, όμως μέσα τους κρύβουν ένα μικρό υπολογιστή.

Καθ' όλη τη διάρκεια της δεκαετίας του '90 η σειρά των διαθέσιμων μικροελεγκτών PIC αυξήθηκε, με αποτέλεσμα ο PIC να ξεπεράσει καθιερωμένους ανταγωνιστές στον χώρο των μικροελεγκτών.

Αντίθετα από πολλούς ανταγωνιστές, η Microchip έκανε τα εργαλεία προγραμματισμού απλά, και φτηνά ή δωρεάν για διάθεση. Επιπλέον, έμεινε σταθερά στον κόσμο των 8-bit. Παρά την τεράστια πρόοδο που έχει γίνει, υπάρχουν ακόμα τα χαρακτηριστικά γνωρίσματα του παλαιού μικροελεγκτή της General Instruments, ακόμη και στα πιο πρόσφατα σχέδια.

Σήμερα, η Microchip προσφέρει πέντε κύριες οικογένειες μικροελεγκτών. Η σειρά του μοντέλου προσδιορίζεται από τα πρώτα δύο ψηφία του κωδικού της συσκευής. Ο αλφαβητικός χαρακτήρας που ακολουθεί δίνει κάποια ένδειξη της τεχνολογίας που χρησιμοποιήθηκε. Το γράμμα `C` απευθύνεται στην τεχνολογία CMOS, που είναι η κύρια τεχνολογία ημιαγωγών για εφαρμογή συστημάτων λογικής χαμηλής ισχύος. Το γράμμα `F` δείχνει την ενσωμάτωση της τεχνολογίας μνήμης flash (ακόμα χρησιμοποιώντας το CMOS ως βασική τεχνολογία). Ένα `A` μετά από τον αριθμό δείχνει μια τεχνολογική αναβάθμιση επάνω στο προηγούμενο μοντέλο. Για παράδειγμα, το 16C84 ήταν το πρώτο μοντέλο του είδους του. Στην συνέχεια επανεκδόθηκε ως 16F84, περιλαμβάνοντας την τεχνολογία μνήμης flash. Αργότερα, επανεκδόθηκε ως 16F84A, με περαιτέρω τεχνολογικές αναβαθμίσεις.



ΕΙΚΟΝΑ V: ΜΙΚΡΟΕΛΕΓΚΤΗΣ 16C84 ΤΗΣ MICROCHIP ΕΠΑΝΩ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΤΗ ΤΟΥ

Η Microchip συνηθίζει να δίνει ένα όνομα σε κάθε κατηγορία μικροελεγκτών. Κατά συνέπεια, η πρώτη οικογένειά τους, το 16C5XX, ονομάστηκε "baseline". Η ανάπτυξη αυτού, με τους αριθμούς συσκευών που αρχίζουν με `16C' ή 16F'ονομαστηκε "mid-range". Η ισχυρή εξέλιξη αυτού, με αριθμούς συσκευών που αρχίζουν από `17C', ονομάστηκε "high-end". Δεδομένου ότι αναπτύχθηκαν περαιτέρω μοντέλα και κατηγορίες, με πολύ απλές ή και προηγμένες αρχιτεκτονικές, πλέον αυτή η ορολογία έχει χάσει το νόημα της, αν και χρησιμοποιείται ακόμα.

1.4.3 Η ΕΤΑΙΡΕΙΑ INTEL

Το 1976 η Intel παρουσίασε την οικογένεια MCS -48 με τους πρώτους μικροελεγκτές 8048,8748 και 8035 που περιλαμβάνουν μια CPU των 8 bits,μνήμη προγράμματος ROM 1 K η μνήμη EPROM, μνήμη δεδομένων RAM 64Kbytes, πόρτες εισόδου-εξόδου και χρονιστή/μετρητή των 8 bits.Όλα τα παραπάνω είναι ενσωματωμένα σε ένα chip των 40pins.

Τα πιο εξελιγμένα προϊόντα της Intel, επέκτειναν την αρχιτεκτονική του MCS-48 σε διάφορες κατευθύνσεις, όπως το 8049,8050 και 8039 που έχουν διπλάσια μνήμη πάνω στο chip και είναι κατά 83 γρηγορότερα. Το 8022 έχει πάνω στο chip έναν A/D Converter 8 bits και μπορεί να συνδεθεί κατευθείαν με αναλογικά αισθητήρια.

Σήμερα οι νέοι μικροελεγκτές υψηλής αποδοτικότητας, επεκτείνουν κατά πολύ τα πλεονεκτήματα της ολοκληρωμένης ηλεκτρονικής. Είναι η σειρά MCS-51 σε τεχνολογία HMOS η HMOS 2 και με τετραπλάσια μνήμη προγράμματος και διπλάσια μνήμη δεδομένων. Επιπλέον, νέες πόρτες εισόδου/εξόδου και δυνατότητες περιφερειακών συνδέσεων, αυξάνουν τον αριθμό των εφαρμογών που μπορούν να πραγματοποιήσουν και παράλληλα μειώνουν το ολικό κόστος, καθώς πολλές λειτουργίες γίνονται με ένα και μόνο chip.

Παράλληλα, η ταχύτητα λειτουργίας(μέχρι και 16 MHz) είναι ανάλογη με τη χρήση από δυόμιση μέχρι πέντε φορές μεγαλύτερη. Το πρώτο μέλος της οικογένειας ο 8051, θεωρείται ο πυρήνας για όλα τα άλλα μέλη και είναι στην παραγωγή από το 1982.

Η οικογένεια MCS-51 των μικροελεγκτών σε ένα ολοκληρωμένο της Intel αποτελείται από τα μέλη που φαίνονται παρακάτω:

ΠΙΝΑΚΑΣ Ι: ΜΕΛΗ ΟΙΚΟΓΕΝΕΙΑΣ MCS-51 ΜΙΚΡΟΕΛΕΓΚΤΩΝ ΤΗΣ INTEL

Part	On-Chip Program Memory	On-Chip Data Memory	
8031	None	128 Bytes	
80C31	None	128 Bytes	HCMOS
8051	4 K ROM	128 Bytes	HMOS
8051 AH	4 K ROM	128 Bytes	HMOS II
80C51	4 K ROM	128 Bytes	
8751	4 K EPROM	128 Bytes	
8032	None	256 Bytes	
8052	8 K	256 Bytes	
8052 AH	8 K ROM BASIC	256 Bytes	HMOS II
8752	8 K EPROM	256 Bytes	
80C51FA	None	256 Bytes	CHMOS
83C51FA	8 K ROM	256 Bytes	

Όπως φαίνεται πιο πάνω οι 805X διαθέτουν εσωτερική ROM ενώ οι 803X δεν διαθέτουν. Επίσης οι 80X1 διαθέτουν εσωτερική RAM 128 bytes και οι 80X2, 256 bytes.

Η Intel διαθέτη επίσης μικροελεγκτές βασισμένους στον 8051, ειδικούς για τηλεπικοινωνιακές εφαρμογές, όπως ο 8044 και ο 83C152. Ακόμα υπάρχει και η σειρά των 16μπιτων MCS-96.

Σημειωτέων ότι και το 8052 AH είναι προγραμματισμένο μέσω μάσκας. Τούτο σημαίνει ότι διατίθεται μόνο σε μεγάλες ποσότητες, επειδή η ενσωματωμένη μνήμη προγράμματος (ROM) μπορεί να γραφτεί μόνο από τον κατασκευαστή του chip. Πρόκειται ουσιαστικά για για ένα ισχυρό και γρήγορο μεταφραστή BASIC φορτωμένο στην ενσωματωμένη ROM 8K. Ειδικά αυτό το ολοκληρωμένο έχει ιδιαίτερο ενδιαφέρον για μεμονωμένες εφαρμογές. Το 8031 όμως διαφέρει από το 8051 κατά το ότι δεν έχει ενσωματωμένη ROM. Έναντι αυτού, παίρνει τις εντολές του από μια εξωτερική μνήμη και συνεπώς είναι ιδανικό για την εγγραφή και τον έλεγχο προγραμμάτων που προορίζονται για το 8051.

Για την οικογένεια MCS-51 της Intel υπάρχουν και διάφοροι DOS-based compilers σε γλώσσες υψηλού επιπέδου όπως ASM-51, PL/M-51, C-51 και PASCAL που παίρνουν τον πηγαίο κώδικα προγράμματος , τον μεταφράζουν και παράγουν ένα αντικειμενικό κώδικα 8051 (object module format) ικανό να χρησιμοποιηθεί από τις άλλες εφαρμογές .

ΕΙΚΟΝΑ VI: Ο ΠΡΩΤΟΣ 32-bit ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΗΣ ΤΗΣ INTEL

ΚΕΦΑΛΑΙΟ 2

ΔΟΜΗ ΚΑΙ ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΤΩΝ ΜΙΚΡΟΕΛΕΓΚΤΩΝ PIC

Στο προηγούμενο κεφάλαιο έγινε μια πρώτη επαφή με τους μικροελεγκτές PIC. Αναλύθηκε το που μπορεί να συναντήσει κανείς τους μικροελεγκτές, το που αυτοί βρίσκουν εφαρμογή αλλά και τα κριτήρια επιλογής ενός μικροελεγκτή για μία εφαρμογή, ενώ συγχρόνως έγινε και μια αναφορά στις εταιρείες παραγωγής τέτοιων μικροελεγκτών ενώ αναφέρθηκαν τα πλεονεκτήματα και οι δυνατότητες τους. Σε αυτό το κεφάλαιο, θα αναπτυχθεί η δομή των μικροελεγκτών καθώς επίσης και οι γλώσσες προγραμματισμού τους έτσι ώστε να γίνει πιο ξεκάθαρος ο λόγος της κυριάρχησης τους.

2.1 ΔΟΜΗ ΜΙΚΡΟΕΛΕΓΚΤΩΝ PIC

Η δομή του μικροελεγκτή PIC μπορεί να χωριστεί σε δύο μέρη, τον πυρήνα(core) και τις περιφερειακές μονάδες του (peripheral units). Ο πυρήνας του μικροελεγκτή αποτελείται από όλα εκείνα τα στοιχεία, τα οποία είναι απολύτως απαραίτητα για την λειτουργία του. Οι περιφερειακές μονάδες βρίσκονται ενσωματωμένες στον μικροελεγκτή και είναι αυτές που τον κάνουν να διαφέρει από έναν μικροεπεξεργαστή.

Στον πυρήνα του PIC ανήκουν:

- Κεντρική μονάδα επεξεργασίας
- Μνήμη
- Εντολές
- Λειτουργίες διακοπών

Εδώ, πρέπει να σημειωθεί ότι, λόγω της σημαντικότητας τους, έχουν συμπεριληφθεί και οι εντολές του πυρήνα του PIC παρόλο που πρόκειται για κάποιο λογικό παρά υλικό στοιχείο του μικροελεγκτή.

Στις περιφερειακές μονάδες ανήκουν:

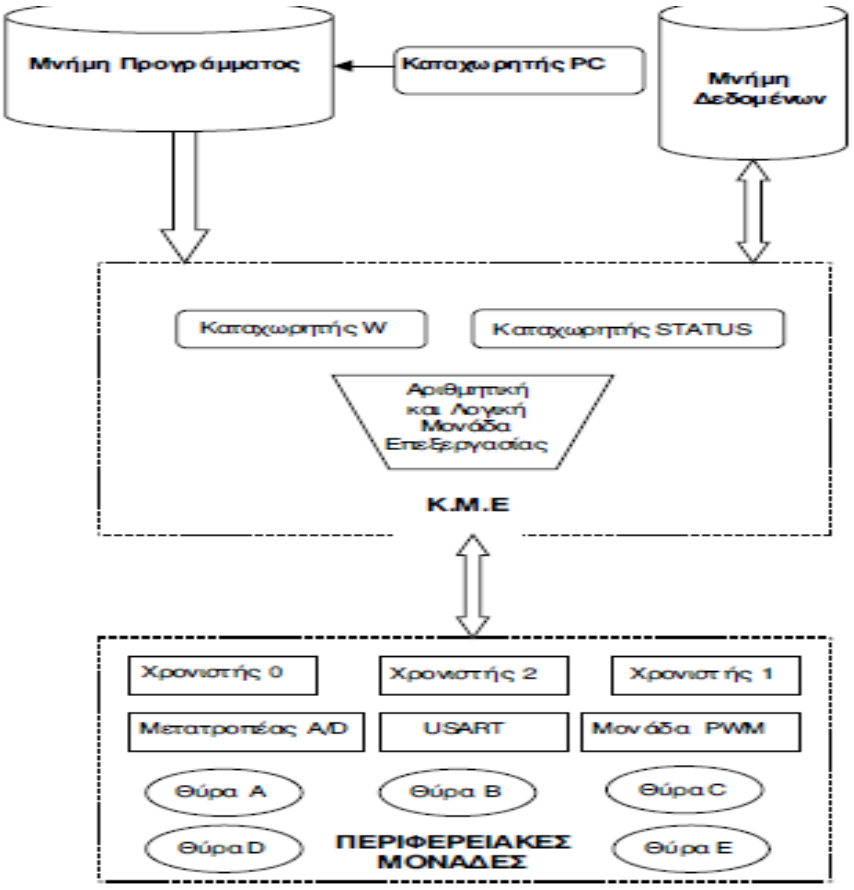
- Οι θύρες εισόδου/εξόδου γενικής χρήσης
- Οι μετρητές χρόνου(τρεις μονάδες)
- Η μονάδα διαμόρφωσης πλάτους
- Οι θύρες σειριακής επικοινωνίας(τρεις θύρες)
- Η θύρα παράλληλης επικοινωνίας
- Η μονάδα παραγωγής τάσης αναφοράς
- Οι συγκριτές
- Ο μετατροπέας αναλογικού σήματος σε ψηφιακό

Στη συνέχεια θα αναλύθούν τα παραπάνω στοιχεία και θα εξετασθεί το πως αυτά λειτουργούν και συνεργάζονται στις διάφορες εφαρμογές του PIC.

2.1.1 ΚΕΝΤΡΙΚΗ ΜΟΝΑΔΑ ΕΠΕΞΕΡΓΑΣΙΑΣ

Η κεντρική μονάδα επεξεργασίας γνωστή και ως CPU, εκτελεί τις εντολές του προγράμματος που έχουν αποθηκευτεί σε μια μνήμη η οποία ονομάζεται μνήμη προγράμματος. Από την μνήμη αυτή, η κεντρική μονάδα επεξεργασίας φέρνει, με τη σειρά τις εντολές του προγράμματος, τις αποκωδικοποιεί και τις εκτελεί. Εδώ πρέπει να σημειωθεί ότι ο PIC αναγνωρίζει τριάντα πέντε εντολές προγραμματισμού.

Μέσα στην κεντρική μονάδα επεξεργασίας, βρίσκεται και η αριθμητική και λογική μονάδα (Α.Λ.Μ). Το σχήμα 2 παρουσιάζει την κεντρική μονάδα επεξεργασίας μαζί με τα άμεσα με αυτήν, συνδεδεμένα στοιχεία του PIC. Οι αριθμητικές πράξεις που μπορεί να εκτελεί είναι η πρόσθεση και η αφαίρεση. Επίσης έχει την δυνατότητα να εκτελεί λογικές πράξεις (AND. OR, XOR). Η μονάδα επεξεργάζεται δεδομένα μήκους οκτώ δυαδικών ψηφίων (8-bit).

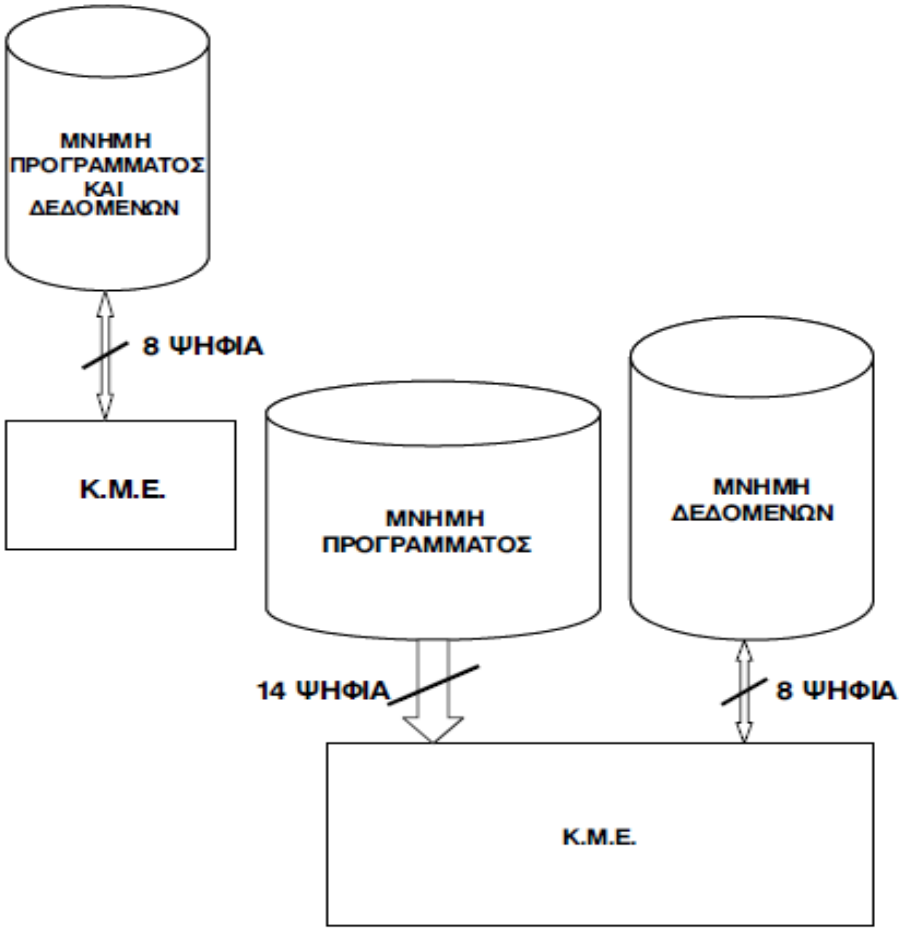


ΣΧΗΜΑ ΙΙ: Η ΔΟΜΗ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ PIC

Για την σωστή ανεύρεση των εντολών, που πρέπει να εκτελεστούν, η κεντρική μονάδα επεξεργασίας χρησιμοποιεί ένα μετρητή προγράμματος. Στην πραγματικότητα πρόκειται για έναν καταχωρητή, ο οποίος περιέχει την διεύθυνση της μνήμης στην οποία βρίσκεται αποθηκευμένη η εντολή που πρέπει να εκτελεστεί. Ο καταχωρητής αυτός ονομάζεται μετρητής προγράμματος (Program Counter ή PC).

2.1.2 ΜΝΗΜΗ

Στη σχεδίαση μικροελεγκτών και μικροεπεξεργαστών ακολουθούνται δύο αρχιτεκτονικές. Στην πρώτη χρησιμοποιείται μια μνήμη τόσο για την αποθήκευση του προγράμματος όσο και για την αποθήκευση των δεδομένων. Στη δεύτερη χρησιμοποιούνται δύο ξεχωριστές μνήμες. Η μία χρησιμοποιείται για την αποθήκευση του προγράμματος και λέγεται μνήμη προγράμματος ενώ η άλλη για την αποθήκευση των δεδομένων και λέγεται μνήμη δεδομένων. Οι δύο αρχιτεκτονικές παρουσιάζονται στο σχήμα ΙΙΙ.



ΣΧΗΜΑ ΙΙΙ: ΑΡΧΙΤΕΚΤΟΝΙΚΕΣ ΜΝΗΜΗΣ ΜΙΚΡΟΕΛΕΓΚΤΩΝ. (α) Ενιαία μνήμη προγράμματος δεδομένων. (β) Ξεχωριστή μνήμη προγράμματος και δεδομένων(μικροελεγκτής PIC)

Ο μικροελεγκτής PIC ακολουθεί την δεύτερη αρχιτεκτονική. Στην δεύτερη αρχιτεκτονική, εντολές και δεδομένα κινούνται σε ξεχωριστούς διαδρόμους(διαύλους) με αποτέλεσμα αυτό να μπορεί να γίνει όχι μόνο με πολύ μεγαλύτερη ταχύτητα αλλά και την ίδια χρονική στιγμή. Αντίθετα στην πρώτη αρχιτεκτονική, εντολές και δεδομένα μοιράζονται τον ίδιο διάδρομο με αποτέλεσμα να ελαττώνεται η ταχύτητα μεταφοράς τους. Επιπλέον, το πλεονέκτημα της

δεύτερης αρχιτεκτονικής, να χρησιμοποιεί ξεχωριστούς χώρους μνήμης για την αποθήκευση των δεδομένων και του προγράμματος δίνει την δυνατότητα χρησιμοποίησης μνήμης με διαφορετικό μήκος λέξης. Έτσι στην περίπτωση του PIC η μνήμη προγράμματος έχει μήκος λέξης δεκατεσσάρων δυαδικών ψηφίων(bits) αντί των οκτώ , της μνήμης των δεδομένων με σκοπό όλες οι εντολές να κωδικοποιούνται σε μια λέξη. Γενικά οι εντολές των μικροελεγκτών μπορεί να έχουν μήκος μιας, δύο ή, ακόμη, περισσότερων λέξεων, με αντίστοιχη βέβαια αύξηση του χρόνου εκτέλεσης.

Τα μέγεθος της μνήμης προγράμματος κυμαίνεται από 2 ως 8 Kbytes και συνήθως είναι τύπου Flash. Η συγκεκριμένη τεχνολογία επιτρέπει όχι μόνο την εγγραφή αλλά και το σβήσιμο της μνήμης να γίνεται με ηλεκτρικό τρόπο. Αυτό σημαίνει ότι ο προγραμματισμός του μικροελεγκτή γίνεται εύκολα ενώ αυτός βρίσκεται συνδεδεμένος στο κύκλωμα της εκάστοτε εφαρμογής.

Το μέγεθος της μνήμης δεδομένων αποτελείται από τρία τμήματα, με μέγεθος 128bytes το κάθε ένα, δηλαδή 384bytes συνολικά. Το κάθε τμήμα αποτελείται τόσο από καταχωρητές γενικού όσο και ειδικού σκοπού. Μερικοί από τους καταχωρητές ειδικού σκοπού χρησιμοποιούνται για τον έλεγχο του πυρήνα του PIC ενώ άλλοι για τον έλεγχο των περιφερειακών του.

2.1.3 ΕΝΤΟΛΕΣ

Όσον αφορά την σχεδίαση εντολών οι μικροελεγκτές ακολουθούν είτε την αρχιτεκτονική RISC είτε την αρχιτεκτονική CISC. Ενδεικτικά αναφέρω ότι οι μεγάλοι κεντρικοί υπολογιστές βασίζονται σε μικροεπεξεργαστές αρχιτεκτονικής RISC. Οι μικροελεγκτές αρχιτεκτονικής RISC χαρακτηρίζονται από το γεγονός ότι διαθέτουν μικρό αριθμό εντολών μπορούν όμως να ικανοποιήσουν όλες τις πιθανές απαιτήσεις προγραμματισμού.

Ο μικροελεγκτής PIC ακολουθεί την αρχιτεκτονική RISC και έχει συνολικά τριάντα πέντε εντολές, μήκους λέξης (14bit). Έτσι σε αντίθεση με τους μικροελεγκτές CISC, ο PIC εκτελεί την κάθε εντολή σε έναν κύκλο μηχανής με αποτέλεσμα την σημαντική βελτίωση της ταχύτητας επεξεργασίας.

2.1.4 ΛΕΙΤΟΥΡΓΙΕΣ ΔΙΑΚΟΠΩΝ

Την ώρα που ο PIC εκτελεί ένα πρόγραμμα που του έχουμε δώσει, μπορούν να συμβούν διάφορα γεγονότα στο περιβάλλον που αυτός ελέγχει. Ανάλογα με την σημασία του κάθε γεγονότος, ενδεχομένως, να χρειαστεί να διακόψει την εκτέλεση του κυρίως προγράμματος για να ασχοληθεί με το γεγονός αυτό. Ο PIC για να αντιληφθεί την ύπαρξη των γεγονότων αυτών έχει αναθέσει σε κάποιες από τις περιφερειακές του μονάδες να διενεργούν διάφορους ελέγχους. Αυτό μπορεί να γίνεται ανεξάρτητα από τις λοιπές λειτουργίες του μικροελεγκτή. Όταν μια από τις μονάδες εντοπίσει το γεγονός, τότε, στέλνει ένα σήμα στην κεντρική μονάδα επεξεργασίας του μικροελεγκτή να διακόψει την εκτέλεση του προγράμματος που εκτελεί. Το σήμα λέγεται διακοπή και προκαλεί την άμεση εκτέλεση ενός τμήματος κώδικα το οποία λέγεται ρουτίνα εξυπηρέτησης της διακοπής.

Σε αυτό το σημείο θα γίνει αναφορά σε ένα παράδειγμα για να γίνει κατανοητή η χρήση και η σημαντικότητα των διακοπών για τον έλεγχο μιας κατεργασίας μετάλλων σε ένα βιομηχανικό περιβάλλον. Ο PIC πρέπει να ρυθμίζει την θερμοκρασία σε κάποιον κλίβανο. Ακόμη για λόγους ασφαλείας, μετρά την θερμοκρασία σε διάφορα σημεία της κατεργασίας. Όταν εκτελείται μια συγκεκριμένη διαδικασία όπως η αποστολή κάποιων δεδομένων από την μνήμη του σε ένα κεντρικό ηλεκτρονικό υπολογιστή, διαπιστώνεται επικίνδυνη αύξηση της θερμοκρασίας σε κάποιο σημείο της κατεργασίας. Σε αυτό το σημείο το περιφερειακό που την εντόπισε, προκαλεί μια διακοπή στην κεντρική μονάδα επεξεργασίας. Τότε η αποστολή σταματά και η κεντρική μονάδα επεξεργασίας καλείται να εκτελέσει την ρουτίνα που συνοδεύει την διακοπή. Η ρουτίνα περιέχει εντολές που σβήνουν τον κλίβανο, ανοίγουν κάποιους ανεμιστήρες ψύξης και ενεργοποιούν την σειρήνα κινδύνου.

Τέλος θα πρέπει να τονιστεί ότι το γεγονός αναφέρεται σε μια κατάσταση κινδύνου αλλά αυτό δεν σημαίνει ότι μόνο τότε έχουμε διακοπές. Διακοπές συμβαίνουν και σε άλλες απλές περιπτώσεις. Ο PIC δέχεται ένα πλήθος διακοπών οι οποίες κατά κύριο λόγω, προέρχονται από τις διάφορες περιφερειακές του μονάδες. Ωστόσο υπάρχουν μονάδες που δίνουν περισσότερες διακοπές όπως για παράδειγμα το περιφερειακό της σειριακής επικοινωνίας. Υπάρχει η δυνατότητα όμως , αν είναι σκόπιμο να απενεργοποιηθούν κάποιες από τις διακοπές ή και όλες , όταν δηλαδή πρέπει να μην διακοπεί η εκτέλεση κάποιας εργασίας.

2.1.5 ΠΕΡΙΦΕΡΕΙΑΚΕΣ ΜΟΝΑΔΕΣ

Οι μικροελεγκτές είναι κατάλληλοι για πολλές εφαρμογές που το ζητούμενο τους είναι η χρήση περιφερειακών μονάδων όπως ο A/D μετατροπέας, η θύρα σειριακής επικοινωνίας και άλλα, απλούστατα διότι έχουν ενσωματωμένα διάφορα περιφερειακά. Μια τέτοια εφαρμογή μπορεί να είναι ο έλεγχος θερμοκρασίας νερού ενός βιομηχανικού λέβητα βιομάζας. Συνήθως ένας ελεγκτής επιλέγεται για μια εφαρμογή με βάση το είδος και τις δυνατότητες των περιφερειακών που διαθέτει. Έτσι και στην περίπτωση του ελεγκτή PIC έχει ενσωματωμένα αρκετά περιφερειακά που του δίνουν την δυνατότητα να επιλέγεται με ευκολία για πολλές εφαρμογές.

Ο μικροελεγκτής έχει πέντε θύρες εισόδου/εξόδου των οχτώ δυαδικών ψηφίων(bits) οι οποίες μπορούν να χρησιμοποιηθούν είτε σαν απλές θύρες είτε σαν θύρες των υπόλοιπων περιφερειακών που διαθέτει. Η τέταρτη και πέμπτη θύρα μπορεί να χρησιμοποιηθεί για παράλληλη επικοινωνία.

Επίσης διαθέτει τρεις μετρητές χρόνου, που του δίνουν ένα ακόμη πλεονέκτημα σε εφαρμογές όπου οι πολλαπλές μετρήσεις χρόνου είναι απαραίτητες. Ακόμη είναι εφικτή η παραγωγή παλμών ελεγχόμενης διάρκειας, κάτι που είναι πολύ χρήσιμο για την παραγωγή ρυθμιζόμενης συνεχούς τάσης.

Επιπροσθέτως ο μικροελεγκτής παρέχει έξοδο παλμοσειράς με ρυθμιζόμενο εύρος(PWM). Αυτή μπορεί να χρησιμοποιηθεί για την ρύθμιση διαφόρων βιομηχανικών συσκευών.

Ο μικροελεγκτής επίσης έχει την δυνατότητα της σειριακής επικοινωνίας. Διαθέτει δύο περιφερειακά. Το ένα για σύγχρονη και το άλλο για ασύγχρονη επικοινωνία. Τέλος διαθέτει ένα μετατροπέα αναλογικού σήματος σε ψηφιακό. Ο μετατροπέας αυτός είναι οχτώ καναλιών, που σημαίνει ότι έχει την δυνατότητα να λαμβάνει και μετατρέπει οχτώ διαφορετικά αναλογικά σήματα και κάθε κανάλι παράγει έναν αριθμό 8 bit.

2.2 ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΤΩΝ PIC

Μια οικογένεια μικροελεγκτών για να θεωρηθεί επιτυχής θα πρέπει σε μεγάλο βαθμό να διαθέτει ευχρηστία των σχετικών εργαλείων ανάπτυξης, όπως μεταφραστές από γλώσσες υψηλού επιπέδου σε γλώσσα κατανοητή από τον μικροελεγκτή (assembly), προγραμματιστές της εσωτερικής μνήμης και εργαλεία εκσφαλμάτωσης (debuggers). Στους μικροελεγκτές, τα εργαλεία αυτά δεν αποτελούνται ποτέ μόνο από λογισμικό, καθώς δεν υπάρχει τυποποιημένος τρόπος επικοινωνίας με αυτούς. Στον τομέα των εργαλείων ανάπτυξης, δραστηριοποιούνται όχι μόνο οι ίδιοι οι κατασκευαστές μικροελεγκτών αλλά και εξειδικευμένες εταιρείες οι οποίες διανέμουν το κατάλληλο λογισμικό για τον προγραμματισμό του κάθε τύπου μικροελεγκτή το οποίο θα αναλυθεί αρκετά στη συνέχεια για να γίνει κατανοητός ο τρόπος προγραμματισμού των PIC.

Η πιο διαδεδομένη γλώσσα προγραμματισμού των μικροελεγκτών είναι η C, η C++ και οι παραλλαγές τους. Σε τμήματα του λογισμικού όπου απαιτείται ταχύτητα ή μικρό μέγεθος χρησιμοποιούμενης μνήμης, μπορεί να χρησιμοποιείται η Assembly. Όμως οι μεγαλύτερες απαιτήσεις σε λειτουργικότητα και η ευκολία προγραμματισμού της C έναντι της assembly, σε συνδυασμό με την επάρκεια μνήμης των σύγχρονων μικροελεγκτών, έχουν γενικά εκτοπίσει την Assembly από τις περισσότερες εφαρμογές.

2.2.1 ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ASSEMBLY ΜΕ ΤΟ ΛΟΓΙΣΜΙΚΟ MPLAB

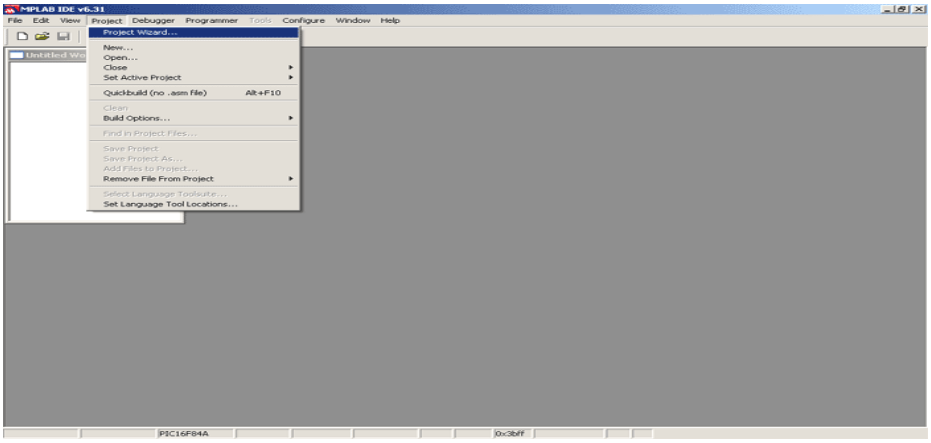
Το ***MPLAB*** είναι ένα ελεύθερο λογισμικό της εταιρείας MICROCIP που επιτρέπει την δημιουργία κώδικα assembly για τον προγραμματισμό των μικροελεγκτών PIC. Με την δημιουργία μιας τέτοιας εφαρμογής δημιουργούνται αρκετά αρχεία με τα σημαντικότερα από αυτά να είναι τα ακόλουθα:

- Αρχεία .ASM : είναι τα αρχεία που περιέχουν τον κώδικα του προγράμματος σε γλώσσα ASSEMBLY για τους μικροελεγκτές της MICROCHIP. Τα αρχεία αυτά δημιουργούνται με ένα πρόγραμμα συντάκτη (editor) και συνήθως χρησιμοποιείται ο ενσωματωμένος συντάκτης του MPLAB.

- Αρχεία .HEX: είναι τα αρχεία που περιέχουν τον κώδικα του προγράμματος σε γλώσσα μηχανής (δεκαεξαδική μορφή) μετά την συμβολομετάφραση τους από τον συμβολομεταφραστή (assembler) που εμπεριέχεται στο MPLAB. Αυτό πετυχαίνετε με την επιλογή Build All από το μενού Project και είναι απαραίτητα για την προσομοίωση της εκτέλεσης του προγράμματος καθώς και για την διαδικασία προγραμματισμού του μικροελεγκτή.

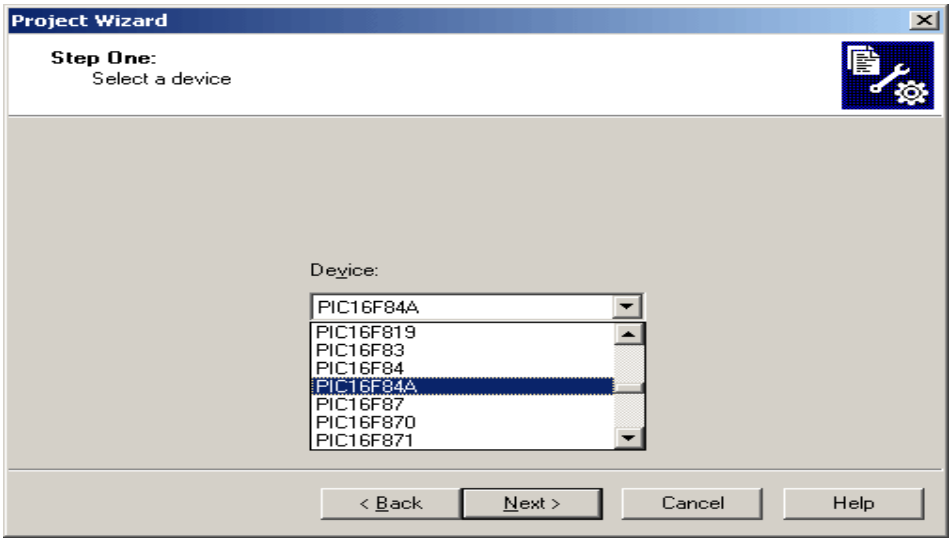
- Αρχεία .MCP: είναι τα αρχεία που περιέχουν τις πληροφορίες για την εφαρμογή που θα αναπτυχθεί στα οποία ενσωματώνονται τα αρχεία τύπου .ASM που περιέχουν τον κώδικα σε γλώσσα assembly.
- Αρχεία .MCW: είναι τα αρχεία που περιέχουν γενικές πληροφορίες όπως παραδείγματος χάρη ο τύπος του μικροελεγκτή.

Ακολουθεί μια σύντομη ανάλυση της χρήσης του παραπάνω λογισμικού.



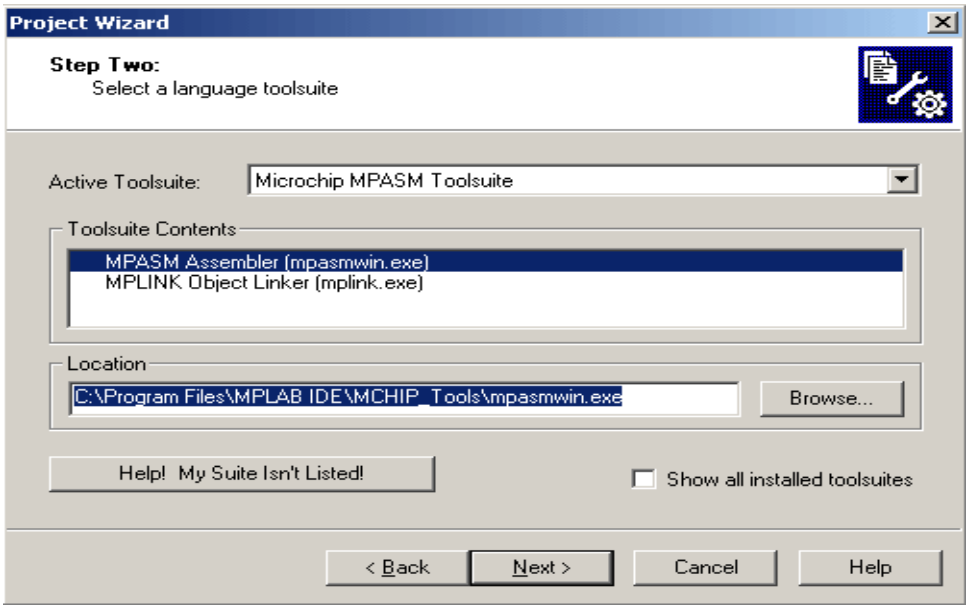
ΕΙΚΟΝΑ VII: ΠΑΡΑΘΥΡΟ ΕΝΑΡΞΗΣ PROJECT ΣΤΟ MPLAB

Μετά από αυτές τις ρυθμίσεις γίνεται η επιλογή του κατάλληλου μικροελεγκτή .



ΕΙΚΟΝΑ VIII: ΕΠΙΛΟΓΗ ΜΙΚΡΟΕΛΕΓΚΤΗ PIC

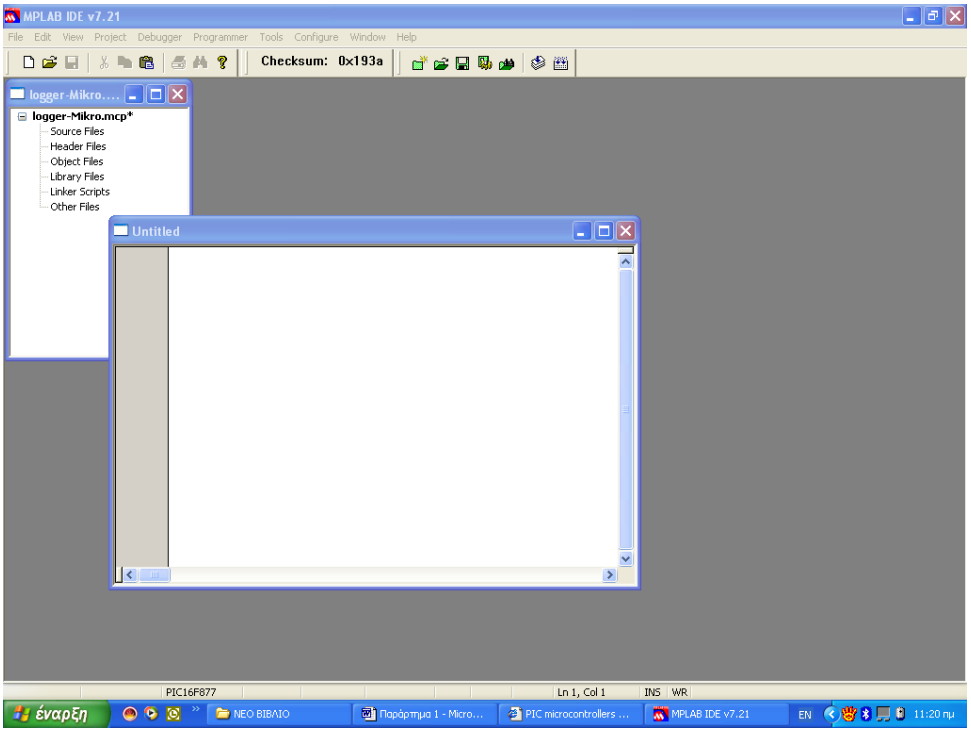
Στη συνέχεια επιλέγεται η γλώσσα assembly ως γλώσσα προγραμματισμού.



ΕΙΚΟΝΑ IX: ΕΠΙΛΟΓΗ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

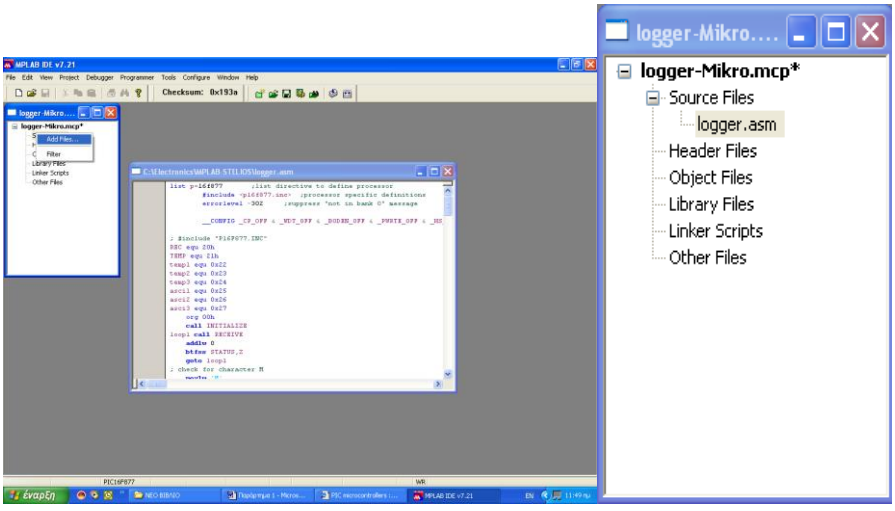
Ως τελευταία ενέργεια είναι να δοθεί ένα όνομα στο Project.

Στη συνέχεια για να γραφτεί ο κώδικας assembly πρέπει να δημιουργηθεί ένα αρχείο με την βοήθεια του συντάκτη (editor) του MPLAB.



ΕΙΚΟΝΑ Χ: ΠΑΡΑΘΥΡΟ ΓΙΑ ΤΗΝ ΣΥΓΓΡΑΦΗ ΤΟΥ ΚΩΔΙΚΑ

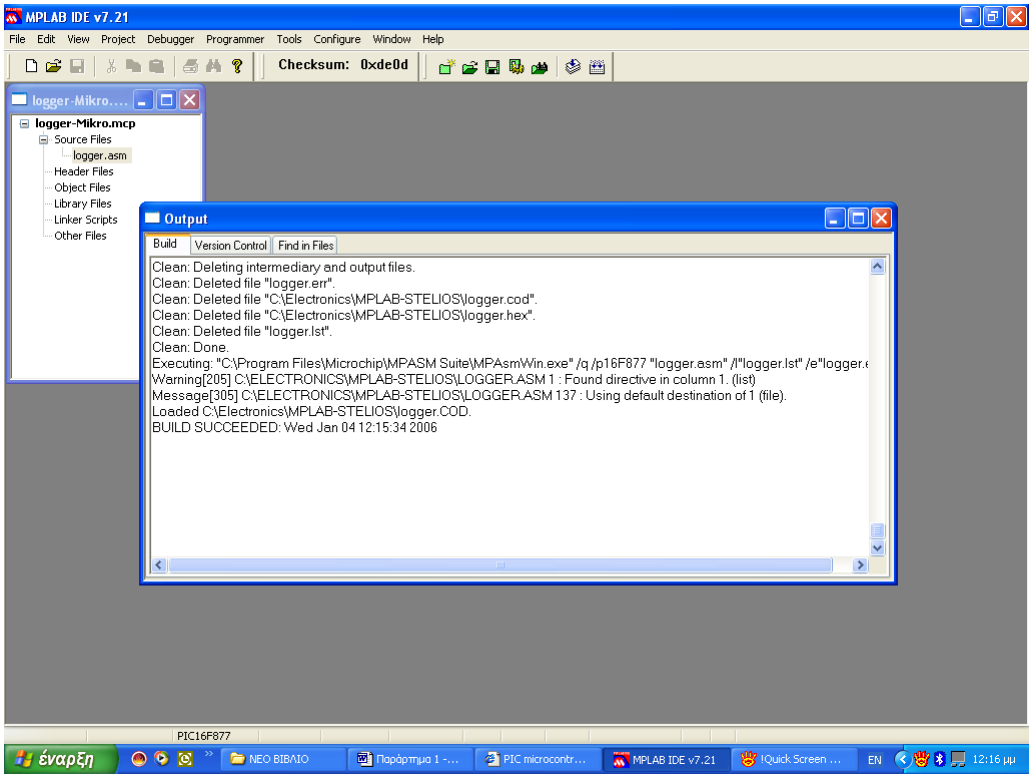
Αφού γραφτεί ο κώδικα πρέπει να γίνει μια ενημέρωση στο αρχείο έργου με τα αρχεία κώδικα που θα περιλαμβάνει.



ΕΙΚΟΝΑ ΧΙ: ΕΝΗΜΕΡΩΣΗ PROJECT ΜΕ ΤΟΝ ΚΩΔΙΚΑ ASSEMBLY

Το επόμενο βήμα που πρέπει να γίνει είναι η συμβολομετάφραση του κώδικα assembly σε γλώσσα μηχανής. Τα αποτελέσματα ελέγχου της συμβολομετάφρασης αλλά και τυχόν συντακτικά λάθη εμφανίζονται σε ένα παράθυρο εξόδου.

Από αυτή την διαδικασία της συμβολομετάφρασης, παράγεται ένα αρχείο που περιέχει τον κώδικα του προγράμματος σε γλώσσα μηχανής (δεκαεξαδική μορφή) και το οποίο μπορεί εύκολα κανείς να δει με την εφαρμογή Notepad των Windows. Το αρχείο αυτό είναι πολύ σημαντικό γιατί είναι το αρχείο με τον κώδικα μηχανής που θα φορτωθεί στο ολοκληρωμένο του μικροελεγκτή κατά την διαδικασία προγραμματισμού, δηλαδή όταν μεταφερθεί ο κώδικας μηχανής από τον προσωπικό υπολογιστή στη μνήμη προγράμματος του μικροελεγκτή.

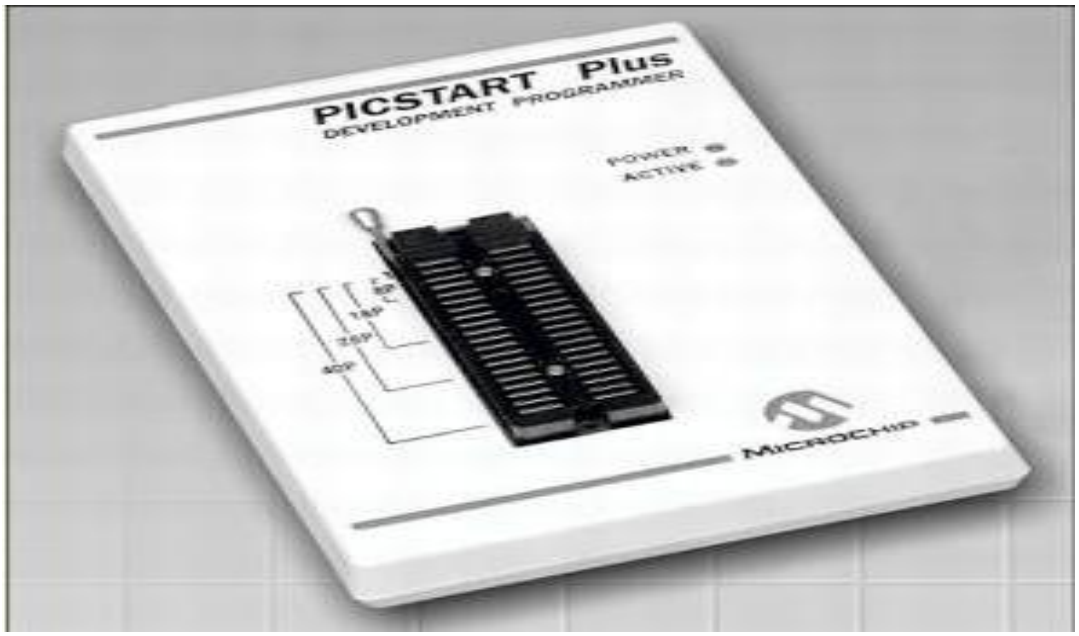


ΕΙΚΟΝΑ XII: ΑΠΟΤΕΛΕΣΜΑΤΑ ΣΥΜΒΟΛΟΜΕΤΑΦΡΑΣΗΣ ΤΟΥ ΚΩΔΙΚΑ ASSEMBLY

Στη συνέχεια της διαδικασίας αφού έχει δημιουργηθεί το αρχείο που περιέχει τον κώδικα του προγράμματος σε γλώσσα μηχανής μεταφέρεται στην μνήμη προγράμματος του μικροελεγκτή. Συνήθως, η εκτέλεση του προγράμματος αρχίζει με την τροφοδότηση του ολοκληρωμένου του μικροελεγκτή. Από εκεί και πέρα ο προγραμματισμός μπορεί να γίνει με διάφορες μονάδες προγραμματισμού οι οποίες συνδέονται στον Η/Υ είτε μέσω της σειριακής, είτε μέσω της παράλληλης διασύνδεσης είτε τέλος μέσω της διασύνδεσης USB οι οποίες αναλύονται παρακάτω.

Προγραμματισμός μέσω σειριακής με την μονάδα PICSTART Plus:

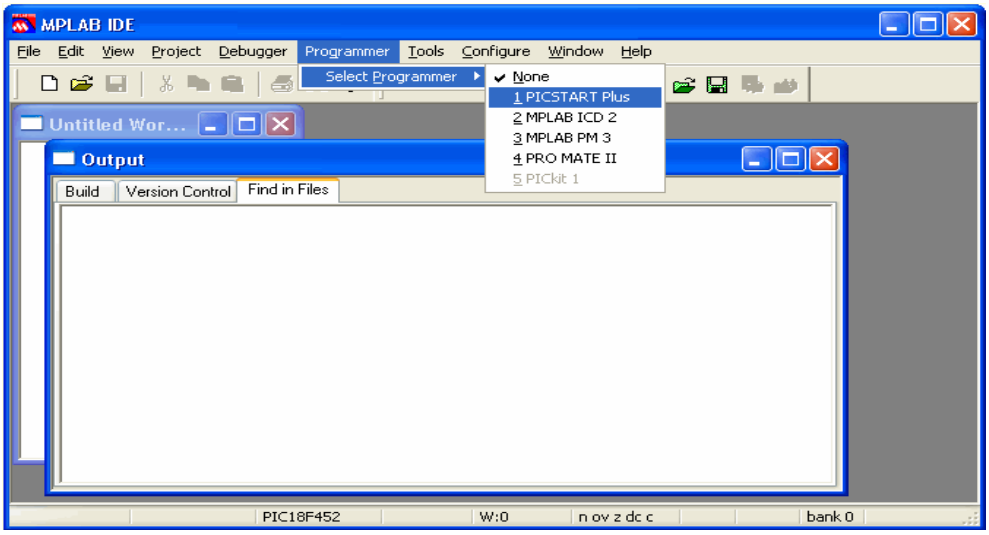
Η μεταφορά του κώδικα στην μονάδα αυτή γίνεται με ένα καλώδιο διασύνδεσης RS232 που μπορεί να συνδεθεί στην σειριακή θύρα ενός προσωπικού Η/Υ.



ΕΙΚΟΝΑ XIII: ΜΟΝΑΔΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ PICSTART Plus

Κατά την σύνδεση της ηλεκτρικής τροφοδοσίας (τροφοδοτικό 9 Volts) και την σύνδεση του σειριακού καλωδίου δεν πρέπει να έχει τοποθετηθεί το ολοκληρωμένο του μικροελεγκτή στην αντίστοιχη βάση. Η τοποθέτηση γίνεται στην συνέχεια. Για τον προγραμματισμό, χρησιμοποιείται πάλι το λογισμικό MPALB στο οποίο προφανώς έχει επιλεγεί ήδη ο τύπος του μικροελεγκτή που θα χρησιμοποιηθεί.

Στη συνέχεια, από το μενού Programmer γίνεται η επιλογή του προγραμματιστή PICSTART Plus και στη συνέχεια ενημερώνεται για την σειριακή θύρα του προσωπικού Η/Υ η οποία έχει πλέον συνδεθεί με τη μονάδα προγραμματισμού. Τα περιεχόμενα του μενού Programmer αλλάζουν, οπότε και επιλέγεται η ενεργοποίηση του προγραμματιστή από την επιλογή Enable Programmer.



ΕΙΚΟΝΑ XIV: ΕΠΙΛΟΓΗ PICSTART Plus

Για την μεταφορά του κώδικα, τοποθετείται και ασφαλίζεται το ολοκληρωμένο του μικροελεγκτή στην αντίστοιχη βάση και μετά μπορεί να ακολουθήσει η παρακάτω διαδικασία:

1) Ελέγχεται αν η μνήμη του ολοκληρωμένου είναι κενή, με την επιλογή Blank Check All.

2) Σβήνεται η μνήμη στην περίπτωση που περιέχει ήδη κώδικα, με την επιλογή Erase Flash Device .

3) Έχοντας ήδη φορτωμένο στη μνήμη του προσωπικού Η/Υ τον κώδικα του προγράμματος σε hex μορφή , με την επιλογή Program αρχίζει να μεταφέρεται ο κώδικας από τον προσωπικό Η/Υ στην μνήμη προγράμματος του ολοκληρωμένου.

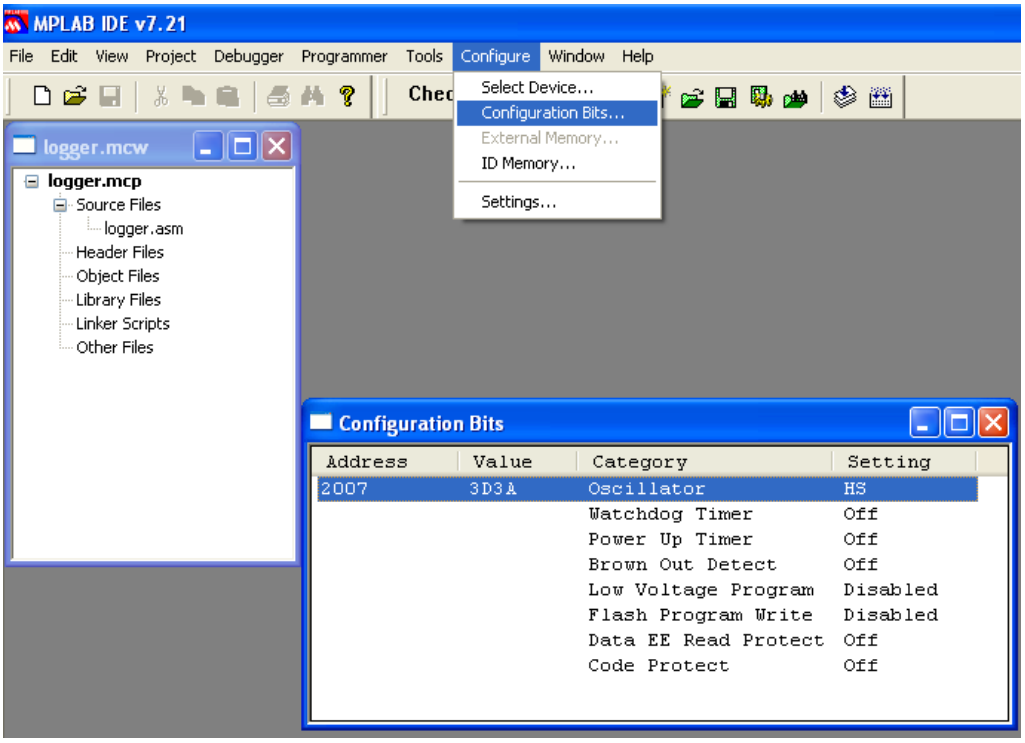
4) Εάν δεν βγάλει σχετικό μήνυμα ελέγχου και επιβεβαίωσης του κώδικα με τον οποίο προγραμματίστηκε ο μικροελεγκτής στο παράθυρο μηνυμάτων Output του MPLAB, θα πρέπει να ενεργοποιηθεί η επιλογή Verify μέχρι να ολοκληρωθεί ο έλεγχος για την σωστή μεταφορά του κώδικα. Τα δύο τελευταία βήματα ολοκληρώνονται με την ένδειξη Successfully στο παράθυρο μηνυμάτων Output του MPLAB.

5) Απασφαλίζεται το ολοκληρωμένο από την βάση προγραμματισμού και τοποθετείται στο ηλεκτρονικό κύκλωμα που έχει κατασκευαστεί. Το πρόγραμμα στον μικροελεγκτή θα αρχίσει να τρέχει με την έναρξη της τροφοδοσίας του.

Ο κώδικας hex φορτώνεται στην μνήμη του προσωπικού Η/Υ κατά την διαδικασία Build All για την συμβολομετάφραση του πηγαίου κώδικα που είναι σε μορφή assembly. Μπορεί επίσης να φορτωθεί απευθείας μέσω του αρχείου Hex από το μενού File.

Μία πολύ σημαντική λειτουργία για τον προγραμματισμό των μικροελεγκτών PIC είναι και η ρύθμιση των Configuartion Bits (εσωτερικές συνδέσεις). Η διαδικασία αυτή είναι πολύ σημαντική γιατί αφορά ιδιαίτερα σημαντικές πληροφορίες για την λειτουργία του μικροελεγκτή στο ηλεκτρονικό κύκλωμα της εφαρμογής . Στις πιο συνηθισμένες περιπτώσεις η μόνη τροποποίηση που πρέπει να γίνει είναι να επιλεγεί για την περίπτωση του ταλαντωτή (Oscillator) η επιλογή XT ή HS όταν στο κύκλωμα χρονισμού του μικροελεγκτή χρησιμοποιείται ένας κρύσταλλος.

Οι ρυθμίσεις των Configuration Bits πρέπει να γίνουν οπωσδήποτε πριν την διαδικασία προγραμματισμού του μικροελεγκτή και καθορίζονται από το μενού Configure / Configuration Bits.



ΕΙΚΟΝΑ XV: CONFIGURATION BITS ΓΙΑ ΤΟΝ ΜΙΚΡΟΕΛΕΓΚΤΗ

Ωστόσο, προτείνεται για την διαδικασία των ρυθμίσεων των Configuration Bits να χρησιμοποιούνται οι ειδικές εντολές αρχικών δηλώσεων που προβλέπονται για τον κώδικα assembly επειδή με τον τρόπο αυτό οι ρυθμίσεις ενσωματώνονται στο αρχείο hex και δεν χρειάζεται να επιλέγονται κάθε φορά που θα προγραμματίζεται το ολοκληρωμένο. Επίσης με τον τρόπο αυτό οι ρυθμίσεις διατηρούνται και στον προγραμματισμό των μικροελεγκτών με άλλες μονάδες προγραμματισμού, σαν και αυτές που θα παρουσιαστούν στην συνέχεια. Παρακάτω, φαίνεται το τμήμα με το οποίο ξεκινά ο κώδικας assembly μιας εφαρμογής που περιλαμβάνει τις ρυθμίσεις για τα Configuration Bits.

```
list p=16f877 ;                 list directive to define processor

#include <p16f877.inc>;   processor specific definitions

errorlevel -302;                suppress "not in bank 0" message

__CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_OFF & _HS_OSC & _WRT_ENABLE_OFF & _LVP_OFF & _DEBUG_OFF & _CPD_OFF
```

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕΣΩ ΤΗΣ ΔΙΑΣΥΝΔΕΣΗΣ USB :

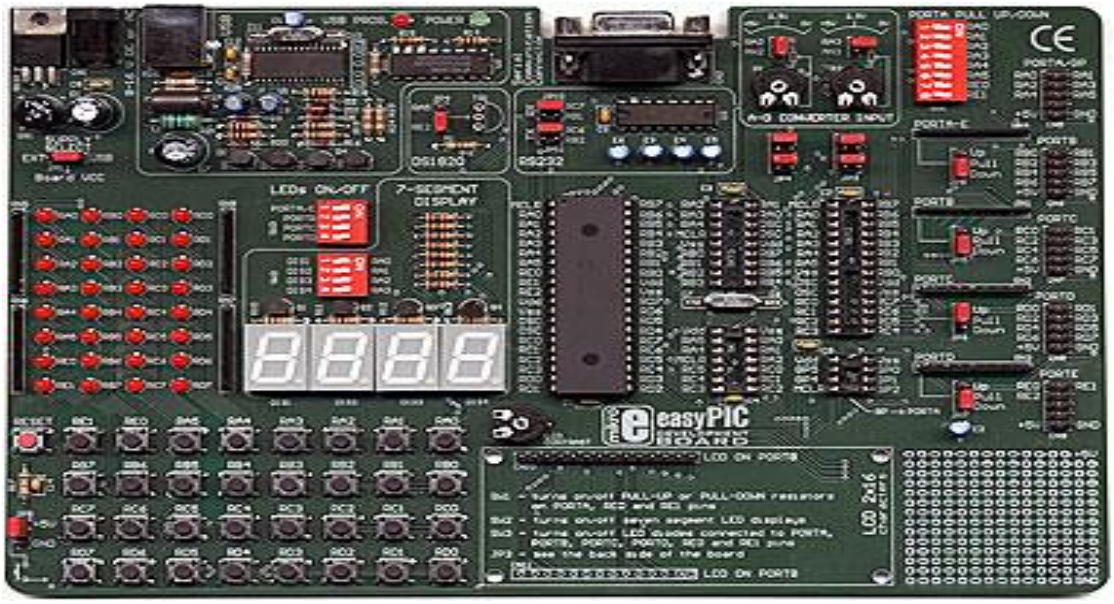
Αυτού του είδος ο προγραμματισμός γίνεται με την αναπτυξιακή μονάδα EasyPIC2 της MIKROELEKTRONIKA και συνιστάται για τους εξής λόγους:

1) Έχει ενσωματωμένο αρκετό περιφερειακό εξοπλισμό για την δοκιμή των προγραμμάτων όπως LEDs, διακόπτες, 7-segment displays, A/D converterts και RS232 διασύνδεση και μπορεί να προγραμματίσει μεγάλο πλήθος μικροελεγκτών PIC.

2) Ο προγραμματισμός και η τροφοδοσία γίνεται μέσω ενός απλού καλωδίου USB (δεν χρειάζεται δηλαδή ξεχωριστή ηλεκτρική τροφοδοσία).

3) Έχει την δυνατότητα επιπλέον προσαρμογής και άλλων πολλών περιφερειακών μονάδων όπως LCD display, RS232/RS485 διασύνδεση, IrDA μετατροπείς και άλλα.

4) Έχει πολύ μικρό κόστος και μπορεί να γίνει η παραγγελία μέσω διαδικτύου από τον δικτυακό τόπο της εταιρείας www.mikroelektronika.com στο οποίο μπορεί να δει κανείς και άλλες αναπτυξιακές μονάδες για τους μικροελεγκτές PIC.



ΕΙΚΟΝΑ XVI: ΑΝΑΠΤΥΞΙΑΚΗ ΜΟΝΑΔΑ EasyPIC2

Αυτή η εταιρεία παραχωρεί ένα συνοδευτικό CD που εκτός από το πλήθος των παραδειγμάτων και το ηλεκτρονικό υλικό για την γλώσσα assembly, υπάρχει και το λογισμικό PICFLASH με το οποίο γίνεται ο προγραμματισμός των ολοκληρωμένων μέσω της αναπτυξιακής πλακέτας EasyPIC2.

Έτσι τα βήματα που πρέπει να γίνουν για να προγραμματιστεί αυτή η πλακέτα είναι:

1. Δημιουργία με το λογισμικό MPLAB του κώδικα assembly (αρχείο .asm) και συμβολομετάφραση του. Από την συμβολομετάφραση δημιουργείται ένα αρχείο τύπου .hex.

2. Φόρτωση του λογισμικού PICFLASH και στη συνέχεια από το μενού File φορτώνεται το αρχείο με το κώδικα μηχανής (αρχείο .hex) που δημιουργήθηκε από το MPLAB

Στη συνέχεια γίνονται διαδοχικά οι παρακάτω ενέργειες:

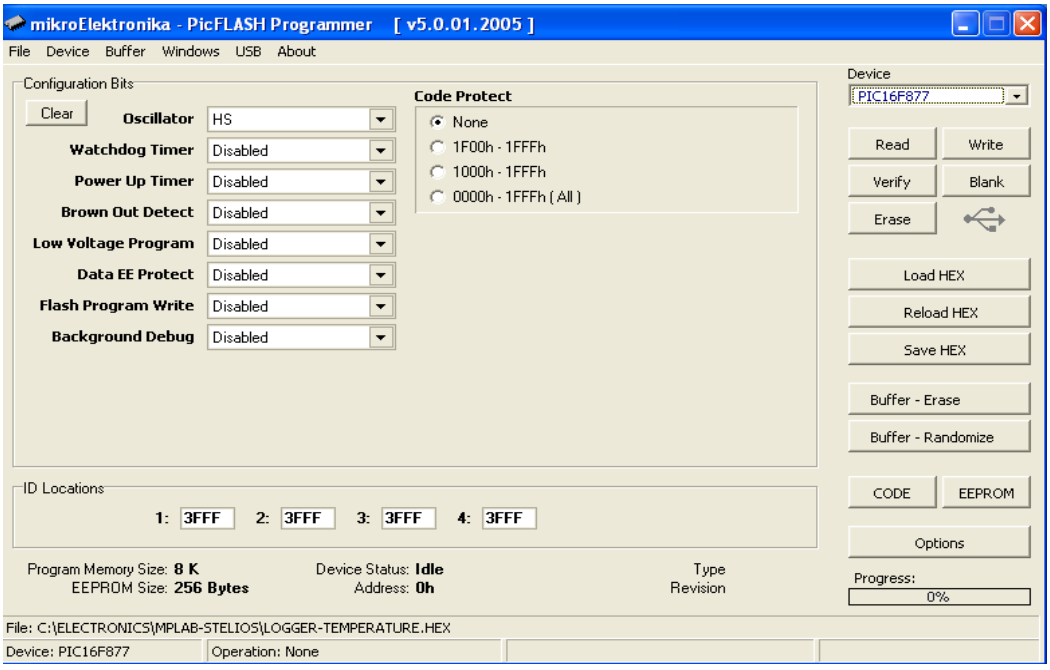
3. έλεγχος για το αν είναι κενή μνήμη του μικροελεγκτή (button Blank).

4. διαγραφή της μνήμης εάν αυτό απαιτείται (button Erase)

5. προγραμματισμός του ολοκληρωμένου με τον δεκαεξαδικό κώδικα (button Write).

Στο ίδιο παράθυρο διακρίνονται και τα μενού επιλογής ρυθμίσεων των Configuration Bits για τους μικροελεγκτές PIC στην περίπτωση που απαιτείται να αλλάξουν αυτές οι ρυθμίσεις. Συνήθως γίνεται αλλαγή στον τύπο του ταλαντωτή σε XT ή HS όταν χρησιμοποιείται κρύσταλλο στο κύκλωμα χρονισμού του μικροελεγκτή. Πάντως όπως αναφέρθηκε και στην προηγούμενη παράγραφο οι ρυθμίσεις αυτές μπορούν να ορισθούν μέσα στον κώδικα με τις εντολές αρχικών δηλώσεων .

Το αρχείο .hex από το οποίο φορτώνεται ο κώδικας μηχανής καθώς και ο τύπος του μικροελεγκτή φαίνονται στην κάτω αριστερή γωνία του παραθύρου του λογισμικού PICFLASH.



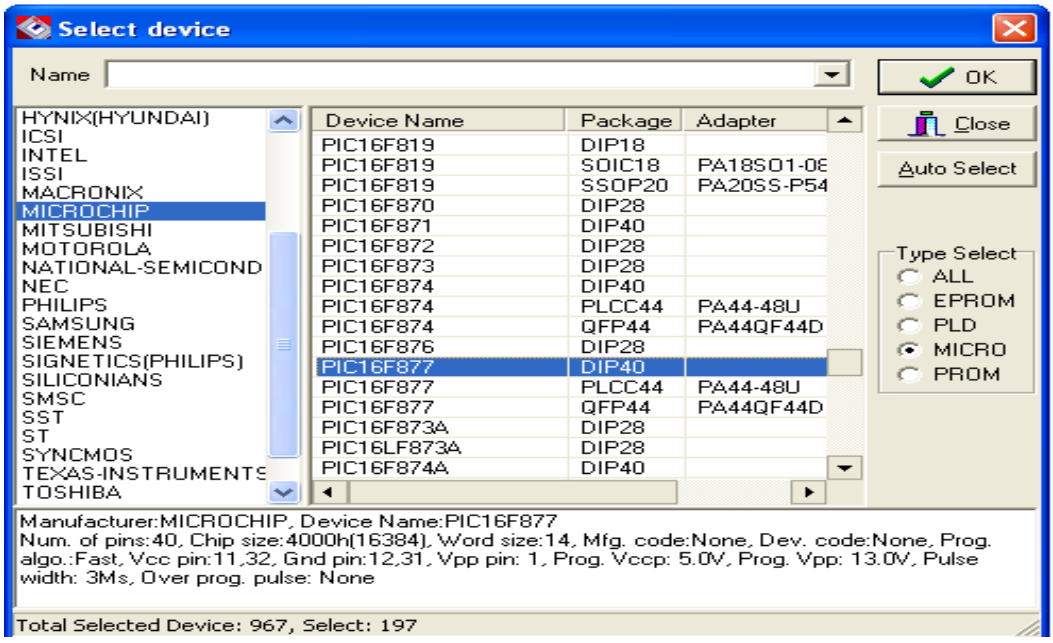
ΕΙΚΟΝΑ XVII: ΛΟΓΙΣΜΙΚΟ PICFLASH

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΜΕ ΤΗΝ ΜΟΝΑΔΑ TOPMAX:

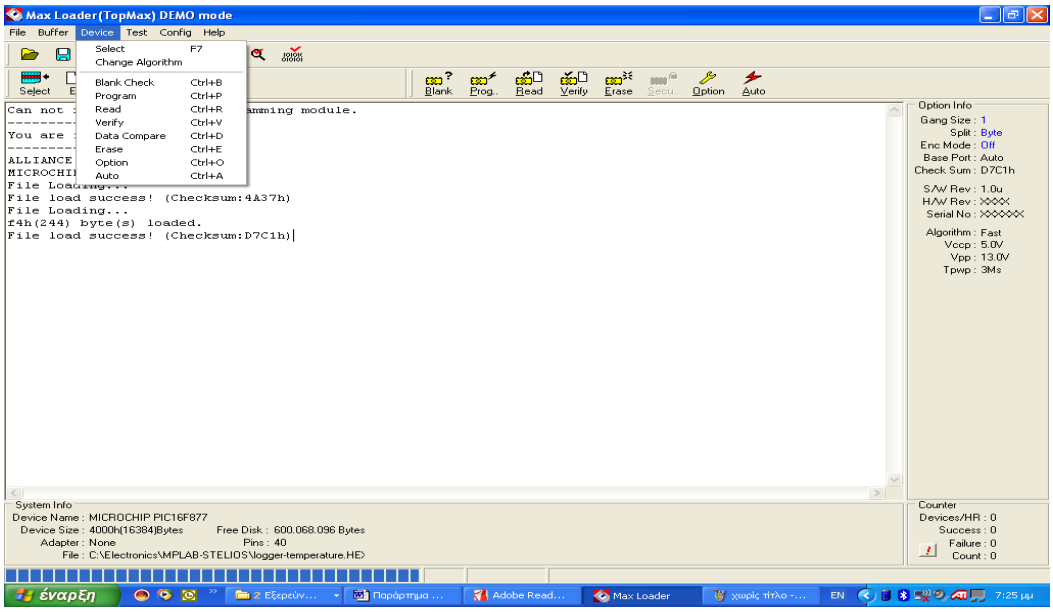
Από την στιγμή που με το λογισμικό MPLAB έχει παραχθεί το αρχείο με τον κώδικα HEX μπορεί να χρησιμοποιηθεί και οποιαδήποτε άλλη προγραμματίστρια μονάδα γενικής χρήσης που θα υποστηρίζει πέραν όλων των άλλων ολοκληρωμένων και μικροελεγκτές της εταιρείας MICROCHIP. Μια τέτοια προγραμματίστρια είναι και μονάδα TopMax της εταιρείας EETools η οποία υποστηρίζει τον προγραμματισμό πάνω από 4000 ολοκληρωμένων.

ΕΙΚΟΝΑ XVIII: TOPMAX

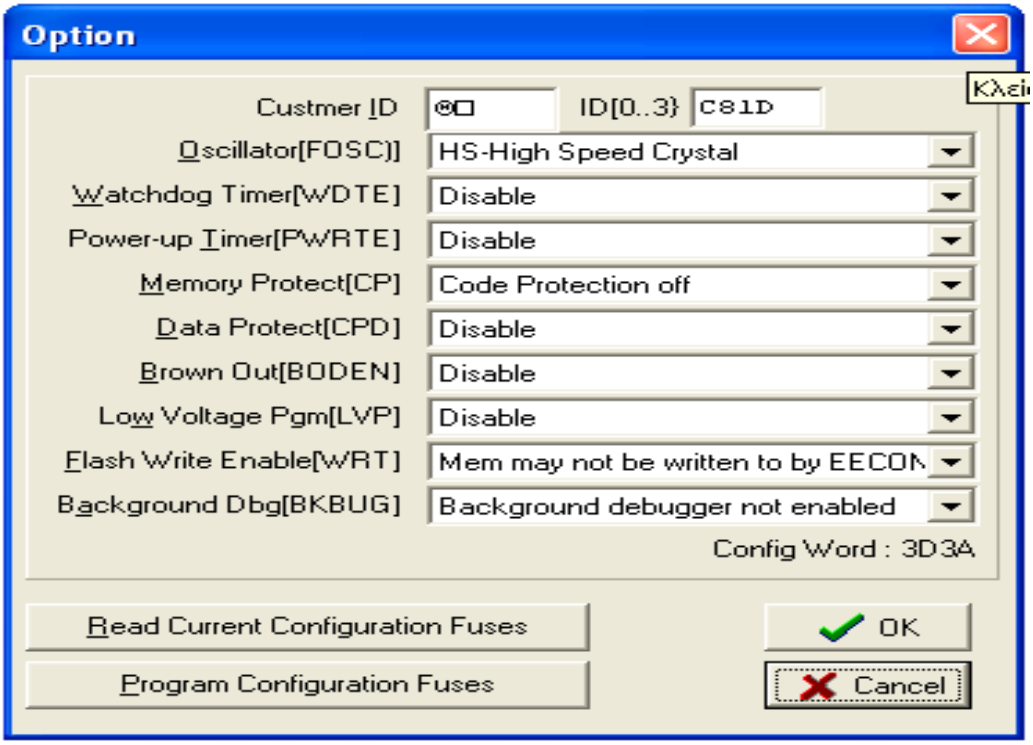
Από το συνοδευτικό CD της εταιρείας γίνεται εγκατάσταση του αντίστοιχου λογισμικού MAX για την μονάδα TopMax και η πρώτη ενέργεια που πρέπει να γίνει είναι η επιλογή της εταιρείας και του τύπου του ολοκληρωμένου που είναι να προγραμματιστεί. Τότε εμφανίζεται ένα παράθυρο για την επιλογή του τύπου του ολοκληρωμένου (EPROM, MICRO, PROM, PLD) και της αντίστοιχης εταιρείας κατασκευαστή.



ΕΙΚΟΝΑ XIX: ΕΠΙΛΟΓΗ ΟΛΟΚΛΗΡΩΜΕΝΟΥ ΠΡΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ



ΕΙΚΟΝΑ XX : ΜΕΝΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΟΛΟΚΛΗΡΩΜΕΝΟΥ



ΕΙΚΟΝΑ XXI: ΠΑΡΑΘΥΡΟ CONFIGURATION BIT

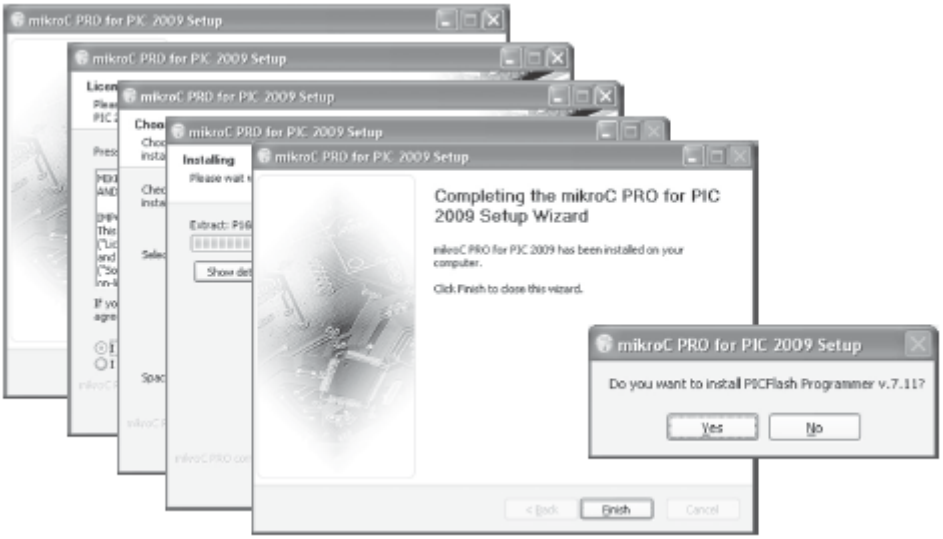
Τέλος είναι προφανές ότι η προγραμματίστρια TopMax είναι μια γενικής χρήσης μονάδα με πάρα πολλές δυνατότητες.

2.2.2 ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C ΜΕ ΤΟ ΛΟΓΙΣΜΙΚΟ microC PRO For PIC Compiler

Το πρώτο πράγμα που χρειάζεται για να γραφτεί ένα πρόγραμμα για τον μικροελεγκτή είναι ένα λογισμικό το οποίο να αντιλαμβάνεται τη γλώσσα προγραμματισμού που χρησιμοποιείται, που σε αυτήν την περίπτωση είναι η γλώσσα C, και στο οποίο να γράφεται ο κώδικας του προγράμματος. Επιπλέον, το λογισμικό πρέπει να «γνωρίζει» την αρχιτεκτονική του μικροελεγκτή που χρησιμοποιείται. Στην περίπτωση αυτή, είναι χρήσιμος ένας μεταγλωττιστής για τη γλώσσα C.

Δεν υπάρχει μεταγλωττιστής που να χρησιμοποιείται μόνο για ένα συγκεκριμένο μικροελεγκτή όπως επίσης δεν υπάρχει μεταγλωττιστής που να χρησιμοποιείται για όλους τους μικροελεγκτές. Όσον αφόρα το λογισμικό συνηθίζεται να προγραμματίζει μια ομάδα παρόμοιων μικροελεγκτών ενός κατασκευαστή .

Το λογισμικό που θα αναλυθεί αφορά τους μικροελεγκτές PIC και ονομάζεται microC PRO for PIC. Όπως υποδηλώνει το όνομα του, ο μεταγλωττιστής προορίζεται για τη εγγραφή προγραμμάτων για μικροελεγκτές PIC σε γλώσσα C. Το πρόγραμμα αυτό είναι εφοδιασμένο με όλα τα δεδομένα από την εσωτερική αρχιτεκτονική αυτών των μικροελεγκτών, με τη λειτουργία των συγκεκριμένων κυκλωμάτων, με ένα σύνολο εντολών, με τα ονόματα των καταχωρητών και τις ακριβείς διευθύνσεις τους, και άλλα. Με την έναρξη του προγράμματος επιλέγεται ο τύπος του μικροελεγκτή που θα χρησιμοποιηθεί στην εφαρμογή αλλά και η C ως η γλώσσα που θα χρησιμοποιηθεί για τον προγραμματισμό του.



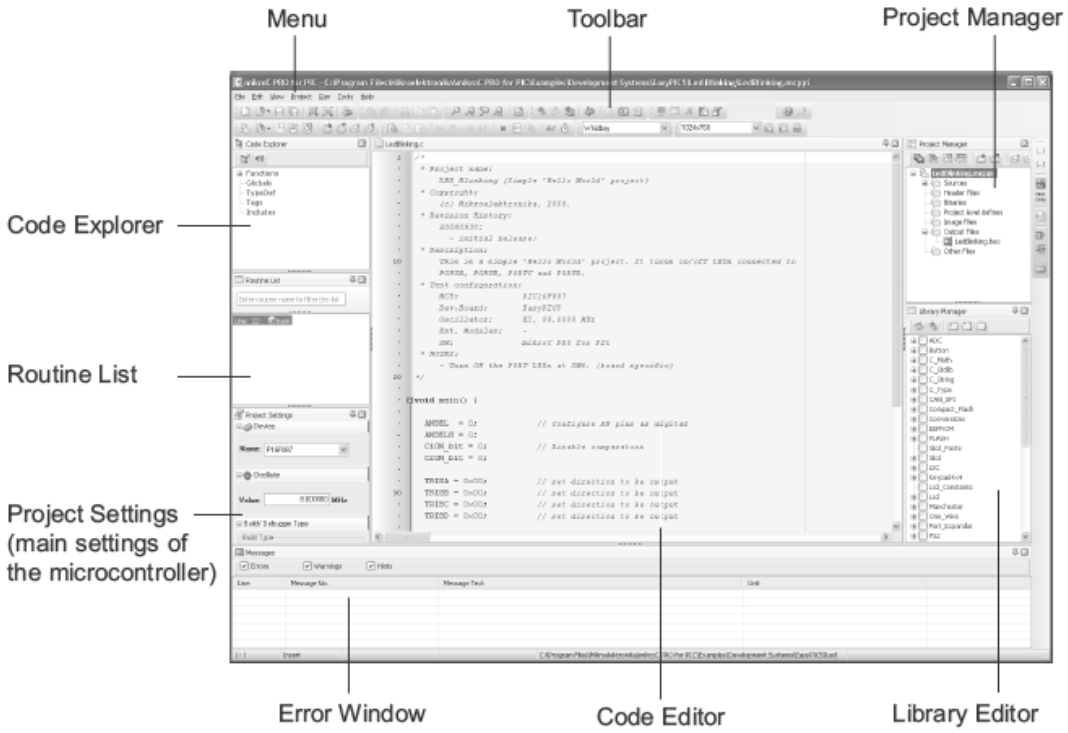
ΕΙΚΟΝΑ XXII : ΟΘΟΝΗ ΕΓΚΑΤΑΣΤΑΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Καθήκον του μεταγλωττιστή είναι να μετατρέψει ένα πρόγραμμα γραμμένο σε γλώσσα C σε hex κώδικα. Όταν ολοκληρωθεί η διαδικασία εγκατάστασης του προγράμματος ζητείται και η εγκατάσταση ενός άλλου λογισμικού που είναι χρήσιμο για τον προγραμματισμό μιας ειδικής ομάδας μικροελεγκτών PIC που λειτουργούν σε κατάσταση χαμηλής κατανάλωσης(3,3volt).

Οι οδηγοί(Drivers) που ζητούνται να περαστούν στην συνέχεια είναι υπεύθυνοι για την επικοινωνία του λογισμικού του προγραμματιστή με το υλικό(Hardware).

ΕΙΚΟΝΑ XXIII: PICFlash Drivers

Με την εκκίνηση του προγράμματος εμφανίζετε το ακόλουθο παράθυρο:

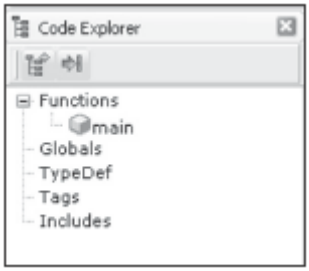


ΕΙΚΟΝΑ XXIV : ΟΘΟΝΗ ΕΝΑΡΞΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Στη συνέχεια θα περιγραφεί η διαδικασία της συγγραφής ενός προγράμματος σε γλώσσα C, καθώς και η φόρτωσή του στη μνήμη του μικροελεγκτή.

Ένα πρόγραμμα γραμμένο σε αυτό το λογισμικό δεν είναι ένα ξεχωριστό έγγραφο, αλλά μέρος ενός έργου που περιλαμβάνει hex κώδικα, κώδικα σε γλώσσα assembly, κεφαλίδα και άλλα αρχεία. Μερικά από αυτά έχουν δημιουργηθεί κατά τη διάρκεια της λειτουργίας του μεταγλωττιστή, ενώ μερικά εισάγονται από άλλα προγράμματα. Ωστόσο, ο διαχειριστής παραθύρων του έργου δίνει τη δυνατότητα χειρισμού όλων αυτών.

Το παράθυρο εξερεύνησης του κώδικα επιτρέπει στον προγραμματιστή να εντοπίζει με ευκολία διάφορες συναρτήσεις και διαδικασίες που έχει συμπεριλάβει στο πρόγραμμα του.



ΕΙΚΟΝΑ XXV: ΠΑΡΑΘΥΡΟ ΕΞΕΡΕΥΝΗΣΗΣ ΤΟΥ ΚΩΔΙΚΑ

Για να μπορέσει ο μεταγλωττιστής να λειτουργήσει με επιτυχία, είναι απαραίτητο να γνωρίζει βασικές πληροφορίες για τον μικροελεγκτή που χρησιμοποιείται καθώς και πληροφορίες σχετικά με το τι περιμένει κανείς από αυτόν μετά από τη διαδικασία της μεταγλώττισης.

ΕΙΚΟΝΑ XXVI: ΡΥΘΜΙΣΕΙΣ ΓΙΑ ΤΟΝ ΜΙΚΡΟΕΛΕΓΚΤΗ

- Συσκευή : Όταν γίνεται επιλογή του μικροελεγκτή, ο μεταγλωττιστής γνωρίζει αυτόματα το αρχείο προορισμού που περιέχει όλους τους SFR καταχωρητές για συγκεκριμένη MCU[Multipoint Control Unit (είναι μια συσκευή που χρησιμοποιείται συνήθως ως γέφυρα συνδέσεων τηλεδιάσκεψης)] καθώς και τις διευθύνσεις μνήμης που θα χρησιμοποιηθούν.

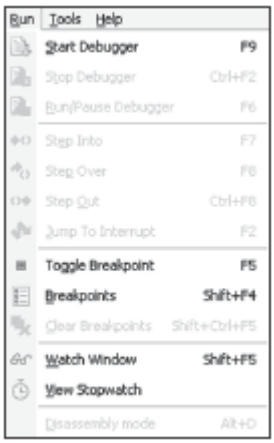
- Ταλαντωτής : Η επιλογή αυτή χρησιμοποιείται για να επιλεγεί η ταχύτητα λειτουργίας του μικροελεγκτή. Στη βάση αυτού, ο μεταγλωττιστής κάνει αλλαγές στο configuration word.. Η ταχύτητα λειτουργίας έχει οριστεί έτσι ώστε να γίνει δυνατή η εσωτερική ταλάντωση του μικροελεγκτή για να λειτουργεί με τα επιλεγμένους χαλαζίες κρυστάλλων.

- Είδος κατασκευής : Αφού η διαδικασία της μεταγλώττισης έχει ολοκληρωθεί, ο μεταγλωττιστής δεν έχει καμία συμμετοχή στην εκτέλεση του προγράμματος. Για τους σκοπούς της αποσφαλμάτωσης μπορεί να χρησιμοποιηθεί ένας προσομοιωτής.

- ICD(In Circuit Debugger) κύκλωμα εντοπισμού σφαλμάτων: Όταν η διαδικασία της μεταγλώττισης έχει ολοκληρωθεί και ο μικροελεγκτής έχει προγραμματιστεί, ο μεταγλωττιστής παραμένει συνδεδεμένος με το μικροελεγκτή και μπορεί ακόμα να επηρεάσει τη λειτουργία του. Η σύνδεση πραγματοποιείται μέσω του προγραμματιστή που συνδέεται στον υπολογιστή μέσω καλωδίου USB. Ένα λογισμικό είναι υπεύθυνο γι' αυτή την διαδικασία και ονομάζεται ICD (κύκλωμα εντοπισμού σφαλμάτων). Επιτρέπει το πρόγραμμα να εκτελείται βήμα προς βήμα και παρέχει πρόσβαση στους καταχωρητές του μικροελεγκτή. Αν η προσομοίωση δεν πραγματοποιηθεί, το περιεχόμενό τους διαβάζεται από μία MCU συσκευή.

ΛΟΓΙΣΜΙΚΟ ΠΡΟΣΟΜΟΙΩΣΗΣ

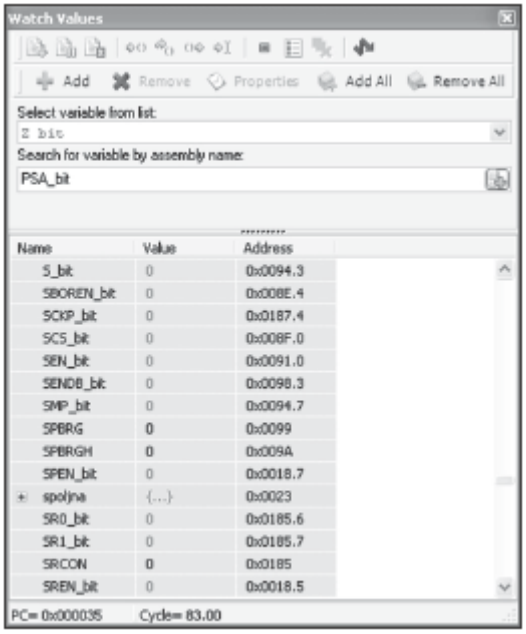
Πριν από την έναρξη της προσομοίωσης, μπορεί να επιλεγεί η κατάλληλη λειτουργία στο παράθυρο Project Settings (Build type) και συγκεκριμένα στο Run / Start Debugger που είναι η επιλογή εντοπισμού σφαλμάτων.Ο μεταγλωττιστής θα ρυθμιστεί αυτόματα σε λειτουργία προσομοίωσης. Ως εκ τούτου, παρακολουθεί την κατάσταση όλων των bits των καταχωρητών. Επιτρέπεται επίσης η εκτέλεση του προγράμματος βήμα προς βήμα καθώς και η παρακολούθηση της λειτουργίας του μικροελεγκτή στην οθόνη.



ΕΙΚΟΝΑ XXVII: ΕΠΙΛΟΓΗ ΠΡΟΣΟΜΟΙΩΣΗΣ

Μερικές ακόμη από τις δυνατότητες που υπάρχουν είναι να τοποθετηθεί ο κέρσορας σε μία συγκεκριμένη γραμμή του κώδικα και να εκτελεστεί όπως επίσης μπορεί να γίνει η εκτέλεση και σε ένα συγκεκριμένο τμήμα κώδικα.

Ο προσομοιωτής και ο debugger έχουν την ίδια λειτουργία για την παρακολούθηση της κατάστασης των καταχωρητών κατά τη διάρκεια εκτέλεσης του προγράμματος. Η διαφορά είναι ότι ο προσομοιωτής εκτελεί το πρόγραμμα στον υπολογιστή, ενώ ο debugger χρησιμοποιεί έναν αληθινό μικροελεγκτή. Οποιαδήποτε αλλαγή κατάστασης σε ένα pin αντανακλάται στον αντίστοιχο καταχωρητή. Καθώς το παράθυρο παρακολούθησης επιτρέπει τον έλεγχο της κατάστασης όλων των καταχωρητών είναι εύκολο να ελεγχθεί αν ένα pin είναι ρυθμισμένο στο μηδέν ή στο ένα. Επίσης υπάρχει η δυνατότητα να γίνει μια λίστα με συγκεκριμένους καταχωρητές μέσα από την οποία θα παρακολουθείται η κατάσταση τους.



ΕΙΚΟΝΑ XXVIII : ΕΛΕΓΧΟΣ ΤΙΜΩΝ ΚΑΤΑΧΩΡΗΤΩΝ

Τέλος μπορεί να εντοπιστεί ο χρόνος που χρειάζεται ο μικροελεγκτής για να εκτελέσει ένα μέρος του προγράμματος.

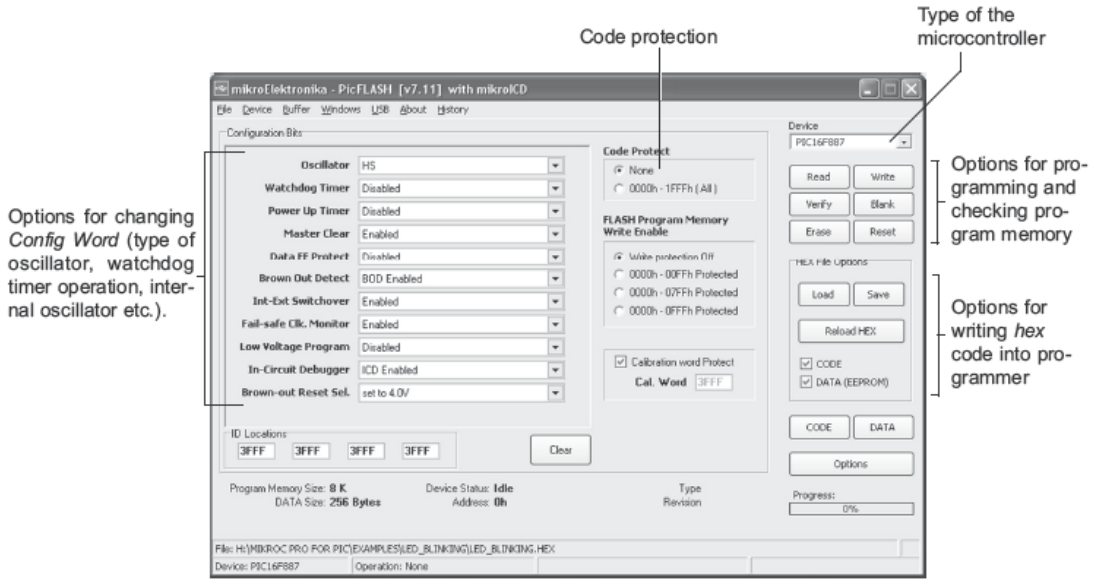
ΕΙΚΟΝΑ XXIX: ΧΡΟΝΟΣ ΕΚΤΕΛΕΣΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΑΠΟ ΤΟΝ ΜΙΚΡΟΕΛΕΓΚΤΗ

ΕΡΓΑΛΕΙΑ ΠΡΟΣΟΜΟΙΩΤΗ

Τα εργαλεία που παρέχει ο προσομοιωτής είναι πάρα πολύ χρήσιμα γιατί απλουστεύουν σημαντικά τη διαδικασία της εγγραφής ενός προγράμματος.

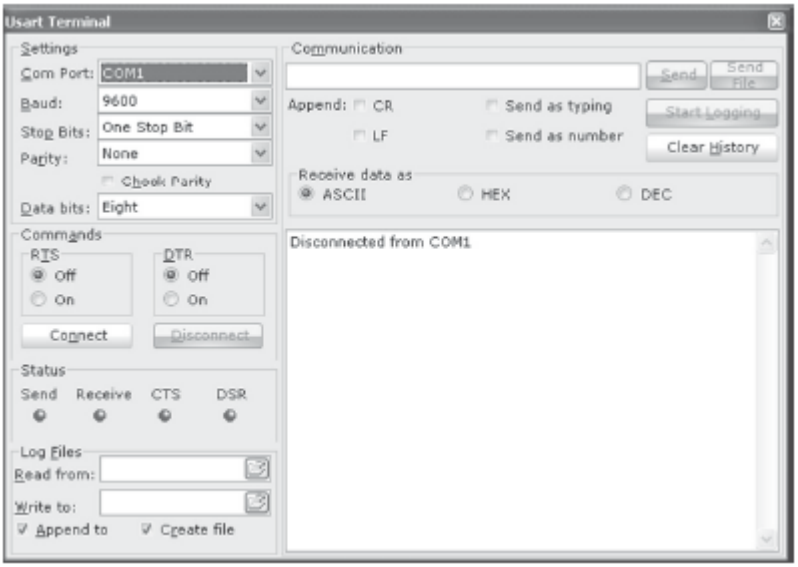
Μερικά από τα σημαντικότερα είναι τα εξής:

- *PICflash programmer*: Είναι ένα αυτόνομο πρόγραμμα το οποίο μπορεί να λειτουργήσει ανεξάρτητα από τον μεταγλωττιστή, δηλαδή μπορεί να χρησιμοποιηθεί ως ένα ξεχωριστό πρόγραμμα. Ωστόσο, στην περίπτωση αυτή, η λειτουργία του είναι στενά συνδεδεμένη με τη λειτουργία του μεταγλωττιστή, ώστε να μπορεί να ενεργοποιηθεί μέσα από τον ίδιο το μεταγλωττιστή. Με την εγκατάσταση του εμφανίζεται ένα παράθυρο που περιέχει διάφορες επιλογές που θα χρησιμοποιηθούν για τη διαδικασία του προγραμματισμού μικροελεγκτών. Ο μεταγλωττιστής είναι ένα λογισμικό που μεταγλωττίζει το πρόγραμμα που είναι γραμμένο σε μια γλώσσα προγραμματισμού υψηλού επιπέδου σε εκτελέσιμο κώδικα, δηλαδή κώδικα Hex διότι αυτός είναι ο κώδικας που κατανοεί και εκτελεί ο μικροελεγκτής. Το λογισμικό που είναι εγκατεστημένο στον ηλεκτρονικό υπολογιστή χρησιμοποιείται για να περάσει τον hex κώδικα στο υλικό μέσω καλωδίου USB.



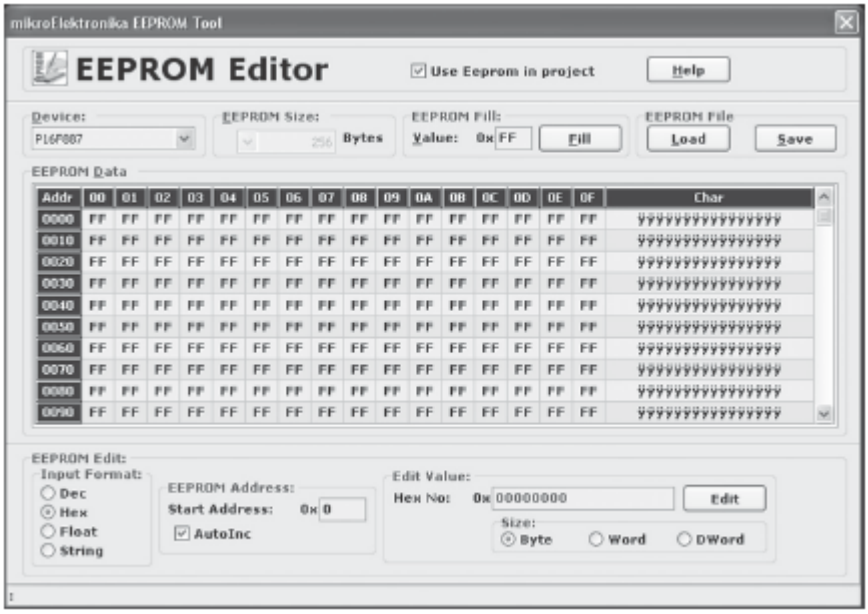
ΕΙΚΟΝΑ XXX : ΠΑΡΑΘΥΡΟ ΕΠΙΛΟΓΩΝ PICFLASH

- *Usart terminal:* Είναι μια αντικατάσταση για το πρότυπο των Windows Hyper Terminal. Μπορεί να χρησιμοποιηθεί για τον έλεγχο της λειτουργίας του μικροελεγκτή που χρησιμοποιεί USART επικοινωνία. Ένας τέτοιος μικροελεγκτής συνδέεται με την RS232 θύρα του υπολογιστή μέσω σειριακού καλωδίου. Το παράθυρο του τερματικού USART, περιέχει επιλογές για τη σειριακή επικοινωνία, για την αποστολή και για την λήψη δεδομένων.



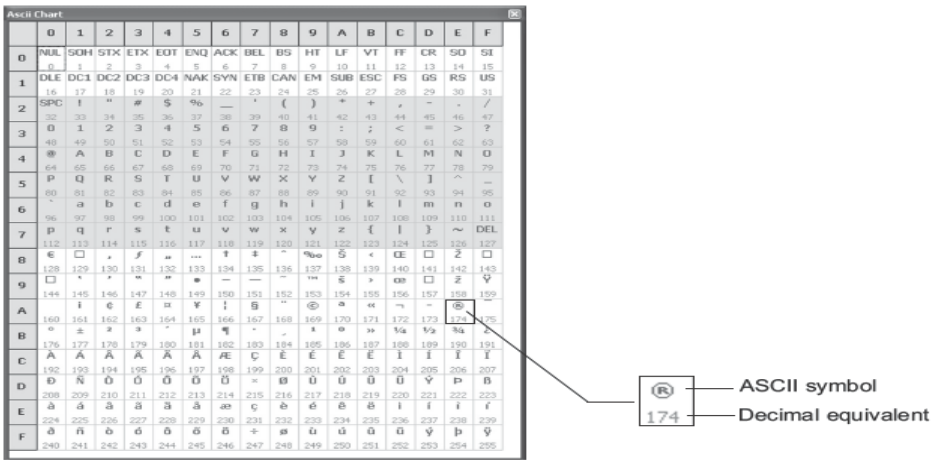
ΕΙΚΟΝΑ XXXI :ΠΑΡΑΘΥΡΟ USART TERMINAL

- *EEPROM Editor*: Είναι το πώς η μνήμη EEPROM του μικροελεγκτή μοιάζει εσωτερικά. Εάν θέλει να αλλάξει κανείς το περιεχόμενό της μετά τη φόρτωση του προγράμματος στο μικροελεγκτή , το πραγματοποιεί με τον *EEPROM Editor*. Εάν το νέο περιεχόμενο είναι ένα σύνολο δεδομένων ενός συγκεκριμένου τύπου όπως (char, int ή double), τότε θα πρέπει να το επιλέξει πληκτρολογώντας την τιμή σε ένα συγκεκριμένο πεδίο και στη συνέχεια να το αποθηκεύσει ως έγγραφο με την κατάληξη .hex. Εάν η χρήση EEPROM στο έργο είναι ενεργή, τα δεδομένα θα πρέπει να φορτώνονται αυτόματα στο τσιπ κατά τη διαδικασία του προγραμματισμού.



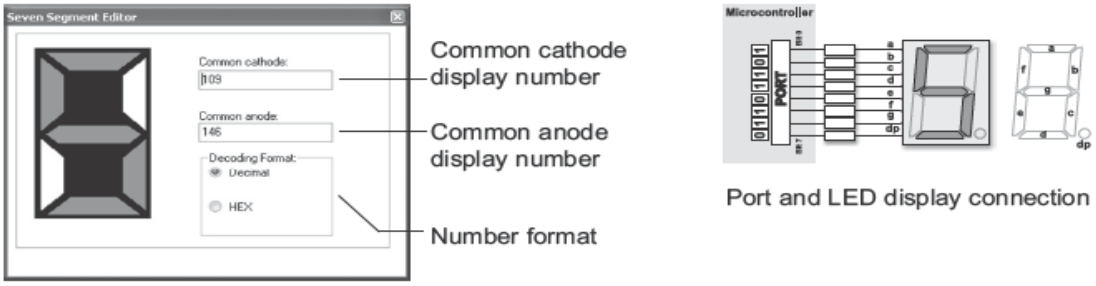
ΕΙΚΟΝΑ XXXII : ΜΝΗΜΗ EEPROM ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ

- *ASCII chart :* Χρησιμοποιείται για την αριθμητική απεικόνιση του κάθε χαρακτήρα ASCII. Οι χαρακτήρες που αντιπροσωπεύουν αριθμοί είναι περιέργως ισοδύναμοι. Για το λόγω αυτό, με την εντολή του προγράμματος για την εμφάνιση του αριθμού 7 παραδείγματος χάριν σε μια οθόνη LCD δεν θα εμφανίζεται τίποτα παρόμοιο με αυτό τον αριθμό. Αντίθετα, το ισοδύναμο της εντολής BEL θα εμφανιστεί. Αν σταλθεί ο ίδιος αριθμός με ένα χαρακτήρα, θα έχει το αναμενόμενο αποτέλεσμα δηλαδή τον αριθμό 7. Κατά συνέπεια, εάν θέλει κανείς να εμφανίσει έναν αριθμό, χωρίς προηγουμένως να το μετατρέψει σε χαρακτήρα, τότε είναι απαραίτητο να προσθέσει τον αριθμό 48 σε κάθε ψηφίο που αποτελείται ο αριθμός.



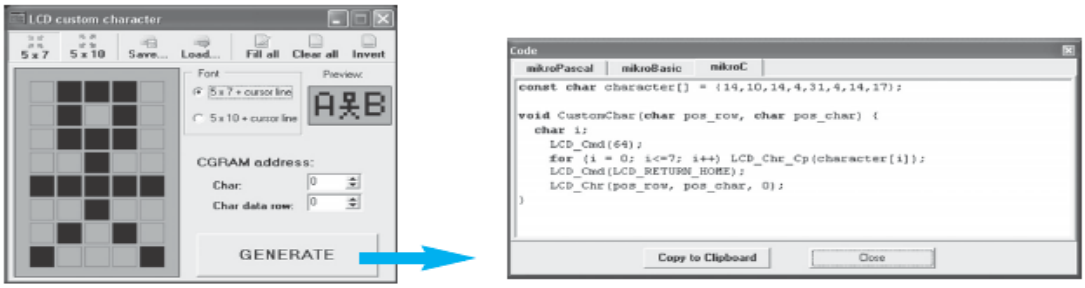
ΕΙΚΟΝΑ XXXIII:ΠΙΝΑΚΑΣ ASCII

- *Seven Segment Editor :* Επιτρέπει σε κάποιον να βρεί εύκολα ποιος αριθμός είναι απαραίτητο να τεθεί σε μια θύρα εξόδου για να εμφανιστεί ένα σύμβολο που θέλει.



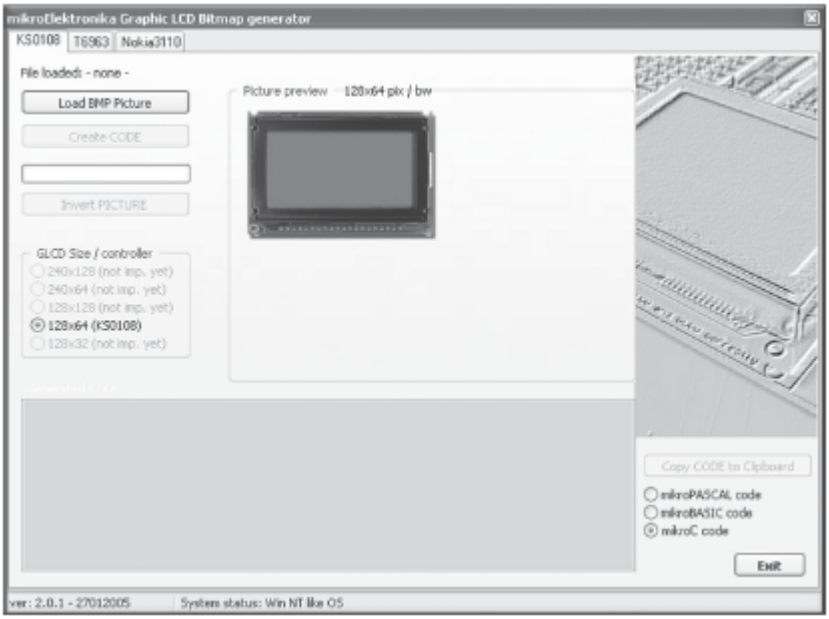
ΕΙΚΟΝΑ XXXIV : ΠΑΡΑΘΥΡΟ Seven Segment Editor

- *LCD Custom Character:* Εκτός από τους βασικούς χαρακτήρες, ο μικροελεγκτής μπορεί να στείλει χαρακτήρες που δημιουργήθηκαν από το χρήστη σε μια οθόνη. Με την επιλογή αυτή απαλλάσσεται από την σύνταξη μιας συνάρτησης και την αποστολή κατάλληλου κώδικα σε μια οθόνη. Ο απαιτούμενος κωδικός αναγράφεται σε άλλο παράθυρο.



ΕΙΚΟΝΑ XXXV : ΠΑΡΑΘΥΡΟ *LCD Custom Character*

- *Graphic LCD Bitmap Generator :* Αυτό είναι άλλο ένα αναντικατάστατο εργαλείο στην περίπτωση που για τον μικροελεγκτή για τον οποίο γράφει ο χρήστης το πρόγραμμα χρησιμοποιεί γραφική οθόνη LCD (GLCD). Αυτό το εργαλείο δίνει τη δυνατότητα εμφάνισης οποιουδήποτε bitmap εύκολα. Στην συνέχεια γίνεται επιλογή του τύπου της οθόνης που θα χρησιμοποιηθεί και η φορτώση ενός αρχείου bitmap. Το bitmap πρέπει να είναι μονοχρωματικό



.

ΕΙΚΟΝΑ XXXVI : ΠΑΡΑΘΥΡΟ *Graphic LCD Bitmap Generator*

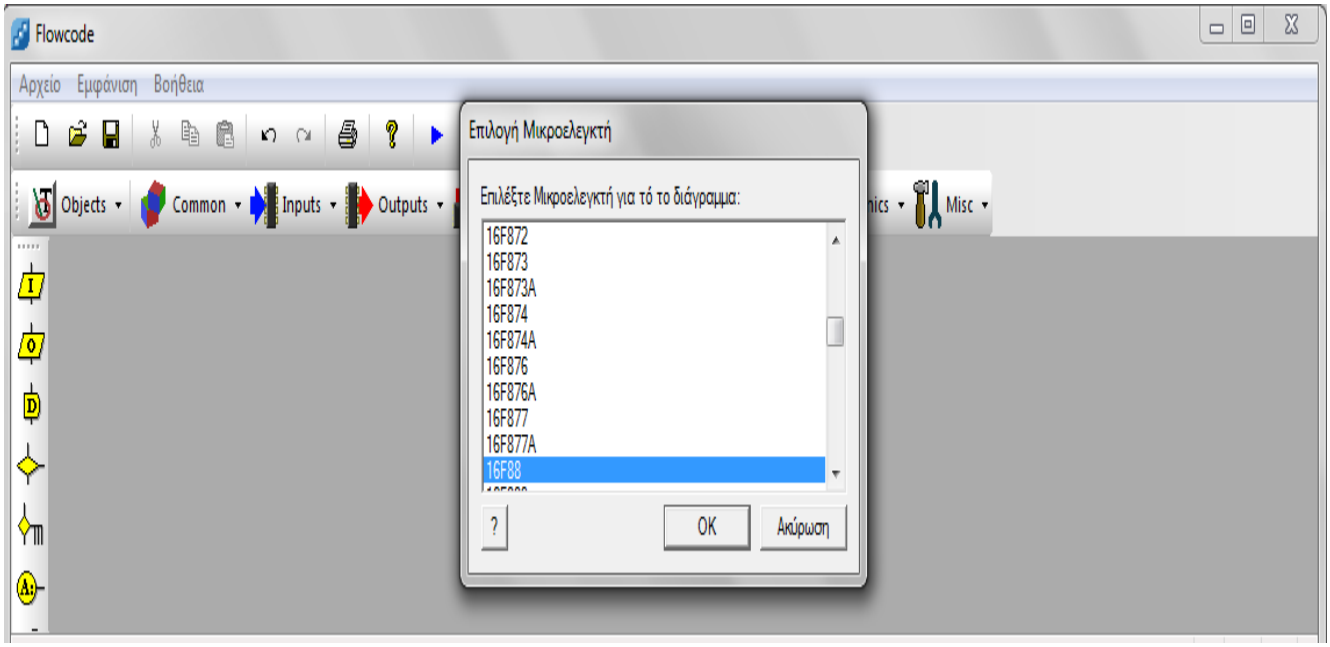
- *Libraries:* Όπως αναφέρθηκε προηγουμένως το κύριο πλεονέκτημα των υψηλότερων γλωσσών προγραμματισμού όπως η C είναι ότι δίνουν τη δυνατότητα να χρησιμοποιούνται ουσιαστικά οι γνώσεις και η εργασία των άλλων ανθρώπων. Εάν χρειάζεται για παράδειγμα μια συνάρτηση για την εκτέλεση ορισμένων εργασιών στο πρόγραμμα του χρήστη, απλά πρέπει να το ψάξει σε κάποια από τις βιβλιοθήκες, οι οποίες ενσωματώνονται στον μεταγλωττιστή και να την χρησιμοποιήσει. Για παράδειγμα, εάν χρειάζεται μια συνάρτηση για να παράγει ήχο σε μερικές από τις ακίδες, πρέπει να ανοίξει τη βιβλιοθήκη ήχου. Επίσης ακόμη και για κάποιον που δεν γνωρίζει τι ακριβώς θέλει να κάνει υπάρχει μια λεπτομερής περιγραφή για τις

διάφορες συναρτήσεις της βιβλιοθήκης. Έτσι το μόνο που χρειάζεται να γίνει είναι να ρυθμιστούν οι κατάλληλες παράμετροι αφού πρώτα γίνει εισαγωγή της συνάρτησης που χρειάζεται. Εάν αυτή η βιβλιοθήκη έχει ελεγχθεί, τα καθήκοντά της θα πρέπει να αναγνωρίζονται αυτόματα κατά τη διαδικασία της κατάρτισης, ώστε να μην είναι απαραίτητο να χρησιμοποιηθεί η εντολή # include.

2.2.3 ΔΙΑΓΡΑΜΜΑΤΙΚΗ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΜΕ ΤΟ ΛΟΓΙΣΜΙΚΟ FLOWCODE

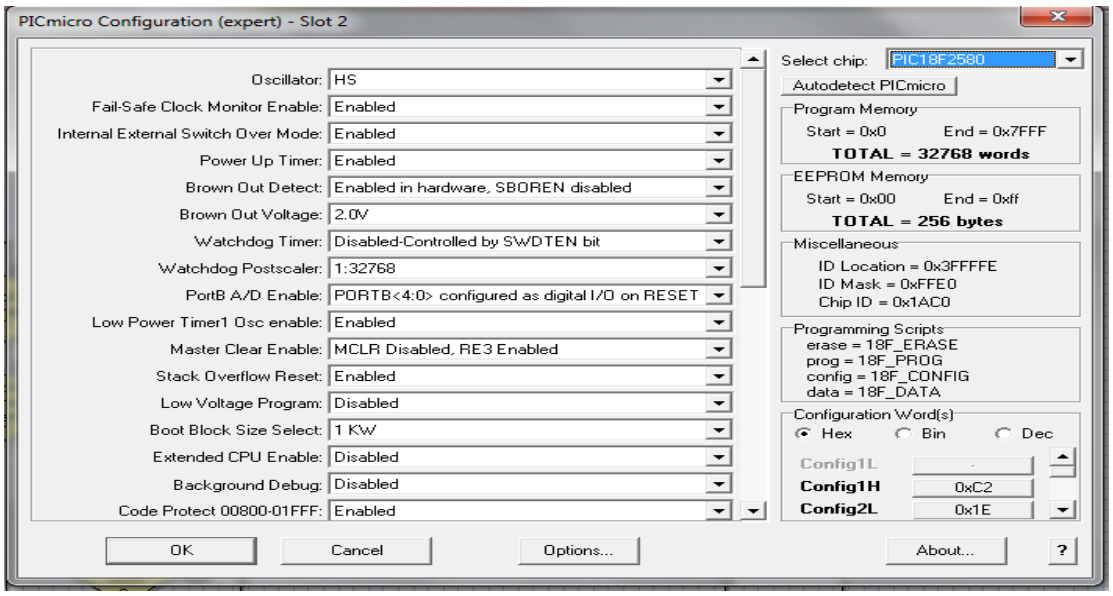
Η διαγραμματική γλώσσα προγραμματισμού με το λογισμικό Flowcode είναι μια πολύ υψηλού επιπέδου γλώσσα προγραμματισμού συστήματος για PIC μικροελεγκτές με βάση τα διαγράμματα ροής. Το Flowcode επιτρέπει την σχεδίαση και την προσομοίωση πολύπλοκων εφαρμογών και συστημάτων ελέγχου απλά σχεδιάζοντας ένα διάγραμμα ροής του επιθυμητού προγράμματος σε λίγα λεπτά, ακόμη και χωρίς καμία προηγούμενη γνώση προγραμματισμού. Εκτός απο τους μικροελεγκτές PIC με αυτό το λογισμικό μπορούν να προγραμματιστούν και μικροελεγτές τύπου AVR και ATMEL.

Με την έναρξη του προγράμματος επιλέγεται ο τύπος του μικροελεγκτή που θα χρησιμοποιηθεί στην εφαρμογή.



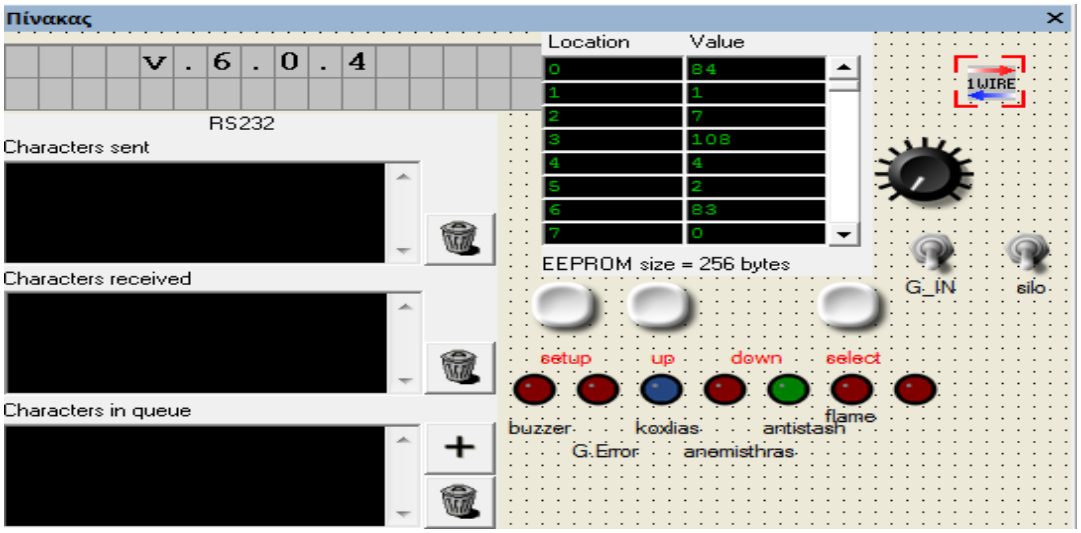
ΕΙΚΟΝΑ XXXVII : ΕΠΙΛΟΓΗ ΜΙΚΡΟΕΛΕΓΚΤΗ ΓΙΑ ΤΟ ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ

Στη συνέχεια ακολουθεί μια περιγραφή των σημαντικότερων χαρακτηριστικών του προγράμματος που βοηθούν στον σωστό προγραμματισμό του μικροελεγκτή.



ΕΙΚΟΝΑ XXXVIII : ΠΑΡΑΘΥΡΟ Configuration

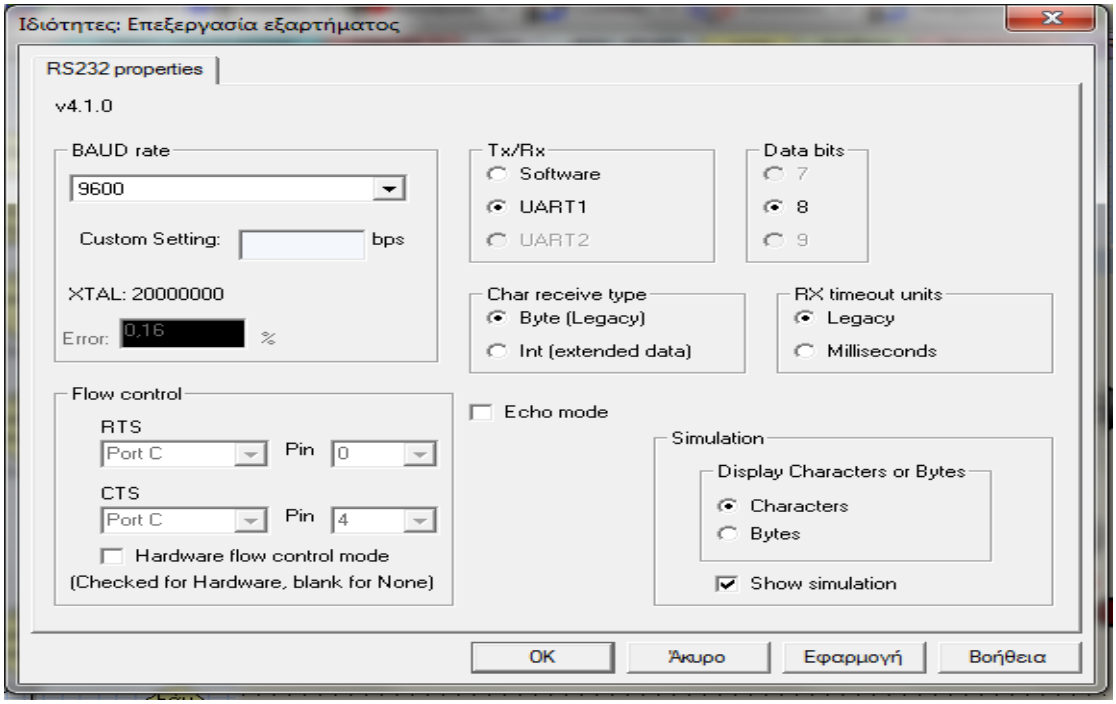
Με βάση τον επεξεργαστή, το hardware διασύνδεσης του και του software που έχει επιλεγεί, ορίζονται οι ρυθμίσεις του έτσι ώστε να υπάρχουν τα αναμενόμενα αποτελέσματα κατά την λειτουργία.



ΕΙΚΟΝΑ XXXIX : ΠΑΡΑΘΥΡΟ ΣΤΟΙΧΕΙΩΝ ΠΟΥ ΕΠΙΛΕΞΑΜΕ

Στο παραπάνω παράθυρο εμφανίζεται ο τύπος κάθε εισόδου και εξόδου, όπως έχει οριστεί. Επίσης, εμφανίζονται τα πρωτόκολλα επικοινωνίας που έχουν επιλεγεί αλλά και χαρακτηριστικά του επεξεργαστή που χρησιμοποιείται .

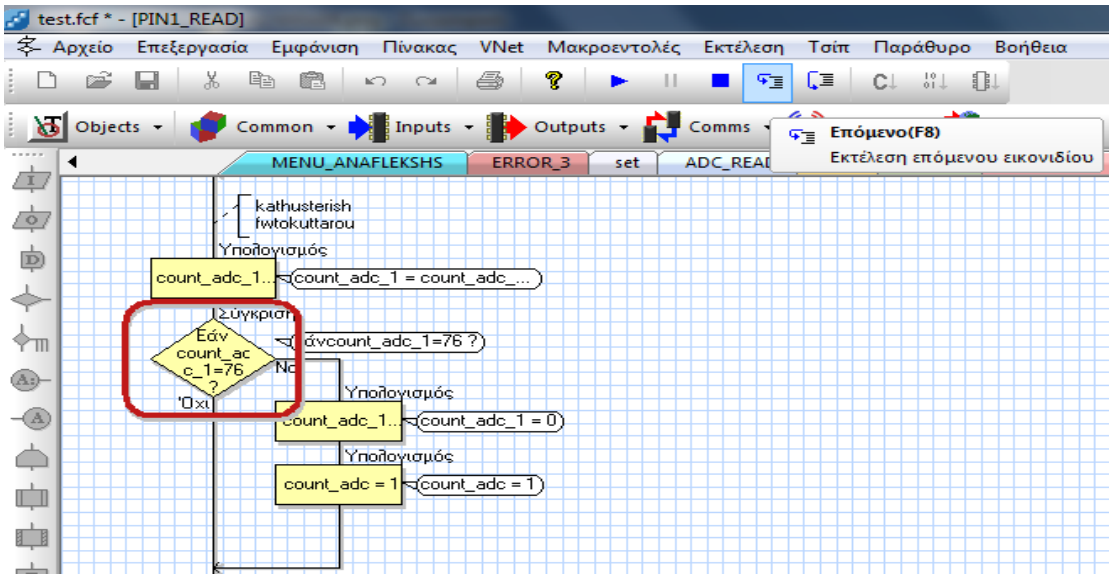
Κάνοντας δεξί κλικ σε οποιαδήποτε συσκευή του προηγούμενου παραθύρου ανοίγει ένα νέο παράθυρο ρυθμίσεων από το οποίο και πραγματοποιούνται οι απαραίτητες ρυθμίσεις για το πρωτόκολλο διασύνδεσης που χρησιμοποιείται που στην περίπτωση αυτή είναι η σειριακή επικοινωνία RS232 .



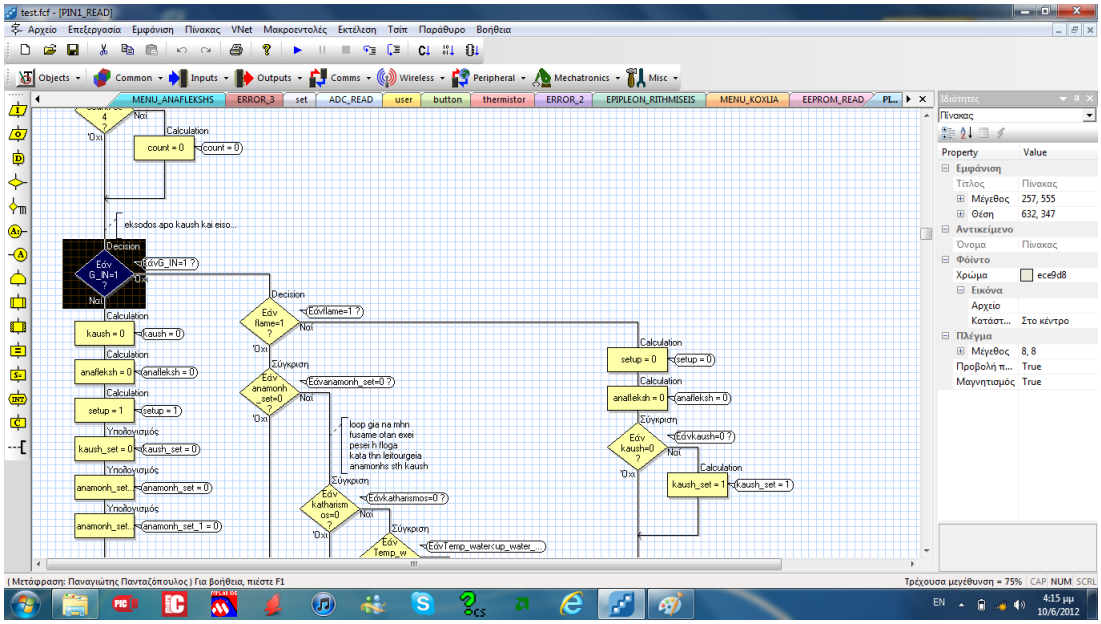
ΕΙΚΟΝΑ XL: ΠΙΝΑΚΑΣ ΡΥΘΜΙΣΕΩΝ ΕΠΕΞΕΡΓΑΣΤΗ ΜΕ ΤΗΝ ΣΕΙΡΙΑΚΗ ΕΠΙΚΟΙΝΩΝΙΑ RS232

Με την εισαγωγή κάθε νέας λειτουργίας εμφανίζεται και ένα σύνολο προκατασκευασμένων μακροεντολών που χρησιμοποιούνται έτσι ώστε να γίνει απλούστερη η ανάπτυξη του προγράμματος.

Για την εκτέλεση σε επίπεδο λογισμικού δίνεται η δυνατότητα να εκτελεστεί ο κώδικας εντολή προς εντολή έτσι ώστε να εντοπιστούν λάθη όπως φαίνεται στην παρακάτω εικόνα. Όλες οι είσοδοι και οι έξοδοι απεικονίζονται στο πίνακα, όπως επίσης και οι τιμές όλων των μεταβλητών και των καταχωρητών του επεξεργαστή.



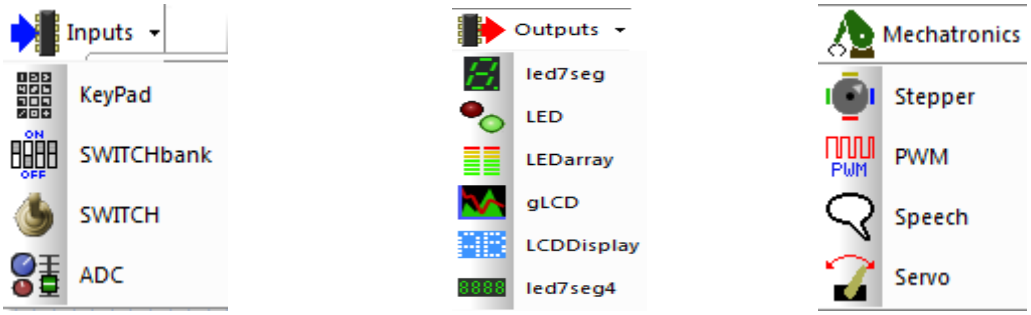
ΕΙΚΟΝΑ XLI: ΕΚΤΕΛΕΣΗ ΚΩΔΙΚΑ ΕΝΤΟΛΗ ΠΡΟΣ ΕΝΤΟΛΗ



ΕΙΚΟΝΑ XLII: ΠΕΡΙΒΑΛΛΟΝ ΛΕΙΤΟΥΡΓΙΑΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

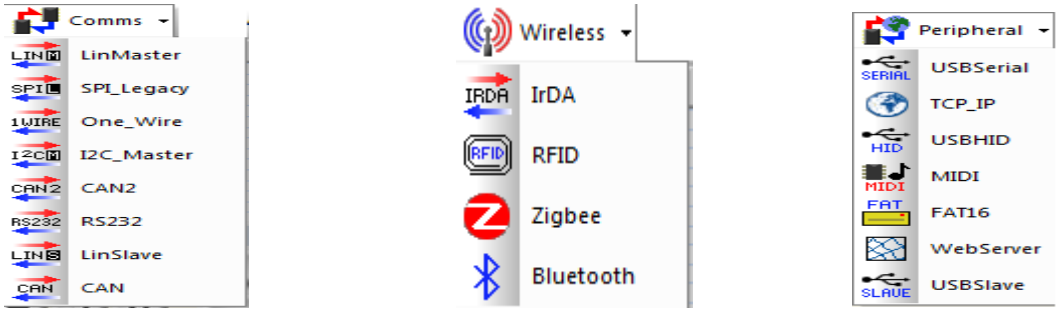
Στη μπάρα που υπάρχει στα αριστερά υπάρχουν τα εικονίδια προγραμματισμού. Το πρόγραμμα δίνει τη δυνατότητα εισαγωγής κώδικα σε γλώσσα C ή Assembly. Κατά την εξομοίωση, το συγκεκριμένο τμήμα του κώδικα δεν λαμβάνεται υπόψη Στον οριζόντιο άξονα υπάρχουν οι επιλογές εισαγωγής των περιφερειακών και των πρωτοκόλλων επικοινωνίας.

Συγκεκριμένα στην επόμενη εικόνα φαίνονται οι επιλογές που υπάρχουν όσον αφορά τους τύπους εισόδων, εξόδων, και λειτουργιών όπως έλεγχος βηματικού κινητήρα που θα χρησιμοποιηθεί.



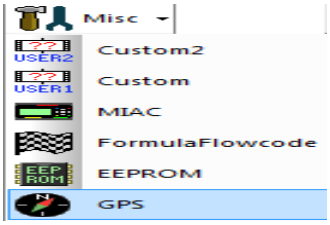
ΕΙΚΟΝΑ XLIII: ΕΠΙΛΟΓΗ ΤΥΠΟΥ ΕΙΣΟΔΩΝ-ΕΞΟΔΩΝ-ΛΕΙΤΟΥΡΓΙΩΝ

Όπως και προηγουμένως έτσι και τώρα φαίνονται κάποιες σημαντικές επιλογές σχετικά με το πρωτόκολλο ενσύρματης επικοινωνίας, ασύρματης επικοινωνίας και δικτυακής επικοινωνίας.



ΕΙΚΟΝΑ XLIV: ΕΠΙΛΟΓΗ ΠΡΩΤΟΚΟΛΛΟΥ ΕΠΙΚΟΙΝΩΝΙΑΣ

Τέλος υπάρχει η εισαγωγή λειτουργιών της μνήμης eeprom του επεξεργαστή αλλά και λειτουργιών που αφορούν την γκάμα υλικού (hardware) που παράγει η συγκεκριμένη εταιρία λογισμικού.



ΕΙΚΟΝΑ XLV: ΛΕΙΤΟΥΡΓΙΕΣ ΜΝΗΜΗΣ EEPROM

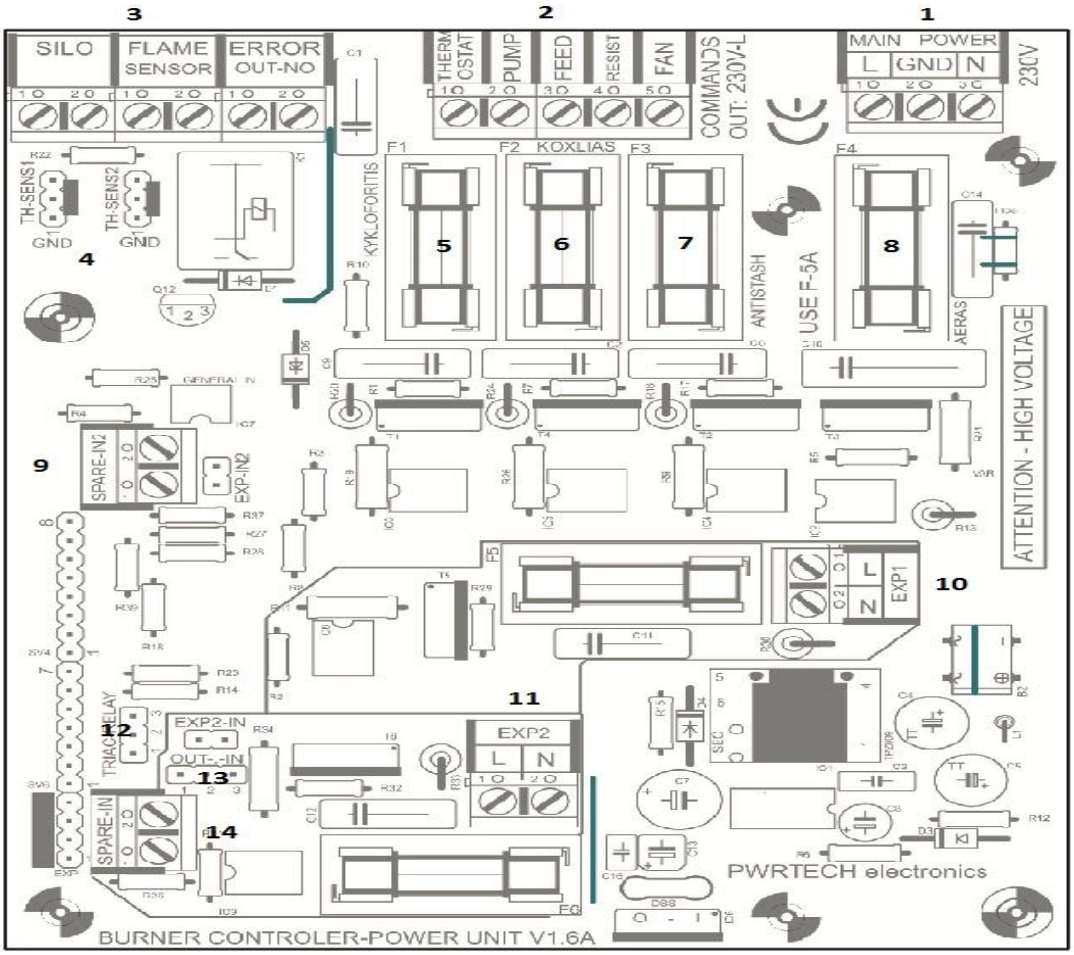
ΚΕΦΑΛΑΙΟ 3

ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΦΑΡΜΟΓΩΝ ΤΩΝ ΜΙΚΡΟΕΛΕΓΚΤΩΝ PIC ΜΕ ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ

3.1 ΠΙΝΑΚΑΣ ΛΕΒΗΤΩΝ ΒΙΟΜΑΖΑΣ HARDWARE

Σε αυτήν την ενότητα θα αναλυθεί τα υλικό του πίνακα λεβήτων βιομάζας που χρησιμοποιήθηκε σε αυτήν εργασία καθώς επίσης θα δοθούν και οι οδηγίες χρήσης του ώστε να γίνει κατανοητή η χρησιμότητα του μικροελεγκτή PIC και οι διάφορες λειτουργίες που μπορεί να εκτελέσει.

Στο σχηματικό που ακολουθεί φαίνεται μια άποψη του πίνακα λεβήτων βιομάζας και το hardware από το οποίο αποτελείται ενώ ακόμη σημειώνονται οι βαθμίδες με τους αριθμούς 1-14 και ακολουθεί επεξήγηση για την κάθε μια αναλυτικά.



ΕΙΚΟΝΑ XLVI : ΠΙΝΑΚΑΣ ΛΕΒΗΤΩΝ ΒΙΟΜΑΖΑΣ

1. Εδώ βρίσκεται η **Κεντρική τροφοδοσία MAINPOWER** από αριστερά προς τα δεξιά με Φάση L, Γείωση GND και ουδέτερος N.

2. Εδώ βρίσκονται οι **Εντολές COMMANDS .**Είναι η **έξοδος** FAN για τον ανεμιστήρα, η RESIST για την αντίσταση, η FEED για τον κοχλία και η PUMP για τον κυκλοφορητή. Επίσης να σημειωθεί πως όλες οι έξοδοι δίνουν κατά την ενεργοποίηση φάση. Τέλος φαίνεται και η **είσοδος** THERMOSTAT που είναι εντολή στον θερμοστάτη και εδώ συνδέεται η επιστροφή του θερμοστάτη η οποία πρέπει να είναι φάση.

3. Εδώ βρίσκεται το **I/O**(input/output)**.** Αρχικά υπάρχει το **Error OUT NO** που είναι μια επαφή normally open και διεγείρεται σε κάθε περίπτωση που ενεργοποιείται κάποιο ERROR στον ελεγκτή. Έπειτα ακολουθεί το **FLAME SENSOR** στο οποίο γίνεται η σύνδεση του αισθητηρίου της φλόγας σε περίπτωση που υπάρξει αυτόματη έναυση, ενώ αν δεν υπάρξει αυτόματη έναυση τότε δεν χρειάζεται το αισθητήριο φλόγας και βραχυκυκλώνετε αυτή η επαφή οπότε το σύστημα είναι σε λογική καύσης.

4. Εδώ υπάρχουν δύο θέσεις τις: **TH1 & TH2 SENS.** Στη βαθμίδα αυτή γίνεται η σύνδεση των αισθητηρίων θερμοκρασίας περιβάλλοντος και νερού. Αν υπάρχει ένα αισθητήριο τοποθετείται σε οποιαδήποτε από τις δύο θέσεις.

5. Στις θέσεις **5,6,7** και **8** φαίνεται η ένδειξη **FUSES** που είναι οι ασφάλειες προστασίας των φορτίων. Συγκεκριμένα στο No5 είναι η ασφάλεια του κυκλοφορητή, στο No6 του κοχλία, στο No7 η αντίσταση και στο No8 του ανεμιστήρα.

9. Ακολουθεί η ένδειξη **SPARE IN2** που είναι μια σύνδεση για βοηθητική είσοδο 2.

10. Εδώ υπάρχει το **EXP1** που είναι μία βοηθητική έξοδος 1 με μόνιμο ουδέτερο και φάση στην διέγερση.

11. Ακολουθεί το **EXP2** που είναι μια άλλη βοηθητική έξοδος 2 με μόνιμο ουδέτερο και φάση στην διέγερση.

12. Εδώ βρίσκεται το **RELAY/TRIAC** που είναι ένα Jumper επιλογής εξόδου error στο relay βαθμίδα 3 ή στην EXP1.

13. Εδώ είναι το **IN/OUT** που είναι ένα Jumper επιλογής λειτουργίας της βοηθητικής βαθμίδας 2 σαν IN ή OUT

14. Τέλος υπάρχει το **SPARE IN1** που είναι μία βοηθητική είσοδος 1 και συγκεκριμένα αυτή που ελέγχεται από το προηγούμενο.

Οι λειτουργίες των επιπλέον βαθμίδων και η χρηστικότητα τους κατά περίπτωση αναφέρονται παρακάτω όπου παρατίθενται και οδηγίες για την σωστή χρήση του ελεγκτή.

3.1.1 ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ ΜΙΚΡΟΕΛΕΓΚΤΗ

Ο ελεγκτής περιλαμβάνει δύο επίπεδα καταλόγων επιλογής. Ένα του χρήστη και ένα του εγκαταστάτη. Ο χρήστης εισέρχεται στον κατάλογο επιλογής του πατώντας το πλήκτρο menu και στην οθόνη του εμφανίζεται η ένδειξη FLAME SETUP (+++--) που είναι η default ρύθμιση. Πατώντας το + ή – μετακινείται η ένδειξη αυξάνοντας την ρίψη ή μειώνοντας την αντίστοιχα, κατά 10% σε κάθε μετατόπιση, σε σχέση με την αρχική του εγκαταστάτη.

Κατάλογος Επιλογής Εγκαταστάτη:

Για να εισέλθει κανείς στον κατάλογο επιλογής του εγκαταστάτη πρέπει να πατήσει διαδοχικά τα εξής πλήκτρα. (↑ ,↓ ,↓ ,↑, menu). Στον κατάλογο επιλογής του εγκαταστάτη υπάρχουν οι παρακάτω ομάδες ρυθμίσεων.

1. ΕΠΙΠΛΕΟΝ ΡΥΘΜΙΣΕΙΣ.
2. MENU ΚΟΧΛΙΑ.
3. MENU ΑΝΑΦΛΕΞΗΣ.
4. MENU ΣΒΗΣΙΜΑΤΟΣ.
5. MENU ΚΑΥΣΗΣ.

Ακολουθεί αναλυτική περιγραφή των παραπάνω.

1. ΕΠΙΠΛΕΟΝ ΡΥΘΜΙΣΕΙΣ

- Στις επιπλέον ρυθμίσεις υπάρχει η ***αντιπαγωτική λειτουργία*** του λέβητα την οποία μπορεί να ενεργοποιήσει κανείς με προεπιλεγμένη όμως την μη ενεργή κατάσταση.
- Επίσης υπάρχει η ***θερμοκρασία της αντιπαγωτικής λειτουργίας*** όπου μπορεί να επιλεχτεί το όριο θερμοκρασίας κάτω από την οποία

θα ενεργοποιείται η αντιπαγωτική λειτουργία εφόσον έχει ενεργοποιηθεί από το παραπάνω μενού με προεπιλεγμένη θερμοκρασία τους **3ºC.**

- Ακολουθεί η ***επιλογή του αισθητηρίου***. Τα αισθητήρια θερμοκρασίας είναι ψηφιακά και απολύτως παράλληλα ως προς τον ελεγκτή. Εδώ ορίζετε πιο από τα δύο είναι των νερών και πιο του περιβάλλοντος. Χωρίς τον ορισμό τους δεν εκκινεί η διαδικασία ανάφλεξης για λόγους ασφαλείας.

- Στην συνέχεια υπάρχει η ***λειτουργία της αντιστάθμισης*** όπου ενεργοποιείται ή όχι η αντιστάθμιση θερμοκρασίας των νερών του λέβητα σε σχέση με την θερμοκρασία περιβάλλοντος. Το Διαφορικό της μεταβολής είναι προκαθορισμένο και ξεκινάει να μεταβάλλεται από θερμοκρασία περιβάλλοντος 5 βαθμούς και πάνω μέχρι τους 20 βαθμούς. Κάτι σημαντικό που πρέπει να προσέξει κάποιος εδώ είναι ότι δεν πρέπει να ενεργοποιηθεί η αντιστάθμιση αν δεν έχει συνδεθεί δεύτερο αισθητήριο.

- Μια επιπλέον λειτουργία είναι η ***ένδειξη φλόγας*** όπου εμφανίζεται με την ένδειξη της φλόγας στην οθόνη η οποία ένδειξη θα βοηθήσει να αποφασιστεί το άνω και κάτω όριο φλόγας στην καύση, αλλά και στην διάγνωση με προεπιλεγμένη κατάσταση ενεργή**.**

- Το ***αισθητήριο φλόγας*** που ακολουθεί είναι επίσης σημαντικό γιατί εδώ γίνεται η ρύθμιση του τύπου του αισθητηρίου φλόγας που έχει συνδεθεί. Υπάρχουν δύο τύποι θετικής και αρνητικής μεταβολής ως προς την φωτεινότητα με προεπιλεγμένη κατάσταση θετικής.

- Η ***λειτουργία του κυκλοφορητή*** που είναι η επόμενη ρύθμιση είναι αυτή που επιτρέπει την επιλογή αν ο κυκλοφορητής θα σταματάει με το που σταματάει η εντολή του θερμοστάτη χώρου ή θα συνεχίσει να λειτουργεί κανονικά μέχρι τα νερά να πέσουν κάτω από την θερμοκρασία εκκίνησης που έχουμε ορίσει. Οι πιθανές επιλογές είναι συσχετισμός ή μη συσχετισμός με το θερμοστάτη χώρου με προεπιλεγμένη την κατάσταση συσχετισμός με θερμοστάτη. Το σημαντικό που πρέπει να προσεχθεί σε αυτή την ρύθμιση είναι αν η εγκατάσταση έχει ηλεκτροβάνα και δεν υπάρχει bypass θα πρέπει ο κυκλοφορητής να σταματάει με την διακοπή εντολής του θερμοστάτη.

- Η τελευταία ρύθμιση σε αυτόν τον κατάλογο επιλογής είναι ο ***τύπος ανεμιστήρα*** όπου μπορεί να επιλεχτεί ο τύπος του ανεμιστήρα που θα χρησιμοποιηθεί για πιο ομαλή λειτουργία μετά από την ρύθμιση στροφών. Οι επιλογές είναι σύγχρονος ή ασύγχρονος με προεπιλεγμένη κατάσταση την ασύγχρονος.

2. ΜΕΝΟΥ ΚΟΧΛΙΑ

- Στο κατάλογο επιλογής κοχλία υπάρχει η ***περίοδος του κοχλία.*** Η καύση περικλείεται στην έννοια ενός χρονικού κύκλου, στον οποίο περιμένει κανείς y χρόνο και ρίχνει x. Το σύνολο του χρόνου x+y = περίοδος κοχλία με τον χρόνο να μετριέται σε δευτερόλεπτα.

- Ακολουθεί ο ***χρόνος τροφοδοσίας*** που είναι ο χρόνος τον οποίο θα ρίξει υλικό κατά την διάρκεια μιας περιόδου (όπως x παραπάνω) με τον χρόνο να μετριέται και εδώ σε δευτερόλεπτα.

- Μία άλλη ρύθμιση είναι ο ***χρόνος τροφοδοσίας High Temp.*** Σκοπός της λειτουργίας αυτής είναι η επιβράδυνση της καύσης κατά τις υψηλές θερμοκρασίες. Εδώ ορίζεται η τροφοδοσία όπως ο χρόνος x παραπάνω η οποία ισχύει 10 βαθμούς πριν την μέγιστη θερμοκρασία που ορίζει κανείς. Το σημαντικό εδώ είναι πως αν δεν υφίσταται καμία μεταβολή τότε ρυθμίζετε και αυτή η τιμή ίδια με το χρόνο τροφοδοσίας.

- Για το πόσο χρόνο θα τροφοδοτηθεί το υλικό στο λέβητα κατά την διαδικασία έναυσης με τον χρόνο να μετριέται και εδώ σε δευτερόλεπτα υπάρχει η ρύθμιση με όνομα ***χρόνος πρώτης τροφοδοσίας***.

- Στην περίπτωση που δεν υπάρξει φλόγα κατά την διάρκεια της πρώτης τροφοδοσίας μπορεί να προστεθεί επιπλέον υλικό με την επιλογή της ***δεύτερης τροφοδοσίας*** που ουσιαστικά είναι μια δεύτερη ρίψη η οποία ενεργοποιείτε με προεπιλεγμένη την κατάσταση ενεργή ενώ επίσης ορίζεται ο χρόνος που θα ρίχνει το υλικό κατά την δεύτερη ρίψη με την επιλογή ***χρόνος δεύτερης τροφοδοσίας.***

- Τέλος υπάρχει η προεπιλογή του ***χρόνου καθυστέρησης της δεύτερης τροφοδοσίας*** όπου φαίνεται πόσος χρόνο θα περάσει από την έναρξη της έναυσης μέχρι την δεύτερη ρίψη.

3. ΜΕΝΟΥ ΑΝΑΦΛΕΞΗΣ

- Στον κατάλογο επιλογής ανάφλεξης βρίσκεται ο ***χρόνος ανεμιστήρα on.*** Κατά την προσπάθεια έναυσης ο ανεμιστήρας φυσάει παλμικά στο θάλαμο. Το πόσο φυσάει σε κάθε προσπάθεια για να ανάψει ορίζεται από εδώ.
- Ακολουθεί ο ***χρόνος ανεμιστήρα off*** όπου ορίζεται το πόσο θα περιμένει κατά την παλμική λειτουργία.
- Μια άλλη ρύθμιση είναι η ***αναμονή ανεμιστήρα*** που είναι ο χρόνος που περιμένει ο ανεμιστήρας από την έναρξη της διαδικασίας ανάφλεξης μέχρι να ξεκινήσει το παλμικό φύσημα που αναφέραμε παραπάνω.
- Επίσης υπάρχει η επιλογή της ***καθυστέρησης καύσης*** όπου γίνεται η ρύθμιση για το χρόνο παράβλεψης του αισθητηρίου φλόγας από την πρώτη ενεργοποίηση του και μετά κάτι που είναι χρήσιμο κατά τα πρώτα δευτερόλεπτα λόγω αστάθειας φλόγας.
- Η Ρύθμιση της ταχύτητας του ανεμιστήρα κατά την διάρκεια της παλμικής λειτουργίας γίνεται με την επιλογή ***στροφές ανάφλεξης.***
- Η προτελευταία μας ρύθμιση είναι το ***άνω όριο φλόγας*** όπου ρυθμίζεται το επίπεδο μέτρησης που θεωρείτε φλόγα στο θάλαμο.
- Τέλος υπάρχει το ***κάτω όριο φλόγας*** όπου ρυθμίζεται το κάτω επίπεδο όπου θεωρείτε πως η φλόγα έχει σβήσει.

4.ΜΕΝΟΥ ΣΒΗΣΙΜΑΤΟΣ

- Εδώ η μόνη επιλογή που υπάρχει είναι η ***αναμονή ανεμιστήρα*** όπου γίνεται η ρύθμιση της καθυστέρησης του ανεμιστήρα κατά το σβήσιμο. Δηλαδή το πόσο θα δουλεύει από την εντολή σβησίματος και μετά με τον χρόνο να μετριέται σε λεπτά.

5.ΜΕΝΟΥ ΚΑΥΣΗΣ

- Στον κατάλογο επιλογής καύσης υπάρχει η ***περίοδος καθαρισμού*** όπου ρυθμίζεται ανά πόσο χρόνο θα γίνεται ο καθαρισμός του λέβητα /καυστήρα κατά την διάρκεια της καύσης. Στη λειτουργία αυτή διακόπτεται η ρίψη υλικού και φυσάει με την μέγιστη ισχύ του ανεμιστήρα για καθαρισμό του χώρου καύσης.
- Ακολουθεί ο ***χρόνος καθαρισμού*** που ουσιαστικά φαίνεται η διάρκεια της παραπάνω λειτουργίας και σε περίπτωση που η παραπάνω λειτουργία δεν πρέπει να είναι ενεργή μπορεί να οριστεί μια περίοδος με μια πολύ υψηλή τιμή και τον χρόνο της λειτουργίας στο 1sec.
- Μια άλλη ρύθμιση είναι ***στροφές καύσης*** ανεμιστήρα που δείχνει την ταχύτητα ανεμιστήρα κατά την καύση.
- Επίσης υπάρχει η ***θερμοκρασία εκκίνησης του κυκλοφορητή*** που είναι η θερμοκρασία στην οποία θα εκκινεί η λειτουργία του κυκλοφορητή ενώ ταυτόχρονα θα ανάβει και το LED. Πάνω αριστερά από την οθόνη.
- Η μέγιστη θερμοκρασία που θα φτάνουν τα νερά του λέβητα κατά την καύση ρυθμίζεται από την επιλογή ***άνω όριο θερμοκρασίας νερού***.
- Τέλος υπάρχει και η ***επιλογή ΔΤ καύσης*** όπου εφόσον τα νερά φτάσουν στο max δείχνει πόσο πρέπει να πέσουν για να ξαναμπεί σε καύση.

Ο controller μπορεί να δεχθεί και να διαχειριστεί επιπλέον εντολές εξόδου και εισόδου κατ’ επιλογήν και επιθυμία του εκάστοτε κατασκευαστή.

Οι διαφοροποιήσεις που μπορούν να υλοποιηθούν ταυτόχρονα είναι:

1. Extra εντολή κατά το σβήσιμο όπως για παράδειγμα καθαρισμό φλογοαυλών .
2. Extra εντολή για χρήση δεύτερου κοχλία.
3. Συνδυασμός των παραπάνω.
4. Ενσωμάτωση λειτουργίας διαχείρισης και αυτόματης φόρτωσης από κεντρικό σιλό.
5. Επιπλέον τερματικά ελέγχου όπως για παράδειγμα το άνοιγμα πόρτας, υπερπλήρωσης και άλλα.

6. Ελεγχόμενο άνοιγμα πόρτας.

7. Οποιαδήποτε άλλη μορφή εντολής ή ελέγχου.

Κάποιες χρήσιμες συμβουλές για την διευκόλυνση και την καλύτερη χρήση του ελεγκτή είναι:

Αν κατά τη διάρκεια της ανάφλεξης, πατηθεί ταυτόχρονα ↑ και set θα μεταβεί στο menu ανάφλεξης ή αν πατηθεί ταυτόχρονα ↓ και set θα μπεί στο μενού κοχλία.

Αν κατά τη διάρκεια της καύσης, πατηθεί ταυτόχρονα ↑ και set θα μεταβεί στο μενού καύσης ή αν πατηθεί ταυτόχρονα ↓ και set θα μεταβεί στο μενού κοχλία.

Να σημειωθεί ότι οι στροφές καύσης και ανάφλεξης είναι καλύτερο να ρυθμιστούν κατά την διάρκεια λειτουργίας καθώς η φλόγα μεταβάλει τον εξολκισμό της καμινάδας και επηρεάζει την λειτουργία του ανεμιστήρα.

Παρακάτω ακολουθεί μια επεξήγηση των σφαλμάτων που μπορεί να προκύψουν:

Error 1: Αποτυχία ανάφλεξης.

Συμβαίνει όταν περάσουν 10 λεπτά από την έναρξη της διαδικασίας ανάφλεξης και δεν έχει δει το αισθητήριο φλόγα.

Error 2

Η επαφή του σιλό μας έδωσε εντολή ότι έχει αδειάσει.

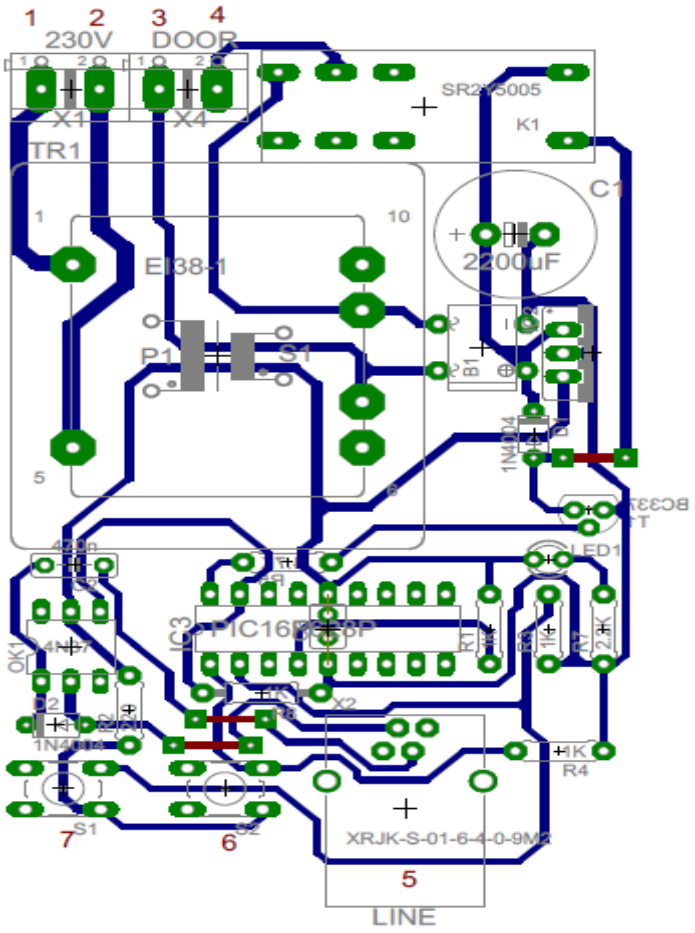
Error 3: Αισθητήριο θερμοκρασίας Off.

Συμβαίνει στη περίπτωση που το ψηφιακό αισθητήριο θερμοκρασίας δεν μπορεί να μεταδώσει σωστά τα δεδομένα. Για το λόγο αυτό σταματάει η λειτουργία έτσι ώστε να αποφευχθεί τυχόν υπερθέρμανση του δικτύου νερού του λέβητα.

Σε όλες τις περιπτώσεις Error σταματάει πλήρως η λειτουργία του πίνακα και λειτουργεί ο ανεμιστήρας με πλήρη ισχύ για δύο λεπτά έτσι ώστε να καεί το υπάρχον υλικό στο χώρο καύσης, και να μην μείνουν κατάλοιπα στο θάλαμο.

3.2 ΑΝΟΙΓΜΑ ΠΟΡΤΑΣ ΜΕΣΩ ΤΗΛΕΦΩΝΟΥ ΜΕ ΧΡΗΣΗ PIC

Στο σχηματικό που ακολουθεί φαίνεται μια εφαρμογή διαχείρισης εισόδου με ηλεκτρική κλειδαριά και το hardware από το οποίο αποτελείται ενώ ακόμη σημειώνονται οι βαθμίδες με τους αριθμούς 1-7 και ακολουθεί επεξήγηση για την κάθε μια αναλυτικά.



ΕΙΚΟΝΑ XLVII: ΠΙΝΑΚΑΣ ΓΙΑ ΑΝΟΙΓΜΑ ΠΟΡΤΑΣ ΜΕ ΧΡΗΣΗ PIC

1-2. Φαίνεται η τάση που πρέπει να δοθεί για να ανοίξει η πόρτα.

3-4. Είσοδος της ηλεκτρικής κλειδαρίας.

5. Είσοδοι όπου ανιχνεύετε ο τηλεφωνικός παλμός με την βοήθεια του μικροελεγκτή και στη συνέχεια δίνεται η εντολή για το άνοιγμα της πόρτας.

6-7. Δύο κουμπιά (push-buttons) που αυξομειώνουν τη διάρκεια της εντολής στο relay έτσι ώστε ο χρόνος να ανταποκρίνεται στις προτιμήσεις του καθενός.

Για τη ρύθμιση του χρόνου, σε αντίθεση με ένα απλό αναλογικό κύκλωμα , με το μικροελεγκτή ρυθμίζουμε με ακρίβεια τους χρόνους καθώς ο χρονισμός του γίνεται από κρύσταλλο υψηλής ακρίβειας. Έτσι είναι εμφανές το πόσες πολλές λειτουργίες μπορεί να κάνει ένας μικροελεγκτής αλλά και η ακρίβεια και απλότητα που τον χαρακτηρίζει.

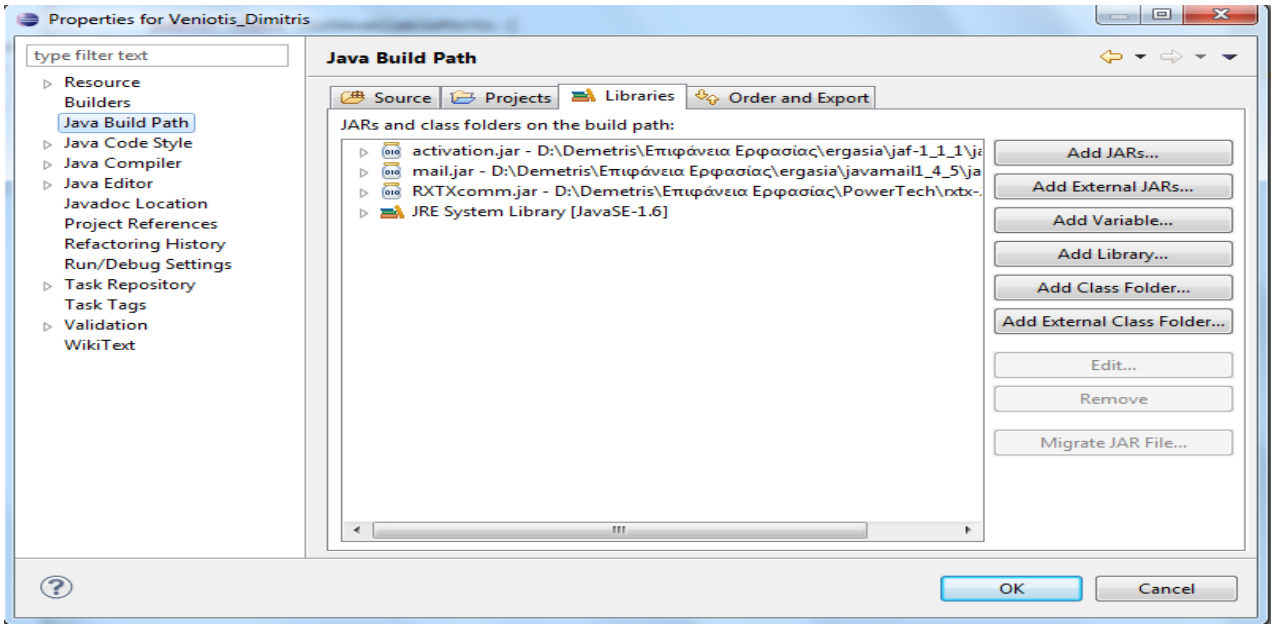
ΚΕΦΑΛΑΙΟ 4

ΚΑΤΑΣΚΕΥΗ ΛΟΓΙΣΜΙΚΟΥ

Ο προγραμματισμός για την επικοινωνία και την γραφική απεικόνιση των διαφόρων τιμών του ελεγκτή των λεβήτων βιομάζας με τον ηλεκτρονικό μας υπολογιστή έγινε μέσω της σειριακής θύρας RS232 με το λογισμικό Eclipse και την αντικειμενοστραφή γλώσσα προγραμματισμού Java. Ακολουθεί ανάλυση και επεξήγηση του κώδικα που παράχθηκε για την υλοποίηση του προγράμματος.

4.1 ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΕΙΡΙΑΚΗΣ ΘΥΡΑΣ RS232

Το πρώτο πράγμα που πρέπει να γίνει είναι να χρησιμοποιηθεί το eclipse για την δημιουργία ενός νέου project. Στη συνέχεια πρέπει να σταλθεί η καινούργια έκδοση της βιβλιοθήκης RXTX η οποία παρέχει σειριακές και παράλληλες επικοινωνίες για την Java Development Toolkit (JDK) και να αντιγραφούν τα αρχεία RXTXcomm.jar, rxtxSerial.dll and rxtxParallel.dll στον κατάλογο lib. Έπειτα από τις ιδιότητες του project γίνεται η επιλογή του java build path και libraries για να εισαχθεί η βιβλιοθήκη όπως φαίνεται παρακάτω από την επιλογή Add JARs.



ΕΙΚΟΝΑ XLVIII : ΕΙΣΑΓΩΓΗ ΒΙΒΛΙΟΘΗΚΗΣ RXTX

ΚΛΑΣΗ ListAvailablePorts

Για να ελεγχθεί τώρα ότι έγινε σωστή ρύθμιση στο περιβάλλον του eclipse για την εκτέλεση εφαρμογών που βασίζονται στην βιβλιοθήκη RXTX δημιουργείται και εκτελείται η ακόλουθη κλάση.

```
import gnu.io.CommPortIdentifier;

import java.util.Enumeration;

 public class ListAvailablePorts {

     public void list() {

         Enumeration ports = CommPortIdentifier.getPortIdentifiers();

         while(ports.hasMoreElements())
System.out.println(((CommPortIdentifier)ports.nextElement()).getName());

      } public static void main(String[] args) {

         new ListAvailablePorts().list();   } }
```

Σαν αποτέλεσμα θα πρέπει να εκτυπωθούν τα ακόλουθα :

Stable Library

==========

Native lib Version = RXTX-2.1-7

Java lib Version = RXTX-2.1-7

COM1

LPT1

Στα αποτελέσματα παρατηρείτε ο αριθμός σειριακών θυρών που διαθέτει ο υπολογιστής που είναι με το ακρωνύμιο COM1 δηλαδή μία σειριακή. Αν διέθετε κι άλλες θα εμφανιζόταν COM2, COM3 και τα λοιπά. Σε περίπτωση που διαθέτει δύο σειριακές μπορεί να γίνει σύνδεση και των δύο μαζί με ένα ειδικό καλώδιο DB9-DB9.

ΚΛΑΣΗ RS232Example ΒΟΗΘΗΤΙΚΗ

Το επόμενο που αναλύεται είναι η κλάση RS232Example για να φανεί πως γίνετε η επικοινωνία και σε άλλη ενότητα πως εμπλουτίστηκε για τις ανάγκες της εργασίας.

```
import gnu.io.CommPortIdentifier;

 import gnu.io.SerialPort;

   public class RS232Example {

     public void connect(String portName) throws Exception {
      CommPortIdentifier portIdentifier = CommPortIdentifier.getPortIdentifier
(portName);

      if (portIdentifier.isCurrentlyOwned()) {

      System.out.println("Port in use!");

      } else {
    SerialPort serialPort = (SerialPort) portIdentifier.open("RS232Example", 2000);
```

```
 serialPort.setSerialPortParams( 38400, SerialPort.DATABITS_8, SerialPort.STOP
BITS_1, SerialPort.PARITY_NONE);

        CommPortSender.setWriterStream(serialPort.getOutputStream());

        new CommPortReceiver(serialPort.getInputStream()).start();   }  }

public static void main(String[] args) throws Exception {
new RS232Example().connect(args[0]);

     CommPortSender.send(new ProtocolImpl().getMessage("HELLO"));

  }
```

Εδώ ουσιαστικά ελέγχεται η σειριακή θύρα COM1 ή αν έχουμε κάποια άλλη αν είναι απασχολημένη από κάποια άλλη εφαρμογή . Αν όχι μπορεί να χρησιμοποιηθεί. Αυτή την δουλεία την κάνει ο CommPortIdentifier. Μετά υπάρχει η ρύθμιση των απαραίτητων παραμέτρων για την επικοινωνία όπως baudrate, number of data bits, number of stop bits and possible parity bit.. Τέλος καλούνταιοι InputStream και OutputStream για αποστολή και λήψη δεδομένων. Μέσα στην main κλάση διακρίνουμε την χρήση της κλάσης CommPortSender για την αποστολή ενός μηνύματος που στην συγκεκριμένη περίπτωση είναι το HELLO το οποίο όμως δεν χρησιμοποιείτε στην εργασία αυτή γιατί δεν στέλνεται κάτι στον ελεγκτή αλλά μόνο λαμβάνει. Παρ' όλα αυτά κρίθηκε ότι έπρεπε να παρατεθεί και η λειτουργία αυτής της κλάσης η οποία αναλύεται παρακάτω για κάποιον που θέλει να δει και την αμφίδρομη επικοινωνία.

ΚΛΑΣΗ CommPortReceiver

Αυτή η κλάση είναι ένα Thread με μία run() μέθοδο η οποία εκτελείται σε ένα βρόχο που δεν σταματάει ποτέ μέχρι να γίνει διακοπή από τον χρήστη. Ενώ έχει ανοίξει η InputStream η μέθοδος in.read() περιμένει την εισερχόμενη σειρά από bytes που θα φτάσουν. Κάθε τέτοια σειρά bytes περνάει από τον διαχειριστή πρωτοκόλλου ο οποίος διαχειρίζεται όλα τα θέματα πρωτοκόλλου όπως για παράδειγμα το να αναγνωρίσει ένα μήνυμα από την ακολουθία των bytes. Όταν η end-point συσκευή κάνει break την ανοιχτή InputStream η μέθοδος in.read() αρχίζει να επιστρέφει την τιμή -1 χωρίς να την μπλοκάρει. Επειδή η αναμονή για μια εξωτερική συσκευή στον άπειρα εκτελέσιμο βρόχο χωρίς η in.read() να μπλοκάρει μπορεί να είναι καταστροφική για την επίδοση του λειτουργικού συστήματος καλούμε την μέθοδο sleep(100) για να μειώσει τον αριθμό των εκτελέσεων της μεθόδου in.read(). Παρακάτω ακολουθεί και ο κώδικας.

```
import java.io.IOException;
import java.io.InputStream;
public class CommPortReceiver extends Thread {
        InputStream in;
        Protocol protocol = new ProtocolImpl();
        CommPortReceiver(InputStream in) {
                this.in = in;    }
        public void run() {
          try {   int b;
                while(true) {
                    while((b = in.read()) != -1) {
                       protocol.onReceive((byte) b);   }
                     protocol.onStreamClosed();
                    sleep(10);   }

                } catch (IOException e) {
                e.printStackTrace();
              } catch (InterruptedException e) {
             e.printStackTrace();  } } }
```

ΚΛΑΣΗ Protocol

Η κλάση αυτή χρησιμοποιείται για να χειριστεί όλα τα εισερχόμενα bytes, διαχωρίζοντας τα και δημιουργώντας τα μηνύματα που αναγνωρίζονται. Το interface της εφαρμογής του πρωτοκόλλου επιτρέπει την ενημέρωση του διαχειριστή του πρωτοκόλλου για κάθε εισερχόμενο byte.

```
public interface Protocol {
void onReceive(byte b);
void onStreamClosed();   }
```

Παρακάτω θα αναλυθεί η κλάση ProtocolImpl η οποία υλοποιεί την Protocol και ουσιαστικά θα γίνει κατανοητό πως δημιουργείται το μήνυμα που λαμβάνεται από τον ελεγκτή. Το πως εμπλουτίστηκε για τις ανάγκες της εργασίας θα το αναλυθεί σε άλλη ενότητα αφού πρώτα γίνει κατανοητή η βασική λειτουργία αυτής της κλάσης.

ΚΛΑΣΗ ProtocolImpl

Το μήνυμα θα τελειώνει πάντα με ένα χαρακτήρα αλλαγής γραμμής. Η συλλογή των bytes που λαμβάνονται δημιουργούν ένα μήνυμα τύπου String το οποίο θα μπορεί να αναγνωριστεί. Όταν ένα νέο μήνυμα αναγνωρίζεται (εμφανίζεται ο χαρακτήρας αλαγής γραμμής) τότε η μέθοδος onMessage () καλείται να αναγνωρίσει το μήνυμα. Όταν ένα HELLO μήνυμα έρχεται στη συνέχεια ένα OK μήνυμα πρέπει να σταλεί ως απάντηση.

```
public class ProtocolImpl implements Protocol {

    byte[] buffer = new byte[1024];

    int tail = 0;

  public void onReceive(byte b) {

        if (b=='\n') {

            onMessage();  } else {

            buffer[tail] = b;

            tail++;  }  }

 public void onStreamClosed() {

         onMessage();  }

        private void onMessage() {

          if (tail!=0) {

            String message = getMessage(buffer, tail);

           System.out.println("RECEIVED MESSAGE: " + message);

      if ("HELO".equals(message)) {

          CommPortSender.send(getMessage("OK"));

     } else if ("OK".equals(message)) {

           CommPortSender.send(getMessage("OK ACK")); }
```

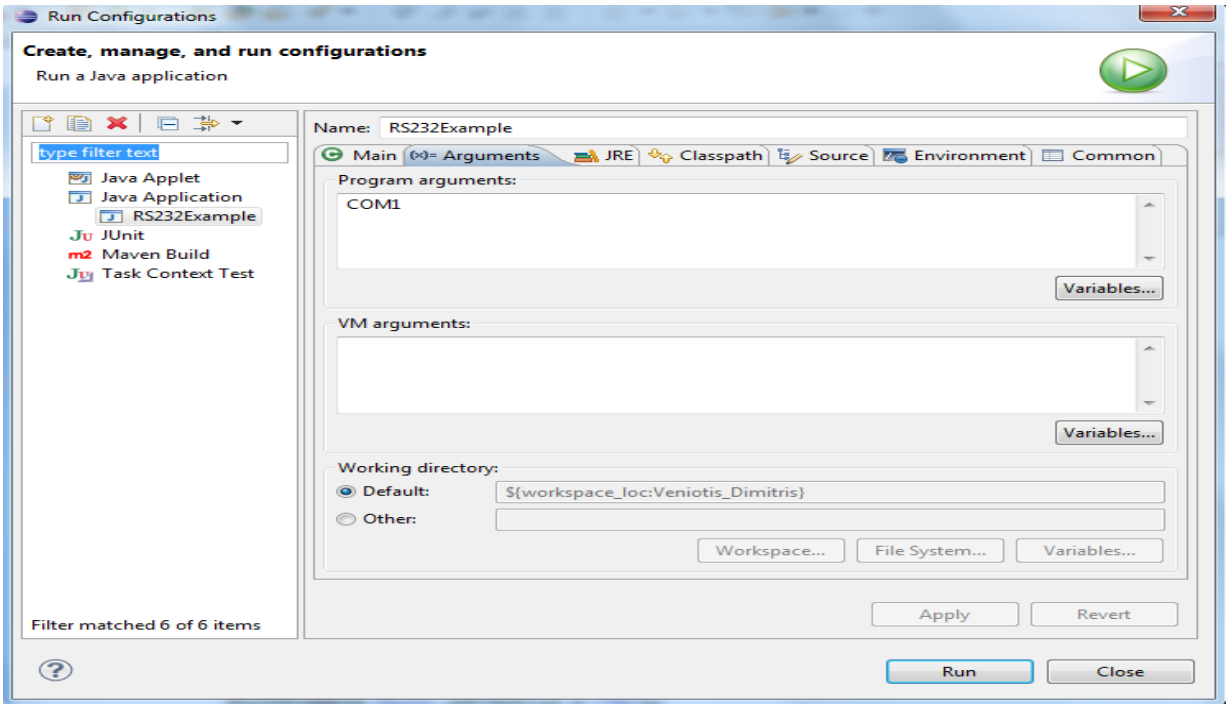
```
            tail = 0;   }  }
public byte[] getMessage(String message) {
          return (message+"\n").getBytes();  }
public String getMessage(byte[] buffer, int len) {
         return new String(buffer, 0, tail);  }  }
```

ΚΛΑΣΗ CommPortSender

Αυτή η κλάση δεν χρησιμοποιείται στην συγκεκριμένη εργασία αλλά είναι καλό να την παρατεθεί για οποιονδήποτε θελήσει να δημιουργήσει μια αμφίδρομη επικοινωνία.

```
import java.io.IOException;
import java.io.OutputStream;
public class CommPortSender {
        static OutputStream out;
    public static void setWriterStream(OutputStream out) {
        CommPortSender.out = out;   }
  public static void send(byte[] bytes) {
     try {
       System.out.println("SENDING: " + new String(bytes, 0, bytes.length));
out.write(bytes);
out.flush();
} catch (IOException e) {
e.printStackTrace();   }  }  }
```

Για να εκτελεστεί τώρα αυτό το πρόγραμμα πρέπει να γίνει δεξί κλικ στην κλάση RS232Example και στην συνέχεια να επιλεχτεί το Run As Configuration και από στο tab argument program argument να γρσφτεί COM1 δηλαδή η διαθέσιμη θύρα σειριακής επικοινωνίας που διαθέτει ο εκάστοτε υπολογιστής.



ΕΙΚΟΝΑ XLIX: ΡΥΘΜΙΣΗ ΠΑΡΑΜΕΤΡΩΝ ΓΙΑ ΤΗΝ ΕΚΤΕΛΕΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ

Αφού έγινε κατανοητό το πώς γίνεται η επικοινωνία και ο προγραμματισμός της σειριακής θύρας RS232 στην επόμενη ενότητα θα αναλυθεί πώς έγινε ο εμπλουτισμός των παραπάνω κλάσεων με τον κατάλληλο κώδικα για τις ανάγκες της επικοινωνίας με τον μικροελεγκτή PIC των βιομηχανικών λεβήτων βιομάζας.

4.2 ΕΠΙΚΟΙΝΩΝΙΑ ΜΕ ΤΟΝ ΜΙΚΡΟΕΛΕΓΚΤΗ PIC

Σε αυτήν την ενότητα θα αναλυθούν οι κλάσεις που πρέπει να τροποποιηθούν για να επιτευχθεί η επικοινωνία με τον ελεγκτή καθώς επίσης και πώς έγινε η σχεδίαση του γραφικού περιβάλλοντος αλλά και το πως μπορεί να σταλθεί κάποιο μήνυμα ηλεκτρονικού ταχυδρομείου μέσω της Java σε περίπτωση που προκληθεί μία δυσλειτουργία στον ελεγκτή.

ΚΛΑΣΗ RS232Example

Σε αυτήν την κλάση φαίνεται η σχεδίαση του γραφικού περιβάλλοντος. Αρχικά γίνεται η εισαγωγή (import) των απαραίτητων βιβλιοθηκών που είναι οι javax.swing.*// java.awt.*// java.awt.Dimension// java.awt.Graphics;// java.awt.Image// javax.swing.ImageIcon// javax.swing.JFrame;// javax.swing.JPanel.

Στη συνέχεια αφού οριστούν πρώτα οι μεταβλητές που θα χρησιμοποιούνται από τον ελεγκτή γίνεται η δημιουργία των αντίστοιχων text πεδίων όπου θα πηγαίνουν οι αντίστοιχες τιμές των μεταβλητών.

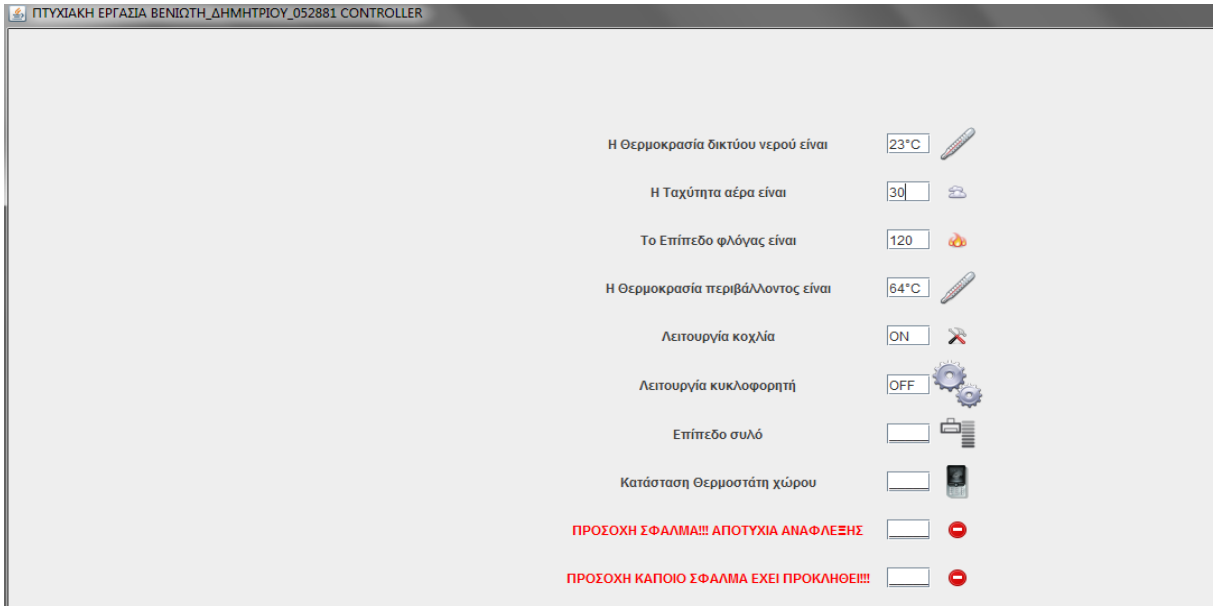
Συγκεκριμένα η δήλωση γίνεται έτσι static JFormattedTextField *text1*= new JFormattedTextField(); διότι πρέπει να είναι static γιατί θα χρησιμοποιηθούν σε άλλη κλάση όπου θα έρχεται το εισερχόμενο μήνυμα και θα τοποθετείτε η τιμή του στο αντίστοιχο πεδίο text.

Μπροστά από το εκάστοτε πεδίο text θα πρέπει να εισαχθείε ένα label για να είναι σαφές ποία μεταβλητή θα περιέχεται στο text πεδίο. Αυτή δήλωση γίνεται έτσι: JLabel label1= new JLabel("Η Θερμοκρασία δικτύου νερού είναι"); ενώ καλό είναι να τοποθετούνται και κάποια εικονίδια για την καλύτερη κατανόηση του χρήστη για το τι περιέχει το text πεδίο ορίζοντας πρώτα την εικόνα που θα χρησιμοποιηθεί ImageIcon icon=new ImageIcon("c:/Thermometer.png"); και στην συνέχεια την τοποθετείτε σε ένα lebel με αυτήν την δήλωση JLabel label1= new JLabel(icon);.

Για να τοποθετηθεί όμως στο σημείο πάνω στο γραφικό περιβάλλον όπως επίσης και το text πεδίο αλλά και το label θα πρέπει να οριστούν τα ακριβή σημεία που θα τοποθετηθούν σύμφωνα με τον άξονα x,y του καρτεσιανού συστήματος συντεταγμένων. Αυτό γίνεται ορίζοντας πρώτα μια μεταβλητή που θα μπαίνουν τα label το ένα κάτω από το άλλο δηλαδή το x θα είναι μηδέν και θα αλλάζει μόνο το y με αυτήν την δήλωση: GridBagConstraints c=new GridBagConstraints(); και στη συνέχεια δύο άλλες μεταβλητές με τις τιμές των txt,icon με τον παραπάνω τρόπο και έπειτα ορίζονται οι ακριβείς συντεταγμένες τους (c.gridx=0; //c.gridy=3;).

Πριν όμως γίνουν όλα αυτά θα πρέπει να δημιουργηθεί το πλαίσιο (frame) και να του δοθεί ένας τίτλος και στη συνέχεια να δημιουργηθεί το πάνελ που θα τοποθετούνται όλα αυτά που αναφέρθηκαν παραπάνω. Για να τοποθετηθεί η background εικόνα θα πρέπει να δημιουργηθεί μια κλάση η οποία θα επεκτείνει (extends) το jpanel και ουσιαστικά θα γίνεται το drawImage εκεί μέσα. Ο κώδικας για όλα αυτά υπάρχει στο παράρτημα.

ΕΙΚΟΝΑ L: ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ



ΕΙΚΟΝΑ LI: ΑΠΕΙΚΟΝΙΣΗ ΤΙΜΩΝ

ΚΛΑΣΗ ProtocolImpl

Στην κλάση αυτή ουσιαστικά διαβάζονται τα μηνύματα που στέλνονται από τον ελεγκτή και εμφανίζεται η αντίστοιχη τιμή τους στο κατάλληλο πεδίο όπως αναφέρθηκε στην παραπάνω κλάση.

Ο ελεγκτής για να γίνει κατανοητό για ποια τιμή μιλάει στέλνει στην αρχή του μηνύματος ένα προσδιοριστικό όπως παραδείγματος χάρη το «water_temp» και στην συνέχεια μία τιμή όπως το 23°C και έτσι γίνεται κατανοητό ότι αναφέρεται στην θερμοκρασία νερού που έχει αυτή την τιμή και έπειτα τοποθετείτε στο αντίστοιχο πεδίο text.

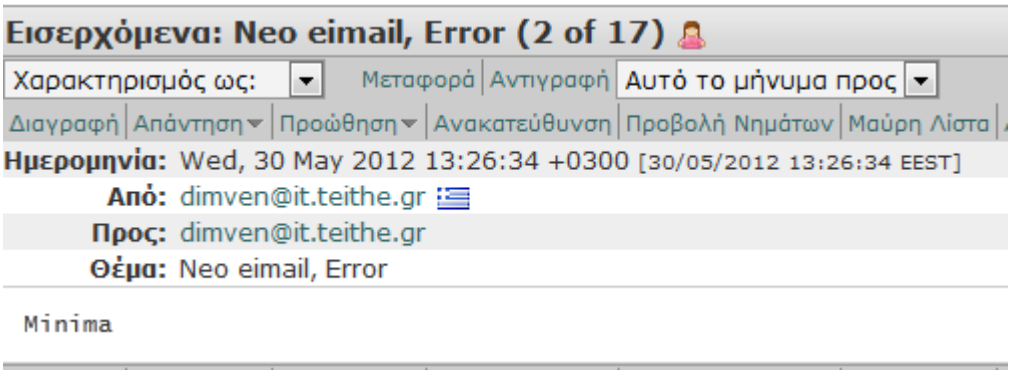
Το μήνυμα που λαμβάνετε είναι τύπου String τόσο το προσδιοριστικό όσο και ο αριθμός που ακολουθεί. Αυτό που πρέπει να γίνει είναι να ελέγχεται το μήκος των μεταβλητών που στέλνει ο μικροελεγκτής και να αποσπάτε το τελευταίο κομμάτι αυτού του μηνύματος που είναι ο αριθμός που θα πρέπει να εμφανιστεί στο αντίστοιχο πεδίο text στο γραφικό περιβάλλον το οποίο δέχεται μόνο μεταβλητές τύπου String. Αυτήν την διαδικασία την κάνουμε και για τις υπόλοιπες μεταβλητές του ελεγκτή. Ακολουθεί ένα απόσπασμα του κώδικα που κάνει αυτή την δουλεία και βλέπουμε επίσης πως καλούμε τα text πεδία από την κλάση RS232Example και το πώς κρατάμε το κομμάτι του μηνύματος που μας ενδιαφέρει.

```
if (message.startsWith("water_temp")) {
                String s1 = message.substring(10);
                 RS232Example.text1.setText(s1 + "°C");

} else if (message.startsWith("air_temp")) {
                String s2 = message.substring(8);
                 RS232Example.text1.setText(s2 + "°C");
```

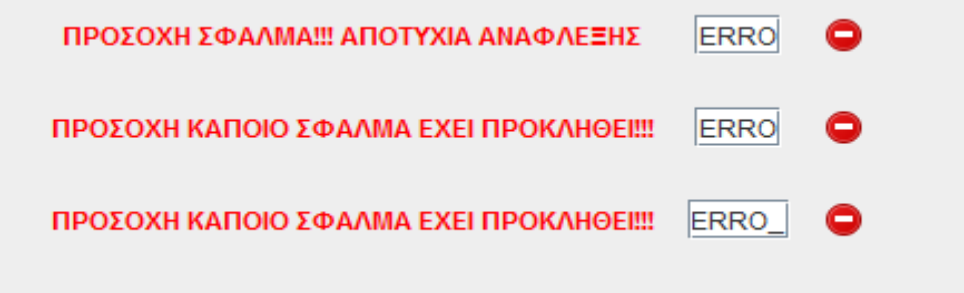
Αν τώρα κάποιες μεταβλητές του ελεγκτή δεν περιέχουν κάποια τιμή αλλά είναι τύπου Boolean όπως δηλαδή η λειτουργία του κοχλία που χρησιμοποιείται σε αυτήν την περίπτωση και ενδιαφέρει αν λειτουργεί σωστά ή όχι είναι ποιο απλά τα πράγματα γιατί αν ο ελεγκτής στείλει «koxliastrue» πρέπει να εμφανίσει στο αντίστοιχο πεδίο του γραφικού περιβάλλοντος ένα «OK» ενώ σε αντίθετη περίπτωση ένα "OFF".

```
} else if ("koxliastrue".equals(message)) {
        RS232Example.text5.setText("OK");
        // System.out.println("Ο κοχλίας λειτουργεί σωστά ");

} else if ("koxliasfalse".equals(message)) {
        RS232Example.text5.setText("OFF");
       //System.out.println("Ο κοχλίας δεν λειτουργεί σωστά ");
```

Το τελευταίο σημαντικό στοιχείο που περιέχεται σε αυτήν την κλάση είναι ότι σε περίπτωση που συμβεί κάποιο σφάλμα στον λέβητα όπως μια αποτυχία ανάφλεξης το στέλνει ο ελεγκτής και εκτός από το να εμφανιστεί στην οθόνη του γραφικού περιβάλλοντος στέλνετε και ένα προειδοποιητικό mail στον διαχειριστή του λέβητα ούτως ώστε εάν δεν μπορεί να είναι εκείνη την ώρα στον λέβητα για να ελέγξει την κατάσταση ή αν δεν ξέρει τη έχει συμβεί να δει τα mail του και να καταλάβει ότι κάτι δεν πάει καλά και στην συνέχεια να διορθώσει το σφάλμα.



ΕΙΚΟΝΑ LII: ΠΡΟΕΙΔΟΠΟΙΗΤΙΚΟ EMAIL



ΕΙΚΟΝΑ LIII:ΜΗΝΥΜΑΤΑ ERROR

Για να γίνει αυτό θα πρέπει να δημιουργηθεί μία ξεχωριστή κλάση με τις κατάλληλες βιβλιοθήκες η θα αναλυθεί παρακάτω αφού πρώτα δούμε πως χρησιμοποιείται στην κλάση αυτή που λαμβάνονται τα μηνύματα του ελεγκτή.

Αφού γίνει λήψη του μηνύματος με τον τρόπο που αναλύθηκε παραπάνω μέσα σε μία try-catch μέθοδο δηλώνεται το mail του παραλήπτη, τον τίτλο του μηνύματος που θα σταλθεί, το περιεχόμενο του ανάλογα με τον τύπο του σφάλματος που προκλήθηκε και το όνομα του αποστολέα.

```
} else if ("injection_error".equals(message)) {
                RS232Example.text9.setText("ERROR");

            try {

        Email.postMail(null, "Neo eimail,ΑΠΟΤΥΧΙΑ ΑΝΑΦΛΕΞΗΣ",
                                        "Minima", "Dimitris Veniotis");

                } catch (MessagingException e) {
                        e.printStackTrace();
                                                }
```

ΚΛΑΣΗ Email

Στην κλάση αυτή πρέπει αρχικά να περαστούν οι κατάλληλες βιβλιοθήκες που είναι η javamail και η jaf. Ορίζονται οι απαραίτητες παράμετροι που είναι ένας πίνακας από String που θα περιέχει τα mail των παραληπτών , μια άλλη μεταβλητή τύπου String για το θέμα, το κυρίως σώμα και τον αποστολέα ενώ επίσης πρέπει να δηλώσουμε την εσωτερική και εξωτερική διεύθυνση του server.

```
import javax.mail.*;
import javax.mail.internet.*;
import java.util.*;

 public class Email {

public static void postMail( String recipients[], String subject, String
message , String from) throws MessagingException    {
    boolean debug = false;

     Properties props = new Properties();
     props.put("mail.smtp.host", "smtp.jcom.net");

    Session session = Session.getDefaultInstance(props, null);
    session.setDebug(debug);


    Message msg = new MimeMessage(session);

    InternetAddress addressFrom = new InternetAddress(from);
    msg.setFrom(addressFrom);

    InternetAddress[] addressTo = new InternetAddress[recipients.length];

        for (int i = 0; i < recipients.length; i++   {
            addressTo[i] = new InternetAddress(recipients[i]);
```

```
            }
                    msg.setRecipients(Message.RecipientType.TO, addressTo);
                     msg.addHeader("MyHeaderName", "myHeaderValue");

                    msg.setSubject(subject);
                    msg.setContent(message, "text/plain");

            Transport.send(msg);
        }

}
```

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στη εργασία αυτή έγινε η ανάπτυξη ενός λογισμικού για την απεικόνιση των διαφόρων καταστάσεων ενός βιομηχανικού καυστήρα βιομάζας. Για την υλοποίηση αυτού απαιτήθηκαν γνώσεις στην γενική δομή και συμπεριφορά ενός μικροελεγκτή τύπου PIC, στις γλώσσες προγραμματισμού που χρησιμοποιούνται για τον προγραμματισμό ενός PIC. Απαραίτητες επίσης κρίθηκαν οι γνώσεις στην αντικειμενοστραφή γλώσσα προγραμματισμού JAVA με την οποία έγινε όλος ο προγραμματισμός στο κατάλληλο περιβάλλον για την γραφική απεικόνιση των διαφόρων καταστάσεων του βιομηχανικού καυστήρα βιομάζας.

Η όλη εργασία έγινε πάνω σε έναν προγραμματιζόμενο μικροελεγκτή τύπου PIC,ο οποίος προγραμματίστηκε με την διαγραμματική γλώσσα προγραμματισμού με το λογισμικό flow code. Αντίστοιχα για την δημιουργία του γραφικού περιβάλλοντος, την διασύνδεση του ελεγκτή με τον υπολογιστή και την απεικόνιση των διαφόρων τιμών του χρησιμοποιήθηκε το λογισμικό Eclipse με την αντικειμενοστραφή γλώσσα προγραμματισμού Java .

Οι μόνοι περιορισμοί της εφαρμογής τέθηκαν από τις ανάγκες της παρουσίασης, καθώς και λόγω της μη πραγματικής εφαρμογής του σε βιομηχανικό λέβητα βιομάζας. Το ίδιο το PIC που χρησιμοποιήθηκε όμως, θέτει ορισμένους περιορισμούς από το μέγεθος της μνήμης και τον αριθμό των θυρών που διαθέτει, για τις οποίες βέβαια δεν προέκυψε θέμα εξάντλησής, αλλά ακόμη και αυτό μπορούσε να ξεπεραστεί με την χρήση βοηθητικών θυρών.

Αυτό που εύκολα διαπιστώνεται στην εφαρμογή αυτή είναι η ύπαρξη περιθωρίων επέκτασής της. Έτσι σαφώς είναι δυνατή η προσθήκη επιπλέον μεταβλητών που θα θελήσει κάποιος να απεικονίσει αλλά και η δημιουργία ενός επιπλέον παραθύρου στο Eclipse όπου θα γίνεται αποθήκευση των τιμών των διαφόρων μεταβλητών όπως για παράδειγμα της θερμοκρασίας νερού ανά τακτά χρονικά διαστήματα που θα ορίζει ο εκάστοτε χειριστής έτσι ώστε με το πέρας της ημέρας να αναλύεται η λειτουργία του λέβητα βλέποντας την διακύμανση των τιμών των διαφόρων μεταβλητών.

Με τον τρόπο αυτό θα εξοικονομεί αρκετό χρόνο αφού δεν θα χρειάζεται να βρίσκεται συνεχώς μπροστά από τον λέβητα για να παρακολουθεί την διακύμανση των τιμών που τον απασχολούν αλλά με μια ματιά στο τέλος της ημέρας θα εντοπίζει τυχόν προβλήματα που προκύπτουν και θα προβαίνει στις κατάλληλες διορθώσειςΕπίσης μπορούν να προστεθούν πολλά ακόμη προγράμματα που θα έχουν έστω και μία ελάχιστη διαφορά μεταξύ τους.

Τέλος μία πάρα πολύ μεγάλη καινοτομία που θα μπορούσε να επιτευχθεί ήταν η όλη εργασία να μπορέσει να γίνει με χρήση wi-fi με αποτέλεσμα να μπορεί

ο χειριστής των εκάστοτε λεβήτων βιομάζας να ενημερώνετε ανά πάσα στιγμή για την κατάσταση ενός λέβητα βιομάζας ακόμη και από το κινητό του τηλέφωνο.

Σε αυτό το σημείο θα ήθελα να αναφερθώ σε μία δυσκολία που συνάντησα κατά την διάρκεια υλοποίησης της εργασίας. Η δυσκολία αυτή προέκυψε όταν θέλησα να απεικονίσω έναν αριθμό τύπου integer. Το μήνυμα που λαμβάνουμε είναι τύπου String και ενώ όντως το προσδιοριστικό είναι τέτοιου τύπου ο αριθμός που ακολουθεί είναι τύπου Integer.

Οπότε αρχικά θα πρέπει να διαβάσουμε το προσδιοριστικό και στην συνέχεια να κρατήσουμε την τιμή που ακολουθεί και να την μετατρέψουμε σε Integer και αφού την κρατήσουμε σε μία μεταβλητή να την ξανά μετατρέψουμε σε String γιατί το πεδίο text στο γραφικό περιβάλλον απεικονίζονται μόνο τιμές τύπου String. Αυτήν η διαδικασία πρέπει να γίνει και για τις υπόλοιπες μεταβλητές του ελεγκτή που εμπεριέχουν κάποιον αριθμό αφού γίνει η δήλωση μίας μεταβλητής (int i;) που θα κρατάει την τιμή της μεταβλητής που ακολουθεί το προσδιοριστικό. Αυτό γίνεται με την εντολή i = message.charAt(message.length() - 1); . Ενώ αυτό λειτουργεί για αριθμούς μονοψήφιους και διψήφιους δεν βρίσκει εφαρμογή σε τριψήφιους αριθμούς. Αυτό ανακάλυψα ότι συμβαίνει γιατί οι αριθμοί είναι σε μορφή Unicode και έτσι αν ο ελεγκτής στείλει έναν τριψήφιο μεταξύ του 100 και 127 θα εμφανιστεί στην οθόνη ένα σύμβολο({,)) και άλλα και θα πρέπει να μετατρέψουμε αυτό το σύμβολο στον αριθμό που αντιστοιχεί με κώδικα Java. Αν όμως ο αριθμός είναι μεγαλύτερος του 127 είναι εκτός ορίων του πίνακα Unicode και στην οθόνη μας εμφανίζεται ένας πολύ μεγάλος αριθμός. Την δυσκολία αυτή την ξεπέρασα μετά από αρκετή μελέτη όσον αφορά τους PIC και την διαπίστωση ότι τελικά αυτού του τύπου οι μικροελεγκτές μπορούν να στέλνουν τους αριθμούς σε μορφή String.

Βέβαια δίνονται και οι δυνατότητες επιλογής τελείως διαφορετικών εφαρμογών για την ανάπτυξη του προγράμματος, καθώς υπάρχει μία πλούσια ποικιλία. Ενώ είναι δυνατή και η επιλογή διαφορετικής γλώσσας προγραμματισμού για την υλοποίηση του προγράμματος ή ακόμη και ο συνδυασμός περισσοτέρων της μίας ή ακόμη και όλων των διατιθέμενων γλωσσών προγραμματισμού PIC.

Με την ανάλυση της εφαρμογής, αλλά και της διαδικασίας που απαιτείται για την λειτουργία της ουσιαστικά δημιουργήθηκε ένα εγχειρίδιο χρήσης της που εξασφαλίζει στον αναγνώστη την δημιουργία μίας ικανοποιητικής εικόνας της λειτουργίας και της δομής της εν λόγω εφαρμογής, έτσι ώστε ο χρήστης να μπορεί να αξιοποιήσει την παρέχουσα από αυτή υπηρεσία, αλλά και γενικότερα τον σκοπό εκπόνησης της παρούσας πτυχιακής εργασίας.

ΠΑΡΑΡΤΗΜΑ

Παρακάτω ακολουθούν ολοκληρωμένες όλες οι κλάσεις που δημιουργήθηκαν για τις ανάγκες της συγκεκριμένης εργασίας.

ΚΛΑΣΗ ListAvailablePorts

```
import gnu.io.CommPortIdentifier;
import java.util.Enumeration;

public class ListAvailablePorts {

    public void list() {
        Enumeration ports = CommPortIdentifier.getPortIdentifiers();

        while(ports.hasMoreElements())

System.out.println(((CommPortIdentifier)ports.nextElement()).getName());
    }

    public static void main(String[] args) {
        new ListAvailablePorts().list();
    }
}
```

ΚΛΑΣΗ CommPortReceiver

```
import java.io.IOException;
import java.io.InputStream;

public class CommPortReceiver extends Thread {

    InputStream in;
    Protocol protocol = new ProtocolImpl();

    public CommPortReceiver(InputStream in) {
        this.in = in;
    }
      public void run() {
       try {
           int b;
           while(true) {

               // if stream is not bound in.read() method returns -1
               while((b = in.read()) != -1) {
```

```
                protocol.onReceive((byte) b);
            }
            protocol.onStreamClosed();

            // wait 10ms when stream is broken and check again
            sleep(10);
          }
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

ΚΛΑΣΗ CommPortSender

```
import java.io.IOException;
import java.io.OutputStream;

public class CommPortSender {

    static OutputStream out;

    public static void setWriterStream(OutputStream out) {
        CommPortSender.out = out;
    }

    public static void send(byte[] bytes) {
        try {
            System.out.println("SENDING: " + new String(bytes, 0,
bytes.length));

            // sending through serial port is simply writing into OutputStream
            out.write(bytes);
            out.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

ΚΛΑΣΗ Protocol

```
public interface Protocol {
  // protocol manager handles each received byte
    void onReceive(byte b);

    // protocol manager handles broken stream
    void onStreamClosed();
}
```

ΚΛΑΣΗ Email

```
        import javax.mail.*;
import javax.mail.internet.*;
import java.util.*;

  public class Email {

public static void postMail( String recipients[], String subject, String
message , String from) throws MessagingException     {
    boolean debug = false;

     //Set the host smtp address
     Properties props = new Properties();
     props.put("mail.smtp.host", "smtp.jcom.net");

    // create some properties and get the default Session
    Session session = Session.getDefaultInstance(props, null);
    session.setDebug(debug);

    // create a message
    Message msg = new MimeMessage(session);

    // set the from and to address
    InternetAddress addressFrom = new InternetAddress(from);
    msg.setFrom(addressFrom);

    InternetAddress[] addressTo = new InternetAddress[recipients.length];
    for (int i = 0; i < recipients.length; i++)
    {
       addressTo[i] = new InternetAddress(recipients[i]);
    }
    msg.setRecipients(Message.RecipientType.TO, addressTo);


    // Optional : You can also set your custom headers in the Email
    msg.addHeader("MyHeaderName", "myHeaderValue");

    // Setting the Subject and Content Type
```

```
    msg.setSubject(subject);
    msg.setContent(message, "text/plain");
    Transport.send(msg);
 }

}
```

ΚΛΑΣΗ ProtocolImpl

```
import javax.mail.MessagingException;

public class ProtocolImpl implements Protocol {

 byte[] buffer = new byte[1024];
 int tail = 0;
 int i;
 String b;

 public void onReceive(byte b) {
        // simple protocol: each message ends with new line
        if (b == '\n') {
                onMessage();
        } else {
                buffer[tail] = b;
                tail++;                 } }

 public void onStreamClosed() {
        onMessage();
 }
  //When message is recognized onMessage is invoked
 private void onMessage() {

        if (tail != 0) {
                // constructing message
             String message = getMessage(buffer, tail);

                if (message.startsWith("water_temp")) {
                        String s1 = message.substring(10);
                         RS232Example.text1.setText(s1 + "°C");

        System.out.println("Η Θερμοκρασία δικτύου νερού είναι: " + i + "°C");

                } else if (message.startsWith("air_temp")) {
                        String s1 = message.substring(10);
                        RS232Example.text4.setText(s1 + "°C");

                } else if (message.startsWith("water_temp-")) {
                        String s1 = message.substring(10);
                        RS232Example.text1.setText(s1 + "°C");
```

```
} else if (message.startsWith("air_temp-")) {
        String s1 = message.substring(10);
        RS232Example.text4.setText(s1 + "°C");

} else if (message.startsWith("voltage")) {
        String s1 = message.substring(10);
         RS232Example.text3.setText(s1);

} else if (message.startsWith("fan_speed")) {
        String s1 = message.substring(10);
         RS232Example.text2.setText(s1);

} else if ("koxliastrue".equals(message)) {
        RS232Example.text5.setText("OK");
        System.out.println("Ο κοχλίας λειτουργεί σωστά ");

} else if ("koxliasfalse".equals(message)) {
        RS232Example.text5.setText("OFF");
         System.out.println("Ο κοχλίας δεν λειτουργεί σωστά ");

} else if ("pumpa".equals(message)) {
        RS232Example.text6.setText("OK");

System.out.println("Ο κυκλοφορητής λειτουργεί σωστά ");

} else if ("pumpfalse".equals(message)) {
        RS232Example.text6.setText("OFF");
System.out.println("Ο κυκλοφορητής δεν λειτουργεί σωστά ");

} else if ("silotrue".equals(message)) {
      RS232Example.text7.setText("OK");
     System.out.println("Το επίπεδο του συλό είναι κανονικό");

} else if ("silofalse".equals(message)) {
        RS232Example.text7.setText("OFF");
System.out.println("Το επίπεδο του συλό δεν είναι κανονικό");

} else if ("general_intrue".equals(message)) {
        RS232Example.text8.setText("OK");

System.out.println("Η κατάσταση του θερμοστάτη χώρου είναι κανονική");

} else if ("general_infalse".equals(message)) {
        RS232Example.text8.setText("OFF");
System.out.println("Η κατάσταση του θερμοστάτη χώρου δεν είναι
κανονική");

} else if ("injection_error".equals(message)) {
        RS232Example.text9.setText("ERROR");
```

```
System.out.println("ΠΡΟΣΟΧΗ ΣΦΑΛΜΑ!!! ΑΠΟΤΥΧΙΑ ΑΝΑΦΛΕΞΗΣ");

                        try {
                    Email.postMail(null, "Neo eimail,ΑΠΟΤΥΧΙΑ ΑΝΑΦΛΕΞΗΣ",
                                            "Minima", "Dimitris Veniotis");
                        } catch (MessagingException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }

                    } else if ("Error_2".equals(message)) {
                        RS232Example.text10.setText("ERROR");
        System.out.println("ΠΡΟΣΟΧΗ ΚΑΠΟΙΟ ΣΦΑΛΜΑ ΕΧΕΙ
ΠΡΟΚΛΗΘΕΙ!!!");
                        try {
                        Email.postMail(null, "Neo eimail, Error", "Minima",
                                            "Dimitris Veniotis");
                        } catch (MessagingException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }

                    } else if ("Error_3".equals(message)) {
                        RS232Example.text11.setText("ERROR");
         System.out.println("ΠΡΟΣΟΧΗ ΚΑΠΟΙΟ ΣΦΑΛΜΑ ΕΧΕΙ
ΠΡΟΚΛΗΘΕΙ!!!");
                        try {
                        Email.postMail(null, "Neo eimail, Error", "Minima",
                                            "Dimitris Veniotis");
                        } catch (MessagingException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }

                    } else
                         System.out.println("");   {
                        }

                      //System.out.println("RECEIVED MESSAGE: " + message);
                        }
                           tail = 0;    }

 // helper methods
 public byte[] getMessage(String message) {
         return (message + "\n").getBytes();
 }
          public String getMessage(byte[] buffer, int len) {

                  return new String(buffer, 0, tail);
         }
```

ΚΛΑΣΗ RS232Example

```
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import javax.swing.*;
import java.awt.*;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Image;
import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JPanel;


public class RS232Example {

        static JFormattedTextField text1= new JFormattedTextField();
        static JFormattedTextField text2= new JFormattedTextField();
        static JFormattedTextField text3= new JFormattedTextField();
        static JFormattedTextField text4= new JFormattedTextField();
        static JFormattedTextField text5= new JFormattedTextField();
        static JFormattedTextField text6= new JFormattedTextField();
        static JFormattedTextField text7= new JFormattedTextField();
        static JFormattedTextField text8= new JFormattedTextField();
        static JFormattedTextField text9= new JFormattedTextField();
        static JFormattedTextField text10= new JFormattedTextField();
        static JFormattedTextField text11= new JFormattedTextField();

      public void connect(String portName) throws Exception {
CommPortIdentifier portIdentifier =
CommPortIdentifier.getPortIdentifier(portName);

          if (portIdentifier.isCurrentlyOwned()) {
              System.out.println("Port in use!");
          } else {
              // points who owns the port and connection timeout
    SerialPort serialPort = (SerialPort) portIdentifier.open("RS232Example",
2000);
              // setup connection parameters
    serialPort.setSerialPortParams(
                  2400, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);
              // setup serial port writer
              CommPortSender.setWriterStream(serialPort.getOutputStream());
              // setup serial port reader
              new CommPortReceiver(serialPort.getInputStream()).start();
          }
      }

      public static void main(String[] args) throws Exception {
```

```
        // connects to the port which name (e.g. COM1) is in the first
argument
        new RS232Example().connect(args[0]);

ImagePanel panel = new ImagePanel(new
ImageIcon("c:/Desert.jpg").getImage());

JFrame frame = new JFrame("ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
ΒΕΝΙΩΤΗ_ΔΗΜΗΤΡΙΟΥ_052881 CONTROLLER");
                        frame.setVisible(true);
                        frame.setSize(600,600);

 frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

                        /*JPanel panel = new JPanel(new GridBagLayout());  */
                        frame.getContentPane().add(panel);
                        frame.add(panel);
                         frame.pack();

                        GridBagConstraints c=new GridBagConstraints();
                        GridBagConstraints w=new GridBagConstraints();
                        ImageIcon icon=new
ImageIcon("c:/Thermometer.png");

        JLabel label1=  new JLabel("Η Θερμοκρασία δικτύου νερού είναι");
                        JLabel label2=  new JLabel(icon);
                        c.gridx=0;
                        c.gridy=0;

                        w.gridx=2;
                        w.gridy=0;
                        c.insets= new Insets (15,15,15,15);
                        panel.add(label2,w);
                        panel.add(label1,c);
                        label1.setForeground(Color.darkGray);
                        panel.add(text1);
                        text1.setValue("_____");


                        GridBagConstraints a=new GridBagConstraints();
                        GridBagConstraints z=new GridBagConstraints();
                        ImageIcon icon1=new ImageIcon("c:/clouds.png");
                        JLabel label3=  new JLabel("Η Ταχύτητα αέρα είναι");
                        JLabel label4=  new JLabel(icon1);
                        c.gridx=0;
                        c.gridy=1;

                        a.gridx=1;
                        a.gridy=1;
```

```
                z.gridx=2;
                z.gridy=1;
                panel.add(label4,z);
                panel.add(label3,c);
                panel.add(text2,a);
                label3.setForeground(Color.darkGray);
                text2.setValue("_____");

                GridBagConstraints b=new GridBagConstraints();
                GridBagConstraints q=new GridBagConstraints();
                ImageIcon icon2=new ImageIcon("c:/fire.png");
        JLabel label5=  new JLabel("Το Επίπεδο φλόγας είναι");
                JLabel label6=  new JLabel(icon2);
                c.gridx=0;
                c.gridy=2;

                b.gridx=1;
                b.gridy=2;

                q.gridx=2;
                q.gridy=2;
                panel.add(label6,q);
                panel.add(label5,c);
                label5.setForeground(Color.darkGray);
                panel.add(text3,b);
                text3.setValue("_____");


                GridBagConstraints d=new GridBagConstraints();
                GridBagConstraints o=new GridBagConstraints();
                ImageIcon icon3=new
        ImageIcon("c:/Thermometer.png");
        JLabel label7=  new JLabel("Η Θερμοκρασία περιβάλλοντος είναι");
                JLabel label8=  new JLabel(icon3);
                c.gridx=0;
                c.gridy=3;

                d.gridx=1;
                d.gridy=3;

                o.gridx=2;
                o.gridy=3;
                panel.add(label8,o);
                panel.add(label7,c);
                panel.add(text4,d);
                label7.setForeground(Color.darkGray);
                text4.setValue("_____");

                GridBagConstraints e=new GridBagConstraints();
```

```
GridBagConstraints p=new GridBagConstraints();
ImageIcon icon4=new ImageIcon("c:/koxlias.png");
JLabel label9=  new JLabel("Λειτουργία κοχλία");
JLabel label10=  new JLabel(icon4);
c.gridx=0;
c.gridy=4;
e.gridx=1;
e.gridy=4;

p.gridx=2;
p.gridy=4;
panel.add(label10,p);
panel.add(label9,c);
panel.add(text5,e);
label9.setForeground(Color.darkGray);
text5.setValue("_____");

GridBagConstraints f=new GridBagConstraints();
GridBagConstraints t=new GridBagConstraints();
ImageIcon icon5=new ImageIcon("c:/engine.png");
JLabel label11=  new JLabel("Λειτουργία κυκλοφορητή");
JLabel label12=  new JLabel(icon5);
c.gridx=0;
c.gridy=5;

f.gridx=1;
f.gridy=5;

t.gridx=2;
t.gridy=5;
panel.add(label12,t);
panel.add(label11,c);
panel.add(text6,f);
label11.setForeground(Color.darkGray);
text6.setValue("_____");


GridBagConstraints g=new GridBagConstraints();
GridBagConstraints r=new GridBagConstraints();
ImageIcon icon6=new ImageIcon("c:/silo.png");
JLabel label13=  new JLabel("Επίπεδο συλό");
JLabel label14=  new JLabel(icon6);
c.gridx=0;
c.gridy=6;

g.gridx=1;
g.gridy=6;

r.gridx=2;
r.gridy=6;
```

```
                    panel.add(label14,r);
                    panel.add(label13,c);
                    panel.add(text7,g);
                    label13.setForeground(Color.darkGray);
                    text7.setValue("_____");

                    GridBagConstraints h=new GridBagConstraints();
                    GridBagConstraints s=new GridBagConstraints();
                    ImageIcon icon7=new ImageIcon("c:/thermostat.png");
          JLabel label15=  new JLabel("Κατάσταση Θερμοστάτη χώρου");
                    JLabel label16=  new JLabel(icon7);
                    c.gridx=0;
                    c.gridy=7;

                    h.gridx=1;
                    h.gridy=7;

                    s.gridx=2;
                    s.gridy=7;
                    panel.add(label16,s);
                    panel.add(label15,c);
                    panel.add(text8,h);
                    label15.setForeground(Color.darkGray);
                    text8.setValue("_____");

                    GridBagConstraints i=new GridBagConstraints();
                    GridBagConstraints n=new GridBagConstraints();
                    ImageIcon icon8=new ImageIcon("c:/error.png");
JLabel label17=  new JLabel("ΠΡΟΣΟΧΗ ΣΦΑΛΜΑ!!! ΑΠΟΤΥΧΙΑ
ΑΝΑΦΛΕΞΗΣ");
                    JLabel label18=  new JLabel(icon8);
                    c.gridx=0;
                    c.gridy=8;

                    i.gridx=1;
                    i.gridy=8;

                    n.gridx=2;
                    n.gridy=8;
                    panel.add(label18,n);
                    panel.add(label17,c);
                    panel.add(text9,i);
                    label17.setForeground(Color.red);
                    text9.setValue("_____");

                    GridBagConstraints j=new GridBagConstraints();
                    GridBagConstraints l=new GridBagConstraints();
                    ImageIcon icon9=new ImageIcon("c:/error.png");
JLabel label19=  new JLabel("ΠΡΟΣΟΧΗ ΚΑΠΟΙΟ ΣΦΑΛΜΑ  ΕΧΕΙ
ΠΡΟΚΛΗΘΕΙ!!!");
```

```
                    JLabel label20=  new JLabel(icon9);
                    c.gridx=0;
                    c.gridy=9;

                    j.gridx=1;
                    j.gridy=9;

                    l.gridx=2;
                    l.gridy=9;
                    panel.add(label20,l);
                    panel.add(label19,c);
                    panel.add(text10,j);
                    label19.setForeground(Color.red);
                    text10.setValue("_____");

                    GridBagConstraints k=new GridBagConstraints();
                    GridBagConstraints m=new GridBagConstraints();
                    ImageIcon icon10=new ImageIcon("c:/error.png");
 JLabel label21=  new JLabel("ΠΡΟΣΟΧΗ ΚΑΠΟΙΟ ΣΦΑΛΜΑ ΕΧΕΙ
ΠΡΟΚΛΗΘΕΙ!!!");
                    JLabel label22=  new JLabel(icon10);
                    c.gridx=0;
                    c.gridy=10;

                    k.gridx=1;
                    k.gridy=10;

                    m.gridx=2;
                    m.gridy=10;
                    panel.add(label22,m);
                    panel.add(label21,c);
                    panel.add(text11,k);
                    label21.setForeground(Color.red);
                    text11.setValue("_____");

                    /*JFormattedTextField text= new
JFormattedTextField();
                    text.setValue(new Integer(temp) ); */
                  }
                 }
        class ImagePanel extends JPanel {

                   private Image img;
                  public ImagePanel(String img) {
                  this(new ImageIcon(img).getImage());
                  }

                  public ImagePanel(Image img) {
                  this.img = img;
```

```
        Dimension size = new Dimension(img.getWidth(null),
img.getHeight(null));
                    setPreferredSize(size);
                    setMinimumSize(size);
                    setMaximumSize(size);
                    setSize(size);
                    setLayout(new GridBagLayout());
                  }
                  public void paintComponent(Graphics g) {
                  g.drawImage(img, 0,0, null);  }
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Αλατσαθιανός Σταμάτης, [2007], Μικροελεγκτές PIC, Εκδόσεις Β. Γκιούρδας Εκδοτική

- Φούρλας Γεώργιος Κ., [2010], Ο μικροελεγκτής PIC16F84A Αρχιτεκτονικά χαρακτηριστικά και προγραμματισμός, Εκδόσεις Φούρλας Γ.
- Predko Myke, [1997], Προγραμματίζοντας Τον Μικροελεγκτή PIC, Εκδόσείς Α. Τζιόλα Ε.
- Verle Milan, [2008], PIC Microcontrollers, mikroElektronika
- Verle Milan, [2009], PIC Microcontrollers - Programming in C, mikroElektronika

ΙΣΤΟΤΟΠΟΙ

- http://www.mikroe.com
- http://www.microchip.com
- http://www.swarthmore.edu/NatSci/echeeve1/Ref/C%20for%20PIC/C_Intro.html
- http://microplanet.gr/tutorials/microcontrollers
- http://www.technologystudent.com/pics/picdex1.htm
- http://www.javabeginner.com/java-swing/java-jtoolbar-class-example
- http://docs.oracle.com/javase/tutorial/uiswing/components/text.html
- http://www.intellog.com/blog/?p=255
- http://www.tex.unipi.gr/undergraduate/notes/diax_dedomenon/course_3.pdf
- http://www.oracle.com/technetwork/java/index-jsp-141752.html