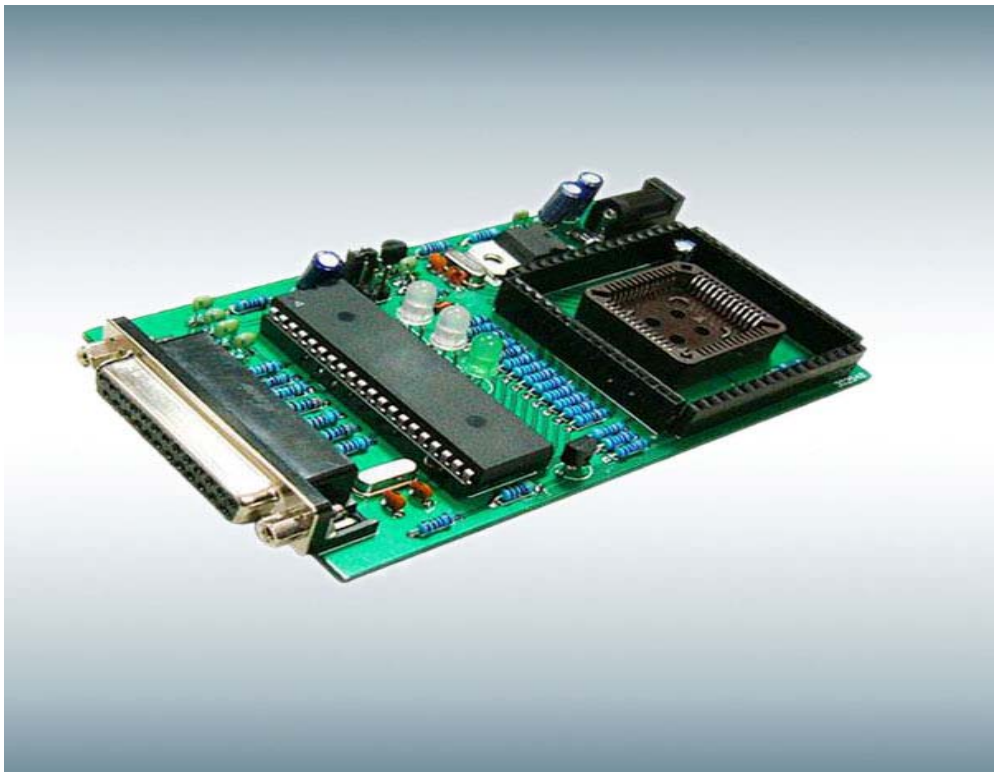
ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΠΕΙΡΑΙΑ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΚΑΤΑΣΚΕΥΗ PROGRAMMER-ΕΦΑΡΜΟΓΗ ΜΙΚΡΟΕΛΕΓΚΤΩΝ»

ΣΠΟΥΔΑΣΤΗΣ
ΓΚΙΚΑΣ Σ. ΝΙΚΟΛΑΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ
ΣΙΝΙΟΡΟΣ ΠΑΝΑΓΙΩΤΗΣ

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2013

Στον άνθρωπο που μου έμαθε πως

«Ζωή σημαίνει 7 φορές να πέφτεις,

να σηκώνεσαι 8»

Ευχαριστίες

«Θα ήθελα να ευχαριστήσω τους Καθηγητές μου, Κο Σινιόρο Παναγιώτη για την τιμή που μου έδωσε να ασχοληθώ με την Πτυχιακή μου Εργασία υπό την επίβλεψή του, την Κα Ζαχμάνογλου Άρτεμις που χωρίς την πολύτιμη βοήθειά της δεν θα είχα καταφέρει τίποτα απ' όλα αυτά, ίσως και να μην έπαιρνα ποτέ το Πτυχίο μου, και τους Κυρίους Νικολή Βασίλη και Κουρελέα Παναγιώτη για όλα τα σημαντικά αυτά μαθήματα. Τέλος θέλω να ευχαριστήσω θερμά όλους τους Καθηγητές του Τμήματος που πέρασα από τις έδρες τους, για όλα τους τα μαθήματα, είτε καλά είτε κακά, ειδικότερα Τον Προϊστάμενο Κο Μαλατέστα για τις άκρως φιλότιμες προσπάθειες που κάνει να ανεβάσει ένα παραπάνω επίπεδο το Τμήμα μας. Και φυσικά την Γραμματεία μας. Έτσι λοιπόν, με μεγάλη χαρά, γιατί παίρνω το Πτυχίο μου, αλλά και μεγάλη λύπη, γιατί κλείνει ένας ευτυχισμένος κύκλος, σας Ευχαριστώ Όλους. Δηλώνω υπεύθυνα ότι το παρόν κείμενο αποτελεί προϊόν προσωπικής μελέτης και εργασίας και πως όλες οι πηγές που χρησιμοποιήθηκαν για τη συγγραφή της δηλώνονται σαφώς είτε στις παραπομπές είτε στη βιβλιογραφία. Γνωρίζω πως η λογοκλοπή αποτελεί σοβαρότατο παράπτωμα και είμαι ενήμερος/η για την επέλευση των νομίμων συνεπειών»

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

ΠΕΙΡΑΙΑΣ ../../2013

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1.
2.
3.

Πρόλογος

Σε αυτήν την Πτυχιακή εργασία πραγματοποιείται μια κατασκευή ενός Pic Programmer για τον έλεγχο και την καύση των Pic Microcontrollers της εταιρίας Microchip.

Στο 1^{ο} Κεφάλαιο αναφέρομαι στην Ιστορική αναδρομή των μικροεπεξεργαστών, την πρώτη εμφάνιση των μικροελεγκτών, τον μετέπειτα διαχωρισμό τους σε οικογένειες ανάλογα με τα bit, το έτος κατασκευάς, όπως και τις πιο διαδεδομένες κατηγορίες μικροελεγκτών. Αναφέρομαι επίσης ονομαστικά στις Γλώσσες Προγραμματισμού και στα Εργαλεία Ανάπτυξης.

Στο 2^{ο} Κεφάλαιο αναφέρομαι αναλυτικά στον Μικροελεγκτή PIC16F84A. Την δομή του, την κατασκευή του, τις μνήμες του, τους καταχωριστές, καθώς και στις εισόδους/εξόδους του μικροελεγκτή.

Στο 3^{ο} Κεφάλαιο αναφέρομαι στην Γλώσσα Assembly. Την δημιουργία της γλώσσας, την κατηγοριοποίησή της, αλλά και σε άλλες γλώσσες προγραμματισμού. Αναλύω επακριβώς τι είναι η Assembly γλώσσα προγραμματισμού, καθώς και τα πλεονεκτήματα και μειονεκτήματά της,

Στο 4^{ο} Κεφάλαιο κάνω λόγο για τον πρόγραμμα MPLAB IDE.Το πρόγραμμα το οποίο είναι υπεύθυνο για τον προγραμματισμό του μικροελεγκτή, καθώς είναι αυτό που διαβάζει σε κώδικα HEX.

Στο 5^{ο} Κεφάλαιο παρουσιάζεται αναλυτικά το πειραματικό μέρος σε τμήματα. Η ανάλυση κυκλώματος, η πλακέτα, τα υλικά, ο σχεδιασμός και η συνδεσμολογία του Pic Programmer. Αναφέρεται αναλυτικά επίσης Το πρόγραμμα το οποίο είναι υπεύθυνο να συγχρονίζει με τον Programmer, να μετατρέπει τον κωδικό και να τον κάνει καύση να αναλύει λάθη και να βαθμονομεί τον Programmer.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

1.1 Η Εμφάνιση των Μικροεπεξεργαστών.[1,2,8]

Το αποτέλεσμα της εμφάνισης της τεχνολογίας των ολοκληρωμένων κυκλωμάτων ήταν η ενσωμάτωση σε ένα μόνο ολοκληρωμένο κύκλωμα όλης της Κεντρικής Μονάδας Επεξεργασίας, η οποία βέβαια θα έπρεπε να προγραμματίζεται για να περιέχει τις βασικότερες λειτουργίες ενός ψηφιακού υπολογιστή. Το κύκλωμα αυτό ονομάστηκε *Μικροεπεξεργαστής*. Πριν την εμφάνιση των μικροεπεξεργαστών , η Κεντρική Μονάδα Επεξεργασίας (CPU) ήταν υπεύθυνη για την εκτέλεση των εντολών σε έναν υπολογιστή. Οι κεντρικές μονάδες επεξεργασίας ήταν ογκώδη σε μέγεθος, αποτελούνταν από πολλά ολοκληρωμένα κυκλώματα και είχαν περιορισμένη ισχύ. Ο πρώτος μικροεπεξεργαστής έκανε την εμφάνιση του στις αρχές του 1972, σχεδόν τρεις δεκαετίες μετά από τους πρώτους ηλεκτρονικούς υπολογιστές. Η εξέλιξη των μικροεπεξεργαστών θυμίζει πολύ την αντίστοιχη εξέλιξη των μεσαίων υπολογιστών. Όπως δηλαδή οι σχεδιαστές των μεσαίων υπολογιστών μετέφεραν σε αυτούς τις ιδέες τους από τη σχεδίαση μεγάλων συστημάτων, έτσι και οι σχεδιαστές των μικροεπεξεργαστών υιοθέτησαν πολλά στοιχεία της οργάνωσης και της αρχιτεκτονικής των μεσαίων και μεγάλων συστημάτων. Η χρήση τους περιορίζονταν σε αριθμομηχανές (calculators) για αριθμητικές πράξεις με δεκαδικούς αριθμούς σε δυαδική αναπαράσταση των 4 bits αλλά σχετικά γρήγορα ακολούθησαν ενσωματωμένοι μικροεπεξεργαστές των 4 και 8 bits σε τερματικά συστήματα (terminals), εκτυπωτές (printers) και διάφορα αυτοματοποιημένα συστήματα. Προς τα τέλη της δεκαετίας του 1970 το κόστος των μικροεπεξεργαστών μειώθηκε σημαντικά και η εμφάνιση μικροεπεξεργαστών 8 bits με 16 bits διευθυνσιοδότηση οδήγησε στην δημιουργία του πρώτου μικροϋπολογιστή γενικής χρήσης. (microcomputer).Ονομάστηκαν μικροεπεξεργαστές λόγω της μείωσης του μεγέθους τους. Χρονολογίες–σταθμοί στην ιστορία των (μικρο)επεξεργαστών μπορούν να θεωρηθούν οι παρακάτω:

• *1971:* Η Intel παρουσιάζει τον πρώτο μικροεπεξεργαστή, τον 4004. Έχει δίαυλο δεδομένων πλάτους 4 bit, κατασκευάζεται με 2.300 τρανσίστορς και έχει συχνότητα λειτουργίας 108 kHz. Μέσα στην επόμενη χρονιά εμφανίζεται ο διάδοχος του 8008.

• *1974:* Εμφάνιση του 8–bit μικροεπεξεργαστή Intel 8080 ως αποτέλεσμα της εξέλιξης του 8008. Έχει συχνότητα λειτουργίας 2 MHz και η κατασκευή του απαιτεί 6.000 τρανσίστορς. Απάντηση της Zilog με τον Z80 και της Motorola με τον 6800, ο οποίος έχει 4.000 τρανσίστορς και ίδια συχνότητα λειτουργίας με τον 8080.

• *1975:* Η Intel αναβαθμίζει τον 8080 σε 8085.

• *1978:* Εμφανίζονται οι πρώτοι 16–bit μικροεπεξεργαστές (δηλαδή ο δίαυλος δεδομένων τους έχει πλάτος 16 bit). Η Intel παρουσιάζει τον 8086/8088, του οποίου η συχνότητα λειτουργίας έχει ανέβει πλέον στα 10 MHz και η κατασκευή του απαιτεί 29.000 τρανσίστορς. Η Motorola εμφανίζει τον 68000 με συχνότητα λειτουργίας 8 MHz, ο οποίος περιέχει 68.000 τρανσίστορς (από αυτό το γεγονός πήρε και το όνομά του).

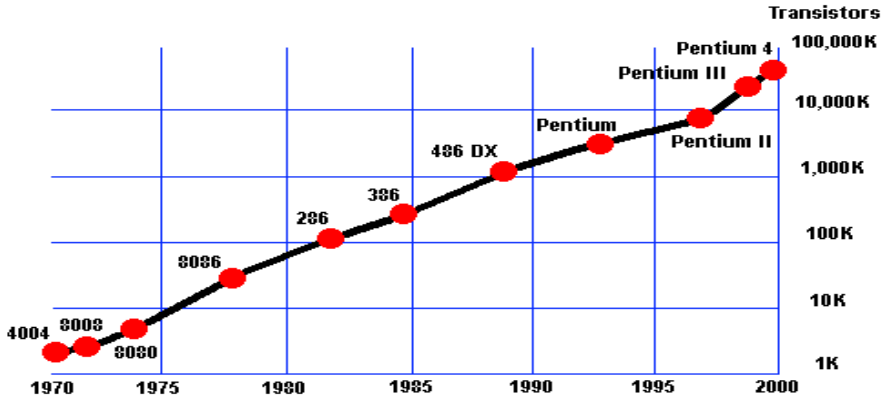
• *1982:* Εμφανίζεται ο Intel 80286, ο οποίος περιέχει 134.000 τρανσίστορς και έχει συχνότητα λειτουργίας 12,5 MHz. Αντίστοιχα η Motorola εμφανίζει τον 68010.

• *1985:* Εμφανίζονται οι πρώτοι 32–bit μικροεπεξεργαστές. Από τη μια ο Intel 80386, ο οποίος περιέχει 275.000 τρανσίστορς και συχνότητα λειτουργίας 33 MHz και από την άλλη ο Motorola 68020 με 200.000 τρανσίστορς και 16 MHz. Οι εξελίξεις πλέον είναι ραγδαίες.
• *1989:* Εμφανίζεται ο 32–bit μικροεπεξεργαστής Intel 80486, ο οποίος έχει 1.200.000 τρανσίστορς και συχνότητα λειτουργίας 50 MHz.
• *1993:* Εμφανίζεται ο Intel Pentium, ο οποίος περιέχει 3.100.000 τρανσίστορς και η συχνότητα λειτουργίας του έχει φτάσει στα 166 MHz.
• *1993:* Η Digital παρουσιάζει τον πρώτο 64–bit μικροεπεξεργαστή Alpha.
• *1997:* Η Intel ανακοινώνει τον Pentium II. Η συχνότητα λειτουργίας του βρίσκεται στα 300 MHz και το ολοκληρωμένο κύκλωμά του αποτελείται από 7.700.000 τρανσίστορς.
• *1999:* Η Intel ανακοινώνει τον Pentium III με συχνότητα λειτουργίας 450 MHz (κ φτάνοντας σταδιακά με το πέρασμα του χρόνου στο 1.13 GHz). Το ολοκληρωμένο κύκλωμα αποτελείται από 9.500.000 τρανσίστορς.
• *2001:* Η Intel ανακοινώνει τον Pentium IV. Η συχνότητα λειτουργίας του βρίσκεται στα 2.0 GHz και το ολοκληρωμένο κύκλωμά του αποτελείται από 55.000.000 τρανσίστορς.
Για μεγάλο χρονικό διάστημα οι μικροεπεξεργαστές αποτελούσαν απλά ολοκληρωμένα κυκλώματα (ICs) μικρού και μεσαίου μεγέθους των μερικών εκατοντάδων τρανζίστορς (transistor). Η ενσωμάτωση όμως ολόκληρης της CPU σε ένα κύκλωμα μείωσε σημαντικά το κόστος παραγωγής και η διαρκώς αυξανόμενη υπολογιστική τους ισχύς, είχε ως αποτέλεσμα οι μικροεπεξεργαστές να χρησιμοποιούνται οπουδήποτε. Από οποιοδήποτε μικρό ενσωματωμένο σύστημα μέχρι σε μεγάλα τερματικά συστήματα (mainframes) και υπερυπολογιστές (supercomputers).

Ο μικροεπεξεργαστή αναλαμβάνει την εκτέλεση των εντολών οι οποίες είναι εντολές σε γλώσσα μηχανής και είναι αποθηκευμένες στην κύρια μνήμη. Η μνήμη του βρίσκεται σε αρκετά ολοκληρωμένα κυκλώματα περιορισμένων αποθηκευτικών δυνατοτήτων, τα οποία το συνοδεύουν. Επίσης υποστηρίζεται και από μια πλειάδα α) ολοκληρωμένων κυκλωμάτων για να διασυνδέεται κατάλληλα και με τον εξωτερικό κόσμο μια και δεν έχει ενσωματωμένες αυτές τις δυνατότητες και β) ολοκληρωμένων κυκλωμάτων, που επιτελούν τις λειτουργίες χρονισμού και προώθησης δεδομένων στον τελικό τους προορισμό. Η ανάπτυξη της τεχνολογίας των ολοκληρωμένων κυκλωμάτων τις τελευταίες δεκαετίες έδωσε τη δυνατότητα να μπορούν να ενσωματωθούν σε ένα ολοκληρωμένο κύκλωμα όλο και πιο πολύπλοκα κυκλώματα (από τον πρώτο μικροεπεξεργαστή, που είχε 2 χιλιάδες τρανσίστορς, έχουμε φτάσει πλέον σε επεξεργαστές με πάνω από 7 εκατομμύρια τρανσίστορς σε ένα και μόνο ολοκληρωμένο κύκλωμα) με αποτέλεσμα τη γρήγορη ανάπτυξη των μικροεπεξεργαστών και την ολοένα και πιο συχνή χρήση τους τόσο σε πολύπλοκες υπολογιστικές συσκευές όσο και σε απλές οικιακές συσκευές ή συστήματα ελέγχου. Μια εντολή σε γλώσσα μηχανής είναι μια σειρά από δυαδικά ψηφία, ένα πλήθος των οποίων ψηφίων χρησιμοποιείται για την κωδικοποίηση της εντολής. Το σύνολο αυτών των εντολών χρησιμεύει ως μία διασύνδεση ανάμεσα στο λογισμικό (software) και το υλικό (hardware), δηλαδή ανάμεσα στα προγράμματα και στους επεξεργαστές.

Η λειτουργικότητα ενός μικροεπεξεργαστή εξαρτάται πλήρως από το σύνολο εντολών που είναι ικανός να εκτελέσει. Ο μικροεπεξεργαστής είναι υπεύθυνος για όλη τη λειτουργία του υπολογιστή.

Το μικρότερο μέγεθος μείωσε επίσης και τον χρόνο μεταγωγής λόγω των φυσικών παραγόντων. Έτσι οι σύγχρονοι μικροεπεξεργαστές έχουν συχνότητα ρολογιού που κυμαίνεται από εκατοντάδες megahertz έως αρκετά gigahertz. Παράλληλα, αυξήθηκε η πολυπλοκότητα και ο αριθμός των τρανζίστορ που αποτελούσαν ένα ολοκληρωμένο κύκλωμα. Ο ρυθμός αύξησης των τρανζίστορ περιγράφεται από τον νόμο του Μουρ, που ισχύει μέχρι σήμερα και προβλέπει τον διπλασιασμό του αριθμό των τρανζίστορ, που ενσωματώνονται σε ένα ολοκληρωμένο κύκλωμα, κάθε 18 μήνες.



Στους μικροεπεξεργαστές της τελευταίας γενιάς άρχισαν ήδη να εφαρμόζονται προχωρημένα στοιχεία αρχιτεκτονικής, με αποτέλεσμα σήμερα να είναι ασαφής ο διαχωρισμός ανάμεσα στους μεσαίους υπολογιστές και σε συστήματα βασισμένα σε μικροεπεξεργαστές.

Αν και η πολυπλοκότητα, το μέγεθος, η κατασκευή, και η γενική μορφή των επεξεργαστών έχει αλλάξει ριζικά τα τελευταία εξήντα χρόνια, είναι αξιοσημείωτο ότι ο βασικός σχεδιασμός και η λειτουργία τους δεν έχει αλλάξει σε μεγάλο βαθμό. Σήμερα σχεδόν όλες οι κοινές ΚΜΕ μπορούν να θεωρηθούν ως μηχανές φον Νόημαν. Καθώς ο νόμος του Μουρ εξακολουθεί να ισχύει, έχουν εκφραστεί ανησυχίες σχετικά με τα όρια της τεχνολογίας ολοκλήρωσης κυκλωμάτων με τρανζίστορ. Οι μεγάλες σμικρύνσεις των ηλεκτρονικών πυλών έχουν ξεπεράσει προβλήματα που παλαιότερα προκαλούνταν από τα υλικά κατασκευής.

Νεότερες όμως ανησυχίες προκαλούν τους ερευνητές να διερευνήσουν νέες μεθόδους υλοποίησης υπολογισμών, όπως ο κβαντικός υπολογιστής, καθώς και να διευρύνουν την χρήση του παράλληλου υπολογισμού και άλλων μεθόδων που επεκτείνουν την χρησιμότητα της υπάρχουσας αρχιτεκτονικής φον Νόημαν.

Συνοψιζοντας:

Ο σχεδιασμός των μικροεπεξεργαστών ξεκίνησε με στόχο τη δημιουργία απλών συστημάτων αυτόματου ελέγχου και τη χρήση τους σε διάφορες συσκευές. Να βελτιστοποιηθεί η απόδοση και να μειωθεί ο όγκος των αρχικων μονάδων. Η ενασχόληση των σχεδιαστων μικροεπεξεργαστών με την ανάπτυξη ολοκληρωμένων ήταν η αρχή καθώς στα επόμενα χρόνια, η ραγδαία εξέλιξη της τεχνολογίας των ολοκληρωμένων κυκλωμάτων έδωσε τη δυνατότητα ενσωμάτωσης εκατομμυρίων τρανζίστορ μέσα σε ένα ολοκληρωμένο κύκλωμα. Αυτό οδήγησε σε μικρότερους και ισχυρότερους μικροεπεξεργαστές και ως αποτέλεσμα τη χρήση του σχεδόν σε όλες τις ηλεκτρονικές συσκευές από αριθμομηχανές μέχρι πανίσχυρους υπερυπολογιστές.

Ενδεικτικά στοιχεία της εξέλιξής των σύγχρονων μικροεπεξεργαστών είναι τα εξής:

· οι σύγχρονοι μικροεπεξεργαστές να υλοποιούν άμεσα και γρήγορα μεγάλους αριθμητικούς υπολογισμούς αφού Το μήκος λέξης του μικροεπεξεργαστή μεγάλωσε από τα 16 δυαδικά ψηφία στα 32 και έπειτα στα 64 δυαδικά ψηφία

· Αυξήθηκε το πλήθος των θέσεων μνήμης που μπορεί να προσπελάσει ο μικροεπεξεργαστής.

· Οι μικροεπεξεργαστές άρχισαν να υποστηρίζουν συστήματα ιεραρχίας μνήμης με κρυφές μνήμες

· Παραλληλία στην εκτέλεση των εντολών των προγραμμάτων (Instruction Level Parallelism) με στόχο την πιο γρήγορη εκτέλεσή των εντολών. Για το σκοπό αυτό πολλοί μικροεπεξεργαστές εφαρμόζουν διοχέτευση (pipeline) ή διαθέτουν πολλαπλούς καταχωρητές και αριθμητικές και λογικές μονάδες (multiple execution units) για να μπορούν να εκτελέσουν περισσότερες από μία εντολές ταυτόχρονα.

Η ραγδαία αυτή εξέλιξη φαίνεται να συνεχίζεται και σύντομα στο μέλλον θα δούμε ακόμη πιο ισχυρούς επεξεργαστές. Η «αλλαγή κατεύθυνσης» σε πολυπύρηνους επεξεργαστές συνέβη γιατί η συνεχής αύξηση της συχνότητας χρονισμού και του αριθμού των τρανζίστορ όπως ορίζει ο Νόμος του Μούρ οδήγησαν τους κατασκευαστές σε εμπόδια που έχουν να κάνουν με την αδυναμία των υπολοίπων κυκλωμάτων να ακολουθήσουν τους ρυθμούς του επεξεργαστή και λόγω της θερμότητας που παράγεται. Οι επεξεργαστές σύντομα θα έχουν καλύτερες επιδόσεις και η χρήση τους θα επεκταθούν σε ακόμη περισσότερες εφαρμογές.

1.2. Βασική Δομή και Αρχιτεκτονική Μικροελεγκτών (Διαχωρισμός Μικροεπεξεργαστών - Μικροελεγκτών).[2,8]

Ο **μικροελεγκτής** (αγγλικά,*microcontroller*) είναι ένας τύπος , ουσιαστικά μια παραλλαγή , ο οποίος μπορεί να λειτουργήσει με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Χρησιμοποιείται ευρύτατα σε όλα τα (embedded systems) ελέγχου χαμηλού και μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρονικά καταναλωτικά προϊόντα (από ψηφιακές φωτογραφικές μηχανές έως παιχνίδια), ηλεκτρικές συσκευές και κάθε είδους αυτοκινούμενα τροχοφόρα οχήματα.Στους σύγχρονους μικροεπεξεργαστές για μη ενσωματωμένα συστήματα (πχ τους μικροεπεξεργαστές των), δίνεται έμφαση στην υπολογιστική ισχύ. Η ευελιξία ανάπτυξης διαφορετικών εφαρμογών είναι μεγάλη, καθώς η λειτουργικότητα του τελικού συστήματος καθορίζεται από τα εξωτερικά περιφερειακά τα οποία διασυνδέονται με την κεντρική μονάδα (μικροεπεξεργαστή), η οποία δεν είναι εξειδικευμένη. Αντίθετα, στους μικροεπεξεργαστές για ενσωματωμένα συστήματα (μικροελεγκτές), οι οποίοι έχουν μικρότερες ή και μηδαμινές δυνατότητες συνεργασίας με εξωτερικά περιφερειακά, αυτού του είδους, η ευελιξία είναι περιορισμένη, καθώς και η υπολογιστική ισχύς. Οι μικροελεγκτές δίνουν έμφαση στο μικρό αριθμό ολοκληρωμένων κυκλωμάτων που απαιτείται για τη λειτουργία μιας συσκευής, το χαμηλό κόστος και την εξειδίκευση.
Αναλυτικά, τα πλεονεκτήματα των μικροελεγκτών είναι:
•Αυτονομία, μέσω της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι πολλοί μικροελεγκτές δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν.
•Η ενσωμάτωση περιφερειακών σημαίνει ευκολότερη υλοποίηση εφαρμογών λόγω των απλούστερων διασυνδέσεων. Επίσης, οδηγεί σε χαμηλότερη κατανάλωση ισχύος, μεγιστοποιώντας τη φορητότητα και ελαχιστοποιεί το κόστος της συσκευής στην οποία ενσωματώνεται ο μικροελεγκτής.
•Χαμηλό κόστος.
•Μεγαλύτερη αξιοπιστία, και πάλι λόγω των λιγότερων διασυνδέσεων.
•Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές. Το πλεονέκτημα αυτό προκύπτει από το μικρότερο αριθμό και μήκος εξωτερικών διασυνδέσεων καθώς και τις χαμηλότερες ταχύτητες λειτουργίας.
•Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος), λόγω της μη δέσμευσής τους για τη σύνδεση εξωτερικών περιφερειακών.
•Μικρό μέγεθος συνολικού υπολογιστικού συστήματος.

Η βασική αρχιτεκτονική των μικροελεγκτών δεν διαφέρει από αυτή των κοινών μικροεπεξεργαστών, αν και στους πρώτους είναι απαντάται συχνά η , η οποία χρησιμοποιεί διαφορετικές αρτηρίες σύνδεσης της μνήμης προγράμματος και της μνήμης δεδομένων (πχ οι σειρές από την Atmel και από την Microchip). Στους κοινούς μικροεπεξεργαστές συνηθίζεται η ενιαία διάταξη μνήμης τύπου
Στον μικροεπεξεργαστή, το ολοκληρωμένο κύκλωμα που τον αποτελεί περιέχει μόνο την *Λογική και Αριθμητική Μονάδα* (ALU), στοιχειώδεις καταχωρητές (registers), προσωρινή RAM πολύ υψηλής ταχύτητας (cache memory) και, κάποιες φορές, τον ελεγκτή μνήμης (memory controller). Όμως, για τη λειτουργία ενός πλήρους ενσωματωμένου υπολογιστικού συστήματος, απαιτούνται πολλά εξωτερικά υποσυστήματα και περιφερειακά.
Τέτοια είναι:
- Κύκλωμα συνδετικής λογικής (glue logic) για τη σύνδεση των εξωτερικών μνημών και άλλων περιφερειακών παράλληλης σύνδεσης στην αρτηρία δεδομένων (bus) του επεξεργαστή.
- Μνήμη προγράμματος (τύπου ROM, FLASH, EPROM κλπ) η οποία περιέχει το του συστήματος. Σε κάποια μοντέλα, είναι δυνατό το κλείδωμα αυτής της μνήμης, μετά την εγγραφή της, ώστε να προστατευτεί το περιεχόμενό της από αντιγραφή.
- Μεγάλο μέγεθος μνήμης .
- Μόνιμη μνήμη αποθήκευσης παραμέτρων λειτουργίας (τύπου EEPROM ή NVRAM) η οποία να μπορεί να γράφεται τον πυρήνα του μικροελεγκτή. Αυτή η μνήμη έχει, έναντι της FLASH, το πλεονέκτημα της δυνατότητας διαγραφής και εγγραφής οποιουδήποτε μεμονωμένου byte.
- Κύκλωμα αρχικοποίησης (reset).
- Διαχειριστή αιτήσεων διακοπής (interrupt request controller) από τα περιφερειακά.
- Κύκλωμα επιτήρησης τροφοδοσίας (brown-out detection) το οποία παρακολουθεί την τροφοδοσία και αρχικοποιεί ολόκληρο το σύστημα όταν αυτή πέσει κάτω από τα ανεκτά όρια, προλαμβάνοντας έτσι την αλλοίωση των δεδομένων.
- Κύκλωμα επιτήρησης λειτουργίας (watchdog timer) το οποίο αρχικοποιεί το σύστημα, αν αυτό εμφανίσει σημάδια δυσλειτουργίας λόγω κολλήματος (hang).
- Τοπικόταλαντωτή για την παροχή παλμών χρονισμού (clock).
- Έναν ή περισσότερους χρονιστές-απαριθμητές υψηλής ταχύτητας (hardware timer-counter) για τη δημιουργία καθυστερήσεων, μέτρηση διάρκειας γεγονότων, απαρίθμηση γεγονότων και άλλων λειτουργιών ακριβούς χρονισμού.
- Ρολόι πραγματικού χρόνου (Real Time Clock, RTC) το οποίο τροφοδοτείται από ανεξάρτητη μπαταρία και γι αυτό πρέπει να έχει πολύ χαμηλή κατανάλωση ρεύματος.
- Σειρά ανεξάρτητων ψηφιακών εισόδων και εξόδων (Parallel Input-Output, PIO).

Γενικά, όλες οι οικογένειες μικροελεγκτών ενσωματώνουν τα περισσότερα από τα παραπάνω περιφερειακά, με διαφοροποιήσεις κυρίως στην ύπαρξη ή μη εσωτερικής μνήμης προγράμματος και στο είδος της.
Έτσι, υπάρχουν:
• Μικροελεγκτές χωρίς μνήμη προγράμματος, οι οποίοι χαρακτηρίζονται ως *ROM-less*. Αυτοί παρέχουν πάντοτε μια παράλληλη αρτηρία (bus) δεδομένων, πάνω στην οποία συνδέονται εξωτερικές μνήμες προγράμματος και RAM. Τέτοιοι τύποι μικροελεγκτών προορίζονται για πιο ισχυρά υπολογιστικά συστήματα ελέγχου, με μεγαλύτερες απαιτήσεις μνήμης.
• Μικροελεγκτές με μνήμη , η οποία κατασκευάζεται με το λογισμικό της (Mask ROM) ή γράφεται μόνο μια φορά (One Time Programmable, OTP). Παρέχουν τη δυνατότητα πολύ χαμηλού κόστους, όταν αγοράζονται σε πολύ μεγάλες ποσότητες.
• Μικροελεγκτές με μνήμη FLASH, οι οποία μπορούν συνήθως να προγραμματιστεί πολλές φορές. Αυτή είναι η πιο διαδεδομένη κατηγορία. Συχνά ο προγραμματισμός της μνήμης μπορεί να γίνει ακόμη και πάνω στο κύκλωμα της ίδιας της ενσωματωμένης (embedded) εφαρμογής (δυνατότητα In Circuit Programming, ISP). Αυτοί οι μικροελεγκτές έχουν ουσιαστικά αντικαταστήσει τους παλαιότερους τύπους EPROM που έσβηναν με υπεριώδη ακτινοβολία (από το ειδικό τζαμάκι).
Ανάλογα με την εφαρμογή για την οποία προορίζεται ένας μικροελεγκτής, μπορεί να περιέχει και:
• Μία ή περισσότερες ασύγχρονες σειριακές θύρες επικοινωνίας (Universal Asynchronous Receiver Transmitter,).
• Σύγχρονες σειριακές θύρες επικοινωνίας (πχ I^2C, SPI,).
• Ολόκληρα υποσυστήματα για την άμεση υποστήριξη από υλικολογισμικό (firmware) των πιο σύνθετων πρωτοκόλλων επικοινωνίας όπως CAN, HDLC,.
• (Floating Point Processing Unit, FPU), η οποία είναι πάντοτε πιο γρήγορη από την ALU του επεξεργαστή. Τέτοιες μονάδες χαρακτηρίζουν τους μικροελεγκτές με δυνατότητες ψηφιακής επεξεργασίας σήματος (Digital Signal Processing, DSP). Τα τελευταία χρόνια, με την ευρύτατη διάδοση των φορητών συσκευών ήχου και εικόνας, παρατηρείται μια τάση σύγκλισης των μικροελεγκτών με τους DSP.
• Περισσότερες από μία εισόδους για μετατροπή αναλογικού σήματος σε ψηφιακό (Analog to Digital converter, ADC).
• Μετατροπέα ψηφιακού σε αναλογικό σήμα (Digital to Analog converter, DAC).
• Ελεγκτή οθόνης υγρών κρυστάλλων (Liquid Crystal Display, LCD).
• Υποσύστημα προγραμματισμού πάνω στο κύκλωμα (τύπου ISP, βλ. παραπάνω). Χάρη σε αυτό το κύκλωμα, είναι δυνατός ο επαναπρογραμματισμός (αναβάθμιση λογισμικού) της εφαρμογής, συνδέοντας στη συσκευή μια εξωτερική συσκευή προγραμματισμού (συνήθως σε θύρα) ή ακόμη και από το διαδίκτυο.
Αυτή η δυνατότητα απαιτεί την προϋπαρξη λογισμικού υποδοχής (bootstrap) μέσα στη μνήμη προγράμματος και επομένως δεν μπορεί να γίνει σε τελείως άδεια μνήμη προγράμματος.

• Υποσύστημα προγραμματισμού (τύπου ISP) και διάγνωσης (συνήθως είναι το καθιερωμένο πρότυπο JTAG). Χάρη σε αυτό, είναι δυνατός ο προγραμματισμός της μνήμης προγράμματος χωρίς να προαπαιτείται κάποιο πρόγραμμα υποδοχής. Γι αυτό το λόγο, είναι ιδιαίτερα χρήσιμο στον αρχικό προγραμματισμό, πχ κατά τη συναρμολόγηση, ή σε περίπτωση σφάλματος (bug) στο λογισμικό υποδοχής το οποίο να καθιστά αδύνατη την κανονική αναβάθμιση.

1.3. Διαδεδομένες Κατηγορίες Μικροελεγκτών.[3,5,6]

Λόγω του ισχυρότατου ανταγωνισμού αλλά και της τάσης ενσωμάτωσης των μικροελεγκτών σε κάθε ηλεκτρική και ηλεκτρονική συσκευή, η βιομηχανία μικροελεγκτών έχει καταλήξει στην παραγωγή ανταγωνιστικών μοντέλων μαζικής παραγωγής καθώς και μικροελεγκτών για πιο εξειδικευμένες εφαρμογές. Έτσι διακρίνονται οι εξής κυρίως κατηγορίες:

• Μικροελεγκτές (καμμιά φορά 4-bit αλλά συνήθως 8-bit) πολύ χαμηλού κόστους, γενικής χρήσης, με πολύ μικρό αριθμό ακροδεκτών (ακόμη και λιγότερους από 8). Σχεδιάζονται με έμφαση στη χαμηλή κατανάλωση ισχύος και την αυτάρκεια, ώστε να χρειάζονται ελάχιστα ή και καθόλου εξωτερικά εξαρτήματα και να μη μπορεί να αντιγραφεί εύκολα το εσωτερικό λογισμικό τους. Απουσιάζει η δυνατότητα επέκτασης της μνήμης τους. Μερικά μοντέλα είναι ευρέως γνωστά στους ερασιτέχνες ηλεκτρονικούς, όπως πχ οι περισσότεροι μικροελεγκτές των σειρών PIC (Microchip), AVR (Atmel) και (Intel, Atmel, Dallas κα).

• Μικροελεγκτές (συνήθως 8-bit αλλά και 16 ή 32-bit) χαμηλού κόστους, γενικής χρήσης, με μέτριο έως σχετικά μεγάλο αριθμό ακροδεκτών. Διαθέτουν μεγάλο αριθμό κοινών περιφερειακών, όπως θύρες UART, I2C, SPI ή CAN, μετατροπείς αναλογικού σε ψηφιακό και ψηφιακού σε αναλογικό. Στους κατασκευαστές της Άπω Ανατολής (Ιαπωνία, Κορέα), συνηθίζεται η ενσωμάτωση ελεγκτών οθόνης υγρών κρυστάλλων και πληκτρολογίου. Μερικές φορές παρέχουν δυνατότητα εξωτερικής επέκτασης της μνήμης τους.

• Μικροελεγκτές (κυρίως 32-bit) μέσου κόστους, γενικής χρήσης, με μεγάλο αριθμό ακροδεκτών. Χαρακτηρίζονται από έμφαση στην ταχύτητα εκτέλεσης εντολών, υψηλή αυτάρκεια περιφερειακών και μεγάλες δυνατότητες εσωτερικής ή εξωτερικής μνήμης προγράμματος (FLASH) και RAM. Στο χώρο αυτό έχουν ισχυρή παρουσία οι αρχιτεκτονικές με υψηλή μεταφερσιμότητα λογισμικού (portability) από τον ένα στον άλλο κατασκευαστή. Πχ μεταξύ των μικροελεγκτών τύπου ή MIPS, το σύνολο των βασικών εντολών που αναγνωρίζει η ALU είναι ακριβώς το ίδιο, μειώνοντας έτσι τις μεγάλες αλλαγές στο λογισμικό, όταν στο μέλλον ο πελάτης υιοθετήσει ένα μικροελεγκτή άλλου κατασκευαστή (αρκεί, φυσικά, να υποστηρίζει κι αυτός το σύνολο εντολών ARM ή MIPS, αντίστοιχα).

• Μικροελεγκτές εξειδικευμένων εφαρμογών, οι οποίοι ενσωματώνουν συνήθως κάποιο εξειδικευμένο πρωτόκολλο επικοινωνίας το οποίο υλοποιείται πάντοτε σε hardware. Τέτοιοι μικροελεγκτές χρησιμοποιούνται σε τηλεπικοινωνιακές συσκευές.

Η μεγάλη μερίδα πωλήσεων των μικροελεγκτών εξακολουθεί να αφορά αυτούς των 8-bit, καθώς είναι η κατηγορία με το χαμηλότερο κόστος και το μικρότερο μέγεθος λογισμικού για το ίδιο αποτέλεσμα, ιδίως επειδή οι σύγχρονες οικογένειες μικροελεγκτών 8-bit έχουν πολύ βελτιωμένες επιδόσεις σε σχέση με το παρελθόν

1.4 Η Εξέλιξη των Μικροεπεξεργαστών.[4,9,10]

4 bit Μικροεπεξεργαστές:

Θεωρείται ότι ο Intel 4004 είναι ο πρώτος μικροεπεξεργαστής, αν και στις αρχές τις δεκαετίας του 1970 τρεις μικροεπεξεργαστές έκαναν την εμφάνιση τους, κατασκευασμένοι από τρεις διαφορετικές εταιρείες. Οι μικροεπεξεργαστές αυτοί ήταν ο 4004 της Intel, ο TMS 1000 της Texas Instruments (TI) και ο Central Air Data Computer (CADC) της Garrett AiResearch. Ο Intel 4004 παρουσιάστηκε το 1971 από τον Ted Hoff και το συνεργάτη του Stan Mazor, μηχανικούς της Intel, στους οποίους ανατέθηκε η ανάπτυξή του. Η Ιαπωνική εταιρεία Busicom είχε ζητήσει από την Intel την ανάπτυξη του για χρήση σε αριθμομηχανές. Ο Intel 4004 ήταν ένας 4bit επεξεργαστής (ο μικροεπεξεργαστής λαμβάνει 4 bit από την μνήμη κάθε φορά με σκοπό να τα επεξεργαστεί), που αποτελούταν από περίπου 2300 τρανζίστορς με συχνότητα ρολογιού 108 KHz. Επιπλέον, εκτελούσε 60000 πράξεις το δευτερόλεπτο και μπορούσε να δει 640 bytes μνήμης. Επιπλέον, το 1971, η Intel ανακοίνωσε τον πρώτο μικροϋπολογιστή, το σύστημα MCS-4, στο οποίο χρησιμοποιήθηκε ο 4004, το 4001 ROM chip, το 4002 RAM chip και το 4003 shift register chip για σειριακή επικοινωνία. Ο 4004 ήταν ένα τεχνολογικό αποκορύφωμα για την εποχή του παρότι ήταν πολύ περιορισμένων δυνατοτήτων. Σχεδόν ταυτόχρονα το 1971, το The Smithsonian Institution ισχυρίζονταν ότι οι μηχανικοί Gary Boone και Michael Cochran είχαν επίσης ολοκληρώσει τον πρώτο μικροεπεξεργαστή, ο οποίος ονομάστηκε TMS 1000 και διατέθηκε στην αγορά το 1974. Λίγα χρόνια πιο πριν, το 1968, ο Ray Holt, απόφοιτος του California Polytechnical University, είχε φτιάξει τον F14 CADC. Αυτό ομώς έγινε γνωστό 30 χρόνια αργότερα, το 1998, όταν το Αμερικανικό Ναυτικό (US Navy) έκανε γνωστά δημοσίως έγγραφα που αποδείκνυαν τη χρήση του μικροεπεξεργαστή σε πολεμικά αεροπλάνα.Ωστόσο επιστημονικές δημοσιεύσεις και βιβλιογραφία αναφέρουν ότι ο MP944 ήταν ο πρώτος μικροεπεξεργαστής, ο οποίος χρησιμοποιούνταν στο F-14 Tomcat πολεμικό αεροσκάφος, αλλά δεν αποτελούνταν από ένα μοναδικό chip και δεν ήταν γενικής χρήσης.

8 bit Μικροεπεξεργαστές:

Τους 4 bits επεξεργαστές ακολούθησαν οι 8 bits, με τον σημαντικότερο από αυτούς τον Intel 8008 ο οποίος ήταν ο πρώτος εμπορικός 8 bit μικροεπεξεργαστής. Η ανάπτυξή του ξεκίνησε το 1971 με επικεφαλή τον Federico Faggin και ολοκληρώθηκε το 1972, αφού ξανασχεδιάστηκε καθώς η αρχική σχεδίαση είχε διαρροές ηλεκτρικού φορτίου από τις συσκευές μνήμης. Η συχνότητα ρολογιού ήταν στα 200 KHz, ενώ το chip χρησιμοποιούσε 3500 κρυσταλλολυχνίες και μπορούσε να δει 16 Kbytes μνήμης. Χρησιμοποιήθηκε στον Micral το 1973, ο οποίος ήταν ο πιο σύγχρονος υπολογιστής που τροφοδοτήθηκε από έναν μικροεπεξεργαστή 8008 ενώ ένα χρόνο αργότερα η Scelbi χρησιμοποιεί τον 8008 στον υπολογιστη 8H. Με τον 8008 ξέσπασε μεγάλο ενδιαφέρον για την ανάπτυξη μικροεπεξεργαστών το οποιο είχε αποτέλεσμα να αυξηθούν οι απαιτήσεις στην ταχύτητα, επικοινωνία με το περιβάλλον, και πιο πολλές εντολές και εισόδους δεδομένων.

Το 1974 ακολούθησε η κυκλοφορία του Intel 8080, ο οποίος έτρεχε στα 2 MHz, μπορούσε να δει 64 Kbytes μνήμης και περιείχε 6.000 τρανζίστορ. Σημαντικά γεγονότα που ακολούθησαν ήταν η ανάπτυξη του λειτουργικού συστήματος CP/M για τον Intel 8080 από τον Gary Kildall της εταιρίας Microcomputer Applications Associates, η σχεδίαση του υπολογιστή Altair 8800 ο οποίος χρησιμοποιούσε Intel 8080 με 256 bytes RAM και η ανάπτυξη της γλώσσας Microsoft Basic, από τον Bill Gates και τον Paul Allen, για τους μικροεπεξεργαστές της Intel. Τα γεγονότα αυτά θεωρούνται από πολλούς ότι οδήγησαν στη ραγδαία ανάπτυξη των προσωπικών υπολογιστών.

Σύντομα μετά τη κυκλοφορία του 8080, η Motorola κυκλοφόρησε τον 6800, 8 bit επεξεργαστη. Είχε 4000 τρανζίστορ, 78 εντολές, σήμα χρονισμού στα 1 ή 2 MHz με 16 bit πλάτος διαύλου διευθύνσεων και ήταν ένας από τους πρώτους μικροεπεξεργαστές με καταχωρητή δείκτη (index register). Ο 6800 ήταν καλοσχεδιασμένος, παρόλα τα προβλήματα με την παραγωγή και γι' αυτό και χρησιμοποιήθηκε σε πολλά συστήματα. Έτσι η εταιρία MITS ξεκίνησε την σχεδίαση ενός Altair βασισμένου στον Motorola 6800 και λίγο αργότερα παρουσιάστηκε ο υπολογιστής Sphere I, με επεξεργαστή τον Motorola 6800, 4Kbytes RAM, πρόγραμμα ROM monitor, πληκτρολόγιο και διασύνδεση βίντεο.

Προς τα τέλη του 1970 η Tektronix σχεδίασε τον 4051 βασισμένο στον επεξεργαστή της Motorola με 32 Kbytes RAM πληκτρολόγιο και high speed data cartridge. Ο 4051 προορίζονταν για χρήστες που προγραμμάτιζαν σε BASIC.

Τον 6800 ακολούθησε ο 6809, ένα από τα πιο ισχυρά σχέδια μικροεπεξεργαστή και επίσης ένα από τον πιο σύνθετα σχέδια λογικής που έγιναν ποτέ στην παραγωγή οποιουδήποτε μικροεπεξεργαστή. Αρχικά χρησιμοποιούσε σήμα χρονισμού στο 1 MHz στο μοντέλο 68A09, ύστερα στο στο 1.5 MHz στο μοντέλο 68A09 και στα 2 MHz στο 68B09. Ο 6809 χρησιμοποιήθηκε, μεταξύ άλλων συστημάτων και στη κονσόλα παιχνιδιών Vectrex.

Το 1975, η ανταγωνίστρια εταιρεία MOS Technology κυκλοφορεί τον επεξεργαστή 6502, μια παραλλαγή του 6800, ο οποίος χρησιμοποιούσε δυο 8 bit καταχωρητές. Είχε 5000 τρανζίστορ, 56 εντολές, σήμα χρονισμού αρχικά 20 KHz μέχρι 4 MHz με 16 bit πλάτος διαύλου διευθύνσεων.

Χρησιμοποιήθηκε σε πολλά συστήματα, μερικά από αυτά οι κονσόλες ηλεκτρονικών παιχνιδιών Atari 2600, Nintendo Entertainment System/NES και οι πρωτοι οικιακοί υπολογιστές Commodore 64 και Apple II.

Το 1976 φτιάχνεται ο Z80 από την εταιρεία Zilog. Επίσης 8bit μικροεπεξεργαστής, βασισμένος στον 8080, του οποίου η γλώσσα μηχανής είναι υπερσύνολο αυτής του Intel 8080. Είχε σήμα χρονισμού στα 3.5 MHz με 16 bit πλάτος διαύλου διευθύνσεων, ενώ μπορούσε να δει 64 Kbytes μνήμης. Το χαμηλό κόστος, η μικρή συσκευασία καθώς και άλλα χαρακτηριστικά τον έκαναν πολύ δημοφιλή στη δεκαετία του 80.

Τέλος, το 1982, ήδη στην αρχή της εποχής των 16 bit επεξεργαστών κυκλοφορεί ο RCA(CDP)1802 της RCA και χρησιμοποιήθηκε για την κατασκευή των δορυφόρων Voyager, Viking και του διαστημοπλοίου Γαλιλαίος. Είχε πολύ μικρή κατανάλωση και ήταν ανθεκτικός στην κοσμική ακτινοβολία και τις ηλεκτροστατικές αποφορτίσεις.

16 bit Μικροεπεξεργαστές

Ο πρώτος 16bit μικροεπεξεργαστής multi-chip ήταν ο IMP 16 της National, που εισήχθη στις αρχές του 1973. Το 1974 εισήχθη h 8 bit έκδοσή του IMP-8.Δύο χρόνια αργότερα, η National εισήγαγε το πρώτο 16bit single-chip μικροεπεξεργαστή, τον PACE, ο οποίος ακολουθήθηκε αργότερα από μια NMOS έκδοση, το INS8900.

Το 1976, εμφανίζεται ο TMS 9900 της Texas Instruments, ένας από τους πρώτους καθαρά 16bit μικροεπεξεργαστές. Ο TMS 9900 δεν είχε καθόλου εσωτερικούς καταχωρητές, εκτός από έναν που όριζε την θέση των καταχωρητών του στην RAM, όπου αποθηκεύονται. Η σχεδίαση επέτρεπε την ταχύτατη αλλαγή context, αφού για να αλλάξουν όλοι οι καταχωρητές και να κληθεί μια συνάρτηση, πρέπει να αλλάξει ο μοναδικός εσωτερικός καταχωρητής. Η συγκεκριμένη σχεδίαση είχε νόημα για την εποχή της, διότι η εσωτερική μνήμη ήταν πιο αργή από την εξωτερική.

Η Intel επανέρχεται στο προσκήνιο αναβαθμίζοντας το σχέδιο του 8080 στον 16bit Intel 8086, το πρώτο μέλος της x86 οικογένειας που χρησιμοποιούν οι περισσότεροι σύγχρονοι υπολογιστές. Ο 8086 είχε 10 φορές καλύτερη απόδοση από τον 8080. Η Intel εισήγαγε τον 8086 ως οικονομικώς αποδοτικό τρόπο μεταφοράς του λογισμικού από τον 8080, και πέτυχε κερδίζοντας την εμπιστοσύνη πολλών επιχειρήσεων με εκείνη την προϋπόθεση. Ο 8086 είχε 29.000 τρανζίστορ, ταχύτητα λειτουργίας στα 10 MHz, ενώ χρησιμοποιούσε καταχωρητές των 16 bit και δίαυλο δεδομένων των 16 bit.

Επιπλέον, μπορούσε να δει 1 Mbyte μνήμης. Τον Ιούνιο του 1979, αποκαλύφθηκε ο μικροεπεξεργαστής 8088, που ήταν μια παραλλαγή του 8086. Πρόκειται για έναν 16bit επεξεργαστή εσωτερικά, του οποίου ο εξωτερικός δίαυλος δεδομένων ήταν των 8 bits. Η σχεδίαση αυτή είχε σκοπό τη χρήση των υπαρχόντων 8bit controller chips για συσκευές. Ο 8088 περιείχε 29000 τρανζίστορς και μπορούσε να δει 1 Mbyte μνήμης. Μετά τον 8088 η Intel απελευθέρωσε τους 16bit μικροεπεξεργαστές 80186 και 80286, παγιώνοντας την κυριαρχία της στην αγορά προσωπικών υπολογιστών.

Ο 80286 παρουσιάστηκε το 1982, και είχε συχνότητα λειτουργίας αρχικά στα 6 MHz και έπειτα στα 12 MHz.Ο ιστορικός αυτός μικροεπεξεργαστής ανήκε στην οικογένεια x86 και περιελάμβανε δίαυλο δεδομένων 16 bit, δίαυλο διευθύνσεων 24 bit. Επιπλέον, μπορούσε να δει μέχρι 16 MBytes μνήμης και περιείχε 130000 τρανζίστορ. Ο 80286 αποτελεί τον πρώτο μικροεπεξεργαστής που είχε τη δυνατότητα να λειτουργεί στην κατάσταση Protected Mode (προστατευμένη κατάσταση λειτουργίας).

32 bit Μικροεπεξεργαστές

Πολύ σύντομα μετά τη κυκλοφορία των 16 bit επεξεργαστών εμφανίστηκαν οι 32 bit επεξεργαστες.

Το 1979 εισήχθη ο MC68000, Ο διασημότερος 32bit μικροεπεξεργαστής και το πρώτο μέλος της οικογένειας m68k. Είχε 32bit καταχωρητές αλλά χρησιμοποίησε 16bit διαδρομές δεδομένων, καθώς και έναν 16bit εξωτερικό δίαυλο δεδομένων. Η Motorola το περιέγραψε γενικά ως 16bit επεξεργαστή, αν και είχε 32bit αρχιτεκτονική.

Ο συνδυασμός της υψηλής ταχύτητας, του μεγάλου χώρου αποθήκευσης (16 Mbyte) και του αρκετά χαμηλού κόστους τον έκανε τον δημοφιλέστερο μικροεπεξεργαστή της κατηγορίας του, με αποτέλεσμα να χρησιμοποιηθεί στους υπολογιστές Apple Lisa και η Macintosh. Τον MC68000 ακολούθησε ο MC68010, ο οποίος πρόσθεσε την υποστήριξη της εικονικής μνήμης ενώ το 1985 ακολούθησε ο MC68020. είχε 200000 τρανζίστορ και συχνότητα λειτουργίας στα 16 MHz. Τα 68020 έγιναν ιδιαίτερα δημοφιλή στη super microcomputer Unix αγορά, ενώ πολλές μικρές επιχειρήσεις, όπως η Altos, παρήγαγαν τα συστήματα desktop. Έπειτα, ακολούθησε ο MC68030, ο οποίος πρόσθεσε τη μονάδα διαχείρισης μνήμης (MMU) στο τσιπ, ο 68040, ο οποίος περιέλαβε τη μονάδα υπολογισμού κινητής υποδιαστολής (FPU) για καλύτερη απόδοση ενώ ο 68050 δεν κατάφερε να επιτύχει τους στόχους απόδοσής του και έτσι δεν απελευθερώθηκε. Το ακόλουθο MC68060 βγήκε στην αγορά την ίδια περίοδου που κυκλοφορούσαν και τα πολύ γρηγορότερα σχέδια RISC όπως οι MIPS R2000 (1984) και R3000 (1989) που ήταν πολύ πετυχημένοι 32-bit RISC επεξεργαστές. Χρησιμοποιήθηκαν σε high-end σταθμούς εργασίας και servers από την SGI, μεταξυ άλλων. Η οικογένεια m68k εξασθένισε από την αγορά υπολογιστή γραφείου στις αρχές της δεκαετίας του 90.

Ο παγκόσμιος πρώτος single-chip 32bit μικροεπεξεργαστής ήταν ο BELLMAC 32A της AT&T Bell Labs, ο οποίος παρουσιάστηκε το 1980 και κυκλοφόρησε το 1982. Μετά από την αποστέρηση του AT&T το 1984 και την αλλαγή της επωνυμίας της εταιρίας, ο BELLMAC 32A μετονομάστηκε σε WE 32000 (WE είναι τα αρχικά της Western Electric). Η επόμενη γενιά μικροεπεξεργαστών που στηρίχτηκε στον WE 32000, είναι ο WE 32100 και ο WE 32200. Αυτοί οι μικροεπεξεργαστές χρησιμοποιήθηκαν στους μίνι-υπολογιστές AT&T 3B5, 3B15 στο 3B2, τον πρώτο super microcomputer γραφείου. Όλα αυτά τα συστήματα έτρεξαν το αρχικό λειτουργικό σύστημα Unix των Bell Labs.

Ο πρώτος 32bit μικροεπεξεργαστής της Intel ήταν το iAPX 432, ο οποίος εισήχθη το 1981 αλλά δεν ήταν μια εμπορική επιτυχία. Παρόλο που είχε μια προηγμένη αντικειμενοστραφή αρχιτεκτονική, είχε κακή απόδοση σε σχέση με άλλες ανταγωνιστικές αρχιτεκτονικές όπως το Motorola 68000.

Ένα άλλο ενδιαφέρον σχέδιο ήταν ο Zilog Z8000, ο οποίος εισήχθη πολύ αργά στην αγορά και εξαφανίστηκε από αυτή πολύ γρήγορα.

Από το 1985 έως τπ 2003, η32-bit x86 αρχιτεκτονική έγινε η κυρίαρχη αρχιτεκτονική στην αγορά dektops, laptop, και server και οι επεξεργαστές έγιναν γρηγορότεροι και πιο ισχυροί. Η σειρά Pentium της Intel είναι πιθανότατα η πιο αναγνωρίσιμη σειρά 32 bit επεξεργαστών.

Πριν τη σειρά Pentium προηγήθηκαν αρκετές σειρές επεξεργαστων. Ξεκινώντας τον Οκτώβριο του 1985, η Intel παρουσιάζει τον απόγονο του 80286 τον μικροεπεξεργαστή 80386. Είχε συχνότητα λειτουργίας αρχικά στα 16 MHz. και χρησιμοποιούσε καθαρούς 32bit καταχωρητές και 32bit διαύλους δεδομένων και διευθύνσεων. Περιείχε 275000 τρανζίστορ, υποστήριζε πολυδιεργασία (multitasking) εικονικής μνήμης με δυνατότητα σελιδοποίησης (paging). Το 1989, εμφανίζεται ο μικροεπεξεργαστής Intel 80486, ο οποίος είχε 1200000 τρανζίστορ και συχνότητα λειτουργίας 50 MHz.

Τη δεκαετία του 1990 έκαναν την εμφάνισή τους οι μικροεπεξεργαστές Intel Pentium, οι οποίοι αποτελούσαν τη συνέχεια του 80486 ενώ είχαν υπερβαθμωτή (superscalar) αρχιτεκτονική και 32bit δίαυλο δεδομένων. Το 1993 εμφανίζεται ο Intel Pentium της οικογένειας P5, ο οποίος περιείχε 3100000 τρανζίστορ και λειτουργούσε στα 60 και 66 MHz. Το 1995, η Intel παρουσιάζει τον Pentium Pro, τον πρώτο στην οικογένεια των P6. Είχε 5500000 τρανζίστορ και ανήκε στην έκτη γενιά των επεξεργαστών της οικογένειας x86. Δύο χρόνια αργότερα, η Intel εισάγει τον μικροεπεξεργαστή Pentium II, έναν Pentium Pro με τεχνολογία MMX (MMX εντολές) για την υποστήριξη πολυμέσων. Ο Pentium II είχε 7500000 τρανζίστορ και η συχνότητα λειτουργίας του βρισκόταν στα 300 MHz.

Το 1999, ακολούθησε ο Pentium III με 9500000 τρανζίστορ και συχνότητα λειτουργίας στα 450 MHz. Την επόμενη χρονιά, εμφανίστηκε ο Pentium IV ο οποίος ήταν σχεδιασμένος σύμφωνα με την μικροαρχιτεκτονική NetBurst, η οποία συνεχίζει να αποτελεί την τεχνολογική καρδιά του Pentium IV και των παραλλαγών του κυκλοφόρησαν αργότερα. Ο Pentium D ήταν ο τελευταίος μικροεπεξεργαστής της σειράς Pentium, η οποία σταμάτησε να κυκλοφορεί το 2008. Λόγω του μεγάλου κόστους παραγωγής του Pentium II η Intel ανεπτυξε και κυκλοφόρησε μια νέα σειρά, τη σειρά Celeron. Ο Celeron είχε πιο προσιτή τιμή, αλλά δεν μπορούσε να λειτουργήσει σε πολύ υψηλές συχνότητες.

Η ανταγωνίστρια της Intel, AMD μπήκε δυναμικά στο χώρο το 1997. Αρκετά αργότερα σε σχέση με την Intel αφου η Intel είχε ήδη πάρει τα δικαιώματα της αρχιτεκτονικής x86 οπότε όλες οι άλλες εταιρείες έπρεπε να απαντύξουν τα δικά τους σχέδια. Η αρχή έγινε με τους επεξεργαστές της σειράς K6, οι οποίοι ήταν εφάμιλλοι αυτών της Intel σε τιμή και επιδόσεις. Το 1999, η AMD προώθησε την καινούργια οικογένεια μικροεπεξεργαστών, Athlon. Ο Athlon Classic, που αποτελεί τον πρώτο επεξεργαστή της σειράς και μεγάλο ανταγωνιστή των Pentium, εισήγαγε την έβδομη γενιά επεξεργαστών της οικογένειας x86.

64 bit Μικροεπεξεργαστές

Αν και οι πρώτοι 64bit επεξεργαστές εμφανίστηκαν στις αρχές του 1990, άρχισαν να εφαρμόζονται στους υπολογιστές γραφείου το 2003. Μέχρι το 2003, οι 64bit επεξεργαστές απευθύνονταν αποκλειστικά στην αγορά των ακριβών σταθμών εργασίας και των διακομιστών. Τον Σεπτέμβριο του 2003 η AMD εισήγαγε “την καλύτερη καινοτομία στους επεξεργαστές για το έτος 2003”, τον Athlon 64 για να ακολουθήσει η Intel με τον Intel 64. Και οι 2 επεξεργαστές μπορούσαν να τρέξουν 32 και 64bit εντολές. Με τη κυκλοφορία των επεξεργαστών αυτών, η γνώμη του ευρύτερου χώρου ήταν ότι θα έβρισκαν εφαρμογή μόνο σε ακαδημαϊκά ή ερευνητικά προγράμματα καθώς εκεί χρειάζονταν μεγάλοι υπολογισμοί, γρήγορες προσβάσεις σε μεγάλες βάσεις δεδομένων αλλά και επίλυση σύνθετων προβλημάτων που μπορούσαν να προσφέρουν οι 64bit επεξεργαστές.

Ο Athlon 64 αποτελεί διαφοροποίηση της αρχιτεκτονικής AMD64 για να μειωθεί το κόστος του. Το 2005, η AMD ανακοίνωσε τους διπλοπύρηνους επεξεργαστές Opteron για servers και workstations, καθώς και τους διπλοπύρηνους επεξεργαστές Athlon 64 για προσωπικούς υπολογιστές. Τον Φεβρουάριο του 2009, η AMD παρουσίασε τον τετραπύρηνο επεξεργαστή Phenom II με το κωδικό όνομα Deneb. Οι πιο πρόσφατοι επεξεργαστές της AMD ανακοινώθηκαν τον Ιούλιο. Συγκεκριμένα η εταιρία ενημέρωσε για τη διάθεση πέντε νέων εξαπύρηνων επεξεργαστών Istanbul, με πολύ χαμηλή κατανάλωση ενέργειας. Όσον αφορά στην απόδοση, τα καινούργια μοντέλα δεν καταφέρνουν να φτάσουν τους Xeon E5504 και E5520 της Intel.

Το 2006, η Intel αλλάζει τα δεδομένα στην απόδοσης και κατανάλωσης των υπολογιστών ανακοινώνοντας τις αρχιτεκτονικές Intel Core 2 Duo και Intel Core 2 Extreme. Οι αρχιτεκτονικές αυτές μπορούν να χρησιμοποιηθούν σε επεξεργαστές για σταθερούς, φορητούς και servers και παρέχουν ισχυρή απόδοση με παράλληλη εξοικονόμηση ενέργειας. Η χρήση των 2 ή 4 πυρήνων διευκολύνει τη ομαλότερη και γρηγορότερη εκτέλεση περισσότερων διεργασιών. Το 2008, κυκλοφόρησε ο Atom, ο μικρότερος σε μέγεθος επεξεργαστής της Intel που υλοποιήθηκε με τα μικρότερα τρανζίστορ του κόσμου. Δημιουργήθηκε ως μία εντελώς νέα σχεδίαση, ειδικά για φθηνές συσκευές, όπως πολύ μικρά notebooks και φορητές συσκευές με πρόσβαση στο Internet. Για τη σχεδίαση του Atom χρησιμοποιήθηκε η μικροαρχιτεκτονική Core, η ίδια δηλαδή τεχνολογία με την οποία η Intel κατασκευάζει τους γνωστούς Core 2 Duo επεξεργαστές για επιτραπέζιους και φορητούς υπολογιστές. Την ίδια χρονιά η Intel ανακοίνωσε τη νέα σειρά επεξεργαστών της με το όνομα Core i7

1.5. Γλώσσες Προγραμματισμού και Εργαλεία Ανάπτυξης.[7]

Η επιτυχία μιας οικογένειας μικροελεγκτών καθορίζεται σε μεγάλο βαθμό από τη διαθεσιμότητα και την 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 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.

Η πιο διαδεδομένη γλώσσα προγραμματισμού των μικροελεγκτών είναι η C, η C++ και οι παραλλαγές τους. Σε τμήματα του λογισμικού όπου απαιτείται ταχύτητα η 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.

Γλώσσα προγραμματισμού μικροελεγκτών.

Οι μικροελεγκτές γενικά προγραμματίζονται σε γλώσσες χαμηλού επιπέδου. Τελευταία όλο και περισσότεροι προγραμματιστές επιλέγουν γλώσσες υψηλότερο επιπέδου.Ως γλώσσα χαμηλού επιπέδου ονομάζεται μια γλώσσα η οποία βρίσκεται πιο κοντά στο υλικό (γλώσσα μηχανής, assembly).Ως γλώσσα υψηλού επιπέδου ονομάζεται μια γλώσσα η οποία είναι αυστηρά δομημένη και υπάρχει συγκεκριμένος compiler ο οποίος μετατρέπει το πρόγραμμα σε γλώσσα μηχανής για το συγκεκριμένο μικροελεγκτή.

Πλεονεκτήματα γλωσσών χαμηλού επιπέδου:
– Ο προγραμματιστής έχει τον απόλυτο έλεγχο της συμπεριφοράς του μικροελεγκτή.
– Μπορεί να επιτύχει με απόλυτη ακρίβεια διάφορους χρονισμούς
– Δεν απαιτείται η δαπάνη για την αγορά assembler καθώς συνήθως διατίθεται δωρεάν από την κατασκευάστρια εταιρεία.

Μειονεκτήματα γλωσσών χαμηλού επιπέδου:
– Απαιτείται μεγαλύτερος κόπος για την εκμάθηση της συμβολικής γλώσσας του εκάστοτε μικροελεγκτή
– Τα προγράμματα που δημιουργούνται σε συμβολική γλώσσα δεν είναι ευανάγνωστα και ο προγραμματιστής δυσκολεύεται να θυμηθεί τη λογική που έχει εφαρμόσει όταν χρειάζεται να κάνει τροποποιήσεις εκ των υστέρων
– Είναι δυσκολότερο να δουλέψουν πολλοί προγραμματιστές στο ίδιο πρόγραμμα.

Πλεονεκτήματα γλωσσών υψηλού επιπέδου:
– Είναι ευκολότερη η ανάπτυξη μεγάλων και σύνθετων προγραμμάτων
– Μπορούν να δουλέψουν πιο εύκολα πολλοί προγραμματιστές στο ίδιο πρόγραμμα

Μειονεκτήματα γλωσσών υψηλού επιπέδου:
– Σε εφαρμογές με κρίσιμους χρονισμούς είναι δυσκολότερη η συγγραφή κώδικα που ανταποκρίνεται στους χρονισμούς αυτούς
– Μερικές φορές η δαπάνη για την αγορά compiler δεν αποτελεί αμελητέο μέγεθος
– Σε παλιότερους compilers ο κώδικας μηχανής που παραγόταν δεν ήταν βελτιστοποιημένος με αποτέλεσμα να απαιτείται μικροελεγκτής με πολύ περισσότερη μνήμη. Οι compilers που κυκλοφορούν σήμερα διαθέτουν εξελιγμένα εργαλεία για βελτιστοποίηση (optimization) του κώδικα και έχουν κερδίσει την εμπιστοσύνη ακόμα και των πιο δύσπιστων προγραμαμτιστών.

Βιβλιογραφία κεφαλαίου

[1]Μικροελεγκτές PIC" Σταμάτης Αλατσαθιανός, Β.Γκιούρδας Εκδοτική
[2]Microcontroller Programming" Julio Sanchez, Maria P. Canton
[3]Mano Morris M., Ψηφιακή Σχεδίαση, (Κεφ. 7), Εκδόσεις Παπασωτηρίου, 1992
[4]Wakerly J. F., Digital Design: Principles and Practices, 2nd Edition, (Chap. 11),
[5]http://cgi.di.uoa.gr/~std06100/Welcome.html
[6]http://robotichellas.forumotion.net/t69-topic
[7]http://www.electronics-lab.com/pic-in-greek/
[8]http://en.wikipedia.org/wiki/PIC_microcontroller
[9]http://el.wikipedia.org/wiki/Reduced_instruction_set_computer
[10]http://www.hlektronika.gr/

ΚΕΦΑΛΑΙΟ 2

PIC16F84A

Εισαγωγή.

Οι μικροεπεξεργαστές είναι ολοκληρωμένα υπολογιστικά συστήματα που βασίζουν την λειτουργία τους σε κάποιες εντολές το σύνολο των οποίων δίνει την λεγόμενη γλώσσα μηχανής. Η συγκεκριμένη πτυχιακή έχει σαν στόχο να επικεντρωθεί στο μοντέλο PIC(Peripheral Interface Controller - Ελεγκτής διασύνδεσης περιφερειακών μονάδων). Οι PIC είναι ολοκληρωμένα κυκλώματα που ανήκουν στην ευρύτερη οικογένεια της εταιρίας Microchip Technology Inc. Κατασκευαστικά εξωτερικά μοιάζουν με ψηφιακά ολοκληρωμένα, ενώ η εσωτερική τους δομή περικλείει όλα τα απαραίτητα στοιχεία για την κατασκευή ενός ψηφιακού προγραμματιζόμενου συστήματος με αποτέλεσμα να τα κατατάσσει σε ένα μικρό υπολογιστή. Όσον αφορά τις εντολές είναι σε μικρό αριθμό, συνολικά 35 και σχεδόν όλες έχουν χρόνο εκτέλεσης ένα κύκλο μηχανής. Είναι ένα εξάρτημα το οποίο μπορεί να βρει χρήση σε πολλές εφαρμογές όπως να μετράει, να ελέγχει διακόπτες, να ανάβει Led, να οπλίζει ρελέ, να βγάζει ήχους, να οδηγεί οθόνη υγρού κρυστάλλου και πολλά άλλα ανάλογα με τις ανάγκες του καθενός, σε μία κατασκευή μικρού μεγέθους.

Ο μικροεπεξεργαστής PIC16F84 κατασκευάζεται από την Microchip και εξωτερικά φαίνεται να είναι ένα συνηθισμένο ολοκληρωμένο με 18 ποδαράκια. Έχει πολλά πλεονεκτήματα που τον κατατάσσουν στους πιο διαδεδομένους μικροεπεξεργαστής της Microchip, λόγω της ευκολίας επαναπρογραμματισμού του, αφού δεν χρειάζεται κάποια λάμπα UV για να τον σβήσουμε, αλλά όλα γίνονται ηλεκτρικά. Αποτέλεσμα είναι το λανθασμένο πρόγραμμα σβήνεται αυτομάτως και πάνω του γράφεται το νέο διορθωμένο φθάνει να διορθώσουμε το πρόγραμμα και απλά να τον ξαναπρογραμματίσουμε .

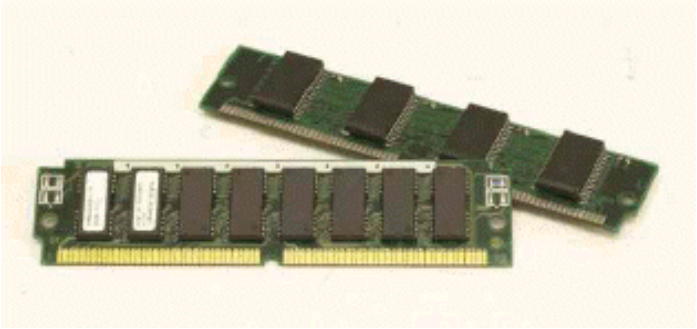
2.1. CPU.[1,2,5]

Η Κεντρική Μονάδα Επεξεργασίας - ΚΜΕ (αγγλικά: Central Processing Unit - CPU) είναι το κεντρικό εξάρτημα ενός ηλεκτρονικού υπολογιστή, και συχνά αναφέρεται απλά ως επεξεργαστής. Η ΚΜΕ ελέγχει τη λειτουργία του υπολογιστή και εκτελεί τις λειτουργίες επεξεργασίας δεδομένων. Αν η ΚΜΕ αποτελείται από ένα μόνο ολοκληρωμένο κύκλωμα τότε ονομάζεται μικροεπεξεργαστής (microprocessor) ή μικροελεγκτής (microcontroller).
Η CPU ως κεντρική μονάδα επεξεργασίας, είναι το πιο πολύπλοκο κύκλωμα του PIC16F84, αφού περιλαμβάνει τέτοια κυκλώματα που μπορούν να κάνουν μαθηματικούς υπολογισμούς ή να επεξεργάζεται δεδομένα, κατά την διάρκεια της εκτέλεσης του προγράμματος.Η παρουσία της είναι απαραίτητη σε όλα τα υπολογιστικά συστήματα και από το μέγεθος και τις δυνατότητές της εξαρτάται η ισχύς αυτών των συστημάτων.Η CPU του PIC16F84 έχει την δυνατότητα να επεξεργάζεται τιμές των 8 bit, δηλαδή αριθμητικές τιμές όχι μεγαλύτερες από 255.

Στο εμπόριο υπάρχουν και μικροεπεξεργαστές με CPU που μπορούν να δουλέψουν με αριθμούς των 16, 32, 64 bit. Τέτοια παραδείγματα έχουμε στις εταιρίες μοντέλων επεξεργαστών όπως οι επεξεργαστές Intel© 80386, 486© και Pentium© διαθέτουν μια CPU πού μπορεί να επεξεργαστεί αριθμού έως και 32 bit. Οι υπολογιστικές δυνατότητες αυτών των επεξεργαστών είναι τεραστίων διαστάσεων αν τους συγκρίνουμε με τον PIC16F84, αλλά αν σκεφθούμε το μέγεθος του ενός και του άλλου, θα δούμε ότι ο PIC16F84 είναι ένα τσιπάκι με 18 ποδαράκια που χρειάζεται ελάχιστα περιφερειακά για να δουλέψει ενώ ο Pentium© για να δουλέψει χρειάζεται μεγάλη επιφάνεια και επιπλέον εξαρτήματα. Απ' ευθείας συνδεδεμένος στην CPU, είναι ο συσσωρευτής γνωστός και σαν Working register ή W, που στην πραγματικότητα είναι μια απλή θέση μνήμης που έχει μήκος 8bit, και είναι απαραίτητος σε όλους υπολογισμούς τηςCPU. Η ουσιαστική διαφορά του, από τους άλλους καταχωρητές είναι ότι η CPU δεν χρειάζεται να έχει υπ' όψιν της κάποια διεύθυνση για να βρεί τον συσσωρευτή W, αλλά έχει άμεση πρόσβαση σ' αυτόν. Κρίνεται λοιπόν απαραίτητος για όλους τους υπολογισμούς τις CPU ο συσσωρευτής W.

2.2. Μνήμη ROM.[3,4]

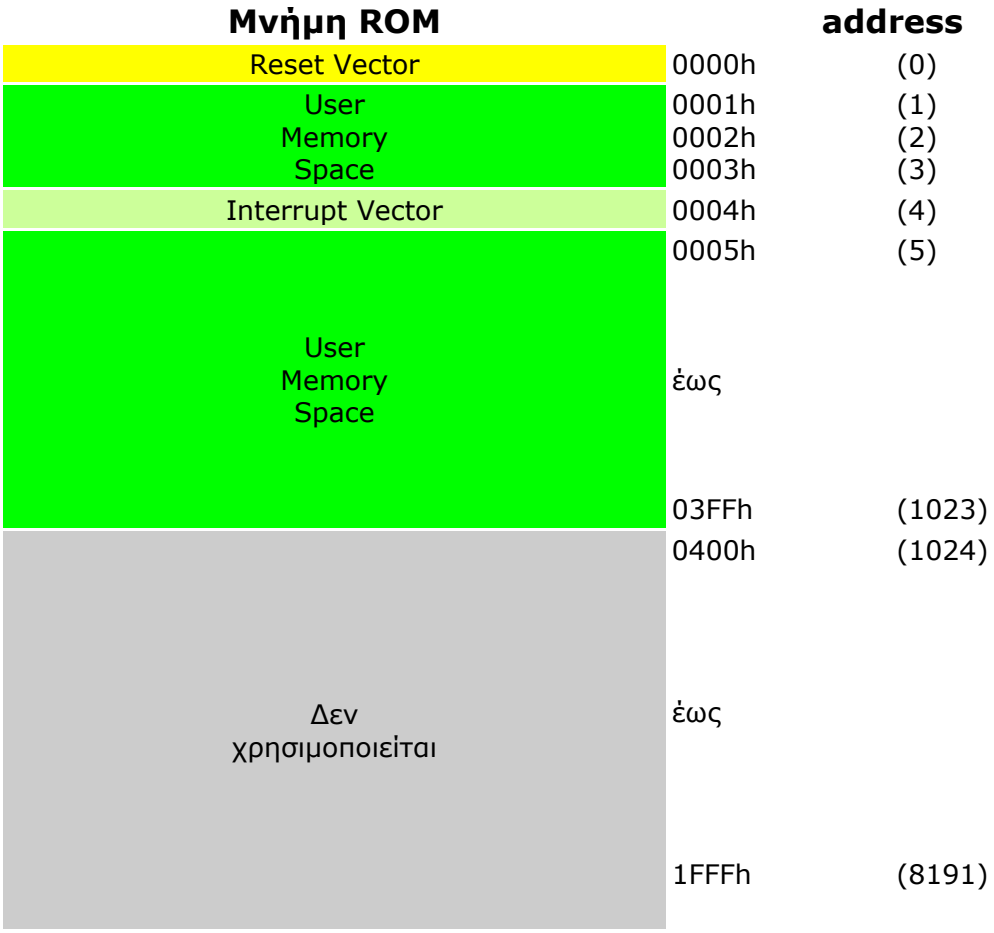
Η μνήμη είναι ηλεκτρονικά κυκλώματα, τα οποία «αποθηκεύουν» προγράμματα και δεδομένα για να χρησιμοποιηθούν από τον μικροεπεξεργαστή. Υπάρχουν δύο είδη μνήμης. Η μνήμη ROM (Read Only Memory) και η μνήμη RAM (Random Access Memory).
Η μνήμη ROM έχει συνήθως μέγεθος γύρω στα 256 Kbytes και χρησιμοποιείται από τον μικροεπεξεργαστή κυρίως κατά την εκκίνηση του Η/Υ, για να του δώσει τις πρώτες εντολές που θα εκτελεστούν. Τα δεδομένα που περιέχονται στην μνήμη ROM είναι αμετάβλητα, δεν μπορούν να αλλαχθούν από εμάς, και καταγράφονται σε αυτήν από τον κατασκευαστή του Η/Υ. Στο παρακάτω σχήμα φαίνεται μια πλήρη απεικόνιση μιας μνήμης ROM.



Εικόνα 2.1 : Πλήρης μορφή μνήμης ROM

Η μνήμη ROM έχει πολλές και διαφορετικές ονομασίες όπως Program Memory, Program Space ή Flash ROM. ROM σημαίνει Read Only Memory (μνήμη μόνο για διάβασμα) δηλαδή είναι μια μνήμη που ό,τι γράψουμε σε αυτή μπορούμε να το διαβάσουμε όσες φορές θέλουμε αλλά δεν είναι δυνατόν να γράψουμε κάτι διαφορετικό τουλάχιστον όταν έχουμε γράψει ένα συγκεκριμένο πρόγραμμα.

Για να γράψουμε θα πρέπει να ξαναπρογραμματίσουμε από την αρχή τον PIC16F84, σβήνοντας ό,τι είχαμε γράψει προηγουμένως.Η μνήμη είναι τύπου Flash δηλαδή επαναπρογραμματιζόμενη. Η μνήμη ROM έχει μέγεθος 8192 θέσεις μνήμης που η κάθε μία ονομάζεται Word (λέξη). Σε κάθε Word, που έχει μήκος (αποτελείται) 14 bits, μπορεί να αποθηκευτεί μία και μόνο μια γραμμή εντολής υπό μορφή κώδικα (opcode). Ένα απλό παράδειγμα είναι όταν γράψουμε την εντολή "bcf portb,0" (μηδένισε το bit0 της portb), αυτή αποθηκεύεται σε μια Word με το κωδικό αριθμό1006h ή 01 0000 0000 0110, ενώ αν γράψουμε "addlw 255" (πρόσθεσε στο συσσωρευτή W την τιμή 255) θα αποθηκευτεί με το κωδικό αριθμό 3FFFh ή 11 1111 1111 1111.Από τις 8192 Words, είναι ενεργοποιημένες μόνο οι πρώτες 1024 και αυτές είναι που καθορίζουν το πραγματικό μέγεθος της μνήμης του PIC16F84. Έτσι το πρόγραμμά μας δεν πρέπει να καταλαμβάνει περισσότερες από 1024 Word, δηλ. 1024 γραμμές διαφορετικών εντολών. Αυτός ο χώρος της μνήμης ονομάζεται User Program Space δηλαδή χώρος προγράμματος που προορίζεται για τον χρήστη, όπως απεικονίζεται παρακάτω.



Εικόνα 2.2: Χάρτης μνήμης ROM

Στο παραπάνω σχήμα λοιπόν μπορούμε να δούμε αναλυτικά ότι κάθε θέση μνήμης έχει και μια συγκεκριμένη διεύθυνση (address). Η διεύθυνση 0 (0000h) ανήκει στην πρώτη Word και εδώ "κάθεται" η πρώτη εντολή του προγράμματος. Άρα το πρόγραμμα θα αρχίσει να εκτελείται από αυτή την Word. Οι επόμενες εντολές αποθηκεύονται στις άλλες Word που ακολουθούν, έως την διεύθυνση 1023 (1FFh).

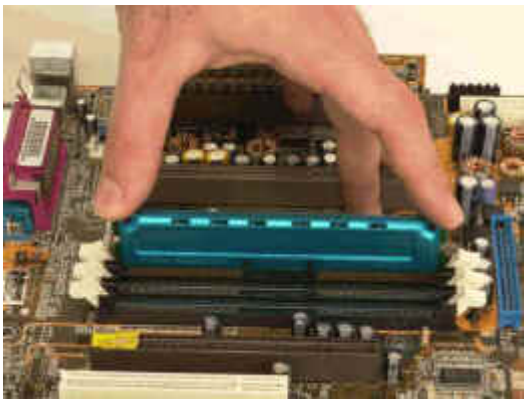
Όλα τα παραπάνω δεν ισχύουν για τη διεύθυνση 4 (0004h) η οποία κατακρατείται από τον PIC16F84 και προορίζεται να αποθηκευτεί μια εντολή διακοπής του προγράμματος, εφ' όσον χρησιμοποιούμε διακοπές στο πρόγραμμα.Από την διεύθυνση 1024 (0200h) έως την 8191 (1FFFh) βρίσκεται ο χώρος της ROM που δεν χρησιμοποιείται και δεν τον λαμβάνουμε υπ' όψιν μας. Συμαντικό είναι ότι ο User Program Space αποτελείται απο 1024 Words και βρίσκεται από την διεύθυνση 0 (0000h) έως την 1023 (01FFh) και αν κάνουμε την αφαίρεση 1023-0 για να βρούμε πόσες Words είναι, θα βρούμε 1023 και όχι 1024 όπως αναφέρουμε πιο πάνω. Το λάθος στην αφαίρεση βρίσκεται στο ότι δεν υπολογίζει και την θέση 0, οπότε όταν την προσθέσουμε στο 1023 θα βρούμε τον πραγματικό αριθμό των Words. Όταν λοιπόν θέλουμε να υπολογίσουμε πόσες Words ή Bytes ή ότι άλλο, βρίσκονται μεταξύ δύο διευθύνσεων θα πρέπει στο αποτέλεσμα να προσθέτουμε 1.
Η διορθωμένη πράξη θα είναι:(1023 - 0) + 1 = 1024 ή (01FFh - 0000h) + 1 = 0200h. Τέλος η μνήμη ROM είναι άμεσα συνδεδεμένη με το μετρητή προγράμματος (Program Counter), καθώς και με την μνήμη Stack (Στοίβα).

Ο μετρητής προγράμματος (Program Counter) είναι ένας απλός μετρητής που μετράει μια μια τις εντολές που εκτελούνται. Κάθε φορά που εκτελείται μια εντολή ο Program Counter αυξάνει την τιμή του κατά 1 και αμέσως μετά εκτελείται η εντολή. Ο Program Counter αποτελείται από 13 bit και αυτό συμβαίνει γιατί το σύνολο των θέσεων μνήμης (Words) ROM είναι 8191 συν η θέση μηδέν.

Η Microchip δεν παράγει μόνο τον PIC16F84, αλλά και άλλους πολλούς τύπους με διαφορετικό μέγεθος της User Memory Space που ανήκουν στην ίδια οικογένεια. Έτσι μπορούμε, για ένα πρόγραμμα που δεν το χωράει ο PIC16F84, να επιλέξουμε κάποιον άλλο μικροεπεξεργαστή της ιδίας οικογενείας, αν και η επιλογή του κατάλληλου μικροεπεξεργαστή είναι πραγματικά δύσκολη καθώς κυκλοφορούν τόσοι πολλοί και διαφορετικοί τύποι.
Σε αυτό το σημείο γνωστοποιούμε ότι ο PIC16F84, αρχίζει την εκτέλεση του προγράμματος από την διεύθυνση 0000h που ονομάζεται Reset Vector. Όταν εκτελέσει την εντολή που βρίσκεται σε αυτή την διεύθυνση, περνά στην επόμενη κ.ο.κ. Άν δεν υπάρχουν κάποιες εντολές για να επηρεάσουν την εκτέλεση του προγράμματος, ο PIC16F84, θα έφθανε πολύ γρήγορα στην τελευταία διεύθυνση.Κάτι τέτοιο δεν ισχύει για οποιοδήποτε σύστημα μικροεπεξεργαστή ή γλώσσα προγραμματισμού, διότι έχει και κάποιες εντολές παραπομπής (jump ή goto), που είναι ικανές να αλλάξουν την ροή της εκτέλεσης του προγράμματος. Η εντολή παραπομπής του PIC16F84 είναι η goto και όταν την συναντά, σταματάει την κανονική εκτέλεση του προγράμματος, χωρίς να εκτελέσει την επόμενη εντολή, και πηδά στην διεύθυνση της ROM που έχει η ετικέτα.

2.3. Μνήμη RAM.[5,7]

Η μνήμη RAM έχει συνήθως μέγεθος από 128 MB και πάνω και χρησιμοποιείται για την προσωρινή αποθήκευση σε αυτήν προγραμμάτων και δεδομένων, τα οποία μετακινούνται μεταξύ του μικροεπεξεργαστή και του σκληρού δίσκου. Ένα πρόγραμμα που πρόκειται να εκτελεσθεί πρέπει πρώτα να μεταφερθεί από τον σκληρό δίσκο στην μνήμη RAM, ώστε να μπορέσει να το «δει» ο μικροεπεξεργαστής. Το ίδιο συμβαίνει και με τα δεδομένα που πιθανώς χειρίζεται το πρόγραμμα. Ο λόγος που επιβάλλει την ύπαρξη της μνήμης RAM σαν ενδιάμεσου μεταξύ του μικροεπεξεργαστή και του σκληρού δίσκου (ή άλλων μέσων αποθήκευσης, είναι ότι τα μέσα μαγνητικής αποθήκευσης εργάζονται με πολύ χαμηλές ταχύτητες σε σχέση με την ταχύτητα με την οποία εργάζεται ο μικροεπεξεργαστής, ενώ η μνήμη RAM έχει τη δυνατότητα να εργάζεται με την ταχύτητα του μικροεπεξεργαστή. Να σημειωθεί ότι η μνήμη RAM δεν έχει την δυνατότητα να αποθηκεύει μόνιμα δεδομένα, ενώ τα δεδομένα που περιέχει «χάνονται» μόλις διακοπεί η τροφοδοσία του Η/Υ με ηλεκτρικό ρεύμα.
Τα νέα λειτουργικά συστήματα, όπως τα Windows XP, απαιτούν τουλάχιστον 128MB μνήμης RAM για να λειτουργήσουν ικανοποιητικά. Μπορεί να πούμε μάλιστα ότι η συνολική απόδοση ενός Η/Υ εξαρτάται περισσότερο από την ύπαρξη περισσότερης μνήμης RAM παρά από την ύπαρξη καλύτερου μικροεπεξεργαστή.



Σχήμα 2.3: Απεικόνιση για τοποθέτηση μνήμης RAM

Η μνήμη ***RAM***, όπως φαίνεται και από το παρακάτω σχήμα, αποτελείται από 256 bytes που είναι ομοδοποιημένα σε δύο banks, των 128 Bytes το καθένα. Το bank0 και το bank1.Το bank0 έχει τις διευθύνσεις από την 00h (0) έως την 7Fh (127).Σύνολο 128 Bytes.

	bank0	bank1	
00h	INDR	INDR	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h			87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2	89h
0ah	PCLATCH	PCLATCH	8ah
0bh	INTCON	INTCON	8bh
0ch 4fh	68 General Purpose Registers	68 Accesses in bank0	8ch cfh
50h 7fh	No implement	No implement	d0h ffh

Εικόνα 2.4: Πίνακας μνήμης RAM

Το bank1 είναι συνέχεια του bank0 και περιλαμβάνει τις διευθύνσεις από την 80h (128) έως την FFh (255). Σύνολο 128 Bytes και σε αυτό το bank.

Special Function Registers:Τα πρώτα 12 bytes τόσο του bank0 όσο και του bank1 έχουν ειδικές λειτουργίες που τα χρησιμοποιεί αποκλειστικά η CPU και τα περιφερειακά κυκλώματα του PIC16F84, και ονομάζονται Special Function Registers. Στο σχήμα, βλέπετε ότι μερικά registers υπάρχουν και στα δύο bank ενώ άλλα βρίσκονται μόνο στο ένα από τα δύο.

Αυτό σημαίνει ότι για να επιλέξουμε ένα register που βρίσκεται καταχωρημένο και στα δύο bank, δεν χρειάζεται προηγουμένως να ενεργοποιήσουμε κάποιο bank. Αντιθέτως για τα registers που βρίσκονται μόνο σε ένα bank πρέπει προηγουμένως να ενεργοποιούμε το bank που βρίσκεται το register.

General Purpose Registers:Από την διεύθυνση 0Ch (12) έως την 4Fh (79), για το bank0 και από 8Ch (140) έως CFh (207), για το bank1, έχουμε 68 bytes για το κάθε bank που είναι registers για δική μας χρήση και ονομάζονται General Purpose Registers. Θα μπορούσαμε να πούμε ότι το σύνολο αυτών των registers είναι 136 αλλά εδώ υπάρχει μια ιδιαιτερότητα και τα προς χρήση registers είναι μόνο 68 αφού όποιο bank και να είναι ενεργό τα registers που βρίσκονται στο bank1 είναι πάντα "ορατά" και από το bank0. Με άλλα λόγια το register με διεύθυνση π.χ. 8Ch (bank1) είναι το ίδιο με το register της διεύθυνσης 0Ch του bank0.

2.4. Μνήμη EEPROM & STACK.[8]

Η μνήμη **EEPROM** αποτελείται από 64 bytes και έχει τα ίδια χαρακτηριστικά με τις γνωστές μνήμες. Σε αυτή την μνήμη μπορούμε να γράψουμε ή σβήσουμε, μέσω του προγράμματος, διάφορα δεδομένα και να τα ανακαλέσουμε όταν τα χρειαστούμε.
Η μνήμη **STACK** (**Στοίβα**) είναι πολύ μικρή - έχει μόνο 8 θέσεις - και μοιάζει με την RAM. Η CPU χρησιμοποιεί αυτή τη μνήμη για να θυμάται πού βρισκόταν (σε ποιό σημείο του προγράμματος) αν της ζητήσουμε να σταματήσει την κανονική ροή του προγράμματος για να εκτελέσει κάποιες άλλες εντολές και μετά να ξανά επιστρέψει στο ίδιο σημείο. Και αυτή η μνήμη χάνει τις πληροφορίες όταν ο PIC16F84 δεν τροφοδοτείται.

2.5. Καταχωρητές.[9]

Ο καταχωρητής (Register) είναι μία συγκεκριμένη θέση μνήμης στο εσωτερικό του μικροελεγκτή στο οποίο μπορούμε να γράψουμε και να διαβάσουμε.
Το παρακάτω σχήμα δείχνει τον χάρτη των καταχωρητών μνήμης του PIC16F84 και μας δείχνει ποια θέση μνήμης καταλαμβάνει ο κάθε καταχωρητής. Το πρώτο που θα παρατηρήσετε είναι ότι είναι χωρισμένος σε δύο επίπεδα (Bank 0 και Bank 1). Η Bank 1 χρησιμοποιείται για τον χειρισμό της πραγματικής λειτουργίας του PIC, για παράδειγμα για να ορίσουμε ποια bits της θύρας A είναι είσοδοι και ποια έξοδοι. Η Bank 0 χρησιμοποιείται για τον χειρισμό δεδομένων. Για παράδειγμα, ας υποθέσουμε ότι θέλουμε να θέσουμε ένα bit της θύρας A «1». Αρχικά θα πρέπει να πάμε στη Bank1 και να δηλώσουμε το αντίστοιχο bit της θύρας A ως έξοδο. Εν συνεχεία θα πρέπει να επιστρέψουμε στη Bank0 και να θέσουμε το αντίστοιχο bit.

Η μνήμη των ειδικών καταχωρητών **BANK0** επιλέγεται όταν το **Bit 5 (RP0)** του **STATUS** είναι **μηδέν (0).** Κάθε σειρά εκτείνεται έως το 17Fh (128 Bytes), οι πρώτες δώδεκα διευθύνσεις κάθε σειράς είναι δεσμευμένες για τους ειδικούς καταχωρητές ενώ οι υπόλοιπες χρησιμοποιούνται για τους καταχωρητές γενικής χρήσης, οι οποίοι έχουν εύρος **8 Bit** και μπορούν να προσπελαστούν άμεσα ή έμμεσα. Η μνήμη αποθήκευσης δεδομένων περιλαμβάνει και τη μνήμη EEPROM (Electrical Eraseble Programable Rom), το τμήμα αυτό προσπελαύνεται έμμεσα μέσο ενός δείκτη (Pointer) ο οποίος υποδεικνύει την διεύθυνση της EEPROM που πρόκειται να εγγραφεί ή να διαβαστεί. Τα 64 Bytes της EEPROM εκτείνονται στις διευθύνσεις 0h-3Fh. Παρακάτω φαίνεται αναλυτικά και ο πίνας με τους ειδικούς καταχωρητές καθώς και την λειτουργία τους.

Address	Bank 0	Bank 1	Address
00h	INDF	←	80h
01h	TMR0	OPTION_REG	81h
02h	PCL	←	82h
03h	STATUS	←	83h
04h	FSR	←	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	Unimplemented	←	87h
08h	EEDATA	EECON1	88h
09h	EEADR	EECON2	89h
0Ah	PCLATH	←	8Ah
0Bh	INTCON	←	8Bh
0Ch - 4Fh	GPR	←	8Ch - CFh

Εικόνα 2.5: Πίνακας Καταχωρητών γενικής χρήσης.

Πίνακας 2.1: Οι Ειδικοί καταχωρητές και η λειτουργία τους

INDF : Καταχωρητής περιεχομένων FSR για έμμεση προσπέλαση της μνήμης

TMR0 : Μετρητής πραγματικού χρόνου (Timer)

PCL : Καταχωρητής αποθήκευσης των 8 λιγότερο σημαντικών bit του program counter

STATUS : Καταχωρητής αποθήκευσης αποτελεσμάτων πράξεων της ALU, του Reset & του Bank select

FSR : Καταχωρητής έμμεσης προσπέλασης της μνήμης (Indirect addressing pointer)

PORTA : Καταχωρητής δεδομένων Εισόδου/Εξόδου (DATA I/O) της PORTA

PORTB : Καταχωρητής δεδομένων Εισόδου/Εξόδου (DATA I/O) της PORTB

EEDATA : Καταχωρητής αποθήκευσης δεδομένων για ανάγνωση/γραφή στην EEPROM

EEADR : Καταχωρητής διευθύνσεις για προσπέλαση της μνήμης EEPROM

PCLATH : Καταχωρητής αποθήκευσης των 5 περισσότερο σημαντικών bit του program counter

INTCON : Καταχωρητής ελέγχου των Interrupt

OPTION_REG : Καταχωρητής αποθήκευσης των ρυθμιστικών Bit για Interrupt, TMR0/WDT Prescaller

TRISA : Καταχωρητής θέσεως σαν Είσοδο ή Έξοδο της PORTA

TRISB : Καταχωρητής θέσεως σαν Είσοδο ή Έξοδο της PORTB

EECON1 : Καταχωρητής ελέγχου της EEPROM

EECON2 : Καταχωρητής ελέγχου και προστασίας της EEPROM από εγγραφή

2.6. Χρονιστές.[9]

Ο PIC διαθέτει τρεις μετρητές: δύο οκτάμπιτους (8bits) χρονιστές/μετρητές (timer/counter) [TMR0,2] και έναν δεκαξάμπιτο (16bits) [TMR1].

Ο TMR0, είναι ένα πλήρως προγραμματιζόμενο κύκλωμα, που μπορεί να λειτουργήσει με δύο τρόπους.

α) *Ως χρονιστής (timer)*: Στην περίπτωση αυτή η πηγή χρονισμού είναι ο εσωτερικός κύκλος των εντολών. Το περιεχόμενο του Timer0 σε λειτουργία χρονιστή αυξάνει κατά ένα σε κάθε κύκλο εντολής. Ο κύκλος της εντολής διαρκεί όσο η περίοδος του εξωτερικού κρυστάλλου επί τέσσερα (4*fosc). Η αντίστοιχη συχνότητα είναι ίση με την συχνότητα του κρυστάλλου διά 4. Χρησιμοποιώντας το TMR0 σε λειτουργία εσωτερικού χρονισμού, έχουμε στη διάθεσή μας ένα αρκετά αξιόπιστο σήμα χρονισμού που μπορεί να χρησιμοποιηθεί για τον ακριβή προσδιορισμό χρονικών διαστημάτων.

β) *Ως απαριθμητής (counter)*: Στην περίπτωση αυτή χρησιμοποιείται ένας εξωτερικός παλμός χρονισμού. Το εξωτερικό σήμα χρονισμού συνδέεται με τον ακροδέκτη TOCKI (RA4). Η χρήση εξωτερικής πηγής χρονισμού επιτρέπει την απαρίθμηση εξωτερικών συμβάντων, με τη μορφή παλμών. Το περιεχόμενο του Timer0 αυξάνει κατά ένα σε κάθε παλμό της εξωτερικής πηγής χρονισμού.

Και στις δύο περιπτώσεις, ο **TMR0** μετρά στο δεκαεξαδικό σύστημα από την αρχική τιμή που καταχωρήσαμε έως 0xFF.

Όπως θα δούμε, η επιλογή του είδους της πηγής χρονισμού του TMR0, επιτυγχάνεται με τη χρήση του bit T0CS του ειδικού καταχωρητή OPTION_REG.

Τα bits του καταχωρητή **OPTION_REG** καθορίζουν εάν θα μπει σε λειτουργία ο απαριθμητής και με ποιον από όλους τους δυνατούς τρόπους θα εργαστεί. Ο πίνακας 6-1 παρουσιάζει τα bits του καταχωρητή OPTION_REG και τη σημασία τους.

Πίνακας 6-1

Τα bits του καταχωρητή OPTION_REG, ο οποίος ρυθμίζει τη λειτουργία του χρονιστή TMR0.

b7	b6	b5	b4	b3	b2	b1	b0
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

- **Bits 7-6**: Δεν έχουν σημασία για την λειτουργία του χρονιστή (τα θέτουμε 11).
- **T0CS**: Επιλογή πηγής ρολογιού
  1 = Πηγή συνδεδεμένη στον ακροδέκτη T0CKI (Λειτουργία μετρητή παλμών)
  0 = Εσωτερική πηγή ρολογιού (Λειτουργία χρονιστή).
- **T0CE:** Επιλογή θετικού ή αρνητικού μετώπου παλμού (αναφέρεται σε εξωτερική πηγή χρονισμού)
  1 = Αύξηση με κατερχόμενο μέτωπο του παλμού στον ακροδέκτη T0CLK
  0 = Αύξηση με ανερχόμενο μέτωπο του παλμού στον ακροδέκτη T0CLK
- **PSA**: "0", όταν θέλουμε να συνδέσουμε τον διαιρέτη συχνότητας.
  "1", όταν θέλουμε να αποσυνδέσουμε τον διαιρέτη συχνότητας.
- **PS2-PS0***:* Επιλογή λόγου διαίρεσης

Εάν επιθυμούμε να αυξηθεί ο χρόνος τον οποίο μετρά ο απαριθμητής **TMR0**, μπορούμε να χρησιμοποιήσουμε έναν διαιρέτη συχνότητας (*prescaler*). Αυτός μπορεί να ρυθμιστεί κατάλληλα, ώστε να διαιρεθεί η συχνότητα αύξησης του μετρητή κατά μία επιθυμητή τιμή. (1:2, 1:4, 1:8, ... 1:256).

Πίνακας 6-2

Διαίρεση της συχνότητας με την οποία αυξάνεται ο TMR0 και ο WDT

Τιμή προμετρητή			Ρυθμός TMR0	Ρυθμός WDT
PS2	PS1	PS0		
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

Η ρύθμιση του διαιρέτη συχνότητας γίνεται ορίζοντας κάποια bits του καταχωρητή **OPTION_REG** (Πίνακας 6-2). Συγκεκριμένα, ορίζουμε τα τρία πρώτα bits (PS0, PS1, και PS2) του καταχωρητή OPTION_REG από p=000 έως p=111 (δηλαδή στο δεκαδικό σύστημα από 0 έως 7), και έτσι μεταβάλλουμε τον ρυθμό του απαριθμητή TMR0 κατά τον παράγοντα 1:2p+1.Εάν το bit PSA του καταχωρητή OPTION_REG τεθεί σε λογικό ένα, τότε ο διαιρέτης αποσυνδέεται από τον απαριθμητή TMR0 και συνδέεται με τον χρονιστή επιτήρησης (WDT). Στην περίπτωση αυτή δεν συμβαίνει καμία καθυστέρηση στο χρόνο αύξησης του περιεχομένου του απαριθμητή **TMR0** και αυτός αυξάνει κατά ένα σε κάθε κύκλο εντολής.
Εάν το bit PSA του καταχωρητή **OPTION_REG** τεθεί σε λογικό μηδέν, τότε ο προμετρητής συνδέεται με τον απαριθμητή **TMR0** και διαιρεί τη συχνότητα αύξησης κατ' ελάχιστο διά δύο. Όταν ο TMR0 λειτουργεί ως απαριθμητής, τότε ο χρόνος που χρειάζεται για να υπερχειλίσει ο Timer0 δίνεται από τη σχέση:

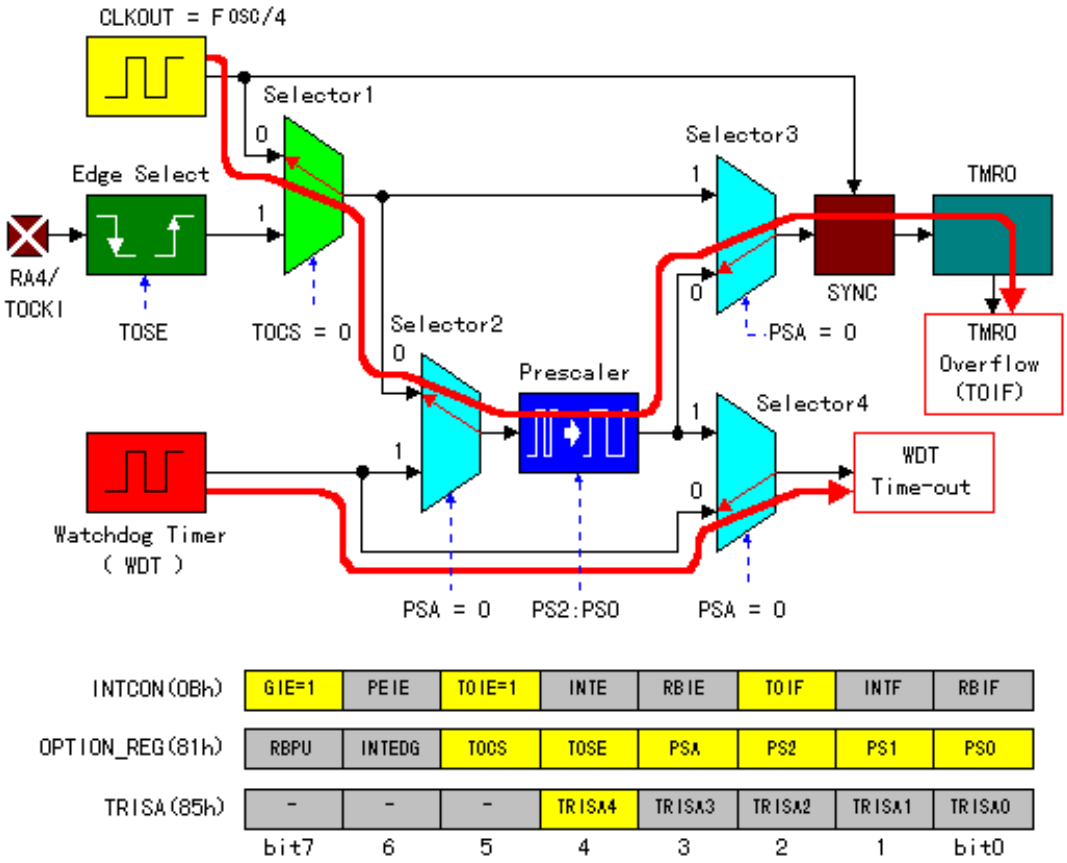
$$\text{Delay} = \frac{(256\text{-αρχικό περιεχόμενο του TMR0})^* \text{ λόγος διαίρεσης συχνότητας}^*4}{\text{Συχνότητα κρυστάλλου}}$$

Όταν ο TMR0 λειτουργεί ως χρονιστής, με εξωτερική πηγή παλμών, τότε ο χρόνος που χρειάζεται για να υπερχειλίσει ο Timer0 δίνεται από τη σχέση:

$$\text{Delay} = \frac{(256\text{-αρχικό περιεχόμενο του TMR0})^* \text{ λόγος διαίρεσης συχνότητας}}{\text{Συχνότητα εξωτερικής πηγής}}$$

Όταν ο απαριθμητής **TMR0** υπερχειλίσει, μεταβαίνοντας από την ανώτερη τιμή 0x0FF στην αρχική τιμή 0x000, δημιουργείται ένα *σήμα διακοπής* (*interrupt*). Αυτό σημαίνει ότι η σημαία (*flag*) που δηλώνει ότι έλαβε χώρα η συγκεκριμένη διακοπή τίθεται σε λογικό ένα. Η σημαία αυτή ονομάζεται T0IF και βρίσκεται στον καταχωρητή INTCON (b2). Στο σημείο αυτό, και εφόσον έχει ενεργοποιηθεί η αντίστοιχη διακοπή, η εκτέλεση του προγράμματος διακόπτεται και καλείται η ρουτίνα εξυπηρέτησης της διακοπής.

Πιο λεπτομερής αναφορά στα σήματα διακοπών στους PIC γίνεται σε επόμενη παράγραφο. Ας σημειωθεί ότι η συγκεκριμένη χρήση του απαριθμητή/χρονιστή **TMR0** για τη σήμανση διακοπών χρησιμοποιείται σε εφαρμογές στις οποίες η καταμέτρηση του χρόνου έχει κρίσιμη σημασία. Παράδειγμα τέτοιας εφαρμογής είναι η χρήση του μικροελεγκτή για λήψη και εκπομπή σημάτων, με τη μέθοδο της ασύγχρονης σειριακής επικοινωνίας.

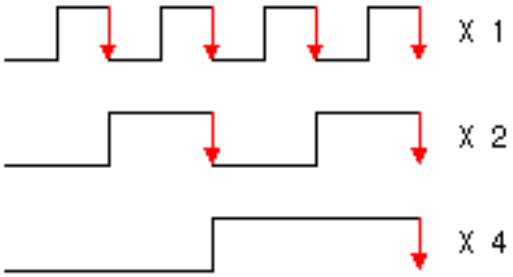


Εικόνα 2.6: Το εσωτερικό του χρονομετρητή (TMR0)

Στην επάνω εικόνα φαίνεται το διάγραμμα του χρονομετρητή (TMR0) και του επιτηρητή (WDT- watchdog timer). Ο προδιαιρέτης (prescaler) χρησιμοποιείται για τον TMR0 και τον WDT. Στο επάνω σχέδιο φαίνεται ο προδιαιρέτης σε χρήση με τον TMR0. Η επιλογή για το αν ο προδιαιρέτης (prescaler) θα χρησιμοποιηθεί για τον χρονομετρητή (TMR0) ή τον επιτηρητή (WDT) γίνεται από το bit PSA του καταχωρητή OPTION_REG. Ο προδιαιρέτης είναι ένας προγραμματιζόμενος ειδικός καταχωρητής του οποίου η τιμή καθορίζεται από τα bit PS0, PS1, PS2 του OPTION_REG. Ο χρονομετρητής TMR0 είναι ένας δυαδικός μετρητής των οκτώ bit (8 bit binary counter) και μπορεί να μετρήσει 256 αριθμούς (από 0 έως 255). Όταν ο TMR0 υπερχειλίσει, δηλαδή από 255(FFh) γίνει 0(00h) δημιουργείται παλμός διακοπής (Interrupt) και το T0IF του καταχωρητή INTCON γίνεται "1" .

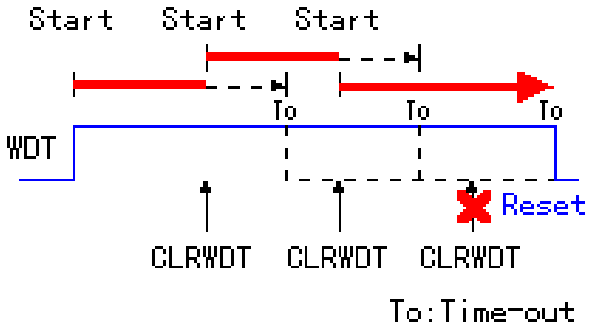
Όταν δημιουργηθεί ο παλμός διακοπής (Interrupt) ο μετρητής προγράμματος (PC - program counter) παίρνει την τιμή 0004h και το πρόγραμμα συνεχίζει να εκτελείται από αυτή την διεύθυνση. Για να ισχύσει η παραπάνω λειτουργία πρέπει τα bit GIE και T0IE του καταχωρητή INTCON να έχουν την τιμή "1".Επειδή μερικές φορές οι 256 αριθμοί που μπορεί να μετρήσει ο χρονομετρητής TMR0 είναι λίγοι, τότε χρησιμοποιείται ο προδιαιρέτης (prescaler). Δείτε το παρακάτω παράδειγμα για να καταλάβετε. Έστω ότι συνδέετε στα ποδαράκια OSC1 & OSC2 ένα κρύσταλλο 4MHz, από τον τύπο 4/Fosc, δηλαδή 4/4 δίνει κύκλο μηχανής 1μsec ή 1000nsec άρα σε 256 κύκλους που υπερχειλίζει ο TMR0 θα έχει περάσει χρόνος ίσος με 256 μsec, που μπροστά στο δευτερόλεπτο (1000000 μsec) είναι πολύ μικρός χρόνος. Αν λοιπόν χρειαζόσαστε περισσότερο χρόνο τότε πρέπει να χρησιμοποιήσετε τον προδιαιρέτη (prescaler). Εκτός από τον εσωτερικό στον μικροεπεξεργαστή ταλαντωτή για την οδήγηση του χρονομετρητή (TMR0), μπορεί να χρησιμοποιηθεί και εξωτερική πηγή όπως ταλαντωτής R/C κ.λ.π.

Για να ενεργοποιηθεί η εξωτερική πηγή χρονισμού στο ποδαράκι RA4/T0CKI πρέπει το bitTOCS του καταχωρητή OPTION_REG και το bit TRISA4 του καταχωρητή TRISA να πάρουν την τιμή "1". Η τιμή που έχει το bit T0SE του καταχωρητή OPTION_REG καθορίζει αν το σήμα σκανδαλισμού για το χρονομετρητή (TMR0) θα είναι το ανερχόμενο ή το κατερχόμενο μέτωπο του παλμού (the rising or falling of the clock pulse). Για σκανδαλισμό στο ανερχόμενο μέτωπο το bit T0SE πρέπει να έχει τιμή "0" ενώ για σκανδαλισμό στο κατερχόμενο μέτωπο πρέπει να έχει την τιμή "1".



Η τιμές που μπορεί να πάρει ο προδιαιρέτης (prescaler) είναι συγκεκριμένες (2, 4, 8, 16, 32, 64, 128, 256). Έστω πως η τιμή του προδιαιρέτη είναι 2 τότε ο TMR0 θα αυξήσει το περιεχόμενο του ανά 2 κύκλους μηχανής ενώ Αν ο προδιαιρέτης έχει τιμή 256 τότε ο TMR0 θα αυξήσει το περιεχόμενο του ανά 256 κύκλους μηχανής. Όσον αφορά το παραπάνω παράδειγμα, με τιμή προδιαιρέτη 256, ο TMR0 θα υπερχειλίσει μετά από 256μsec X 256 = 65535μsec ή 65,5msec ή 0,06sec.

Ο επιτηρητής (WDT - watchdog timer) έχει δικό του ξεχωριστό ταλαντωτή, και υπερχειλίζει δίχως την χρήση προδιαιρέτη (prescaler) κάθε 18msec. Ο προδιαιρέτης (prescaler) που μεγαλώνει την διάρκεια του επιτηρητή (WDT) μπορεί να πάρει τιμές (2, 4, 8, 16, 32, 64, 128). Προσέξτε ότι δεν μπορεί να πάρει την τιμή **256**. Με ανώτερη τιμή το **128** μπορεί να φτάσει χρόνο ίσο με 18msec x 128 = 2304msec ή 2,3sec. Η χρήση του επιτηρητή είναι απαραίτητη όταν τυχόν δυσλειτουργίες του προγράμματος μπορούν να προξενήσουν ζημιά ή καταστροφές. Για παράδειγμα αν ο μικροεπεξεργαστής ελέγχει την λειτουργία ενός αερόθερμου και από λάθος προγραμματισμό "κολλήσει", τότε μπορεί να υπάρξει σταμάτημα του ανεμιστήρα, υπερθέρμανση στης αντίστασης, καταστροφή της ή και πυρκαγιά.Για να απενεργοποιηθεί ο επιτηρητής (WDT) πρέπει το bit WDTE της διαμόρφωσης (configuration word) (διεύθυνση μνήμης ram 2007h) του μικροεπεξεργαστή να δηλωθεί ως "0".



Ο επιτηρητής (WDT) πρέπει να μηδενίζεται πριν υπερχειλίσει, μέσα από το πρόγραμμα με την χρήση της εντολής "CLRWDT". Εάν περάσουν τα 18msec και δεν μηδενιστεί τότε αναγκάζει τον μικροεπεξεργαστή να μηδενιστεί (reset) και να ξεκινήσει την εκτέλεση του προγράμματος από την αρχή. Αν χρειάζεται μεγαλύτερη διάρκεια πριν την υπερχείλιση του επιτηρητή (WDT) μπορεί να χρησιμοποιηθεί ο προδιαιρέτης (prescaler).

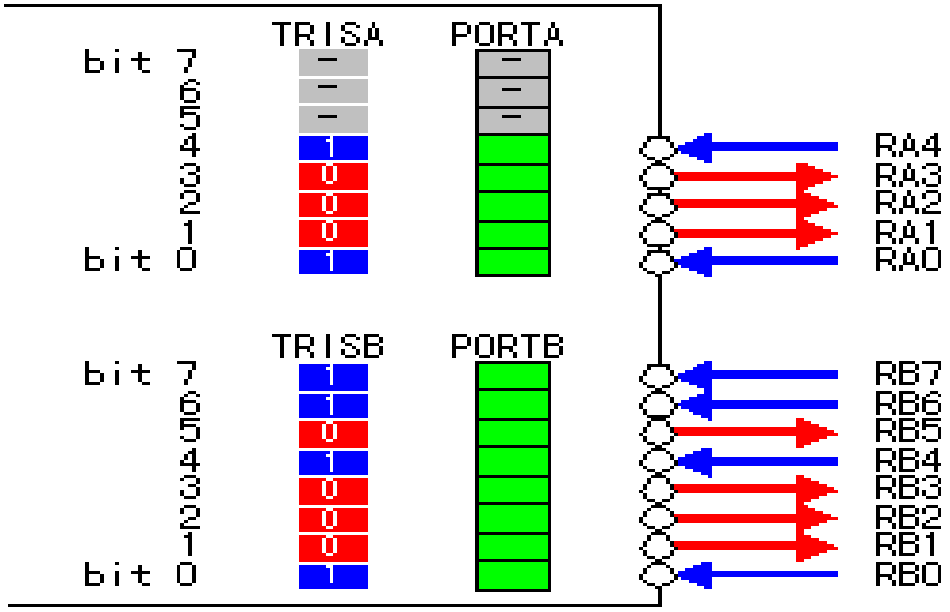
2.7. Είσοδοι/Έξοδοι (I/O).[6,10]

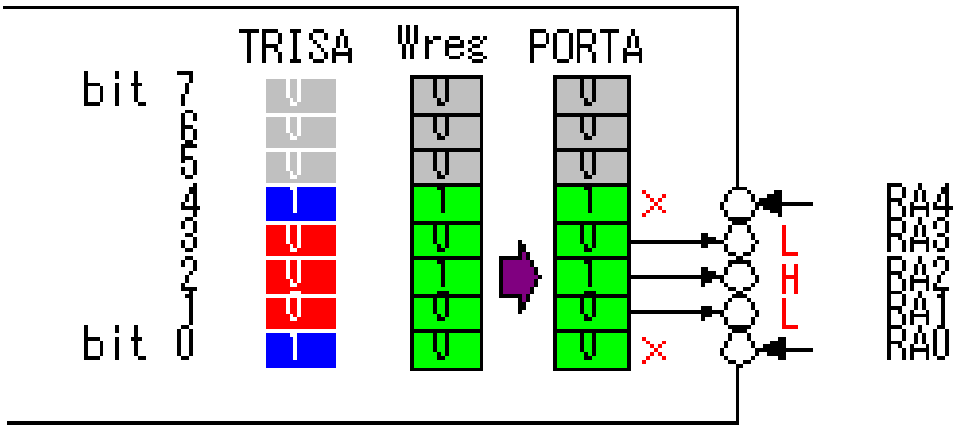
Τις πόρτες εισόδου & εξόδου τις χρησιμοποιεί ο μικροεπεξεργαστής για να επικοινωνεί με το υπόλοιπο κύκλωμα. Ο PIC16F84A έχει 13 ποδαράκια εισόδου - εξόδου (input/output pins). Αυτά χωρίζονται σε ένα σετ των πέντε που ονομάζεται **A PORT** και ένα σετ των οκτώ που ονομάζεται **B PORT.** Η A port αντιστοιχεί στον καταχωρητή**PORTA** και η B port αντιστοιχεί στον καταχωρητή **PORTB**. Οι δύο αυτοί καταχωρητές συγκροτούνται από 8 bit και κάθε ένα ποδαράκι εισόδου - εξόδου τους (input/output pin) αντιστοιχεί σε ένα bit. Για την **PORTA**, χρησιμοποιούνται μόνο πέντε από τα οκτώ bit, δηλαδή το bit 0 έως bit 4. Για την **PORTB**, και τα οκτώ bit χρησιμοποιούνται. Η κατάσταση (αν θα είναι είσοδος ή έξοδος) για κάθε ένα ποδαράκι (pin) καθορίζεται από τον καταχωρητή **TRISA** (για την PORTA) και τον καταχωρητή **TRISB** (για την PORTB). Εάν δηλώσετε κάποιο ποδαράκι ως "0" στον καταχωρητή **TRISx** σημαίνει ότι το ποδαράκι αυτό θα εργαστεί σαν έξοδος ενώ αν το δηλώσετε ως "1" θα εργαστεί σαν είσοδος.

Η δήλωση της κατάστασης για κάθε πόρτα γίνεται με το φόρτωμα (καλλίτερα σε δυαδική μορφή για να αποφύγετε τα λάθη) στον καταχωρητή W του αριθμού που αντιστοιχεί στην επιθυμητή διαμόρφωση και την μεταφορά του

στον καταχωρητή**TRISx** με την χρήση της εντολής MOVWF. Αν για παράδειγμα θέλετε να δηλώσετε στην **PORTB** τα ποδαράκια RB0 έως RB3 σαν εξόδους και τα RB4 έως RB7 σαν εισόδους θα πρέπει να μεταφέρετε μέσο του W στον **TRISB** τον δυαδικό αριθμό **b'11110000** ή τον δεκαδικό **240** ή τον δεκαεξαδικό **F0h.**

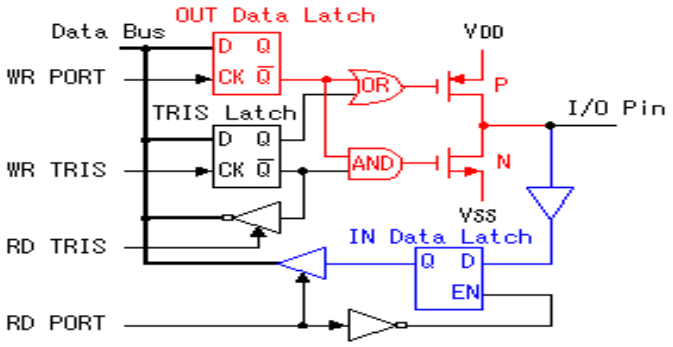
Το ποδαράκι RA4 μπορεί να λειτουργήσει σαν είσοδος - έξοδος ή σαν είσοδος εξωτερικού χρονισμού (clock input) για τον χρονομετρητή TMR0. Στην B port, τα ποδαράκια RB4 έως RB7 μπορούν σαν είσοδοι να παράγουν παλμούς διακοπής ροής προγράμματος (Interrupts), αν μεταβληθεί η κατάσταση τους (από είσοδος σε έξοδο ή αντίστροφα). Επίσης το ποδαράκι RB0 μπορεί να παράγει παλμό διακοπής (external interrupt).





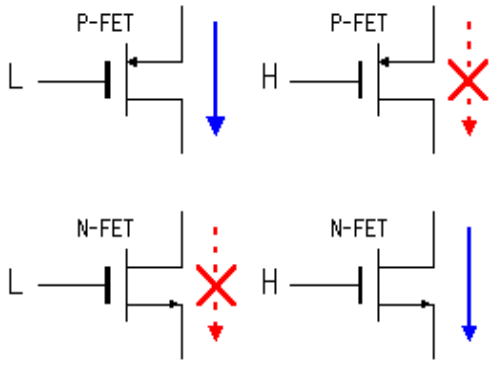
2.7.1. PORT A.

Το εσωτερικό της A port (ποδαράκια RA0-RA3).



Στην πάνω φωτογραφία απεικονίζονται τα τέσσερα ποδαράκια RA0, RA1, RA2 και RA3 της A port.Το κύκλωμα εξόδου συγκροτείται από τον καταχωρητή εξόδου (τύπου συγκράτησης, Latch) και την οδηγό βαθμίδα CMOS.

Λειτουργία του κυκλώματος εξόδου (Γενικά):

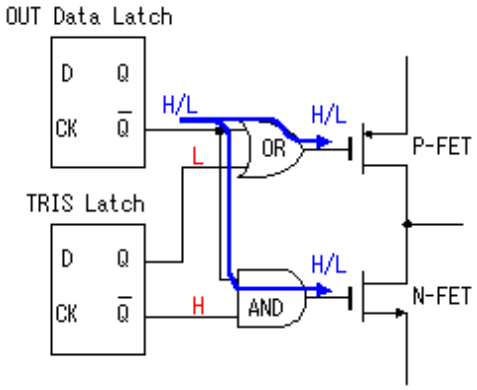


Το κύκλωμα CMOS (Complementary-Metal Oxide Semiconductor) συνδυάζει ένα N-channel MOSFET(N-FET) και ένα P-channel MOSFET(P-FET). Στην περίπτωση του P-FET, όταν η πύλη του πάρει χαμηλό δυναμικό (Low level), τότε αυτό άγει, ενώ όταν πάρει η πύλη υψηλό δυναμικό (Hi level),δεν άγει. Αντίθετα τώρα, στην περίπτωση του, N-FET, όταν η πύλη του πάρει χαμηλό δυναμικό (Low level), δεν άγει, ενώ όταν πάρει η πύλη υψηλό δυναμικό (Hi level), άγει. Όταν η έξοδος σε κάποιο ποδαράκι είναι 1 (+ 5Volt Hi level), η ανεστραμμένη έξοδος (Q̄) του data latch register γίνεται 0 (0Volt Low level), το P-FET άγει ενώ το N-FET δεν άγει και έτσι το ποδαράκι δίνει έξοδο 1 (+ 5Volt Hi level). Αντίστροφα όταν η έξοδος είναι 0 (0Volt Low level),η ανεστραμμένη έξοδος (Q̄) του data latch register γίνεται 1 (+ 5Volt Hi level) το P-FET δεν άγει ενώ το N-FET άγει και δίνει έξοδο 0 (0Volt Low level). Επειδή το κύκλωμα είναι C-MOS η κατανάλωση είναι πολύ μικρή.

Λειτουργία κυκλώματος εισόδου – εξόδου:

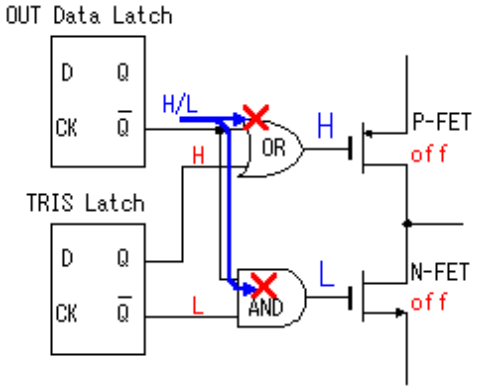
Η κατάσταση για κάθε ποδαράκι της A PORT δηλώνεται στον καταχωρητή TRISA. Το περιεχόμενο του καταχωρητή TRISA αντιγράφεται στον καταχωρητή TRIS latch. Η ανεστραμμένη έξοδος ($\bar{Q}$) του καταχωρητή συγκράτησης δεδομένων εξόδου και η έξοδος (Q) του καταχωρητή TRIS latch εισάγονται στο P-FET διαμέσου της πύλης OR. Η ανεστραμμένη έξοδος ($\bar{Q}$) του καταχωρητή συγκράτησης δεδομένων εξόδου και η ανεστραμμένη έξοδος (Q) του καταχωρητή TRIS latch εισάγονται στο N-FET διαμέσου της πύλης AND.

Λειτουργία κυκλώματος σε κατάσταση Εξόδου:



Όταν ορίζεται ένα ποδαράκι σαν έξοδος, το Bit που του αντιστοιχεί στον καταχωρητή TRISA γίνεται "0".Επειδή η έξοδος (Q) του καταχωρητή TRIS είναι εξαρχής (by default) ορισμένη με 0 (Low level), η ανεστραμμένη έξοδος ($\bar{Q}$) του data latch register εφαρμόζεται δίχως αλλαγή στο P-FET. Επίσης η ανεστραμμένη έξοδος ($\bar{Q}$) του καταχωρητή TRIS είναι 1 (Hi level) και η ανεστραμμένη έξοδος ($\bar{Q}$) του data latch register εφαρμόζεται δίχως αλλαγή στο N-FET. Όταν η έξοδος γίνεται 1 (Hi level), το P-FET άγει, ενώ το N-FET δεν άγει και αντίστροφα όταν η έξοδος γίνεται 0 (Low level).

Λειτουργία κυκλώματος σε κατάσταση Εισόδου:



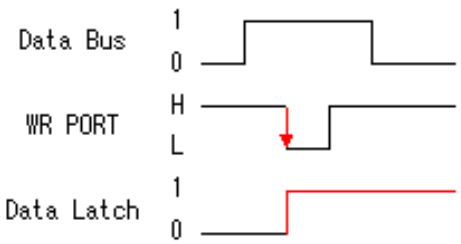
Όταν ορίζεται ένα ποδαράκι σαν έξοδος, το Bit που του αντιστοιχεί στον καταχωρητή TRISA γίνεται "1".Η έξοδος (Q) του καταχωρητή TRIS είναι 1 (Hi level), η έξοδος της πύλης OR είναι 1 (Hi level) και το P-FET δεν άγει. Επίσης η ανεστραμμένη έξοδος ($\bar{Q}$) του καταχωρητή TRIS είναι 0 (Low level), η έξοδος της πύλης AND είναι 0 (Low level), και το N-FET δεν άγει. Παρατηρήστε πώς τα P-FET και N-FET δεν άγουν, το ποδαράκι απομονώνεται από το κύκλωμα εξόδου και λειτουργεί σαν είσοδος.

Λειτουργία κυκλώματος εισόδου:

Το κύκλωμα εισόδου είναι πάντοτε συνδεδεμένο με τα ποδαράκια του μικροεπεξεργαστή (I/O pin), ακόμα και όταν αυτά λειτουργούν σαν έξοδοι. Το σήμα εισόδου για κάθε ποδαράκι καταχωρείται στον καταχωρητή input data latch με την χρήση μιας ενδιάμεσης μνήμης TTL (buffer).

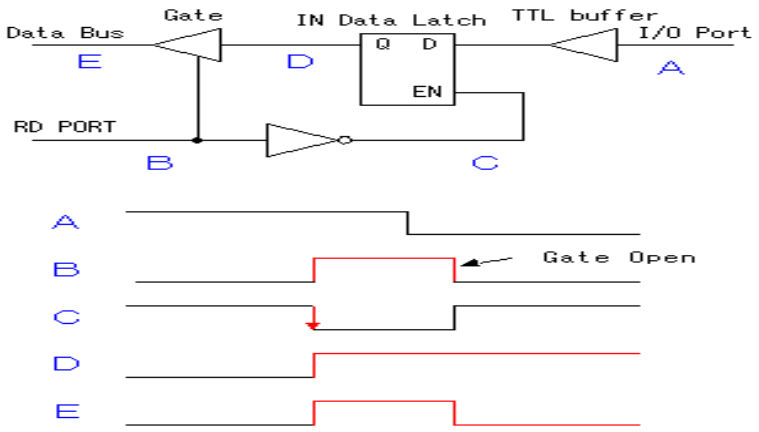
Χρονισμός εισόδου – εξόδου:

Η μετάβαση από τον καταχωρητή input data στον output data latch και οι ρυθμίσεις στον TRIS latch γίνονται με την χρήση ενός διαύλου δεδομένων (data bus). Ο data bus έχει 8 κανάλια και η μεταφορά των δεδομένων από και προς τα ποδαράκια γίνεται παράλληλα (parallel). Επειδή ο data bus χρησιμοποιείται για πολλές λειτουργίες, η μεταφορά των δεδομένων ελέγχεται από παλμό χρονισμού (timing pulse) που παράγει ο κάθε καταχωρητής (control signal).



Όταν εξάγονται τα περιεχόμενα του καταχωρητή PORTA, πρώτα μεταφέρονται στον data bus. Στην συνέχεια το σήμα ελέγχου του output data latch register (WR PORT) γίνεται 0 (Low level) από 1 (Hi level), και τα δεδομένα από το δίαυλο data bus μεταφέρονται στον καταχωρητή data latch. Εκεί μένουν αναλλοίωτα, (ακόμα και αν τα δεδομένα του διαύλου data bus αλλάξουν) έως ότου το σήμα ελέγχου γίνει ξανά 0 (Low level) από 1 (Hi level).

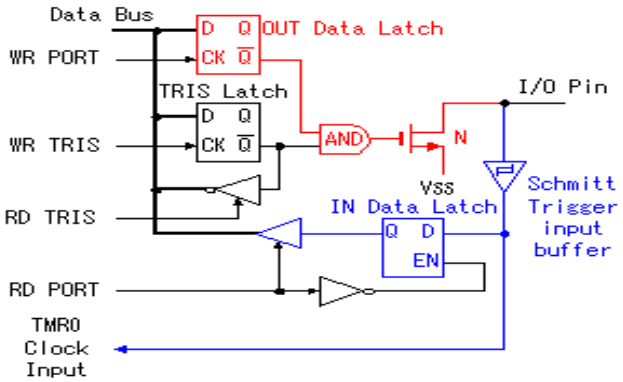
Το ίδιο ακριβώς συμβαίνει και με τον καταχωρητή TRIS latch. Τα περιεχόμενα του καταχωρητή TRISA, πρώτα μεταφέρονται στον data bus. Στην συνέχεια το σήμα ελέγχου του καταχωρητή TRIS latch (WR TRIS) γίνεται 0 (Low level) από 1 (Hi level), και τα δεδομένα από το δίαυλο data bus μεταφέρονται στον καταχωρητή TRIS latch.
Ο αναστροφέας χρησιμοποιείται για την ανάγνωση των περιεχομένων του καταχωρητή TRIS latch. Τα περιεχόμενα του καταχωρητή TRIS latch μεταφέρονται στον δίαυλο data bus όταν το RD TRIS γίνει "1".



Ο αναστροφέας (πύλη NOT) που βρίσκεται ανάμεσα στον καταχωρητή data latch και τον δίαυλο data bus λειτουργεί όταν το σήμα ανάγνωσης (RD PORT) γίνεται 1 (Hi level) από 0 (Low level). Ταυτόχρονα η είσοδος ελέγχου του καταχωρητή input data latch (EN) γίνεται 0 (Low level) από 1 (Hi level), η κατάσταση που έχει το ποδαράκι μεταφέρεται στον καταχωρητή nput data latch και τέλος στο δίαυλο data bus διαμέσου του αναστροφέα (πύλη NOT).
Ο καταχωρητής input data latch χρησιμοποιείται διότι δεν επιδρά στα δεδομένα που μεταφέρονται διαμέσου του δίαυλου data bus ακόμα και όταν η κατάσταση που βρίσκεται το ποδαράκι μεταβληθεί κατά την διάρκεια ανάγνωσης του.

Το εσωτερικό της A port (ποδαράκι RA4):

Το ποδαράκι RA4 της A port έχει σύνθετη λειτουργία. Στο σχήμα φαίνεται το κύκλωμα του. Η διαφορά του από τα άλλα ποδαράκια είναι ότι για την οδήγηση της εξόδου υπάρχει μόνο ένα N-FET και σαν ενδιάμεση μνήμη εισόδου (Input Buffer) χρησιμοποιείται ένας Schmitt trigger.



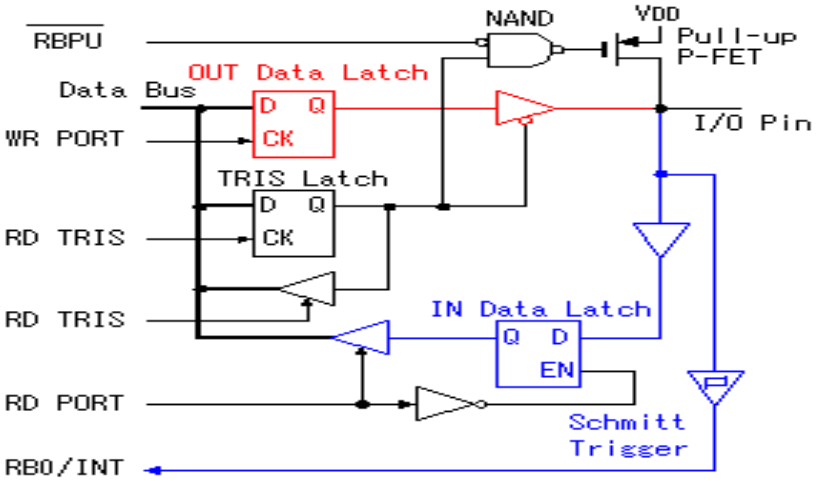
Αυτό το ποδαράκι μπορεί να χρησιμοποιηθεί σαν είσοδος εξωτερικού χρονισμού για τον TMR0. Επειδή χρησιμοποιεί τον Schmitt trigger σαν buffer εισόδου μπορεί εύκολα να ξεχωρίσει το 1 (Hi level) ή το 0 (Low level), όταν το μέτωπο (ανερχόμενο ή κατερχόμενο) του εξωτερικού σήματος χρονισμού δεν είναι ξεκάθαρο.Επειδή δεν υπάρχει P-FET στο κύκλωμα οδήγησης της εξόδου οι αντιστάσεις pull-up (συνδέει το ποδαράκι στην θετική τάση +5Volt μέσο αντίστασης, συνήθως 10K Ohm) πρέπει να συνδεθούν εξωτερικά.

2.7.2. PORT B.

Το εσωτερικό της B port (ποδαράκια RB0-RB3):

Το κύκλωμα εξόδου της B port είναι διαφορετικό από αυτό της A port. Δεν χρησιμοποιεί FET για την οδήγηση της εξόδου και η αλλαγή κατάστασης από είσοδο σε έξοδο και αντίστροφα γίνεται με πύλες. Χαρακτηριστικό της B port είναι πώς έχει ενσωματωμένες μέσα στο σώμα του μικροεπεξεργαστή, pull-up αντιστάσεις που μπορούν να χρησιμοποιηθούν όταν η B port βρίσκεται σε κατάσταση εισόδου.

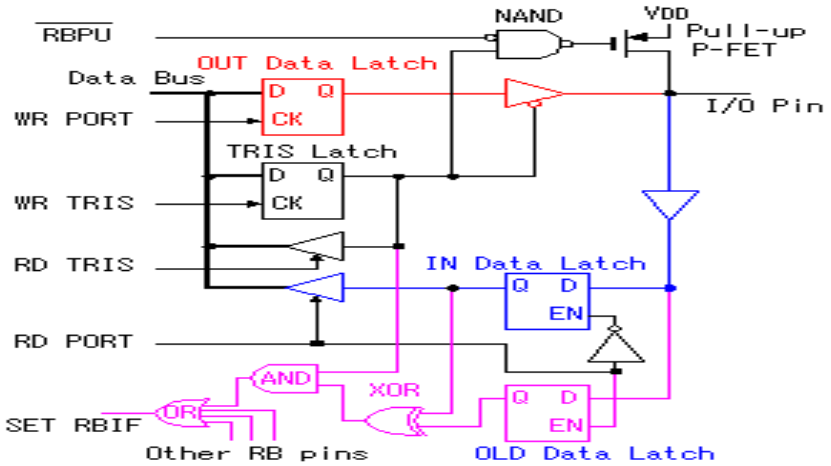
Το ποδαράκι RB0 μπορεί να χρησιμοποιηθεί και σαν ποδαράκι εισόδου εξωτερικού παλμού διακοπής ροής προγράμματος (external input interrupt), και συνδέεται με τα εσωτερικά κυκλώματα του μικροεπεξεργαστή μέσο ενδιάμεσης μνήμης Schmitt trigger buffer.



Για να βρεθεί ένα ποδαράκι της B Port σε κατάσταση εξόδου πρέπει ο καταχωρητής TRIS να γίνει "0". Επειδή υπάρχει αναστροφέας στην γραμμή εξόδου η πύλη λειτουργεί, όταν η έξοδος του καταχωρητή TRIS latch είναι 0 (Low level). Η είσοδος της πύλης NAND για pull-up είναι σε 0 (Low level), και η έξοδος είναι 1 (Hi level). Επομένως το P-FET δεν άγει και οι pull-up δεν λειτουργούν.Όταν ο καταχωρητής TRIS γίνει "1", τότε έχουμε κατάσταση εισόδου. Η έξοδος δεν λειτουργεί, και το σήμα εξόδου απομονώνεται από το ποδαράκι. Εάν θα λειτουργήσουν οι pull-up ή όχι ρυθμίζεται από το RBUP. Το **RBPU** είναι το **bit 7** του καταχωρητή OPTION-REG, για να ενεργοποιηθούν οι εσωτερικές αντιστάσεις "Pull-up" πρέπει το bit να είναι "0" για να απενεργοποιηθούν πρέπει να είναι "1". στην περίπτωση της απενεργοποίησης "0", η πύλη του P-FET γίνεται 0 (Low level) και το P-FET άγει. Στην περίπτωση της ενεργοποίησης "1", συμβαίνει το αντίθετο.Ρυθμίζοντας το RBPU, ενεργοποιούνται ή απενεργοποιούνται όλες οι εσωτερικές αντιστάσεις pull up της B port. Δεν είναι δυνατόν να γίνει ρύθμιση για κάθε ποδαράκι ξεχωριστά.

Το εσωτερικό της B port (ποδαράκια RB4-RB7):

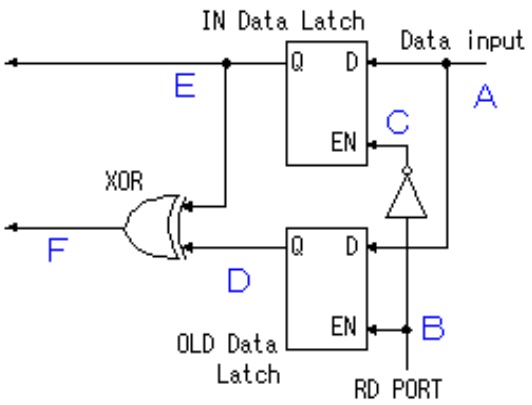
Τα ποδαράκια RB4 έως RB7 έχουν την δυνατότητα να ανιχνεύουν αλλαγή κατάστασης (από 1 (Hi level) σε το 0 (Low level και αντίστροφα)του εισερχόμενου σήματος και να δημιουργούν παλμό διακοπής ροής προγράμματος (Interrupt). Η ανίχνευση γίνεται ταυτόχρονα και στα τέσσερα ποδαράκια. Αυτή η δυνατότητα λειτουργεί μόνο όταν τα ποδαράκια βρίσκονται σε κατάσταση εισόδου.



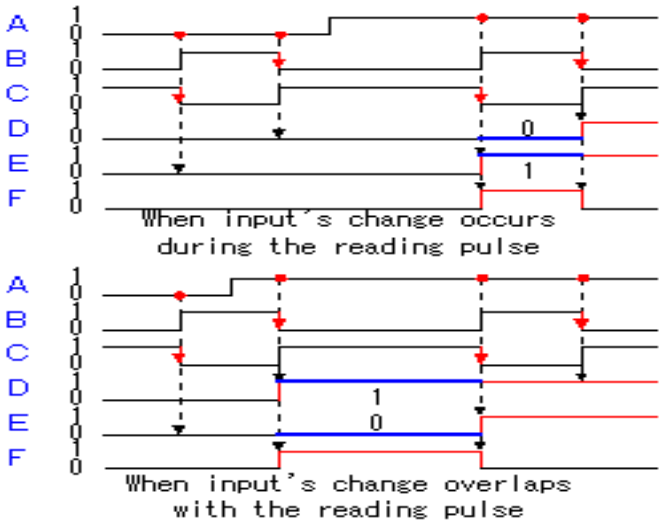
Ο καταχωρητής OLD data latch και η πύλη Exclusive OR(XOR) χρησιμοποιούνται για την ανίχνευση της αλλαγής κατάστασης σε συνδυασμό με τον καταχωρητή input data latch.Ο πίνακας αλήθειας της πύλης XOR είναι ο ακόλουθος.

A	B	Y
L(0)	L(0)	L(0)
L(0)	H(1)	H(1)
H(1)	L(0)	H(1)
H(1)	H(1)	L(0)

Το σήμα εισόδου διαβάζεται από τον καταχωρητή input data latch μέσο του αναστροφέα όταν το RD Port γίνει 1 (Hi level) από 0 (Low level). Στη συνεχεία μεταφέρεται στον καταχωρητή OLD data latch όταν το RD Port γίνει 0 (Low level) από 1 (Hi level).
Επειδή οι έξοδοι των δυο καταχωρητών είναι ίδιες, αν δεν υπάρξει αλλαγή στο σήμα εισόδου, η έξοδος της πύλης XOR είναι 0 (Low level).Το κύκλωμα φαίνεται παρακάτω:



Όταν η RD Port γίνει από 0 (Low level) σε 1 (Hi level) ο καταχωρητής input data latch διαβάζει την κατάσταση που έχει το ποδαράκι εισόδου. Η είσοδος σε κάθε πύλη XOR είναι διαφορετική και η έξοδος είναι 1 (Hi level), αυτή η έξοδος είναι και το σήμα ενεργοποίησης (σκανδαλισμός - trigger) του παλμού διακοπής ροής προγράμματος (Interrupt).
Όταν η RD Port γίνει από 1 (Hi level) σε 0 (Low level) ο καταχωρητής OLD input data latch διαβάζει την κατάσταση που έχει το ποδαράκι εισόδου. Η είσοδος στις πύλες XOR είναι ίδια και η έξοδος είναι 0 (Low level).



Όταν το σήμα εισόδου μεταβληθεί το RD Port γίνεται 0 (Low level) από 1 (Hi level), και διαβάζεται από τον καταχωρητή OLD data latch, σε αυτό το σημείο επειδή η μεταβολή δεν έχει ακόμα διαβαστεί από τον καταχωρητή input data latch η κατάσταση των δύο καταχωρητών είναι διαφορετική, η έξοδος της XOR είναι 1 (Hi level), και δημιουργείται ο παλμός διακοπής (Interrupt).Η έξοδος της πύλης XOR μένει σε αυτή την κατάσταση μέχρι τον επόμενο παλμό ανάγνωσης του RD Port. Παλμός διακοπής μπορεί να δημιουργηθεί ξανά όταν η έξοδος της πύλης XOR γίνει από 0 (Low level) 1 (Hi level), ακόμα και αν το bit RBIF μηδενίζεται εκείνη την στιγμή από προηγούμενο παλμό (Interrupt).

Βιβλιογραφία κεφαλαίου

[1]Μικροελεγκτές PIC" Σταμάτης Αλατσαθιανός, Β.Γκιούρδας Εκδοτική
[2]Microcontroller Programming" Julio Sanchez, Maria P. Canton
[3]Mano Morris M., Ψηφιακή Σχεδίαση, (Κεφ. 7), Εκδόσεις Παπασωτηρίου, 1992
[4]Wakerly J. F., Digital Design: Principles and Practices, 2nd Edition, (Chap. 11),
[5]http://www.pi-schools.gr/lessons/tee/electronic/biblia.php
[6]http://ww1.microchip.com/downloads/en/DeviceDoc/41262c.pdf
[7]http://www.gooligum.com.au/
[8]http://www.electronics-lab.com/pic-in-greek/
[9]http://en.wikipedia.org/wiki/PIC_microcontroller
[10]http://el.wikipedia.org/wiki/Reduced_instruction_set_computer

ΚΕΦΑΛΑΙΟ 3

ΓΛΩΣΣΑ ASSEMBLY

3.1 Γλώσσες Προγραμματισμού.[4,5,7]

Γλώσσα προγραμματισμού λέγεται μια τεχνητή γλώσσα που μπορεί να χρησιμοποιηθεί για τον έλεγχο μιας μηχανής, συνήθως ενός υπολογιστή. Οι γλώσσες προγραμματισμού (όπως άλλωστε και οι ανθρώπινες γλώσσες) ορίζονται από ένα σύνολο συντακτικών και εννοιολογικών κανόνων, που ορίζουν τη δομή και το νόημα, αντίστοιχα, των προτάσεων της γλώσσας.

Οι γλώσσες προγραμματισμού χρησιμοποιούνται για να διευκολύνουν την οργάνωση και διαχείριση πληροφοριών, αλλά και για την ακριβή διατύπωση αλγορίθμων. Ορισμένοι ειδικοί χρησιμοποιούν τον όρο γλώσσα προγραμματισμού μόνο για τυπικές γλώσσες που μπορούν να εκφράσουν όλους τους πιθανούς αλγορίθμους. Μη-υπολογιστικές γλώσσες όπως ηHTML ή τυπικές γραμματικές όπως η BNF δεν λέγονται συνήθως γλώσσες προγραμματισμού.

3.1.1Χαρακτηριστικά των Γλωσσών Προγραμματισμού.

Κάθε γλώσσα προγραμματισμού έχει το δικό της σύνολο τυπικών προδιαγραφών (ή κανόνων) που αφορούν το συντακτικό, το λεξιλόγιο και το νόημα της. Για πολλές γλώσσες που χρησιμοποιούνται ευρέως και έχουν χρησιμοποιηθεί για αρκετό χρονικό διάστημα (π.χ. C, C++, Java, Scheme), υπάρχουν ειδικοί φορείς τυποποίησης, οι οποίοι μέσα από τακτές συναντήσεις δημιουργούν, τροποποιούν ή επεκτείνουν τις τυπικές προδιαγραφές που διέπουν τη χρήση μιας γλώσσας προγραμματισμού. Άλλες γλώσσες δεν περιγράφονται σε κάποιο επίσημο πρότυπο αλλά ορίζονται μόνο με βάση κάποια υλοποίησή τους (που αποτελεί το ντε φάκτο πρότυπο), όπως η Python που περιγράφεται από την υλοποίηση CPython.

Για να διευκολυνθεί η διαδικασία ανάπτυξης λογισμικού δημιουργήθηκαν εργαλεία με στόχο την αυτοματοποίηση της ανάπτυξης ορισμένων τμημάτων του κώδικα. Τα εργαλεία αυτά αποσκοπούν:
– στην αύξηση της παραγωγικότητας των προγραμματιστών αλλά
– και στην μετακίνηση ορισμένων σημείων της κωδικοποίησης από τους προγραμματιστές προς τους αναλυτές, τους σχεδιαστές, και προς τους τελικούς χρήστες των Π.Σ.

Τα περισσότερα από τα εργαλεία αυτά είναι γνωστά ως γεννήτριες γιατί δέχονται την περιγραφή ενός τμήματος του Π.Σ. και αναπτύσσουν τον κώδικα του προγράμματος που αντιστοιχεί στο τμήμα αυτό αυτόματα.
Έχουν αναπτυχθεί γεννήτριες
– αναφορών και
– γεννήτριες οθονών.

Γεννήτριες αναφορών:Η ανάπτυξη κώδικα για την προετοιμασία αναφορών παρόλο που ακολουθεί συγκεκριμένους κανόνες εμπεριέχει αρκετές λεπτομέρειες:
– υπολογισμός μερικών αθροισμάτων,
– αλλαγή σελίδων,
– σελιδοποίηση,
– ολικά αθροίσματα
Χρησιμοποιώντας μία γεννήτρια αναφορών (report generator) ένας προγραμματιστής μπορεί να ορίσει τη μορφή της αναφοράς προσδιορίζοντας τα περιεχόμενα της αναφοράς. Οι γεννήτριες αναφορών έχουν πρόσβαση σε αρχεία ή σε βάσεις δεδομένων από τις οποίες εξάγουν δεδομένα τα οποία μορφοποιούν σε αναφορές.Ο κώδικας για την υλοποίηση των αντίστοιχων υπορουτινών δημιουργείται αυτόματα από την γεννήτρια.

Γεννήτριες οθονών,προγραμμάτων:Οι γεννήτριες οθονών (screen generator) είναι προγράμματα τα οποία επιτρέπουν την εύκολη και γρήγορη ανάπτυξη του interface ενός Π.Σ., προσδιορίζοντας τα περιεχόμενα της κάθε οθόνης, χωρίς την ανάγκη προγραμματισμού. Μία άλλη κατηγορία γεννητριών είναι οι γεννήτριες προγραμμάτων οι οποίες δέχονται ως είσοδο την περιγραφή ενός συστήματος σε μορφή που είναι εύκολο να δοθεί από τον χρήστη και παράγουν αυτόματα τον κώδικα που αντιστοιχεί στο σύστημα.Οι γλώσσες αναζητήσεων επιτρέπουν την εύκολη επικοινωνία του χρήστη με τον υπολογιστή κυρίως για την αναζήτηση δεδομένων που φυλάσσονται σε βάσεις δεδομένων. Βάση δεδομένων είναι μία συλλογή από σχετιζόμενα δεδομένα.

Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ, Database Management System):Είναι ένα σύνολο προγραμμάτων που είναι υπεύθυνο για την δημιουργία και συντήρηση των βάσεων δεδομένων. Το σύστημα διαχείρισης βάσεων δεδομένων ενός Π.Σ προσφέρει δυνατότητες για την αποθήκευση, την ανάκτηση (αναζήτηση) και τον έλεγχο των δεδομένων που χρειάζονται για την
λήψη αποφάσεων.Τα ΣΔΒΔ ανάλογα με τον τρόπο που οργανώνουν τα δεδομένα στη βάση δεδομένων κατηγοριοποιούνται σε :
– ιεραρχικά (hierarchical),
– δικτυωτά (network),
– σχεσιακά (relational) και
– αντικειμενοστραφή (object-oriented).

Λεξικό Δεδομένων:Είναι ένας κατάλογος όλων των δεδομένων που περιεχονται στην βάση δεδομένων. Εκτός από τον ορισμό των δεδομένων μπορεί να περιέχει την περιγραφή και την πηγή τους.

3.1.2 Κατηγοριοποίηση Γλωσσών Προγραμματισμού.

Δεν υπάρχει απλός τρόπος να κατηγοριοποιηθούν οι γλώσσες προγραμματισμού. Αυτό συμβαίνει γιατί συνήθως κάθε γλώσσα προγραμματισμού περιέχει επιρροές από πολλές προηγούμενες γλώσσες, συνδυάζοντας θετικά στοιχεία και προσθέτοντας νέα. Χαρακτηριστικά που εμφανίζονται σε μια γλώσσα και έχουν θετική αποδοχή, συνήθως υιοθετούνται από μεταγενέστερες γλώσσες ακόμα και αν πρόκειται για γλώσσες που ανήκουν σε διαφορετική κατηγορία.

Η κατηγοριοποίηση είναι ακόμα πιο περίπλοκη για το λόγο ότι πολλές γλώσσες συνήθως ανήκουν σε παραπάνω από μία κατηγορίες. Για παράδειγμα, η Java είναι τόσο αντικειμενοστραφής όσο και παράλληλη γλώσσα, δεδομένου ότι υποστηρίζει την οργάνωση των δεδομένων και υπολογισμών σε αντικείμενα, αλλά επιτρέπει επίσης και την δημιουργία προγραμμάτων με ταυτόχρονα νήματα (threads) που εκτελούνται παράλληλα.Δεδομένης της δυσκολίας στην κατηγοριοποίηση, μπορούμε να κατηγοριοποιήσουμε τις γλώσσες προγραμματισμού με διάφορους τρόπους. Οι συνηθέστεροι τρόποι είναι:

• με βάση τον τρόπο οργάνωσης του προγράμματος
• με βάση τον στόχο που έχει η γλώσσα
• με βάση τον τρόπο που περιγράφουν το ζητούμενο αποτέλεσμα.

Στην πρώτη περίπτωση προκύπτουν κατηγορίες όπως:

- Διαδικαστικές γλώσσες (procedural) όπου το πρόγραμμα είναι οργανωμένο σε διαδικασίες, που αποτελούνται από σειρές εντολών που περιγράφουν αλγορίθμους. Παραδείγματα γλωσσών που ανήκουν σε αυτή την κατηγορία είναι η Pascal ή η C.
- Αντικειμενοστρεφείς γλώσσες (object-oriented) όπου το πρόγραμμα είναι οργανωμένο σε αντικείμενα. Ένα αντικείμενο είναι μια μονάδα που αποτελείται από την περιγραφή κάποιων δεδομένων και την περιγραφή των αλγορίθμων που τα επεξεργάζονται. Ένα αντικειμενοστρεφές πρόγραμμα αποτελείται από διάφορα αντικείμενα που αλληλεπιδρούν μεταξύ τους. Παραδείγματα αντικειμενοστρεφών γλωσσών είναι η Java ή η C++.
- Συναρτησιακές γλώσσες (functional) όπου οι υπολογισμοί εκφράζονται ως εφαρμογές μαθηματικών συναρτήσεων, σε αντίθεση με τα άλλα είδη προγραμματισμού όπου οι υπολογισμοί εκφράζονται ως σειρές εντολών, όπου η κάθε μία αλλάζει με κάποιο τρόπο την κατάσταση του συστήματος. Θεωρητικό τους υπόβαθρο είναι ο λ-λογισμός. Χαρακτηριστικές συναρτησιακές γλώσσες είναι η Lisp, η Haskell και η OCaml.

Στην περίπτωση που οι κατηγοριοποίηση των γλωσσών προγραμματισμού γίνει με βάση το στόχο που έχει η γλώσσα, υπάρχουν οι παρακάτω κατηγορίες:

- Γλώσσες γενικής χρήσης. Σε αυτήν την κατηγορία ταξινομούνται γλώσσες που δημιουργήθηκαν για τον προγραμματισμό γενικών εφαρμογών, καθώς και πολλές εκπαιδευτικές γλώσσες που αποδείχτηκαν χρήσιμες για την ανάπτυξη γενικών εφαρμογών, όπως η Pascal.
- Γλώσσες προγραμματισμού συστημάτων, που χρησιμοποιούνται συνήθως για τον προγραμματισμό λειτουργικών συστημάτων ή οδηγών (drivers) υλικού, όπου χρειάζεται πολλές φορές ο προγραμματιστής να έχει έλεγχο και γνώση του πως λειτουργεί το υλικό. Η πιο συχνά χρησιμοποιούμενη γλώσσα προγραμματισμού συστημάτων είναι η C.
- Γλώσσες σεναρίων (scripting). Αυτές οι γλώσσες χρησιμοποιούνται συνήθως για τη γρήγορη ανάπτυξη μικρών προγραμμάτων, σε περιπτώσεις που ο χρόνος του προγραμματιστή είναι πιο πολύτιμος από την ταχύτητα εκτέλεσης του προγράμματος, όπως για παράδειγμα συμβαίνει όταν το πρόγραμμα απλά αυτοματοποιεί απλές λειτουργίες. Παραδείγματα γλωσσών σεναρίων (scripting) είναι η Perl, η Python, η Ruby ή τα κελύφη του λειτουργικού συστήματος Unix (shells).
- Γλώσσες ειδικών εφαρμογών. Σε αυτή την κατηγορία ανήκουν γλώσσες που αναπτύχθηκαν ειδικά για μια συγκεκριμένη εφαρμογή. Για παράδειγμα, η γλώσσα PostScript είναι σχεδιασμένη ειδικά για να περιγράφονται με λεπτομέρεια κείμενα προς εκτύπωση, ενώ η γλώσσα Matlab είναι σχεδιασμένη για την επεξεργασία πινάκων από αριθμητικά δεδομένα.
- Παράλληλες ή κατανεμημένες γλώσσες. Στη συγκεκριμένη κατηγορία ταξινομούνται γλώσσες που επιτρέπουν τη ανάπτυξη παράλληλων προγραμμάτων, όπου πολλές εντολές εκτελούνται ταυτόχρονα σε πολλούς υπολογιστές, έτσι ώστε το τελικό αποτέλεσμα να προκύψει γρηγορότερα. Οι παράλληλες γλώσσες προσφέρουν συνήθως εύκολους τρόπους επικοινωνίας μεταξύ των νημάτων που εκτελούνται παράλληλα, καθώς και τρόπους ώστε να δημιουργούνται καινούριες παράλληλες εκτελέσεις. Παραδείγματα γλωσσών που ανήκουν (και) σε αυτή την κατηγορία είναι η Java, η Erlang, η MultiLisp ή η Cilk.

Τέλος, στην περίπτωση που η κατηγοριοποίηση γίνεται με βάση τον τρόπο που περιγράφεται το ζητούμενο, υπάρχουν οι παρακάτω κατηγορίες:

- Προστακτικές γλώσσες προγραμματισμού (imperative) είναι οι γλώσσες που περιγράφουν το ζητούμενο αποτέλεσμα κατασκευαστικά, δίνοντας μια σειρά εντολών που όταν εκτελεστούν παράγουν το ζητούμενο αποτέλεσμα. Τέτοιες γλώσσες είναι η C, η Java αλλά και η OCaml.
- Δηλωτικές γλώσσες προγραμματισμού (declarative) είναι οι γλώσσες που περιγράφουν το ζητούμενο αποτέλεσμα χρησιμοποιώντας τις ιδιότητες που έχει, και όχι τον τρόπο με τον οποίο υπολογίζεται. Παραδείγματα δηλωτικών γλωσσών είναι η Haskell, η SQL και η Prolog.

3.1.3 Πλήθος Γλωσσών Προγραμματισμού.

κατ' αλφαβητική σειρά:

- Ada
- Algol
- Applescript
- AWK
- BASIC
- C
- C++
- C#
- Cilk
- Clojure
- COBOL
- Datalog
- Erlang
- Forth
- FORTRAN
- Haskell
- Java
- JavaScript
- Lisp
- Logo
- Lua
- Lucid
- Mathematica
- Matlab
- Miranda
- ML
- OBJ / Σύστημα Maude
- Objective-C
- OCaml
- Pascal
- Perl
- PHP
- Prolog
- Python
- Ruby
- Scala
- Scheme
- Simula
- Smalltalk
- SQL
- Tcl
- Visual Basic
- ΓΛΩΣΣΑ

Η παραπάνω λίστα είναι ενδεικτική και σε καμία περίπτωση δεν εξαντλεί το εύρος και την ποικιλία των χιλιάδων γλωσσών που χρησιμοποιούνται στην πράξη.

Θα μπορούσαν *παρόλα αυτά*,οι γλώσσες προγραμματισμού με βάση την ιστορική τους εξέλιξη να ταξινομηθούν και σε "γενιές".

• γλώσσες 1ης γενεάς ή γλώσσες μηχανής (machine languages):Βασίζονται στον δυαδικό κώδικα, είναι άμεσα κατανοητές από τον ηλεκτρονικό υπολογιστή και εξαρτώνται από την συγκεκριμένη μηχανή (machinedependent) δηλ. προγράμματα που γράφονται σε έναν υπολογιστή δεν είναι κατανοητά από άλλον.

• γλώσσες 2ης γενεάς ή συμβολικές γλώσσες (assembly languages):Αναπτύχθηκαν την δεκαετία του '50, απαιτούν μεταφραστές για την μετατροπή τους σε γλώσσα μηχανής, είναι ευκολότερη η εκμάθηση και απομνημόνευση τους.

• γλώσσες 3ης γενεάς ή διαδικαστικές ή υψηλού επιπέδου γλώσσες (procedural languages):Αναπτύχθηκαν από τα τέλη της δεκαετίας του '50, χρησιμοποιούν εκτενώς σύμβολα, υιοθετούν την έννοια της υπορουτίνας, χρησιμοποιούντα για την ανάπτυξη συστημάτων υποστήριξης αποφάσεων παρά το γεγονός ότι δεν περιέχουν ευκολίες για την ανάπτυξη αυτών.

• γλώσσες 4ης γενεάς ή μη διαδικαστικές γλώσσες (nonprocedural languages):Η βασική ιδέα μίας μη διαδικαστικής γλώσσας είναι να μεταφερθεί η ευθύνη της ροής του προγράμματος από τον προγραμματιστή στο λογισμικό. Με τις μη διαδικαστικές γλώσσες ο προγραμματιστής προσδιορίζει τι θέλει να υπολογίσει ο υπολογιστής και όχι τον τρόπο (το πώς) που θα γίνει αυτό. Οι γλώσσες αυτές χρησιμοποιούνται κυρίως για την ανάπτυξη των συστημάτων υποστήριξης αποφάσεων.

• γλώσσες 5ης γενιάς:Είναι συμβολικές γλώσσες που παρέχουν αποτελεσματικούς τρόπους αναπαράστασης αντικειμένων και μεθόδων που χρησιμοποιούνται στην τεχνητή νοημοσύνη.

3.2 Assembly.[1,2,3,6]

3.2.1 Τι είναι η Assembly.

Η Assembly είναι μια γλώσσα προγραμματισμού υπολογιστών. Είναι μια γλώσσα πολύ χαμηλού επιπέδου , αφού επιτρέπει πρόσβαση στις λειτουργίες του επεξεργαστή . Αυτό σημαίνει ότι υπάρχουν πολλές διαφορετικές γλώσσες Assembly , μια για κάθε είδος επεξεργαστή . Στο κείμενο αυτό θα μας απασχολήσει η Assembly του επεξεργαστή x86 της Intel , που είναι σήμερα ο πιο διαδεδομένος επεξεργαστής . Αν έχετε 286 , 386 ή ακόμα και 8086 (γενικώς ότι τελειώνει σε 86) ή ακόμη και Pentium , τότε ο επεξεργαστής σας υποστηρίζει τις εντολές που θα αναφερθούν εδώ .

Η Assembly λέγεται αλλιώς και συμβολική γλώσσα , αφού στην ουσία τα προγράμματα σε αυτή είναι συμβολικές ονομασίες εντολών , που αποτελούν μνημονικά ονόματα (στην Αγγλική γλώσσα) κάθε λειτουργίας του επεξεργαστή. Βέβαια , όπως είναι γνωστό , οι υπολογιστές καταλαβαίνουν μόνο το 0 και το1 , οπότε τα σύμβολα της γλώσσας Assembly πρέπει να μεταφραστούν σε 0 και 1 (δηλαδή σε γλώσσα μηχανής) από ένα πρόγραμμα που καλείται συμβολομεταφραστής (Assembler) , κατά τρόπο ανάλογο με τη μετάφραση προγραμμάτων γλωσσών υψηλού επιπέδου από μεταγλωττιστές και διερμηνείς .

Για να ακολουθήσετε το κείμενο αυτό δεν χρειάζεται να προμηθευτείτε κανένα συμβολομεταφραστή , αφού αυτός υπάρχει ήδη στο σύστημά σας και λέγεται debug . Για την ακρίβεια δεν είναι ακριβώς συμβολομεταφραστής , αλλά μπορούμε να τον χρησιμοποιήσουμε σαν τέτοιο.

3.2.2 Πλεονεκτήματα / Μειονεκτήματα.

Το βασικότερο πλεονέκτημα της Assembly είναι η ταχύτητα των προγραμμάτων που γράφονται σε αυτή . Πραγματικά τα προγράμματα αυτά είναι ταχύτατα και μπορούν να αποδειχθούν εξαιρετικά χρήσιμα σε εφαρμογές όπου η ταχύτητα παίζει σημαντικό ρόλο , όπως πχ στα γραφικά .

Ένα επιπλέον πλεονέκτημα είναι ότι ο προγραμματιστής σε Assembly καταλαβαίνει ακριβώς πως λειτουργεί ο υπολογιστής του . Καταλαβαίνει τον τρόπο σκέψης του και τον προγραμματίζει με τέτοιο τρόπο , ώστε τα προγράμματά του να είναι πιο γρήγορα και σωστά δομημένα , ακόμα και όταν χρησιμοποιεί γλώσσα υψηλότερου επιπέδου από την Assembly . Η Assembly είναι μια γλώσσα ιδανική για αυτούς που ενδιαφέρονται να μάθουν πώς ακριβώς λειτουργούν τα πράγματα.

Υπάρχουν όμως και σοβαρά μειονεκτήματα . Ένα από αυτά είναι η αργή συγγραφή προγραμμάτων . Για παράδειγμα χρειαζόμαστε ένα πρόγραμμα 10 γραμμών για να εκτυπώσουμε ένα αλφαριθμητικό στην οθόνη , πράγμα που σε BASIC γίνεται με μια και μόνο εντολή .

Ένα άλλο μειονέκτημα είναι ότι η γλώσσα Assembly δεν διαθέτει καμία μέριμνα αποφυγής σφαλμάτων και καμία λογική δόμησης , οπότε μπορούν να προκύψουν προγράμματα με σφάλματα και με κώδικα «σπαγγέτι» πολύ εύκολα .
Η μέριμνα σε αυτή την περίπτωση αφήνεται εξολοκλήρου στον προγραμματιστή για καθαρό κώδικα και αποφυγή σφαλμάτων . Αυτός είναι και ο κύριος λόγος που η Assembly δεν συνίσταται για αρχάριους προγραμματιστές .

Παρόλα αυτά τα μειονεκτήματα μπορούν να παρακαμφθούν με διάφορες προσεγγίσεις :
1. Με τη χρήση κανόνων ευταξίας και δόμησης προγραμμάτων , οι οποίοι και θα ακολουθούνται αυστηρά .
2. Με την ενσωμάτωση κώδικα Assembly σε προγράμματα γλωσσών υψηλότερου επιπέδου , ώστε να επιταχύνονται οι αργές διαδικασίες (πχ γραφικά) , ενώ παράλληλα οι συνηθισμένες , μη απαιτητικές διαδικασίες (όπως η εκτύπωση αλφαριθμητικών) να γίνονται εύκολα με τις τυποποιημένες εντολές της κάθε γλώσσας .

3.3 Εντολές PIC.[8]

Η σειρά PIC16, ονομάζεται RISC (Reduced Instruction Set Computer) και χρησιμοποιεί ένα σετ 35 μόνο εντολών.

Επεξήγηση βασικών εννοιών :

Μνημονικό : Το αντίστοιχο δεκαεξαδικής εντολής σε απλή μορφή, κατανοητή από τον άνθρωπο.

Τελεστής : Το λειτουργικό μέρος του μνημονικού [ADDWF f,1 (μνημονικό=πρόσθεσε στον W τον) (τελεστής= f και αποθήκευσε το αποτέλεσμα στον f)]

MSb : Το περισσότερο σημαντικό bit ενός Byte

LSb : Το λιγότερο σημαντικό bit ενός Byte

Flag : Σημαία, κατάσταση των bit του καταχωρητή STATUS

f : Το όνομα του καταχωρητή

d : Η σταθερά που δείχνει τον προορισμό του αποτελέσματος μιας πράξης (αποθήκευση στο w αν d=0 ή αποθήκευση στο f αν d=1)

W : Η συντομογραφία του καταχωρητή γενικής λειτουργίας (Working register)

k : Ένας σταθερός αριθμός που θέλετε να χρησιμοποιήσετε

Επεξήγηση συντομογραφιών :

C : Κρατούμενο, αποτέλεσμα πράξης (Carry)

DC : Ψηφιακό κρατούμενο, αποτέλεσμα πράξης (Digital Carry)

Z : Μηδενικό αποτέλεσμα πράξης (Zero)

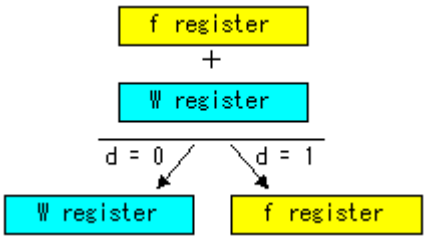
: Ψηφία του καταχωρητή STATUS που δείχνουν Time Out (TO) και Power Down (PD)

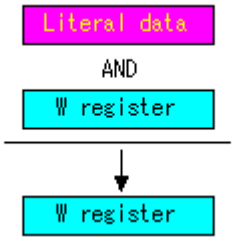
MSb : Most Significant bit: Περισσότερο Σημαντικό Ψηφίο μιας ψηφιολέξης (Byte) Π.Χ : 10101010

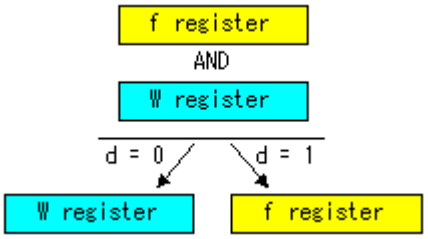
LSb : Least Significant bit: Λιγότερο Σημαντικό Ψηφίο μιας ψηφιολέξης (Byte) Π.Χ : 10101010

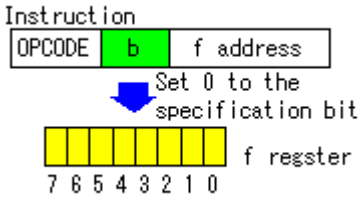
Παρακάτω δίνεται αναλυτικά το σετ των 35 εντολές της σειράς PIC16 XXXX με επεξήγηση και λειτουργεία.

ADDLW	Πρόσθεσε το W και το k	
Σύνταξη	**[Ετικέτα] ADDLW k** (Η ετικέτα μπορεί να παραληφθεί, Το ▲ σημαίνει διάστημα)	
Τελεστές	k : Σταθερός αριθμός (00(00h) έως 255(FFh))	
Λειτουργία	Προσθέτει τον σταθερό αριθμό k με το περιεχόμενο του καταχωρητή W. L 01111010 7Ah(122) + W 00110100 34h(52) 10101110 AEh(174) (Π.Χ)	Literal data + W register ↓ W register
Σημαίες	Όταν το αποτέλεσμα (byte) ξεπεράσει τον αριθμό 255, υπερχειλίζει και βάζει **την τιμή 1** στο ψηφίο **C**. Όταν τα τέσσερα λιγότερο σημαντικά ψηφία υπερχειλίσουν βάζει **την τιμή 1** στο ψηφίο **DC**. Όταν το αποτέλεσμα της πράξης είναι μηδέν, βάζει **την τιμή 1** στο ψηφίο **Z**. Σε κάθε άλλη περίπτωση τα, **C, DC και Z** έχουν την τιμή **0**.	
Κύκλοι μηχανής	1 Κύκλος	

ADDWF	Πρόσθεσε το W και το f
Σύνταξη	**[Ετικέτα] ▲ ADDWF ▲ f, d** (Η ετικέτα μπορεί να παραληφθεί, Το ▲ σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)
Λειτουργία	Προσθέτει τα περιεχόμενα των καταχωρητών f και W. d = 0 : Αποθήκευση στο W d = 1 : Αποθήκευση στο f f 01101000 68h(104) + W 00110100 34h(52) 10011100 9Ch(156) (Π.Χ) 
Σημαίες	Όταν το αποτέλεσμα (byte) ξεπεράσει τον αριθμό 255, υπερχειλίζει και βάζει **την τιμή 1** στο ψηφίο **C**. Όταν τα τέσσερα λιγότερο σημαντικά ψηφία υπερχειλίσουν βάζει **την τιμή 1** στο ψηφίο **DC**. Όταν το αποτέλεσμα της πράξης είναι μηδέν, βάζει **την τιμή 1** στο ψηφίο **Z**. Σε κάθε άλλη περίπτωση τα, **C, DC και Z** έχουν την τιμή **0**.
Κύκλοι μηχανής	1 Κύκλος

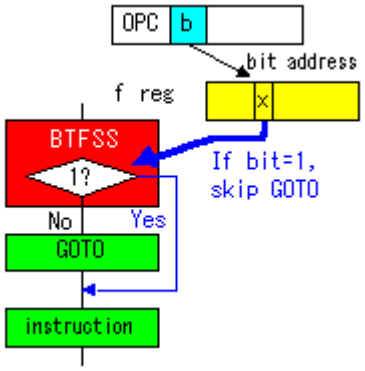
ANDLW	Κάνε την λογική πράξη AND μεταξύ W και k
Σύνταξη	**[Ετικέτα] ▲ ADDLW ▲ k** (Η ετικέτα μπορεί να παραληφθεί, Το ▲ σημαίνει διάστημα)
Τελεστές	k : Σταθερός αριθμός (00(00h) έως 255(FFh))
Λειτουργία	Κάνει την λογική πράξη AND μεταξύ του καταχωρητή W και του σταθερού αριθμού k. L 01101000 AND W 00110100 00100000 (Π.Χ) 
Σημαίες	Όταν το αποτέλεσμα είναι **0**, βάζει την τιμή **1** στο ψηφίο **Z**. Όταν το αποτέλεσμα **δεν** είναι **0**, βάζει την τιμή **0** στο ψηφίο **Z**.
Κύκλοι μηχανής	1 Κύκλος

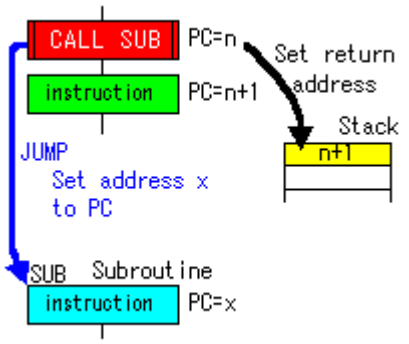
ANDWF	Κάνε την λογική πράξη AND μεταξύ W και f	
Σύνταξη	**[Ετικέτα] ▲ ANDWF ▲ f, d** (Η ετικέτα μπορεί να παραληφθεί, Το ▲ σημαίνει διάστημα)	
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)	
Λειτουργία	Κάνει την λογική πράξη AND μεταξύ των καταχωρητών W και f. d = 0 : Αποθήκευση στο W d = 1 : Αποθήκευση στο f f 01101000 AND W 00110100 00100000 (Π.Χ)	
Σημαίες	Όταν το αποτέλεσμα είναι **0**, βάζει **την τιμή 1** στο ψηφίο **Z**. Όταν το αποτέλεσμα **δεν** είναι **0**, βάζει την τιμή **0** στο ψηφίο **Z**.	
Κύκλοι μηχανής	1 Κύκλος	

BCF	Μηδένισε το ψηφίο b στον καταχωρητή f	
Σύνταξη	**[Ετικέτα] ▲ BCF ▲ f, b** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) b : Αριθμός ψηφίου bit (0 έως 7)	
Λειτουργία	Μηδενίζει το ψηφίο b του καταχωρητή f f 00110100 b=5 f 00010100 (Π.Χ)	
Σημαίες	Καμία	
Κύκλοι μηχανής	1 Κύκλος	

BSF	Κάνε λογικό 1 το ψηφίο b του καταχωρητή f
Σύνταξη	**[Ετικέτα] ▲ BSF ▲ f, b** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) b : Αριθμός ψηφίου bit (0 έως 7)
Λειτουργία	Κάνει **1** το ψηφίο b του καταχωρητή f f 00110100 b=3 f 00111100 (Π.Χ)
Σημαίες	Καμία
Κύκλοι μηχανής	1 Κύκλος

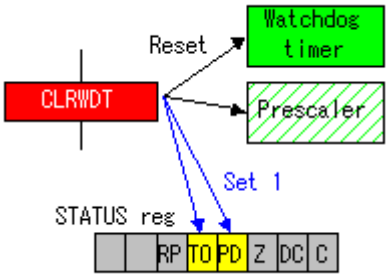
BTFSC	Εξέτασε το ψηφίο b του καταχωρητή f, παρέκαμψε την επόμενη εντολή αν b=0
Σύνταξη	**[Ετικέτα] ▲ BTFSC ▲ f, b** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) b : Αριθμός ψηφίου bit (0 έως 7)
Λειτουργία	Εξετάζει την κατάσταση του ψηφίου b. Αν είναι **1** εκτελεί την επόμενη εντολή, ενώ αν είναι **0** παρακάμπτει την επόμενη εντολή και εκτελεί την μεθεπόμενη.
Σημαίες	Καμία
Κύκλοι μηχανής	1 Κύκλος

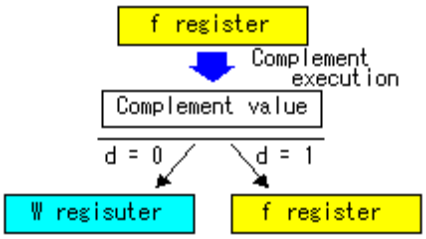
BTFSS	Εξέτασε το ψηφίο b του καταχωρητή f, παρέκαμψε την επόμενη εντολή αν b=1	
Σύνταξη	**[Ετικέτα] ▲ BTFSS ▲ f, b** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) b : Αριθμός ψηφίου bit (0 έως 7)	
Λειτουργία	Εξετάζει την κατάσταση του ψηφίου b. Αν είναι **0** εκτελεί την επόμενη εντολή, ενώ αν είναι **1** παρακάμπτει την επόμενη εντολή και εκτελεί την μεθεπόμενη.	
Σημαίες	Καμία	
Κύκλοι μηχανής	1 Κύκλος	

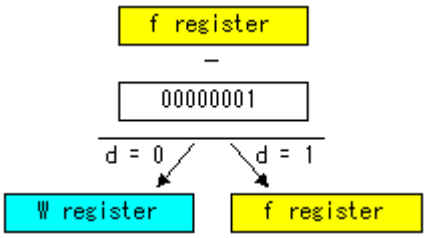
CALL	Κάλεσε την υπορουτίνα k	
Σύνταξη	**[Ετικέτα] ▲ CALL ▲ k** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	k : Σταθερός αριθμός (000(000h) to 2047(7FFh))	
Λειτουργία	Καλεί την υπορουτίνα που υποδεικνύει ο k. Βάζει τα 12 λιγότερο σημαντικά της ψηφία στον καταχωρητή (PC) και τα υπόλοιπα δυο στα ψηφία (bit3) και (bit4) του καταχωρητή PCLATH. Τέλος αποθηκεύει την τιμή του μετρητή προγράμματος + 1 στον καταχωρητή σωρού (stack).	
Σημαίες	Καμία	
Κύκλοι μηχανής	2 Κύκλοι	

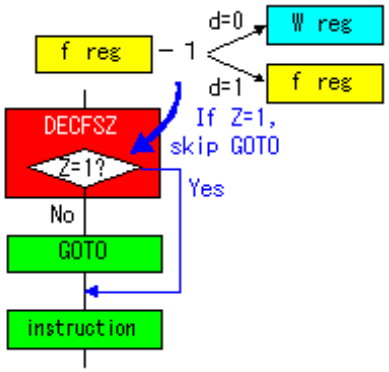
CLRF	Μηδένισε τον καταχωρητή f	
Σύνταξη	**[Ετικέτα] ▲ CLRF ▲ f** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh))	
Λειτουργία	Μηδενίζει όλα τα ψηφία του καταχωρητή f. 00000000 → f 1 → Z	00000000 ↓ f register
Σημαίες	Βάζει την τιμή **1** στο ψηφίο Z του καταχωρητή STATUS.	
Κύκλοι μηχανής	1 Κύκλος	

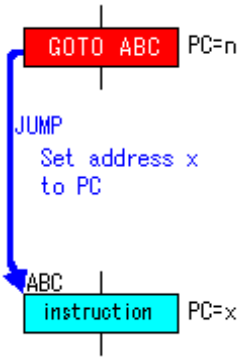
CLRW	Μηδένισε τον καταχωρητή W	
Σύνταξη	**[Ετικέτα] ▲ CLRW** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	Κανένας	
Λειτουργία	Μηδενίζει όλα τα ψηφία του καταχωρητή W. 00000000 → W 1 → Z	00000000 ↓ W register
Σημαίες	Βάζει την τιμή **1** στο ψηφίο Z του καταχωρητή STATUS.	
Κύκλοι μηχανής	1 Κύκλος	

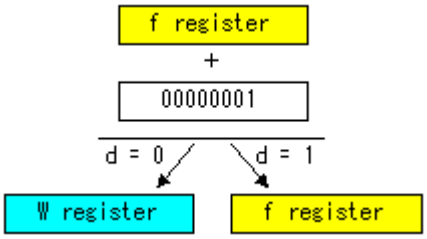
CLRWDT	Μηδένισε τον επιτηρητή Watchdog Timer
Σύνταξη	**[Ετικέτα] ▲CLRWDT** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	Κανένας
Λειτουργία	Μηδενίζει τον επιτηρητή (WDT), και αν χρησιμοποιείται προδιαιρέτης (Prescaler), τον μηδενίζει και αυτόν. 
Σημαίες	Βάζει την τιμή **1** στα ψηφία TO και PD του καταχωρητή STATUS.
Κύκλοι μηχανής	1 cycle

COMF	Φτιάξε το συμπλήρωμα του καταχωρητή f
Σύνταξη	**[Ετικέτα] ▲COMF ▲f, d** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)
Λειτουργία	Δημιουργεί το συμπλήρωμα του καταχωρητή f . (Προσοχή! δεν δημιουργεί το 2'ο Συμπλήρωμα) d = 0 : αποθήκευση στον W d = 1 : αποθήκευση στον f f 01101000 10010111 (Π.Χ) 
Σημαίες	Όταν το αποτέλεσμα είναι **0**, βάζει την τιμή **1** στο ψηφίο **Z**, ενώ όταν **δεν** είναι **0** βάζει την τιμή **0** στο ψηφίο **Z** του καταχωρητή STATUS.
Κύκλοι μηχανής	1 Κύκλος

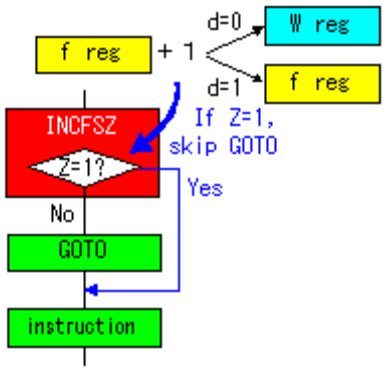
DECF	Μείωσε κατά ένα την τιμή του καταχωρητή f
Σύνταξη	**[Ετικέτα] ▲DECF ▲f, d** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)
Λειτουργία	Μειώνει κατά 1 την τιμή του καταχωρητή f. d = 0 : αποθήκευση στον W d = 1 : αποθήκευση στον f (Π.Χ) f 01100000 60h(96) - 00000001 1h(1) 01011111 5Fh(95) 
Σημαίες	Όταν το αποτέλεσμα είναι **0**, βάζει την τιμή **1** στο ψηφίο **Z**, ενώ όταν **δεν** είναι **0** βάζει την τιμή **0** στο ψηφίο **Z** του καταχωρητή STATUS.
Κύκλοι μηχανής	1 Κύκλος

DECFSZ	Μείωσε κατά ένα την τιμή του καταχωρητή f, παρέκαμψε την επόμενη εντολή αν ο f γίνει 0
Σύνταξη	**[Ετικέτα] ▲DECFSZ ▲f, d** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)
Λειτουργία	Μειώνει κατά 1 την τιμή του καταχωρητή f. d = 0 : αποθήκευση στον W d = 1 : αποθήκευση στον f Στην περίπτωση που ο καταχωρητής f γίνει **0,** παρακάμπτει την επόμενη εντολή και εκτελεί την μεθεπόμενη. 
Σημαίες	Καμία
Κύκλοι μηχανής	1 Κύκλος 2 Κύκλοι, Όταν ο καταχωρητής f γίνει **0**.

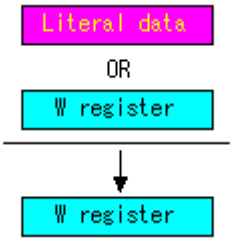
GOTO	Πήγαινε και εκτέλεσε την εντολή που υπάρχει στην διεύθυνση k	
Σύνταξη	**[Ετικέτα] ▲GOTO ▲k** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	k : Σταθερός αριθμός (000(000h) έως 2047(7FFh))	
Λειτουργία	Φορτώνει την διεύθυνση k στον μετρητή προγράμματος (PC) και το πρόγραμμα συνεχίζει να εκτελείται από την νέα διεύθυνση k. Βάζει τα 12 λιγότερο σημαντικά της ψηφία στον καταχωρητή (PC) και τα υπόλοιπα δυο στα ψηφία (bit3) και (bit4) του καταχωρητή PCLATH.	
Σημαίες	Καμία	
Κύκλοι μηχανής	2 Κύκλοι	

INCF	Αύξησε κατά ένα την τιμή του f	
Σύνταξη	**[Ετικέτα] ▲INCF ▲f, d** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)	
Λειτουργία	Αυξάνει κατά 1 την τιμή του καταχωρητή f d = 0 : αποθήκευση στον W d = 1 : αποθήκευση στον f (Π.Χ) f 01100000 60h(96) + 00000001 1h(1) 01100001 61h(97)	
Σημαίες	Όταν το αποτέλεσμα είναι **0**, βάζει την τιμή **1** στο ψηφίο **Z**, ενώ όταν **δεν** είναι **0** βάζει την τιμή **0** στο ψηφίο **Z** του καταχωρητή STATUS.	
Κύκλοι μηχανής	1 Κύκλος	

INCFSZ	Αύξησε κατά ένα την τιμή του καταχωρητή f, παρέκαμψε την επόμενη εντολή αν ο f γίνει 0
Σύνταξη	**[Ετικέτα] ▵INCFSZ ▵f, d** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)
Λειτουργία	Αυξάνει κατά 1 την τιμή του καταχωρητή f. d = 0 : αποθήκευση στον W d = 1 : αποθήκευση στον f Στην περίπτωση που ο καταχωρητής f γίνει **0,** παρακάμπτει την επόμενη εντολή και εκτελεί την μεθεπόμενη.
Σημαίες	Καμία
Κύκλοι μηχανής	1 Κύκλος



IORLW	Κάνε την λογική πράξη IOR ανάμεσα στο k και το W
Σύνταξη	**[Ετικέτα] ▵IORLW ▵k** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	k : Σταθερός αριθμός (00(00h) έως 255(FFh))
Λειτουργία	Κάνει την λογική πράξη IOR μεταξύ του καταχωρητή W και του σταθερού αριθμού k. L 01101000 OR W 00110100 01111100 (Π.Χ)
Σημαίες	Όταν το αποτέλεσμα είναι **0**, βάζει την τιμή **1** στο ψηφίο **Z**, ενώ όταν **δεν** είναι **0** βάζει την τιμή **0** στο ψηφίο **Z** του καταχωρητή STATUS.
Κύκλοι μηχανής	1 Κύκλος

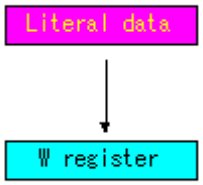


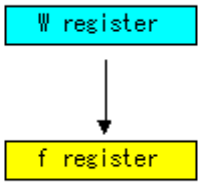
IORWF	Κάνε την λογική πράξη IOR ανάμεσα στο W και το f
Σύνταξη	**[Ετικέτα] ▵IORWF ▵f, d**

	(Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)
Λειτουργία	Κάνει την λογική πράξη IOR μεταξύ του καταχωρητή W και του καταχωρητή f. d = 0 : αποθήκευση στον W d = 1 : αποθήκευση στον f f 01101000 OR W 00110100 01111100 (Π.Χ)
Σημαίες	Όταν το αποτέλεσμα είναι **0**, βάζει την τιμή **1** στο ψηφίο **Z**, ενώ όταν **δεν** είναι **0** βάζει την τιμή **0** στο ψηφίο **Z** του καταχωρητή STATUS.
Κύκλοι μηχανής	1 Κύκλος

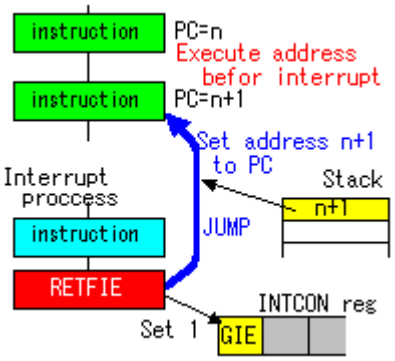
MOVF	Μετέφερε το περιεχόμενο του καταχωρητή f
Σύνταξη	**[Ετικέτα] MOVF f, d** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)
Λειτουργία	Μεταφέρει το περιεχόμενο του καταχωρητή f. d = 0 : αποθήκευση στον W d = 1 : αποθήκευση στον f
Σημαίες	Όταν το αποτέλεσμα είναι **0**, βάζει την τιμή **1** στο ψηφίο **Z**, ενώ όταν **δεν** είναι **0** βάζει την τιμή **0** στο ψηφίο **Z** του καταχωρητή STATUS.
Κύκλοι μηχανής	1 Κύκλος

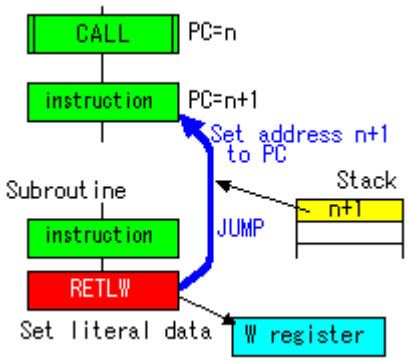
MOVLW	Μετέφερε το περιεχόμενο του k στο W

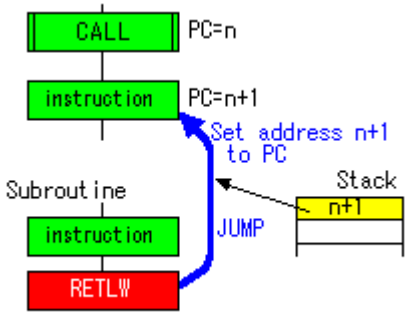
Σύνταξη	**[Ετικέτα] ▲MOVLW ▲k** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	k : Σταθερός αριθμός (00(00h) έως 255(FFh))	
Λειτουργία	Μεταφέρει το σταθερό αριθμό k στον W	
Σημαίες	Καμία	
Κύκλοι μηχανής	1 Κύκλος	

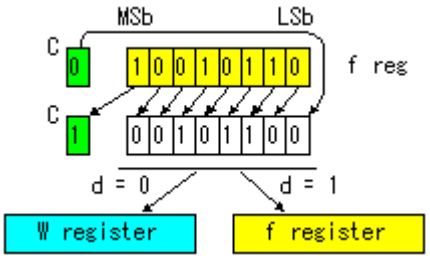
MOVWF	Μετέφερε το περιεχόμενο του W στο f	
Σύνταξη	**[Ετικέτα] ▲MOVWF ▲f** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh))	
Λειτουργία	Μεταφέρει το περιεχόμενο του W στον καταχωρητή f	
Σημαίες	Καμία	
Κύκλοι μηχανής	1 Κύκλος	

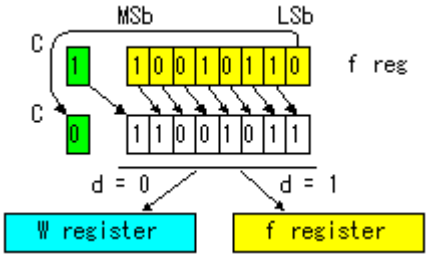
NOP	Εντολή δίχως λειτουργία	
Σύνταξη	**[Ετικέτα] ▲NOP** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	Κανένας	
Λειτουργία	Δημιουργεί απλή χρονική καθυστέρηση ενός κύκλου μηχανής	
Σημαίες	Καμία	
Κύκλοι μηχανής	1 Κύκλος	

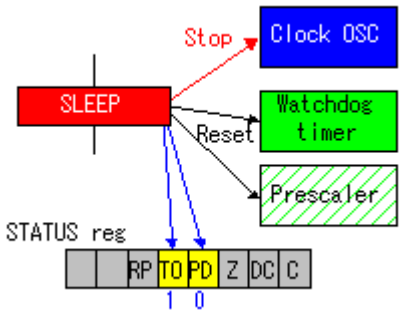
RETFIE	Επέστρεψε στην διεύθυνση που ήσουν πριν συμβεί η διακοπή (interrupt)	
Σύνταξη	**[Ετικέτα] ▲RETFIE** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	Κανένας	
Λειτουργία	Επιστρέφει από την υπορουτίνα του Interrupt. Βάζει στον μετρητή προγράμματος την τελευταία διεύθυνση που αποθηκεύθηκε στον καταχωρητή σωρού (stack), και κάνει **1** το ψηφίο **GIE** του καταχωρητή INTCON	
Σημαίες	Καμία	
Κύκλοι μηχανής	2 Κύκλοι	

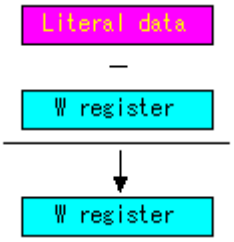
RETLW	Επέστρεψε από υπορουτίνα και φόρτωσε τον σταθερό αριθμό k στο W	
Σύνταξη	**[Ετικέτα] ▲ RETLW ▲ k** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	k : Σταθερός αριθμός (00(00h) έως 255(FFh))	
Λειτουργία	Επιστρέφει από υπορουτίνα. Φορτώνει το σταθερό αριθμό k στον W, και βάζει στον μετρητή προγράμματος την τελευταία διεύθυνση που αποθηκεύθηκε στον καταχωρητή σωρού (stack).	
Σημαίες	Καμία	
Κύκλοι μηχανής	2 κύκλοι	

RETURN	Επέστρεψε από υπορουτίνα	
Σύνταξη	**[Ετικέτα] ▲ RETURN** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	Κανένας	
Λειτουργία	Επιστρέφει από υπορουτίνα, και βάζει στον μετρητή προγράμματος την τελευταία διεύθυνση που αποθηκεύθηκε στον καταχωρητή σωρού (stack).	
Σημαίες	Καμία	
Κύκλοι μηχανής	2 κύκλοι	

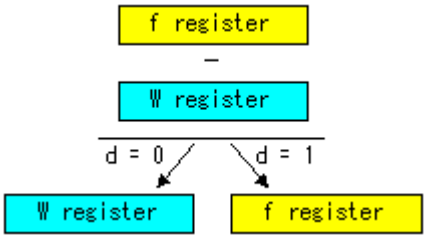
RLF	Μετέφερε προς τα αριστερά το περιεχόμενο του καταχωρητή f μέσο του ψηφίου Carry
Σύνταξη	**[Ετικέτα] ▲ RLF ▲ f, d** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)
Λειτουργία	Μεταφέρει τα οκτώ ψηφία που περιέχει ο καταχωρητής f συμπεριλαμβανομένου του κρατούμενου ψηφίου (Carry), μια θέση αριστερά. d = 0 : αποθήκευση στον W d = 1 : αποθήκευση στον f 
Σημαίες	Τοποθετεί το περισσότερο σημαντικό ψηφίο (MSB) του καταχωρητή f, στο κρατούμενο ψηφίο (Carry).
Κύκλοι μηχανής	1 Κύκλος

RRF	Μετέφερε προς τα δεξιά το περιεχόμενο του καταχωρητή f μέσο του ψηφίου Carry
Σύνταξη	**[Ετικέτα] ▲ RRF ▲ f, d** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)
Λειτουργία	Μεταφέρει τα οκτώ ψηφία που περιέχει ο καταχωρητής f συμπεριλαμβανομένου του κρατούμενου ψηφίου (Carry), μια θέση δεξιά. d = 0 : αποθήκευση στον W d = 1 : αποθήκευση στον f 
Σημαίες	Τοποθετεί το λιγότερο σημαντικό ψηφίο (LSB) του καταχωρητή f, στο κρατούμενο ψηφίο (Carry).
Κύκλοι μηχανής	1 Κύκλος

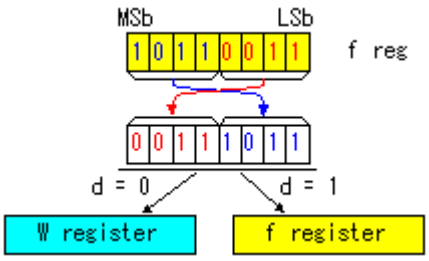
SLEEP	Ενεργοποίησε την λειτουργία χαμηλής κατανάλωσης (Sleep - κατανάλωση 2μA)	
Σύνταξη	**[Ετικέτα] ▲ SLEEP** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	Κανένας	
Λειτουργία	Σταματά η λειτουργία του κυκλώματος χρονισμού και ο μικροεπεξεργαστής μπαίνει σε κατάσταση αναμονής. Η εντολή μηδενίζει τον επιτηρητή προγράμματος (watchdog timer), και αν χρησιμοποιείται προδιαιρέτης (prescaler), τον μηδενίζει και αυτόν.	
Σημαίες	Βάζει την τιμή **1** στο ψηφίο TO και **0** στο ψηφίο PD του καταχωρητή STATUS.	
Κύκλοι μηχανής	1 Κύκλος	

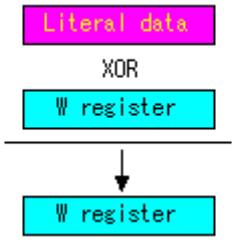
SUBLW	Αφαίρεσε το περιεχόμενο του W από το σταθερό αριθμό k	
Σύνταξη	**[Ετικέτα] ▲ SUBLW ▲ k** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	k : Σταθερός αριθμός (00(00h) έως 255(FFh))	
Λειτουργία	Αφαιρεί το περιεχόμενο του W από το σταθερό αριθμό k. Στην πραγματικότητα δημιουργεί το δεύτερο συμπλήρωμα του W και προσθέτει σε αυτό τον σταθερό αριθμό k. L 01111010 7Ah(122) - W 00110100 34h(52) 01000110 46h(70) (ΠΧ)	
Σημαίες	**Ψηφίο Carry=1, και Zero=0 (Θετικό αποτέλεσμα)** **Ψηφίο Carry=1, και Zero=1 (Μηδενικό αποτέλεσμα)** **Ψηφίο Carry=0, και Zero=0 (Αρνητικό αποτέλεσμα)**	
Κύκλοι μηχανής	1 Κύκλος	

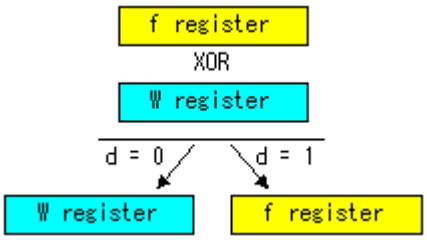
SUBWF	Αφαίρεσε το W από τον καταχωρητή f
Σύνταξη	**[Ετικέτα] ▲ SUBWF ▲ f, d** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)
Λειτουργία	Αφαιρεί το περιεχόμενο του W από το περιεχόμενο του καταχωρητή f. d = 0 : αποθήκευση στον W d = 1 : αποθήκευση στον f Στην πραγματικότητα δημιουργεί το δεύτερο συμπλήρωμα του W και προσθέτει σε αυτό το περιεχόμενο του καταχωρητή f. f 01100001 61h(97) - W 00100010 22h(34) 00111111 3Fh(63) (Π.Χ)
Σημαίες	**Ψηφίο Carry=1, και Zero=0 (Θετικό αποτέλεσμα)** **Ψηφίο Carry=1, και Zero=1 (Μηδενικό αποτέλεσμα)** **Ψηφίο Carry=0, και Zero=0 (Αρνητικό αποτέλεσμα)**
Κύκλοι μηχανής	1 Κύκλος



SWAPF	Αντιμετάθεσε τα δύο μισά της ψηφιολέξης (Byte) του καταχωρητή f
Σύνταξη	**[Ετικέτα] ▲ SWAPF ▲ f, d** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)
Λειτουργία	Αντιμεταθέτει τα δύο μισά της ψηφιολέξης (Byte) του καταχωρητή f
Σημαίες	Καμία
Κύκλοι μηχανής	1 Κύκλος



XORLW	Κάνε την λογική πράξη XOR ανάμεσα στο σταθερό αριθμό k και το W	
Σύνταξη	**[Ετικέτα] ▲ XORLW ▲ k** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	k : Σταθερός αριθμός (00(00h) έως 255(FFh))	
Λειτουργία	Κάνει την λογική πράξη XOR μεταξύ του καταχωρητή W και του σταθερού αριθμού k. L 01111010 XOR W 00110100 01001110 (Π.Χ)	
Σημαίες	Όταν το αποτέλεσμα είναι **0**, βάζει την τιμή **1** στο ψηφίο **Z**, ενώ όταν **δεν** είναι **0** βάζει την τιμή **0** στο ψηφίο **Z** του καταχωρητή STATUS.	
Κύκλοι μηχανής	1 Κύκλος	

XORWF	Κάνε την λογική πράξη XOR ανάμεσα στο W και τον καταχωρητή f	
Σύνταξη	**[Ετικέτα] ▲ XORWF ▲ f, d** (Η ετικέτα μπορεί να παραληφθεί, Το σημαίνει διάστημα)	
Τελεστές	f : Διεύθυνση καταχωρητή (00(00h) έως 127(7Fh)) d : Προορισμός αποτελέσματος (0 ή 1)	
Λειτουργία	Κάνει την λογική πράξη XOR μεταξύ του καταχωρητή W και του καταχωρητή f. d = 0 : αποθήκευση στον W d = 1 : αποθήκευση στον f f 01101000 XOR W 00110100 01011100 (Π.Χ)	
Σημαίες	Όταν το αποτέλεσμα είναι **0**, βάζει την τιμή **1** στο ψηφίο **Z**, ενώ όταν **δεν** είναι **0** βάζει την τιμή **0** στο ψηφίο **Z** του καταχωρητή STATUS.	
Κύκλοι μηχανής	1 Κύκλος	

Αυτές ήταν αναλυτικά οι 35 εντολές του PIC16 XXXX.

Βιβλιογραφία κεφαλαίου

[1]Μικροελεγκτές PIC" Σταμάτης Αλατσαθιανός, Β.Γκιούρδας Εκδοτική
[2]Microcontroller Programming" Julio Sanchez, Maria P. Canton
[3]Mano Morris M., Ψηφιακή Σχεδίαση, (Κεφ. 7), Εκδόσεις Παπασωτηρίου, 1992
[4]Wakerly J. F., Digital Design: Principles and Practices, 2nd Edition, (Chap. 11),
[5]https://el.wikipedia.org/wiki/%CE%93%CE%BB%CF%8E%CF%83%CF%83%CE%B1_%CF%80%CF%81%CE%BF%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1%CF%84%CE%B9%CF%83%CE%BC%CE%BF%CF%8D
[6]http://nuclear.dnsalias.com/tmp/samurai_old_articles/intro_asm.html
[7]http://ww1.microchip.com/downloads/en/DeviceDoc/41262c.pdf
[8]http://www.electronics-lab.com/pic-in-greek/

ΚΕΦΑΛΑΙΟ 4

MPLAB IDE

4.1 Το MPLAB IDE.[1,2,3]

Το MPLAB IDE είναι ένα πρόγραμμα λογισμικού που τρέχει σε έναν υπολογιστή για την ανάπτυξη εφαρμογών για Microchip μικροελεγκτές. Λέγεται ένα ολοκληρωμένο περιβάλλον ανάπτυξης ή IDE,διότι παρέχει ένα ενιαίο, ολοκληρωμένο περιβάλλον για την ανάπτυξη κώδικα για τους ενσωματωμένους μικροελεγκτές.Είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης για επεξεργαστές, με σκοπό το manager και desktop σχεδιασμό για την ανάπτυξη εφαρμογών ενσωματωμένων σχεδίων, χρησιμοποιώντας Microchip PICmicro και dsPIC μικροελεγκτές. Το πλήρες σύνολο δυνατοτήτων του MPLAB IDE είναι αδύνατον να καλυφθεί σε ένα μόνο κεφάλαιο. Οπότε αυτό το κεφάλαιο μας βοηθά στις λεπτομέρειες για την εγκατάσταση και την απεγκατάσταση του MPLAB IDE,όπως επίσης πώς να σχεδιάσουμε έργα, να επεξεργαστούμε κώδικα και δοκιμή, βασικές έννοιες του Project Manager, έκδόσης και Debugger.Θα ακολουθηθεί μία απλή βήμα-προς-βήμα διαδικασία που δημιουργεί ένα project και εξηγεί τις στοιχειώδη debugging δυνατότητες του MPLAB IDE. Καμία προηγούμενη γνώση υποτίθεται, και Ολοκληρωμένες τεχνικές λεπτομέρειες του MPLAB IDE και οι παραμέτρους του παραλείπονται, προκειμένου να παρουσιαστεί το βασικό πλαίσιο για τη χρήση MPLAB IDE. Αυτά τα βασικό πλαίσιο θα πρέπει να καλύπτει ανάγκες χρήσης όπως:

• MPLAB IDE Χαρακτηριστικά και Εγκατάσταση.
• Επισκόπηση Tutorial.
• Επιλογή της συσκευής.
• Δημιουργία ενός έργου.
• Ρύθμιση Γλωσσικών Εργαλείων.
• Ονομασία του Έργου.
• Δημιουργία κώδικα.
• Κατασκευή του έργου.
• Κωδικός Δοκιμής με τον προσομοιωτή.

4.2 MPLAB IDE Χαρακτηριστικά & Εγκατάσταση.

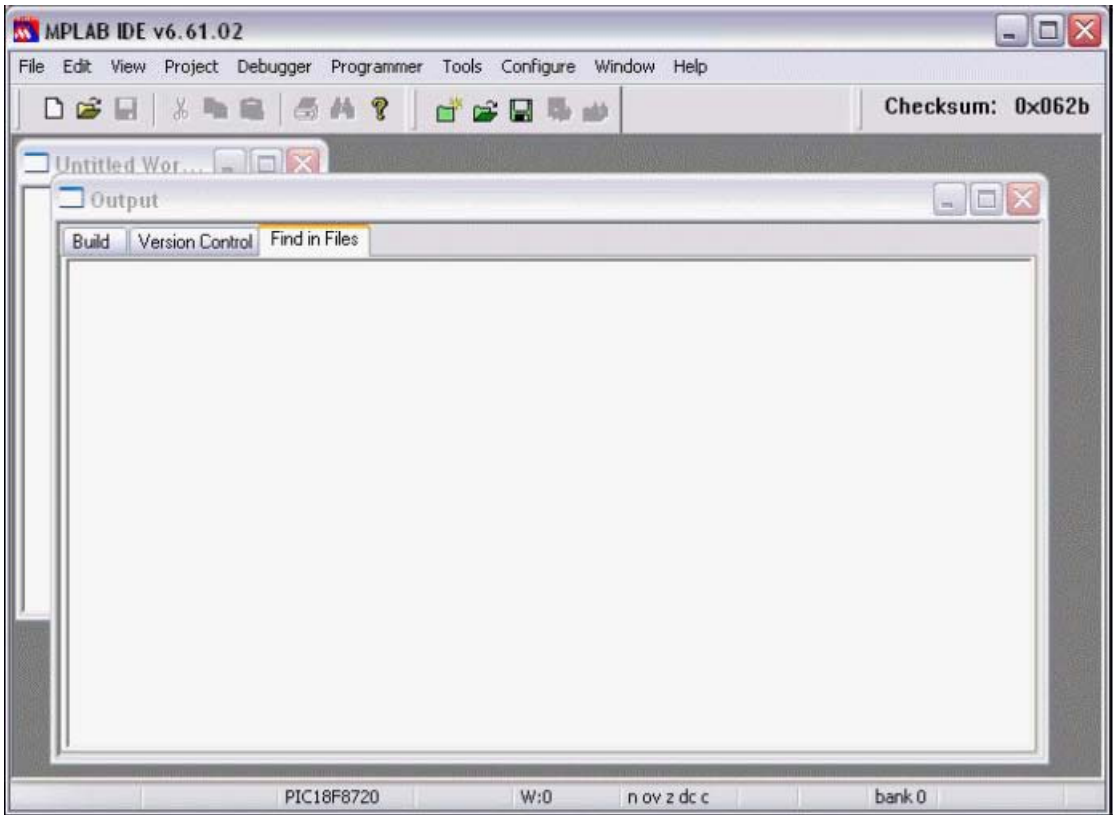
Το MPLAB IDE είναι ένα λειτουργικό σύστημα Windows με βάση το Ολοκληρωμένο Περιβάλλον Ανάπτυξης για τις PICmicro οικογένειες MCU και οι dsPIC Digital Signal Controllers. Το MPLAB IDEπαρέχει τη δυνατότητα:
• Δημιουργίας και επεξεργασίας του πηγαίου κώδικα χρησιμοποιώντας το ενσωματωμένο επεξεργαστή.
• Τοποθετήση, συγκέντρωση και σύνδεση του πηγαίου κώδικα.
• Διόρθωση της λογικής εκτελέσιμου προφράμματος παρακολουθώντας τη ροή του προγράμματος με το ενσωματωμένο προσομοιωτή ή σε πραγματικό χρόνο με το κύκλωμα emulators ή στο κύκλωμα εντοπισμού σφαλμάτων.
• Βεβαίωσης των μετρήσεις χρονισμού με τον προσομοιωτή ή εξομοιωτή.
• Προβολή μεταβλητών στο ρολόι των Windows.
• firmware Προγράμματος σε συσκευές με προγραμματιστές συσκευή.

4.2.1 Εγκατάσταση / Κατάργηση MPLAB IDE.

Για να εγκαταστήσουμε το MPLAB IDE στο σύστημά μας ακολουθούμε τα εξής βήματα:
• Εάν η εγκατάσταση γίνεται από ένα CD-ROM, τοποθετούμε το δίσκο στη μονάδα CD. Ακολουθούμε τις οδηγίες στην οθόνη μενού για να εγκαταστήσουμε το MPLAB IDE. Εάν το μενού δεν εμφανίζεται στην οθόνη, χρησιμοποιηούμε τα Windows για να βρει και να εκτελέσει το CD-ROM μενού, menu.exe.
• Εάν η λήψη MPLAB IDE γίνεται από την ιστοσελίδα Microchip web (www.microchip.com), εντοπίζουμε το αρχείο λήψης (. zip), επιλέγουμε το αρχείο για αποθήκευση στον υπολογιστή. Αποσυμπιέζουμε το αρχείο .zip και να εκτελούμε το αρχείο που προκύπτει για την εγκατάσταση.
Για να καταργήσουμε την εγκατάσταση του MPLAB IDE:
• Επιλέγουμε Έναρξη> Ρυθμίσεις> Πίνακας Ελέγχου για να ανοίξουμε τον Πίνακα Ελέγχου.
• Κάνουμε διπλό κλικ στο Add / Remove Programs. Βρείσκουμε το MPLAB IDE στη λίστα και κάντε κλικ σε αυτό.
• Κάνουμε κλικ στο κουμπί Αλλαγή / Κατάργηση για να καταργήσουμε το πρόγραμμα από το σύστημά σας.

4.2.2 Εκτέλεση MPLAB IDE.

Για να ξεκινήσουμε το MPLAB IDE, κάνουμε διπλό κλικ στο εικονίδιο εκκίνηση στην επιφάνεια εργασίας μετά την εγκατάσταση
ή επιλέξτε Έναρξη> Προγράμματα> Microchip MPLAB IDE vx.x> MPLAB IDE vx.x.Αμέσως μετα ένα παράθυρο θα
εμφανιστεί με το MPLAB IDE logo ακολουθούμενο από την MPLAB IDE επιφάνεια εργασίας (Εικόνα 4-1).



Εικόνα 4.1: MPLAB IDE επιφάνεια εργασίας.

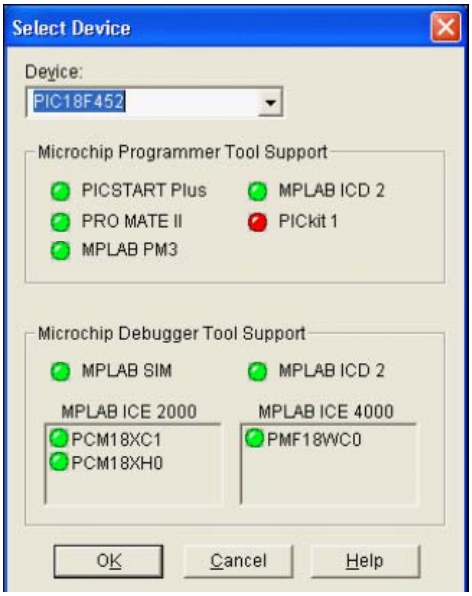
4.3 Επισκόπηση Καθοδήγησης και Οδηγιών Προγράμματος.

Για να δημιουργήσουμε κώδικα που είναι εκτελέσιμο από το PICmicro MCU, αρχεία προέλευσης πρέπει να τεθούν σε ένα έργο. Ο κωδικός μπορεί στη συνέχεια να κατασκευαστεί σε εκτελέσιμο κώδικα χρησιμοποιώντας επιλεγμένα γλωσσικά εργαλεία (assemblers, compilers, συνδέσμους κ.λπ.). Στο MPLAB IDE, ο διαχειριστής του έργου ελέγχει αυτή τη διαδικασία.

Όλα τα έργα θα έχουν αυτά τα βασικά βήματα:
• Επιλογή συσκευής. Οι δυνατότητες του MPLAB IDE ποικίλλουν ανάλογα με το ποια συσκευή έχει επιλεγεί. Η επιλογή συσκευής θα πρέπει να έχει ολοκληρωθεί πριν από την έναρξη ενός έργου.
• Δημιουργία Project. Ο MPLAB Εργοδηγός θα πρέπει να χρησιμοποιηθεί για τη δημιουργία ενός έργου.
• Επιλογή Εργαλείων γλωσσών. Στον Οδηγό έργου τα γλωσσικά εργαλεία που θα επιλεγούν.
• Τοποθέτηση του αρχείου στο Project. Δύο αρχεία θα μπουν στο σχέδιο, ένα πρότυπο αρχείο και ένα συνδετικό σενάριο. Και τα δύο αυτά αρχεία θα υπάρχουν σε υπο-φακέλους μέσα στο φάκελο MPLAB IDE. Χρησιμοποιώντας αυτά τα δύο αρχεία θα είναι εύκολο να ξεκινήσουμε.
• Δημιουργία κώδικα. Κάποιοι κωδικός θα προστεθεί στο πρότυπο αρχείο για να σταλεί σε προσαυξητική αξία ένα προς ένα στις θύρες εισόδου εξόδου (I/O).
• Κατασκευή του έργου. Το έργο θα κατασκευαστεί – δίνοντας εντολή στα αρχεία πηγαίου κώδικα για να συναρμολογηθούν και να συνδεθούν με τον κώδικα μηχανής που μπορεί να τρέξει με την επιλεγμένη PICmicro MCU.
• Κωδικός δοκιμών με Simulator. Τέλος, ο κώδικας θα πρέπει να ελέγχεται με τον προσομοιωτή.

4.4 Επιλογή της Συσκευής.

Για να δούμε τις επιλογές του μενού σε αυτό το έγγραφο, το στοιχείο του μενού από την κορυφή γραμμή στο MPLAB
IDE θα εμφανιστεί με το όνομα του μενού όπως αυτό MenuName> MENUITEM. Για να επιλέξουμε το καταχώρηση της συσκευής στο μενού Διαμόρφωση, θα γράφεται συνήθως ως Διαμόρφωση> Επιλογή Συσκευής. Επιλέξτε Διαμόρφωση> Επιλογή συσκευής.



Εικόνα 4-2: Επιλογή της Συσκευής

Στο παράθυρο διαλόγου Device, επιλέγουμε το PIC16F84A από τη λίστα, αν δεν είναι ήδη επιλεγμένο. Τα «φώτα» αναφέρουν ποια στοιχεία MPLAB IDE υποστηρίζει αυτή τη συσκευή.
• Η πράσινη λυχνία υποδεικνύει την πλήρη υποστήριξη.
• Ένα κίτρινο φως δείχνει την ελάχιστη υποστήριξη για ενα επερχόμενο μέρος που δεν μπορεί να υποστηρίζονται πλήρως σε αυτή την έκδοση από τη συγκεκριμένη συνιστώσα MPLAB IDE. Τα εξαρτήματα με κίτρινο φως αντί για ένα πράσινο φως που συχνά προορίζονται για την πρόωρη υιοθέτηση των νέων εξαρτημάτων που χρειάζονται γρήγορη υποστήριξη και να καταλάβουν ότι ορισμένες λειτουργίες ενδέχεται να μην είναι διαθέσιμες.
• Το κόκκινο φως δείχνει ότι δεν υπάρχει υποστήριξη για τη συγκεκριμένη συσκευή. Η στήριξη μπορεί να συμβεί στο μέλλον ή ακατάλληλο για το εργαλείο, π.χ όπως dsPIC συσκευές που μπορεί να μην υποστηρίζονται σε MPLAB ICE 2000.

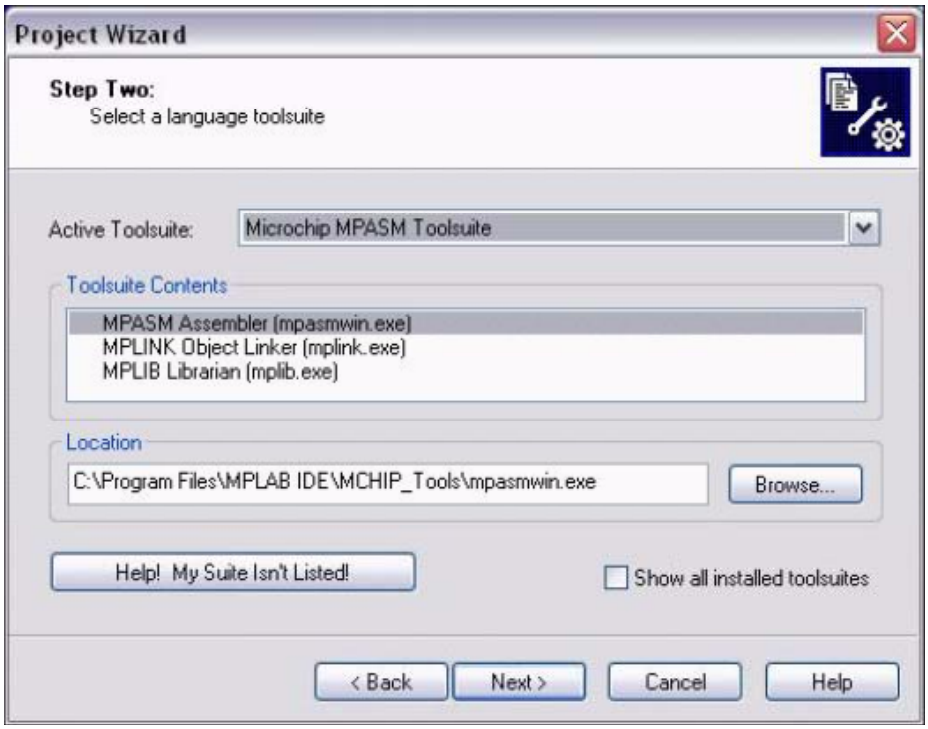
4.5 Δημιουργία του Project.

Το επόμενο βήμα είναι να δημιουργήσουμε ένα έργο χρησιμοποιώντας τον Οδηγό έργου. Ένα έργο είναι ο τρόπος με τον οποίο τα αρχεία οργανώνονται να καταρτιστούν και να συναρμολογηθούν. Θα χρησιμοποιήσουμε ένα αρχείο συναρμολόγησης για το έργο αυτό και ένα συνδετηκό σενάριο. Επιλέγουμε Έργοο> Οδηγός του έργου. Από το Welcome διαλόγου, κάνουμε κλικ στο Επόμενο> για να προχωρήσουμε. Το επόμενο παράθυρο διαλόγου (Βήμα) μας επιτρέπει να επιλέξουμε τη συσκευή που έχουμε ήδη κάνει.Βεβαιωθείτε ότι λέει PIC16F84A. Αν δεν το κάνει, το επιλέγουμε από την drop down μενού. Κάνουμε κλικ στο Επόμενο>.

Εικόνα 4.3: Project Wizard – Επιλογή Συσκευής.

4.6 Εγκατάσταση στα Γλωσσικά Εργαλεία.

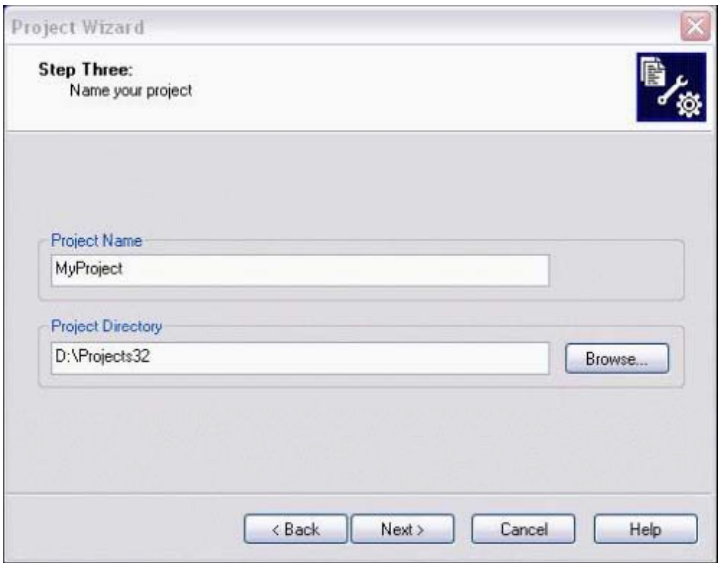
Το δεύτερο βήμα του Οδηγού έργου ορίζει τα γλωσσικά εργαλεία που χρησιμοποιούνται με αυτό του έργου. Επιλέγουμε "Microchip MPASM Toolsuite" στο Active πλαίσιο λίστας Toolsuite. Τότε το "MPASM" και "MPLINK" θα πρέπει να είναι ορατά στο Toolsuite πλαίσιο περιεχομένων. Κάνουμε κλικ σε κάθε μία για να δούμε τη θέση του. Αν το MPLAB IDE έχει εγκατασταθεί στον κατάλογο προεπιλογή, η MPASM εκτελέσιμο συναρμολόγησης θα είναι:
C: \ Program Files \ MPLAB IDE \ MCHIP_Tools \ mpasmwin.exe.
Ο εκτελέσιμος σύνδεσμος MPLINK θα είναι:
C: \ Program Files \ MPLAB IDE \ MCHIP_Tools \ mplink.exe
και η MPLIB εκτελέσιμη βιβλιοθηκονόμος θα είναι:
C: \ Program Files \ MPLAB IDE \ MCHIP_Tools \ mplib.exe
Εάν αυτά δεν εμφανίζονται σωστά, χρησιμοποιούμε το κουμπί Αναζήτηση για να τα ρυθμίσουμε με τα κατάλληλα αρχεία στο MPLAB IDE και στους υποφακέλους. Όταν τελειώνουμε, κάνουμε κλικ στο «Επόμενο>».



Εικόνα 4.4: Project Wizard - Επιλογή Εργαλεία Γλώσσας.

4.7 Ονομασία του Project.

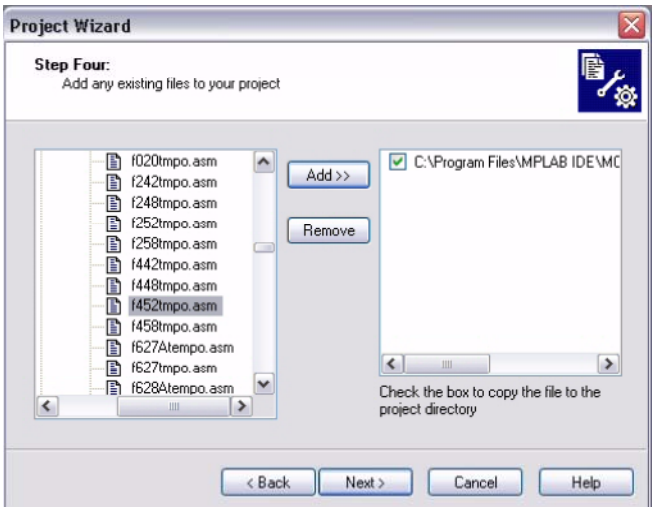
Το τρίτο Βήμα του οδηγού μας επιτρέπει να ονομάσουμε το έργο και να το βάλουμε σε ένα φάκελο. Αυτός ο φάκελος είναι δείγμα έργου που θα ονομάζεται π.χ. MyProject. Χρησιμοποιώντας το κουμπί Αναζήτηση, τοποθετούμε το έργο σε ένα φάκελο που ονομάζεται Projects32. Κάνουμε κλικ στο Επόμενο>.



Εικόνα 4.5: Project Wizard-Ονομασία Project.

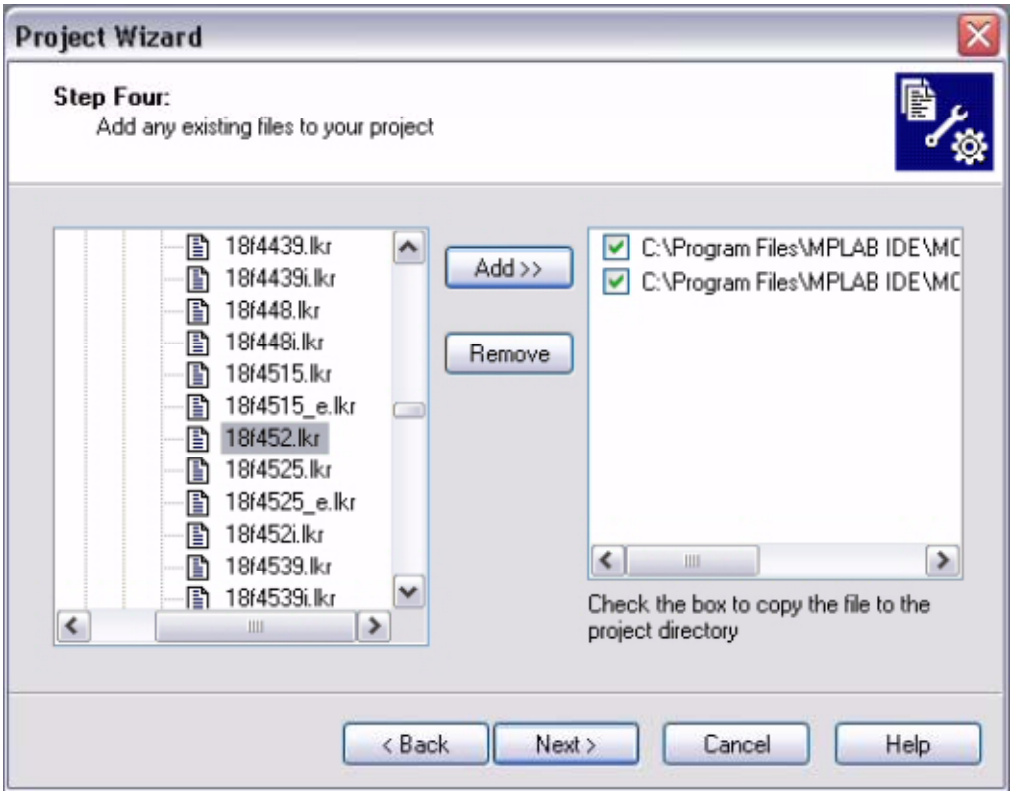
4.8 Προσθέτοντας τα αρχεία για το Project.

Το τέταρτο βήμα του έργου Οδηγού επιτρέπει την επιλογή αρχείων για το έργο. Ένα αρχείο προέλευσης δεν έχει ακόμη επιλεγεί, γι'αυτό θα χρησιμοποιήσουμε ένα αρχείο προτύπου MPLAB IDE. Τα αρχεία προτύπων είναι απλά αρχεία που μπορεί να χρησιμοποιηθούν για να ξεκινήσει ένα έργο. Έχοντας τα απαραίτητα τμήματα για οποιοδήποτε αρχείο προέλευσης και περιέχοντας πληροφορίες που θα μας βοηθήσουν να γράψουμε και να οργανώσουμε τον κωδικό μας. Αυτά τα αρχεία βρίσκονται στο φάκελο MPLAB IDE, το οποίο είναι εξ ορισμού στο φάκελο Program Files φάκελο στον υπολογιστή. Υπάρχει ένα αρχείο πρότυπο για κάθε Microchip PICmicro και dsPIC συσκευή. Επιλέγουμε το αρχείο με το όνομα f452tmpo.asm. Αν το MPLAB IDE είναι εγκατεστημένο στην προεπιλεγμένη θέση, η πλήρης διαδρομή του αρχείου θα είναι:
C: \ Program Files \ MPLAB IDE \ MCHIP_Tools \ ΥΠΟΔΕΙΓΜΑ \ Object \ f452tmpo.asm



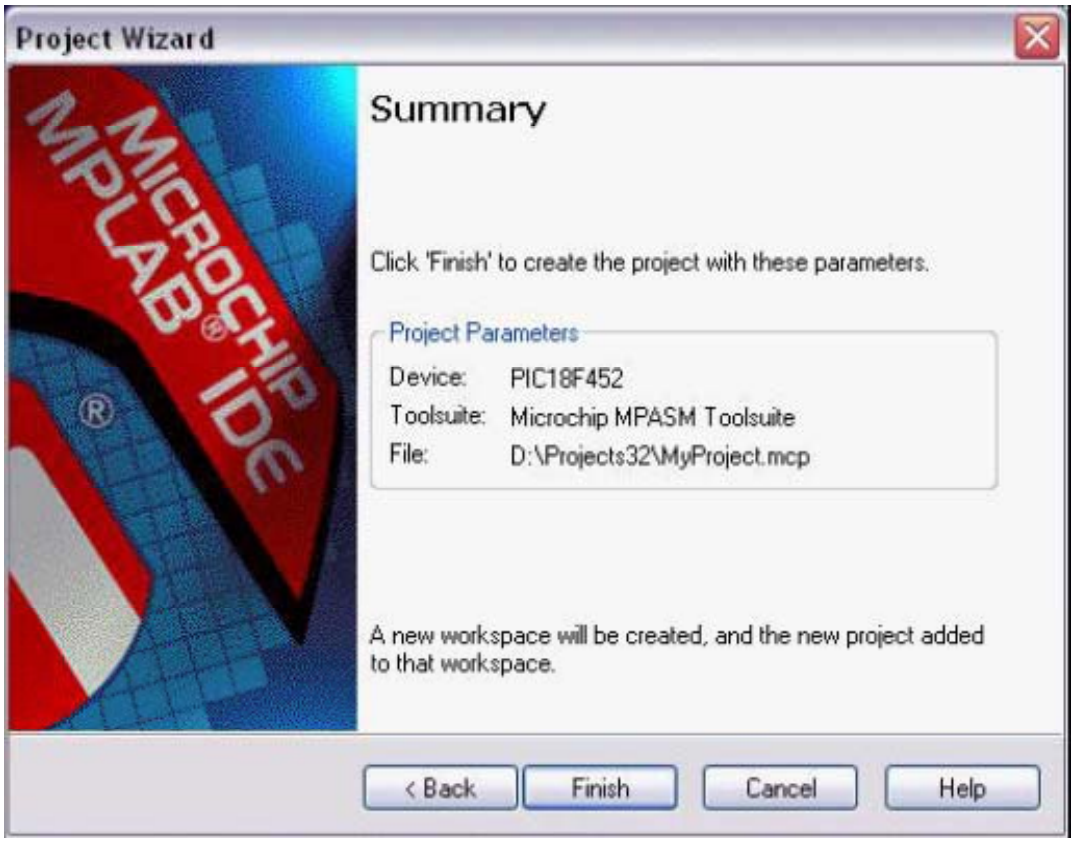
Εικόνα 4.6: Project Wizard - Επιλογή αρχείου προτύπου.

Πατάμε Προσθήκη >> για να μετακινήσουμε το όνομα του αρχείου στο δεξί παράθυρο, και κάνουμε κλικ στο πλαίσιο ελέγχου στην έναρξη της γραμμής με το όνομα του αρχείου για να μπορέσει αυτό το αρχείο να αντιγραφεί στον κατάλογο του έργου μας. Στη συνέχεια, προσθέτουμε το δεύτερο αρχείο για το έργο μας, τον σύνδεσμο σενάριο. Υπάρχει ένας σύνδεσμος σενάριο για την κάθε μια συσκευή. Τα αρχεία αυτά καθορίζουν τη διαμόρφωση μνήμης και καταχωρίζουν ονόματα για τα διάφοραμέρη. Οι σύνδεσμοι σενάρια είναι στο φάκελο με το όνομα LKR κάτω από το φάκελο MCHIP_Tools. Χρησιμοποιούμε το αρχείο με το όνομα 18F452.lkr. Η πλήρης διαδρομή είναι: C: \ Program Files \ MPLAB IDE \ MCHIP_Tools \ LKR \ 18F452.lkr



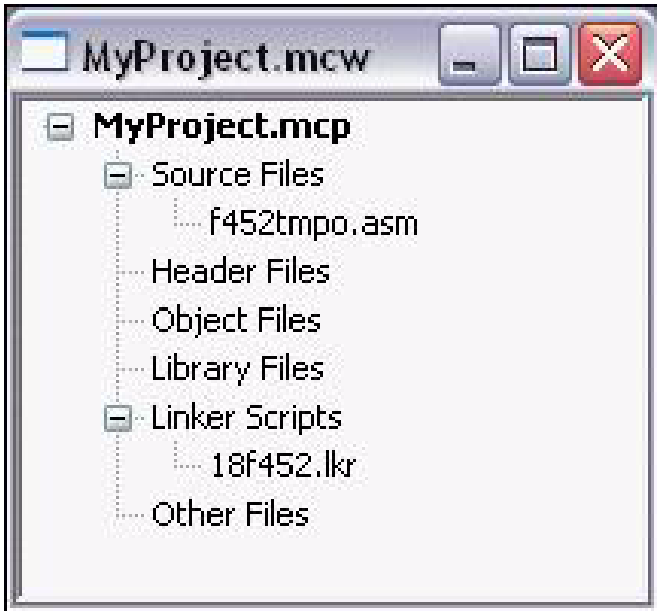
Εικόνα 4.7: Project Wizard - Επιλογή Linker Script

Βεβαιωνόμαστε ότι το παράθυρο διαλόγου μας μοιάζει με την παραπάνω εικόνα, με τα δύο πλαίσια ελέγχου, και στη συνέχεια πατάμε Επόμενο> για να ολοκληρώσουμε τον Οδηγό έργου. Η τελική οθόνη του Οδηγού έργου είναι μια περίληψη που δείχνει την επιλεγμένη συσκευή, το toolsuite και το νέο όνομα αρχείου του έργου.



Εικόνα 4.8: Project Wizard – Περίληψη.

Αφού πατήσουμε το κουμπί Τέλος, εξετάζουμε το παράθυρο του έργου για το MPLAB IDE desktop. Θα πρέπει να φαίνεται όπως στην Εικόνα 2-10. Αν το παράθυρο του έργου δεν είναι ανοιχτό, επιλέγουμε Προβολή> Έργου.



Εικόνα 4.9: Παράθυρο του Project

4.9 Κατασκευή ενός Project.

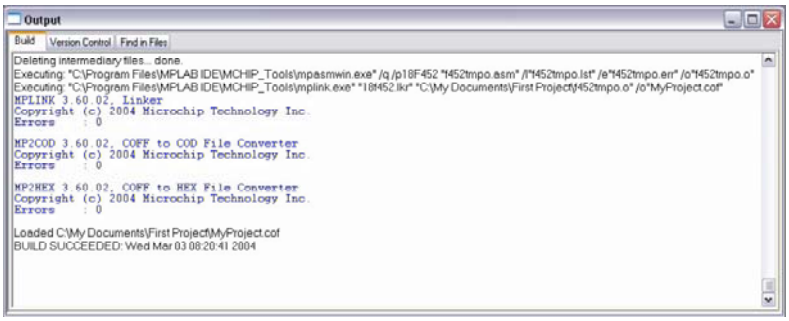
Από το Έργο>μενού, μπορούμε να συγκεντρώσουμε και να συνδέσουμε τα τρέχοντα αρχεία. Δεν έχουμε οποιονδήποτε από τις κώδικές μας σε αυτά ακόμα, αλλά αυτό εξασφαλίζει ότι το έργο έχει ρυθμιστεί σωστά. Για την κατασκευή του έργου, επιλέγουμε είτε:

• Project>Build All

• Κάνουμε δεξί κλικ στο όνομα του έργου στο παράθυρο του έργου και επιλέγουμε την κατασκευή όλων τους.

• Κάνουμε κλικ στο Build All εικονίδιο στη γραμμή εργαλείων του έργου.

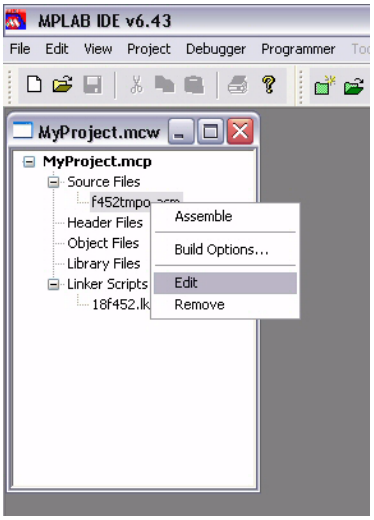
Μετακινούμε το ποντίκι πάνω από τα εικονίδια για να δούμε το pop-up κείμενο και το τι αντιπροσωπεύουν. Το παράθυρο εξόδου δείχνει το αποτέλεσμα της διαδικασίας κατασκευής, και είναι σημαντικό να μην υπάρχουν σφάλματα σε κάθε βήμα.



Εικόνα 4.10: Παράθυρο εξόδου.

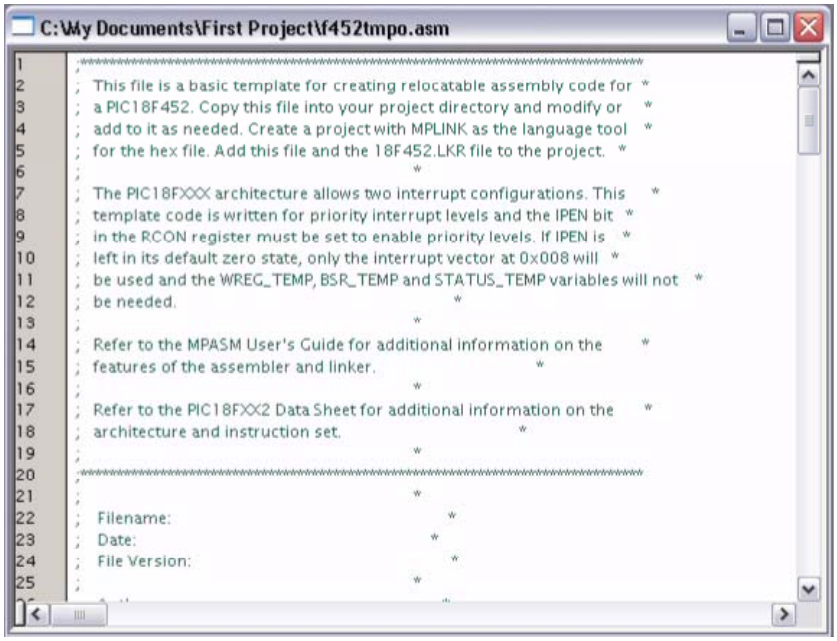
4.10 Δημιουργία Κώδικα.

Ανοίγουμε το αρχείο προτύπου στο πρόγραμμα κάνοντας διπλό κλικ στο όνομά του στο Έργο,Παράθυρο ή επιλέγοντας με τον κέρσορα και χρησιμοποιώντας το δεξί πλήκτρο του ποντικιού για να εμφανιστεί το μενού:



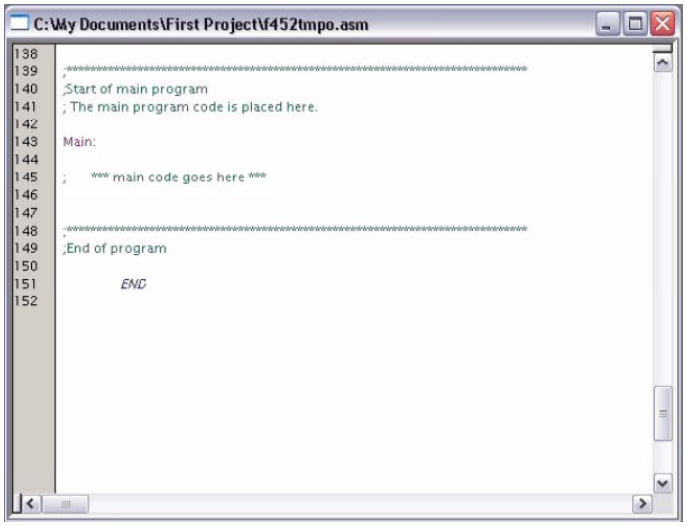
Εικόνα 4.11: Project μενού περιβάλλοντος (δεξί κλικ ποντίκι).

Το αρχείο έχει ορισμένες παρατηρήσεις στην αρχή, και η περιοχή αυτή μπορεί να χρησιμοποιηθεί ως πρότυπο από σχόλιο, κεφαλίδα, πληροφορίες κ.α. για το αρχείο. Για την ώρα θα το αφήσουμε όπως είναι, αλλά μιας και αυτό είναι ένα πραγματικό έργο, παρακάτω θα τοποθετήσουμε πληροφορίες σχετικά με το σχέδιο και τον κώδικά μας εδώ.



Εικόνα 4.12: Αρχείο προτύπου.

Ο κώδικας στο πρώτο μέρος του αρχείου μας είναι για τις πιο προηγμένες λειτουργίες, όπως για την δημιουργία διακοπής και τη διαμόρφωση bits σε μια τελική εφαρμογή. Τα στοιχεία αυτά μπορούν να αγνοηθούν σε αυτό το σημείο, με έμφαση στο γράψιμο του κώδικα. Ο νέος κώδικας θα τεθεί στο αρχείο μετά το σημείο όπου το σύμβολο Main ορίζεται.



Εικόνα 4.13: Αρχείο προτύπου – Main.

Όταν οποιοδήποτε αρχείο προέλευσης ανοίγει, είμαστε αυτόματα στον επεξεργαστή. Που σημαίνει ότι είμαστε έτοιμοι να πληκτρολογήσουμε και να καταγράψουμε τον κωδικό μας:

```
Main:
clr f WREG
movwf PORTC; clear PORTC
movwf TRISC; configure PORTC as all outputs
Init
clrf COUNT
IncCount
incf COUNT
movf COUNT,W
movwf PORTC; display COUNT on PORTC
callDelay
goto IncCount; infinite loop
Delay
movlw 0x40; set outer delay loop
movwf DVAR2
Delay0
movlw 0xFF
movwf DVAR; set inner delay loop
Delay1
decfsz DVAR
goto Delay1
decfsz DVAR2
goto Delay0
return
```

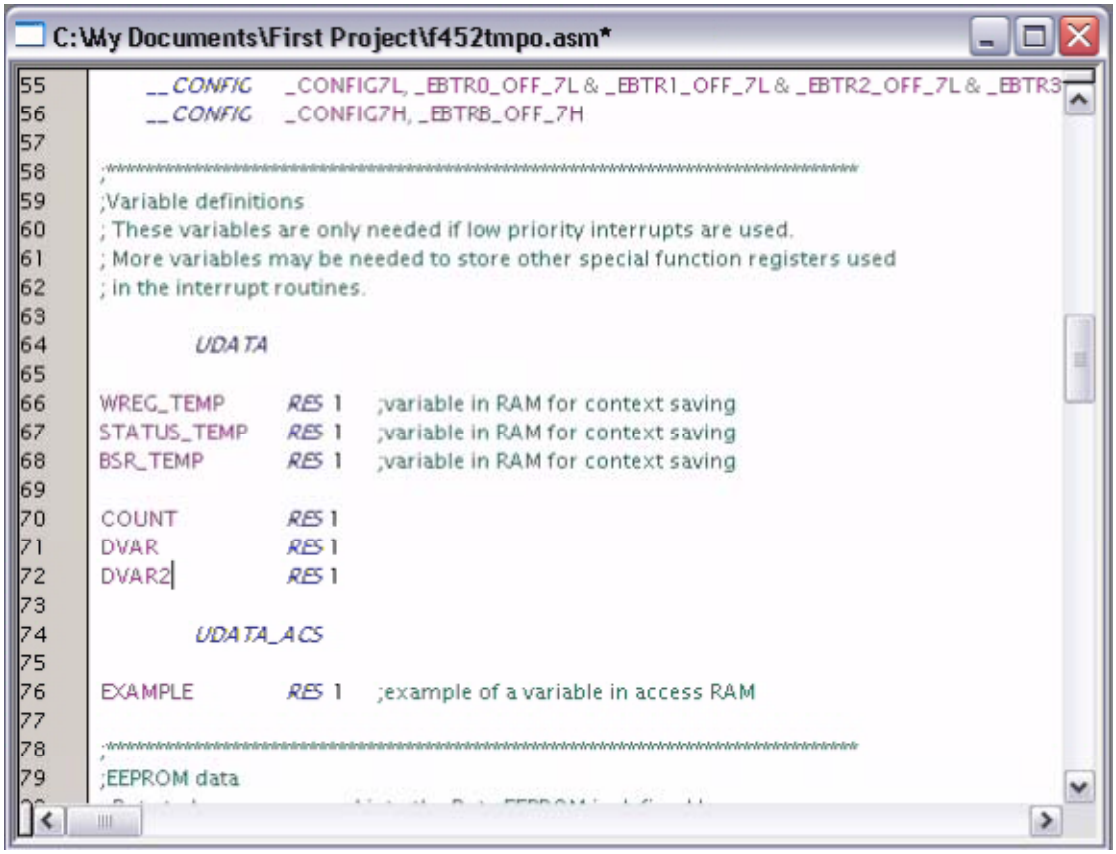
Το πρότυπο αρχείο θα πρέπει τώρα να μοιάζει με την εικόνα 4.14.

C:\My Documents\First Project\f452tmpo.asm

```
146
147  Main:
148
149      clrf    WREG
150      movwf   PORTC       ; clear PORTC
151      movwf   TRISC       ; configure PORTC as all outputs
152
153  Init
154      clrf    COUNT
155  IncCount
156      incf    COUNT
157      movf    COUNT,W
158      movwf   PORTC       ; display COUNT on PORTC
159
160      call    Delay
161      goto    IncCount    ; infinite loop
162
163  Delay
164      movlw   0x40        ; set outer delay loop
165      movwf   DVAR2
166  Delay0
167      movlw   0xFF
168      movwf   DVAR        ; set inner delay loop
169  Delay1
170      decfsz  DVAR
171      goto Delay1
172
173      decfsz  DVAR2
174      goto    Delay0
175      return
176
177  ;*****************************************************************
178  ;End of program
179
180          END
```

Εικόνα 4.14: Πρότυπο αρχείο - Προσθήκη κώδικα.

Σε αυτό το κομμάτι του κώδικα, χρησιμοποιήσαμε τρεις μεταβλητές που ονομάζονται COUNT, DVAR και DVAR2. Αυτές οι μεταβλητές πρέπει να καθορίζονται στο αρχείο του προτύπου στο τμήμα UDATA για τα uninitialized δεδομένα. Υπάρχουν ήδη τρεις μεταβλητές σε αυτό το τμήμα του αρχείου προτύπου, η δική μας μπορεί να είναι προστεθεί στο τέλος χρησιμοποιώντας την ίδια μορφή. Κάθε μεταβλητή είναι μια 8-bit μεταβλητή, έτσι ώστε να πρέπει να κάνει κράτει 1 byte η κάθε μία.

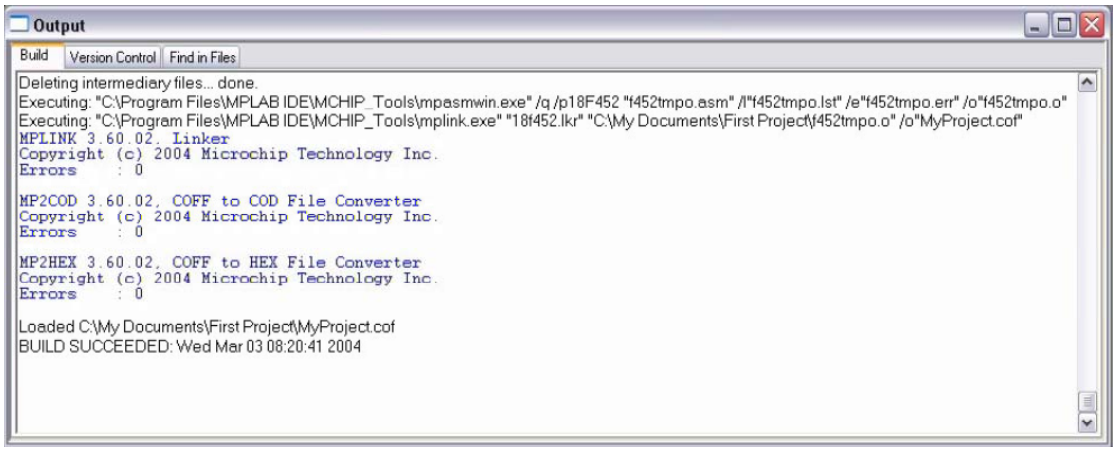


Εικόνα 4.15: Αρχείο προτύπου - προσθήκη μεταβλητών.

4.11 Κατασκευή του Έργου.

Επιλέγουμε Project>Build All για να συγκεντρώσει και να συνδέσει τον κωδικό. Εάν ο κώδικας δεν συναρμολογείται με σφάλματα, το παράθυρο εξόδου θα μοιάζει με την εικόνα 4-17.
Αν αυτά δεν τα συγκεντρώσει και να τα συνδέσει με επιτυχία, ελέγχουμε τα παρακάτω στοιχεία και να ξανακάνει Bulid στη συνέχεια το έργο και πάλι:
• Ελέγχουμε την ορθογραφία και τη μορφή του κώδικα που αναγράφεται στο παράθυρο του επεξεργαστή. Βεβαιωνόμαστε ότι οι νέες μεταβλητές και τα ειδικά μητρώα λειτουργίας TRISC και PORTC, είναι κεφαλαία γράμματα. Εάν η επιχείρηση συναρμολόγησης αναφερθεί πάνω σε λάθη στο παράθυρο εξόδου, κάνουμε διπλό κλικ στο σφάλμα και το MPLAB IDE θα ανοίξει την αντίστοιχη γραμμή στον πηγαίο κώδικα με ένα πράσινο βέλος στο αριστερό περιθώριο του παραθύρου του πηγαίου κώδικα.
• Ελέγχουμε ότι είναι σωστή η συναρμολόγηση (MPASM assembler) και ο σύνδεσμος για τις PICmicro συσκευές που χρησιμοποιούμε. Επιλέγουμε Project> Set Τοποθεσίες Εργαλείο Γλώσσας. Κάνουμε κλικ στα επιπρλοσθετα κελιά για να επεκτείνει την Microchip MPASM toolsuite και να την κάνει εκτελέσιμη. Ύστερα κάνουμε κλικ στο MPASM Assembler (mpasmwin.exe) για να επανεξετάσει τη θέση τους στην οθόνη.

Αν η τοποθεσία είναι σωστή, κάνουμε κλικ στο κουμπί Άκυρο. Αν δεν είναι, το αλλάζουμε και στη συνέχεια κάνουμε κλικ στο κουμπί OK. Τα μονοπάτια για την προεπιλεγμένη μηχανή αναζήτησης μπορεί να είναι κενά. Μετά απο μια επιτυχημένη κατασκευή, το αρχείο εξόδου που παράγεται από το εργαλείο γλώσσα θα φορτωθεί. Αυτό το αρχείο περιέχει τον κώδικα του αντικειμένου που μπορεί να προγραμματιστεί σε ένα PICmicro MCU καθώς και πληροφορίες εντοπισμού σφαλμάτων έτσι ώστε ο πηγαίος κώδικας να μπορεί να διορθωθεί και να προβάλει την πηγή μεταβλητών σε συμβολική γλώσσα των Windows.
Σημείωση: Η πραγματική δύναμη των έργων είναι εμφανής όταν υπάρχουν πολλά αρχεία για να είναι καταρτιζόμενα / συναρμολογημένα και να συνδέονται για να σχηματίσουν την τελική εκτελέσιμη εφαρμογή - όπως σε μια πραγματική εφαρμογή.



Εικόνα 4.16: Build παράθυρο εξόδου.

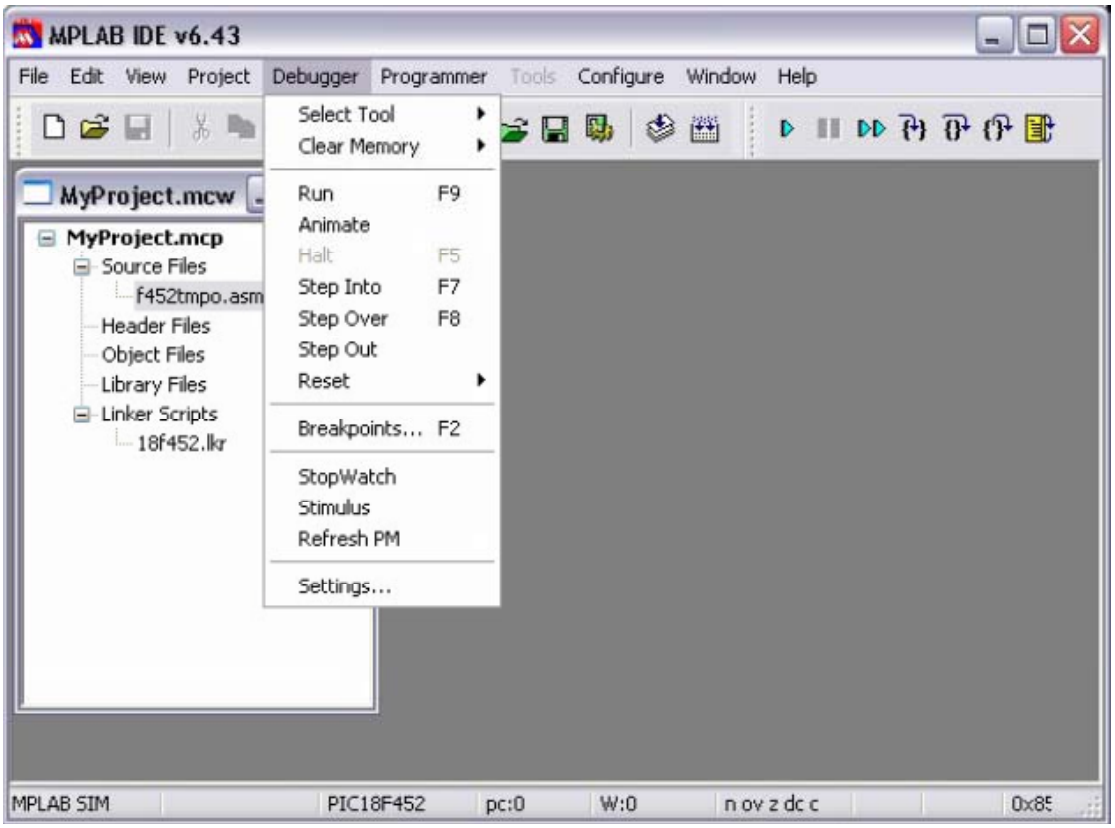
4.12 Δοκιμή του Κώδικα με τον Προσομοιωτή.

Στο κομμάτι αυτό θα δοκιμάσουμε τον κώδικα, το λογισμικό ή το υλικό που χρειάζεται που θα εκτελέσει τις PICmicro οδηγίες. Ένα εργαλείο εντοπισμού σφαλμάτων εκτέλεσης είναι ένα υλικό ή εργαλείο λογισμικού που χρησιμοποιείται για την επιθεώρηση του κωδικού που εκτελεί ένα πρόγραμμα (σε αυτή την περίπτωση cnt452.asm). εργαλείου του υλικού όπως MPLAB ICE ή MPLAB ICD 2 και να μπορεί να εκτελέσει κώδικα σε πραγματικές συσκευές. Εάν το υλικό δεν είναι διαθέσιμο, ο προσομοιωτής MPLAB μπορεί να χρησιμοποιηθεί για τη δοκιμή του κώδικα. Για αυτή τη χρήση έχουμε στην διάθεσή μας τον MPLAB SIM προσομοιωτή. Ο προσομοιωτής είναι ένα πρόγραμμα λογισμικού που τρέχει στον υπολογιστή για να προσομοιώσουν τις οδηγίες του PICmicro MCU. Δεν τρέχει σε «πραγματικό χρόνο», δεδομένου ότι ο πρόγραμμα-προσομοιωτής εξαρτάται από την ταχύτητα του υπολογιστή, την πολυπλοκότητα του κώδικα, από το εναέριο λειτουργικό σύστημα καθώς και πόσα άλλα καθήκοντα εκτελεί.

Ωστόσο, ο προσομοιωτής μετρά με ακρίβεια το χρόνο που θα χρειαζόταν για να εκτελέσει τον κώδικα, αν λειτουργούσε σε πραγματικό χρόνο σε μια εφαρμογή. Επιλέγουμε τον προσομοιωτή ως εργαλείο εντοπισμού σφαλμάτων εκτέλεσης. Αυτό γίνεται από το Debugger> Εργαλείο Επιλογή pull down μενού. Μετά την επιλογή MPLAB SIM, οι ακόλουθες αλλαγές θα πρέπει να θεωρηθούν ως εξής (βλ. αντίστοιχους αριθμούς στην εικόνα 4-18).

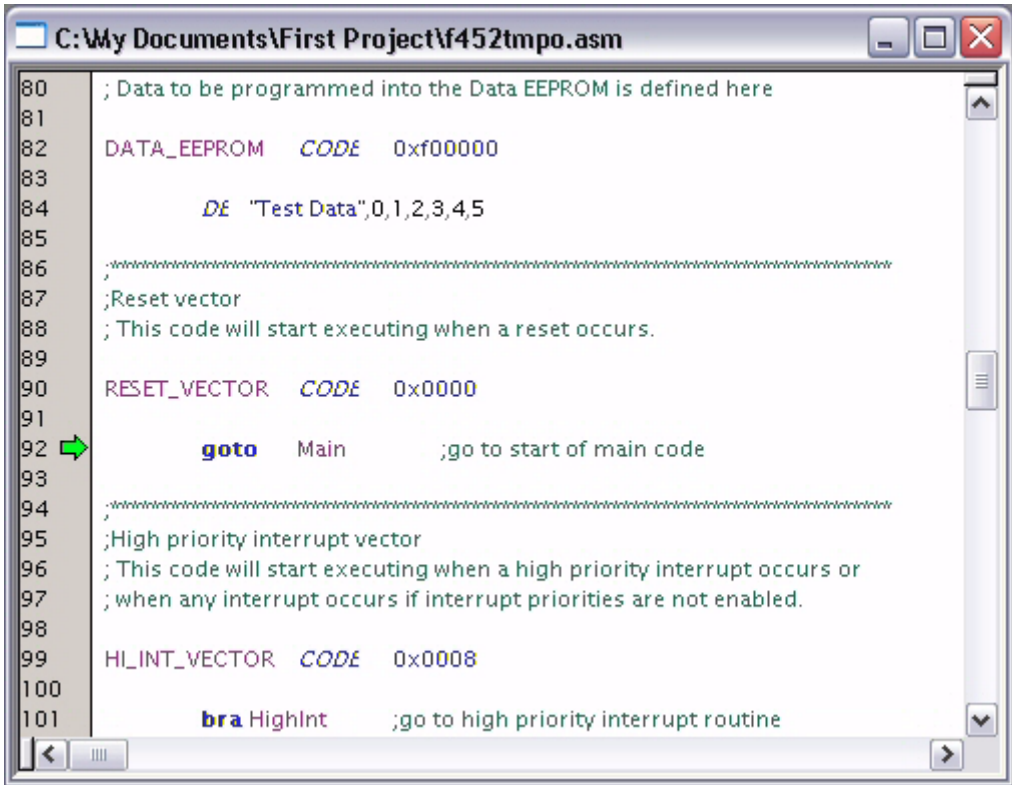
1. Η γραμμή κατάστασης στο κάτω μέρος του παραθύρου του MPLAB IDE θα πρέπει να αλλάξει σε "MPLAB SIM ".
2. Τα πρόσθετα στοιχεία μενού θα πρέπει να εμφανίζονται στο μενού Debugger.
3. Πρόσθετες εικονίδια της γραμμής εργαλείων θα πρέπει να εμφανίζονται στο Debug Bar Tool.

TIP: Τοποθετούμε το δείκτη του ποντικιού πάνω από ένα κουμπί γραμμής εργαλείων για να δούμε μια σύντομη περιγραφή της λειτουργίας του πλήκτρου.
Σημείωση: Άλλα εργαλεία εκτέλεσης debug περιλαμβάνουν το MPLAB ICE 2000, MPLAB ICE 4000 και το MPLAB ICD 2. Αυτά είναι προαιρετικά εργαλεία υλικού για τη δοκιμή κώδικα για την εφαρμογή υπολογιστή του σκάφους. Οι περισσότερες από τις εργασίες MPLAB IDE είναι debugging το ίδιο με το προσομοιωτή, αλλά σε αντίθεση με τον προσομοιωτή, αυτά τα εργαλεία επιτρέπουν το PICmicro MCU να τρέχει σε πλήρη ταχύτητα στην πραγματική εφαρμογή στόχο.



Εικόνα 4.17: MPLAB IDE επιφάνεια εργασίας με MPLAB SIM ως Debugger.

Στη συνέχεια, επιλέγουμε Επαναφορά Debugger> και ένα πράσινο βέλος δείχνει ότι το πρόγραμμα θα ξεκινήσει. Αυτό ήταν μέρος του αρχείου προτύπου. Η πρώτη εντολή σε άλματα μνήμης με την ετικέτα ονομάζεται Main, όπου ο κώδικάς μας εισήχθη. Αυτή η εντολή πηδά πάνω από το PIC16XXXX περιοχές του φορέα σε χαμηλότερη μνήμη.



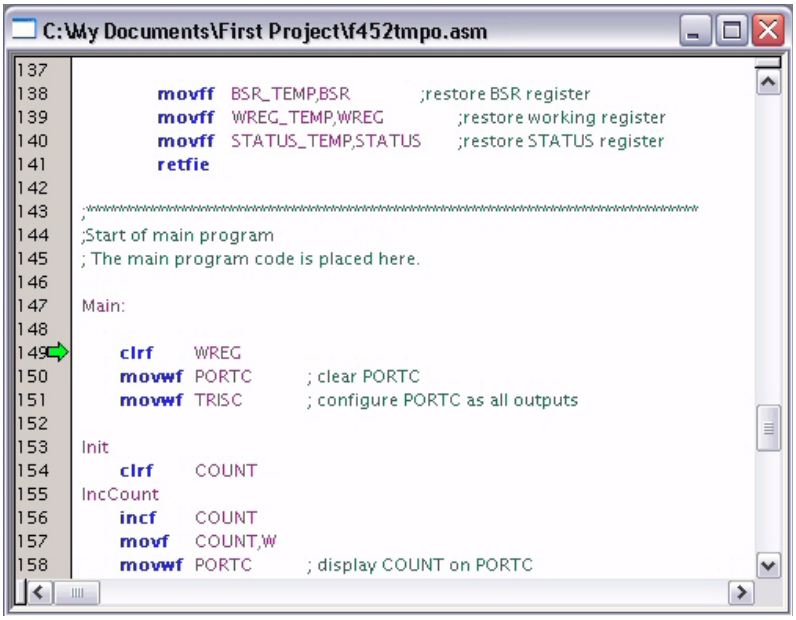
Εικόνα 4.18: Debug > Επαναφορά.

Για να κάνουμε ένα βήμα μέσα από το πρόγραμμα εφαρμογής, επιλέγουμε Debugger> Step Into. Αυτό θα εκτελέσει την παρούσα ενδεικτική γραμμή του κώδικα και θα μετακινήσει το βέλος στην επόμενη γραμμή κώδικα για να εκτελεστεί. Υπάρχουν συντομεύσεις για τις συχνότερα χρησιμοποιούμενες λειτουργίες στο Debug Bar Tool,όπως αναφέρονται στον πίνακα 4-1.

Debugger Menu	Toolbar Buttons	Hot Key
Run		F9
Halt		F5
Animate		
Step Into		F7
Step Over		F8
Step Out Of		
Reset		F6

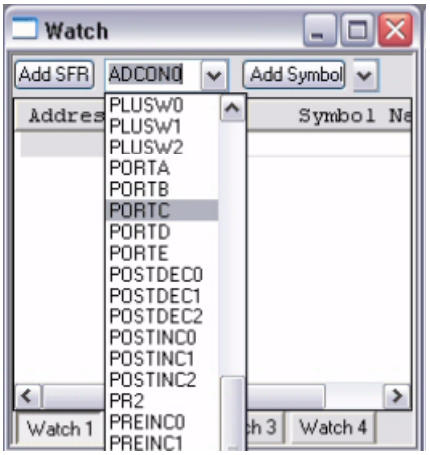
ΠΙΝΑΚΑΣ 2-1: Debug εικονίδια συντόμευσης.

Στη συνέχεια, πατάμε το Step Into εικονίδιο ή επιλέγουμε το Debugger> Step Into σε ενιαίο βήμα για τον κώδικα στο Main.



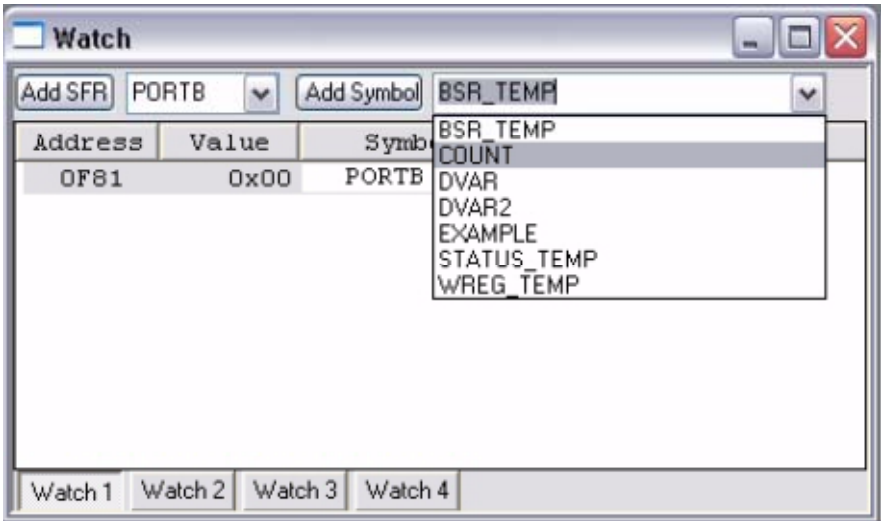
Εικόνα 4.19: Debug >Step Into.

Για να ελέγξουμε αν ο κώδικας λειτουργεί όπως προβλεπόταν, στέλνουμε προσαύξηση στις τιμές out PORTC, για να παρακολουθήσουμε τις τιμές που αποστέλλονται σε PORTC. Επιλέγουμε Προβολή> Παρακολουθήστε για να εμφανιστεί ένα άδειο Παράθυρο παρακολούθησης. Υπάρχουν δύο pull downs στην κορυφή του παραθύρου Watch. Ο πρώτος στα αριστερά με την ένδειξη "Προσθήκη SFR" μπορεί να χρησιμοποιηθεί για να προσθέσουμε το Ειδικό Μητρώο Λειτουργίας, PORTC, στο ρολόι. Επιλέγουμε PORTC από τη λίστα και στη συνέχεια κάνουμε κλικ στο κουμπί Προσθήκη SFR για να προσθέσουμε το παράθυρο.



Εικόνα 4.20: Watch – Επιλογή PORTC.

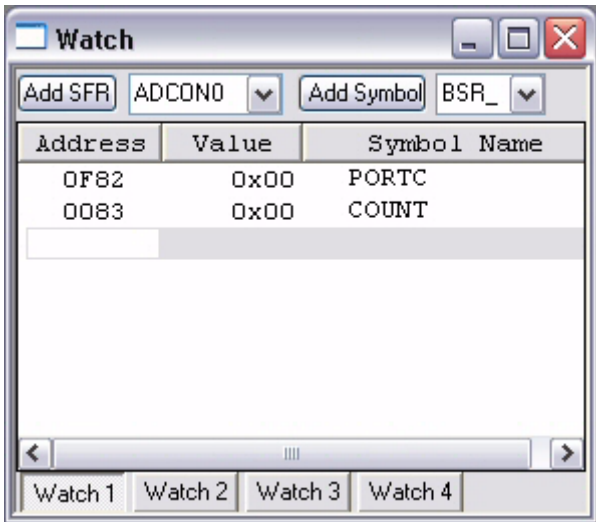
Η επιλογή ¨pull down to the right¨, επιτρέπει στα σύμβολα για να προστεθούν από το πρόγραμμα. Χρησιμοποιούμε αυτή την έλξη κάτω για να προσθέσουμε τη μεταβλητή COUNT στο παράθυρο Watch. Επιλέγουμε COUNT από τη λίστα και στη συνέχεια κάνουμε κλικ στο κουμπί Προσθήκη σύμβολο για να το προσθέσουμε στο παράθυρο.



Εικόνα 4.21: Watch - Επιλογή Προσαρμογής "Count".

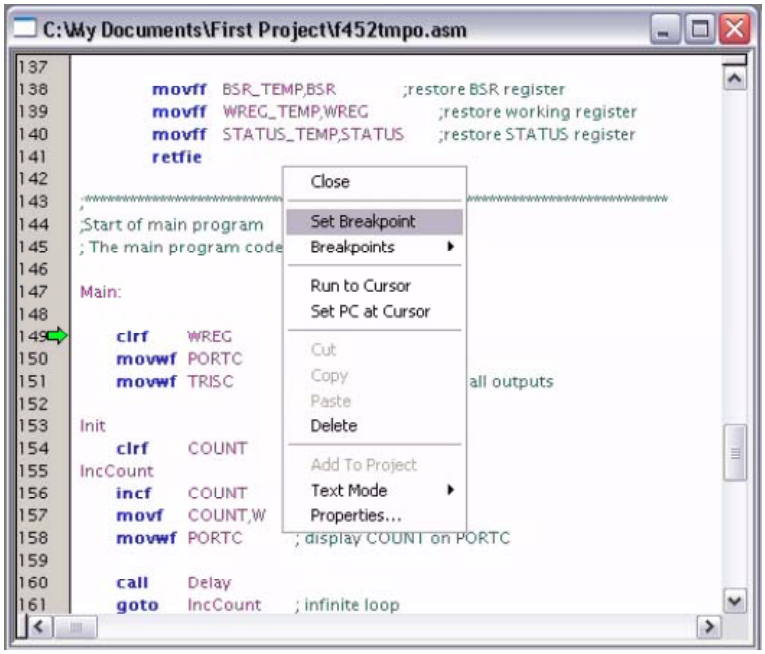
Το παράθυρο “Watch” θα πρέπει τώρα να δείξει τη διεύθυνση, την αξία και το όνομα των δύο μητρώων. Στο σημείο αυτό του προγράμματος, και οι δύο θα είναι μηδέν.

Σημείωση: Τα στοιχεία μπορούν επίσης να προστεθούν στο παράθυρο “Watch” είτε σέρνοντάς το από την SFR, Αρχείο Εγγραφείτε ή Εκδότης παράθυρο ή να κάνουμε κλικ απευθείας στο παράθυρο με το όνομα σύμβολο και πληκτρολογώντας το στοιχείο.



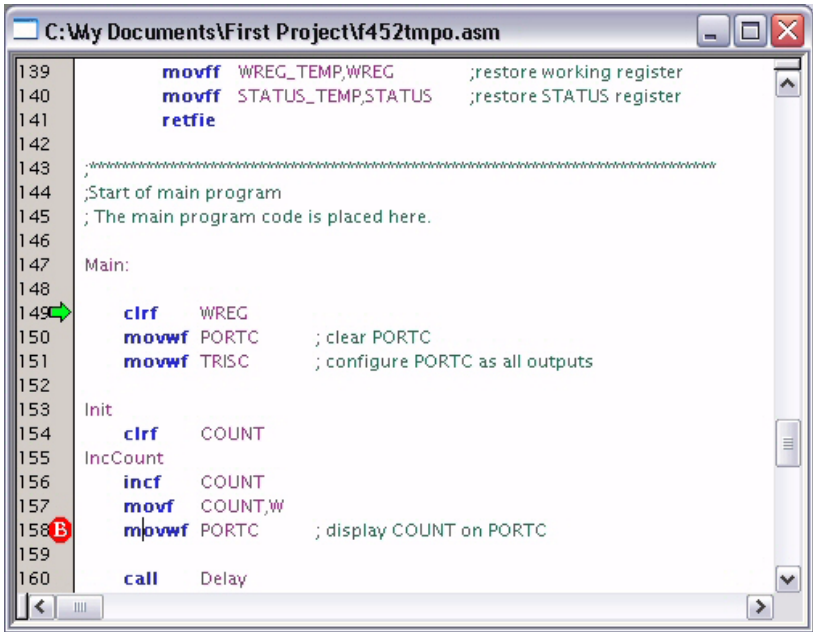
Εικόνα 4.22: Watch – Επαναφορά τιμών.

Θα μπορούσε να συνεχίσει μια ενιαία ενίσχυση μέσα από τον κώδικα, αλλά αντ 'αυτού, θα ορίσουμε ένα breakpoint ακριβώς πριν από την πρώτη τιμή που στέλνεται στο PORTC. Για να ορίσετε ένα σημείο διακοπής, τοποθετήστε το δρομέα στην ενδιαφερόμενη γραμμή και πατάμε το δεξί πλήκτρο του ποντικιού.



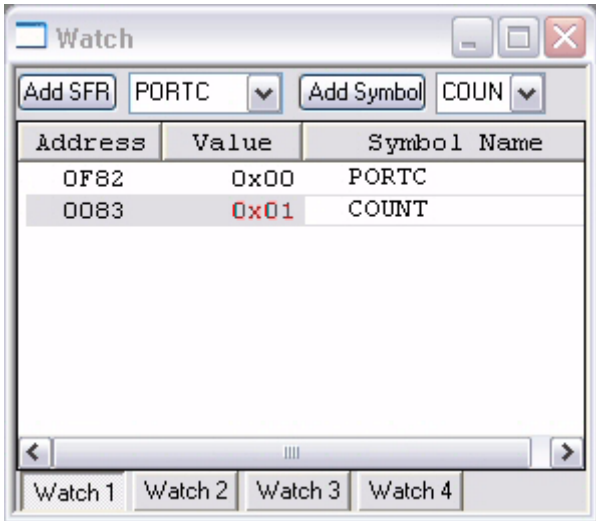
Εικόνα 4.23: DEbug Μενού περιβάλλοντος (δεξί κλικ).

Επιλέγουμε το σημείου διακοπής Set από το μενού περιβάλλοντος. Ένα κόκκινο "B" θα δείξει στη γραμμή. Βέβαια μπορούμε επίσης να κάνουμε διπλό κλικ σε μια γραμμή για να προσθέσουμε ένα σημείο διακοπής.



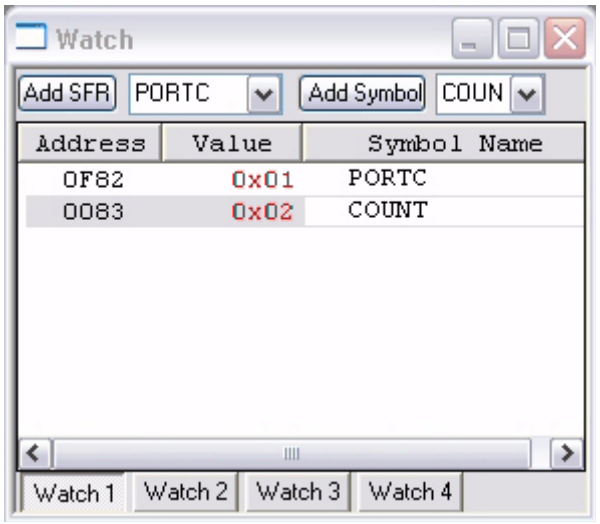
Εικόνα 4.24: Παράθυρο του Επεξεργαστή – Breakpoint Set.

Επιλέγουμε Debugger> Εκτέλεση για να εκτελέσουμε την εφαρμογή. Ένα μήνυμα κειμένου "Running ..." θα είναι σύντομα εμφανιζόμενο στη γραμμή κατάστασης πριν από τις στάσεις εφαρμογής σε αυτό το πρώτο σημείο διακοπής. Το παράθυρο “Watch” θα πρέπει να δείχνει σε αυτό το σημείο ότι η μεταβλητή COUNT αυξάνεται κατά ένα, αλλά δεδομένου ότι η ρήξης είναι στη γραμμή πριν από τη μετάβαση στο PORTC που εκτελεί, η PORTC ακόμα έχει μια τιμή μηδέν.



Εικόνα 4.25(a): Η Watch στο σημείο του Breakpoint.

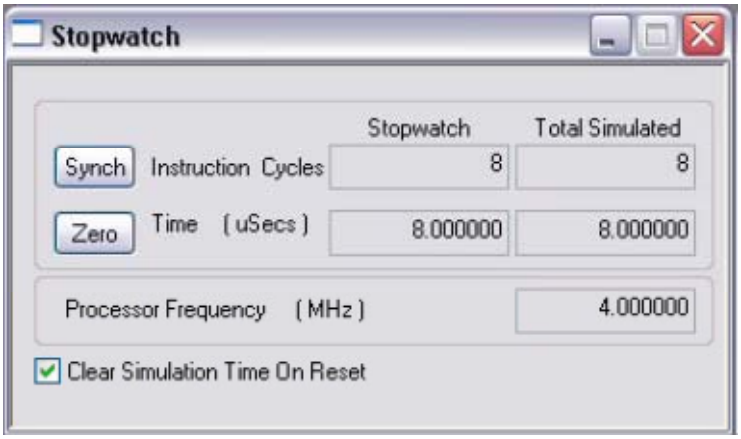
Πατάμε το κουμπί Run εικονίδιο για να εκτελέσει τον κώδικα μέχρι να φτάσει σε αυτό το σημείο και πάλι. Το Παράθυρο παρακολούθησης πρέπει τώρα να δείξει τις δύο τιμές προσαυξανόμενες κατά ένα.



Εικόνα 4.25(b): Η Watch στο επόμενο σημείο του Breakpoint.

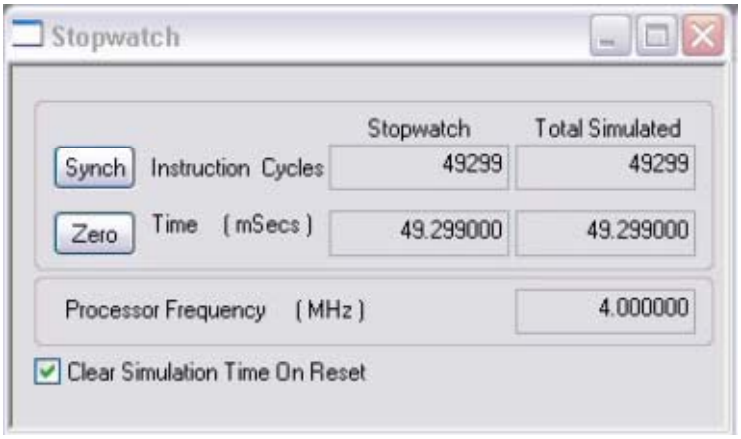
Αυτό φαίνεται να δείχνει ότι το πρόγραμμα λειτουργεί όπως έχει σχεδιαστεί. Μπορούμε να ξεχωρίσουμε βήμα βήμα μέσα στον κώδικα, ή να του δώσουμε την εντολή να εκτελέσει τον κώδικα περισσότερες φορές για να βεβαιωθούμε ότι εκτελείται κανονικά. Εάν τρέξουμε τον κώδικα βήμα βήμα στο βρόχο καθυστέρηση, θα κολλήσουμε στην εκτέλεση χιλιάδων βημάτων μέχρι να φτάσει στο τέλος. Για να παρακάμψουμε το βρόχο καθυστέρησης, χρησιμοποιούμε το Debugger> Step Out. Αν ενδιαφέρόμαστε στο να υπολογίσουμε τον χρόνο καθυστέρησης, η βάση των δεδομένων θα μπορούσε να χρησιμοποιηθεί για να καθορίσει πόσο καιρό κάθε εντολή θα λάβει βρόχο καθυστέρηση και θα έρθει με έναν αρκετά ακριβή αριθμό. Μπορούμε επίσης να χρησιμοποιήσετε το MPLAB χρονόμετρο για τη μέτρηση ή καθυστέρηση. Το κύριο ενδιαφέρον μας θα πρέπει να είναι η ώρα που κάθε νέα τιμή του COUNT εμφανίζεται. Εάν ορίσουμε το breakpoint μας, όπως είχε γίνει αρχικά, σύμφωνα με τις οδηγίες που μετακινούν τους COUNT και PORTC, μπορούμε να το εκτελέσουμε στο επόμενο σημείο διακοπής στο ίδιο μέρος για τη μέτρηση του χρόνου.

Χρησιμοποιούμε τον Debugger> Χρονόμετρο για να εμφανιστεί το παράθυρο διαλόγου χρονόμετρο. Βεβαιωνόμαστε ότι μια ενιαία breakpoint βρίσκεται στην εντολή COUNT movwf, και στη συνέχεια πατάμε Debug> Επαναφορά και τότε Debug> Τρέξτε να σταματήσει κατόπιν εντολής το COUNT movwf. Με τον επεξεργαστή default και συχνότητα 4 MHz, το χρονόμετρο πρέπει να αποδείξει ότι έλαβε 8 μικροδευτερόλεπτα για να φθάσει το πρώτο σημείο διακοπής.



Εικόνα 4.26: Stopwatch – Σε πρώτο Breakpoint.

Εκτελούμε το Run και πάλι για να πάει γύρω από το βρόχο μια φορά, και σημειώστε ότι το χρονόμετρο δείχνει
ότι χρειάστηκαν περίπου 49 χιλιοστά του δευτερολέπτου. Για να το αλλάξετε αυτό, μπορείτε να αλλάξετε τις τιμές στην
καθυστέρηση βρόχου.



Εικόνα 4.27: Stopwatch - Μετά από καθυστέρηση.

4.13 Περίληψη και Επίλογος MPLAB IDE.

Με την ολοκλήρωση αυτού του κεφαλαίου, έχουμε εκτελέσει τα σημαντικότερα βήματα για τη δημιουργία, την οικοδόμησηκαι τη δοκιμή για ένα απλό Project. Η ολοκλήρωση των εργασιών αυτών περιελάβανε:

- Επιλέγοντας τη συσκευή - η PIC16F84A.
- Χρησιμοποίηση του Οδηγού έργου για τη δημιουργία ενός έργου, και χρησιμοποιώντας τον οδηγό για:
- Την επιλογή του MPLAB IDE χτισμένο σε MPASM συναρμολόγησης και MPLINK γλώσσα συνδέσμου εργαλείων,
- Πρόσθεση αρχείων για το έργο: ένα αρχείο πρότυπο για την επιλεγμένη συσκευή και ένα συνδετικό σενάριο για την σωστή οικοδόμηση.
- Γράφοντας κάποιο απλό κώδικα για να στείλουμε μία μεταβαλλόμενο αξία από μία ή παραπάνω I / O θύρες.
- Η οικοδόμηση του έργου.
- Και τέλος, τον έλεγχο του κώδικα με τον προσομοιωτή.

Βιβλιογραφία κεφαλαίου

[1]Μικροελεγκτές PIC" Σταμάτης Αλατσαθιανός, Β.Γκιούρδας Εκδοτική
[2]Microcontroller Programming" Julio Sanchez, Maria P. Canton
[3]http://ww1.microchip.com/downloads/en/DeviceDoc/41262c

ΚΕΦΑΛΑΙΟ 5

ΠΕΙΡΑΜΑΤΙΚΟ ΜΕΡΟΣ

Εισαγωγή

Ο PIC PROGRAMMER είναι μία κατασκευή που χρησιμοποιείται για τον εύκολο προγραμματισμό του PIC μικροτσίπ και για τους Επεξεργαστές με μνήμη flash του προγράμματος.Ο Programmer διαβάζει αρχεία Intel Hex ως παράδειγμα Μικροτσίπ που δημιουργούνται από το πρόγραμμα, και εκτελεί την καύση στη μνήμη flash του PIC, στον περιέχοντα hex αρχείο,όπως επίσης και είναι υπεύθυνο για ρυθμίσεις και καύση γενικά μέσα στο PIC. Έχει μία 40-pin υποδοχή ελέγχου και ένα 5-pin ICSPVerbinder. Προορίζεται κυρίως για PIC στην υποδοχή δοκιμής για να εκτελέσει καύση κώδικα. Σε ICSP υποδοχή μπορεί να συνδεθεί και να τροφοδοτηθεί μέσω του προσαρμογέαι.
Για να λειτουργήσει ο PIC PROGRAMMER θα πρέπει να έχουμε:
· Τον Programmer
· Το firmware για το PIC στον έλεγχο του Programmer
· PC με θύρα USB και καλώδιο USB
· Windows XP (nt/2k/Vista32/7)
· Το πρόγραμμα των Windows: US Burn
· Τα αρχεία βάσης δεδομένων με τις πληροφορίες για US-Burn

5.1 Υποστηριζόμενοι Τύποι PIC.[5]

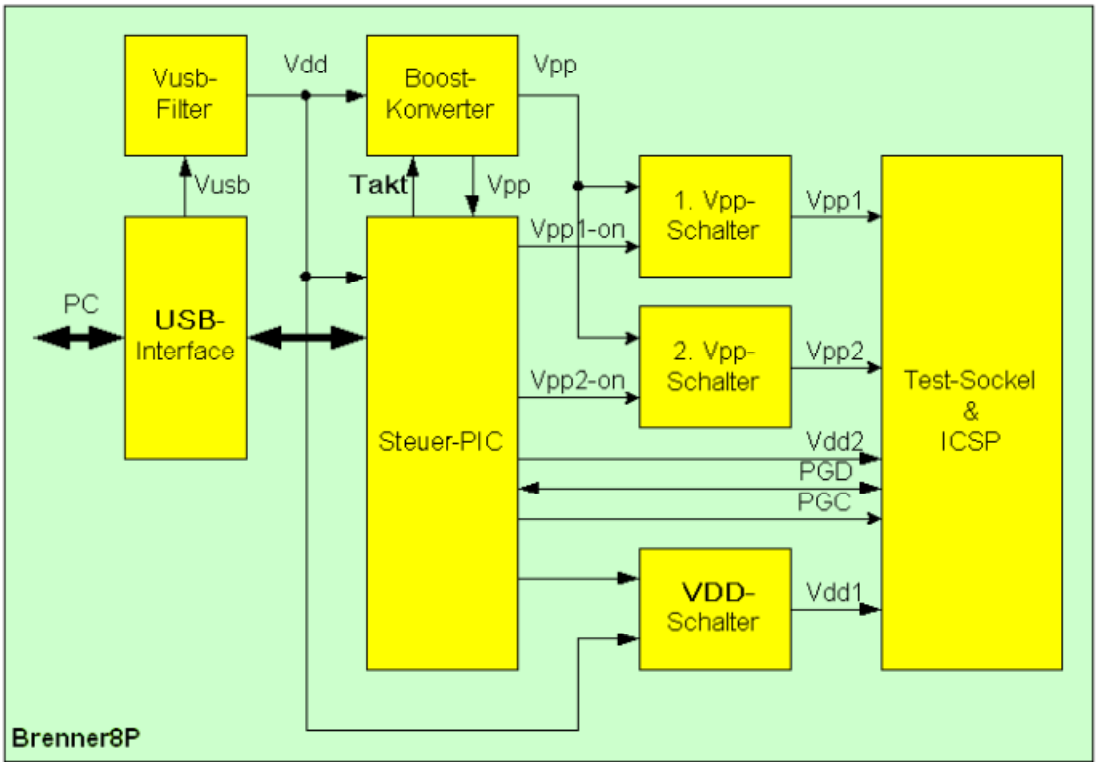
Ο Programmer έχει σχεδιαστεί για να καλύψει όλους τους μικροελεγκτές PIC με φλας
Μνήμη προγράμματος και μπορεί να προγραμματιστεί για τάση λειτουργίας 5V. Η οικογένεια είναι οι εξής:
· Όλα PIC18FXXX και PIC18Fxxxx (δεν PIC18FxxJxx ή PIC18FxxKxx)
· Όλα PIC16Fxx και PIC16Fxxx
· Όλα PIC12Fxxx
· Όλα PIC10Fxxx
· Όλα dsPIC30Fxxxx.

5.2 Pic Programmer.[1,2,4]

Το υλικό του Porgrammer.

Αυτή η ενότητα αναφέρεται στα τεχνικά χαρακτηριστικά του Programmer. Παρακάτω είναι μια σύντομη περιγραφή του υλικού του Programmer:

· USB interface
· Watchdog Timer
· Πηγή τάσης αναφοράς
· Τάση Προγραμματισμού
· 2 διακόπτες τάση προγραμματισμού
· Test Socket



Εικόνα 5.1: Μπλοκ Διάγραμμα του Programmer.

5.2.1 USB interface.

Η θύρα USB χρησιμοποιείται για την επικοινωνία με τον υπολογιστή, καθώς και για την
τροφοδοσία του καυστήρα. Η υποδοχή USB χρησιμοποιείται για τη σύνδεση με τον υπολογιστή. Ο πυκνωτής για τον έλεγχο pin Vusb PIC αποτελεί μέρος του V 3.3 στην παραγωγή PIC18F2550. Αυτό το 3.3 V ως στάθμη σήματος στη θύρα USB ένα πηνίο και δύο πυκνωτές είναι η VBUS τάση USBInterfaces, και στη συνέχεια διατίθεται ως τάση λειτουργίας Vdd του καυστήρα προς λειτουργεία(4.5 .. 5 V).

5.2.2 Παραγωγή Timer.

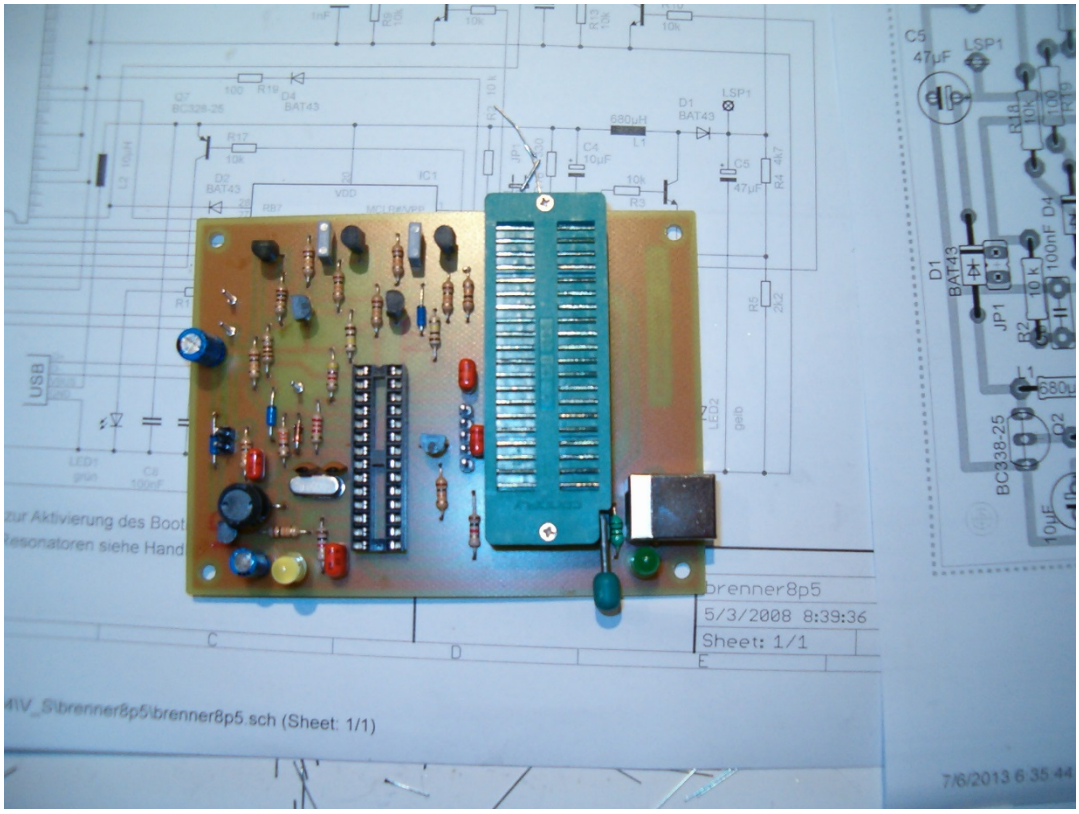
Rsonator με χαλαζία ή των πυκνωτών 2 και 20 MHz ρολόι, που παράγεται τόσο από τον εγκέφαλο του PIC18F2550 καθώς και τον πυρήνα PIC,και από το Clock USB που μπορεί να προέλθει.

5.2.3 Τάση Αναφοράς.

Μια δίοδος Zener 3.3V με μια σειρά αντιστάσεων που παράγουν μια σταθερή τάση (περίπου 3.3 V) του επιπέδου τάσης λειτουργίας (και συνεπώς, από το VBUS επίπεδο), ανεξαρτήτως. Χρησιμεύει ως σημείο αναφοράς για τη μέτρηση (και συνεπώς για την ορθή ρύθμιση) της τάσης προγραμματισμού.

5.2.4 Τάση Προγραμματισμού.

Είναι υπεύθυνο για την τάση προγραμματισμού των 10 .. 13V (ανάλογα με τον τύπο). Η τάση παράγεται από την λειτουργίας μετατροπέα ανύψωσης. Η PIC18F2550 είναι pin 13, 100kHz και έχει ένα σήμα τετράγωνο του οποίου το πλάτους παλμού είναι ρυθμιζόμενο.
Η τάση που παράγεται στο μετατροπέα ώθηση είναι θεωρητικά Vpp = 4.7 V * Περίοδος / off του χρόνου όπου "περίοδος" ο χρόνος ενός τετραγωνικού κύματος, την στιγμή τη της "off-time" της χαμηλής από την περίοδο αυτή. Με την αλλαγή από το "off-χρόνου", η τάση Vpp
μπορεί να ρυθμιστεί ανάλογα. Στην πράξη είναι αρκετές οι παρενέργειες που αντισταθμίζουν και αλλάζουν από τη βαθμονόμηση. Δύο αντιστάσεις σχηματίζουν ένα διαιρέτη τάσης μέσω της οποίας το PIC18F2550 μπορεί να μετρήσει μια τάση που δημιουργείται στο Vpp (στο pin 3).



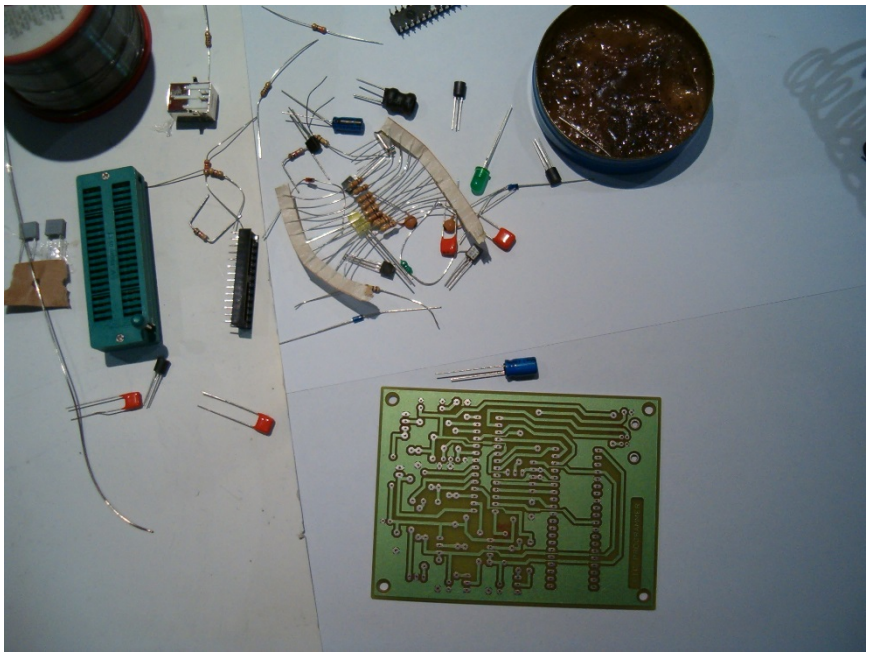
Εικόνα 5.2: Κατασκευή Programmer.

5.2.5 Διακόπτης Τάσης Προγραμματισμού.

Υπάρχουν 2 διακόπτες που μπορούν να αλλάξουν την Vpp στην υποδοχή δοκιμής. Από διαφορετική τάση προγραμματισμού του PIC αναμένουν διαφορετικές ακίδες, ανάλογα με τον τύπο PIC που διαθέτει μία. Κάθε διακόπτης αποτελείται από ένα NPN και ένα PNP τρανζίστορ.

5.2.6 Socket Test.

Στις υποδοχή ελέγχου ή στο σύνδεσμο ICSP στα 5 σήματα Vpp, Vdd, Vss, καθώς και τα δεδομένα. Με την εξαίρεση όλων των σημάτων Vpp απευθείας από το Port-Παράγεται πινέζες του 18F2550. Ανάλογα με το περίβλημα από τους PIC- ο στόχος είναι τα σήματα να εφαρμοστούν στις καρφίτσες μέτρησης. Μία δίοδος είναι αναγκαία, διότι μερικές φορές η δοκιμή pin υποδοχής 1 Vpp και Vss είναι μερικές φορές (ανάλογα με το PIC) δυσανάλογη. Η RB7 δημιουργεί Vss επίπεδο, εάν είναι απαραίτητο.



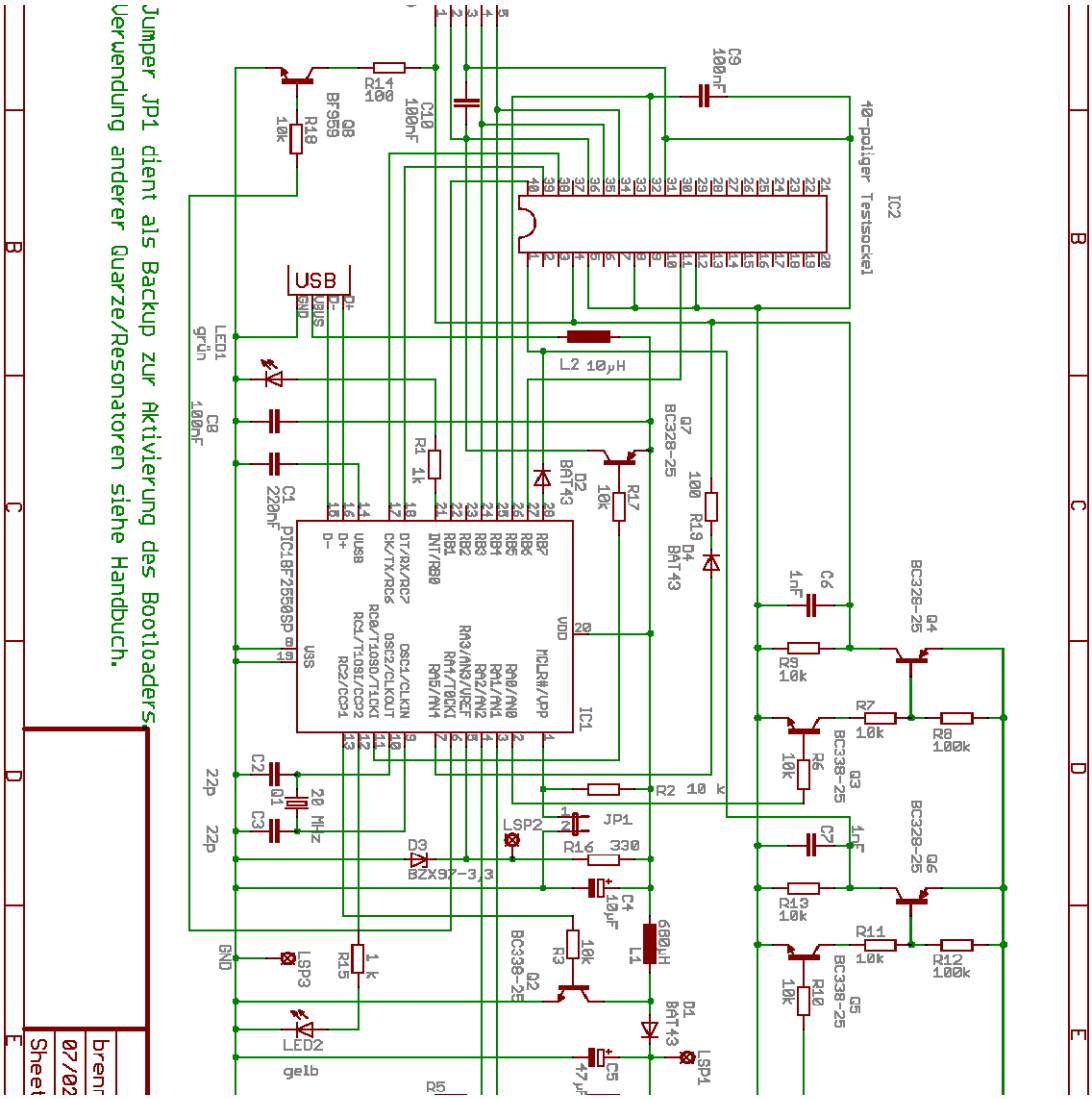
Εικόνα 5.3: Διαδικασία Κατασκευής

5.2.7 Υπόλοιπα.

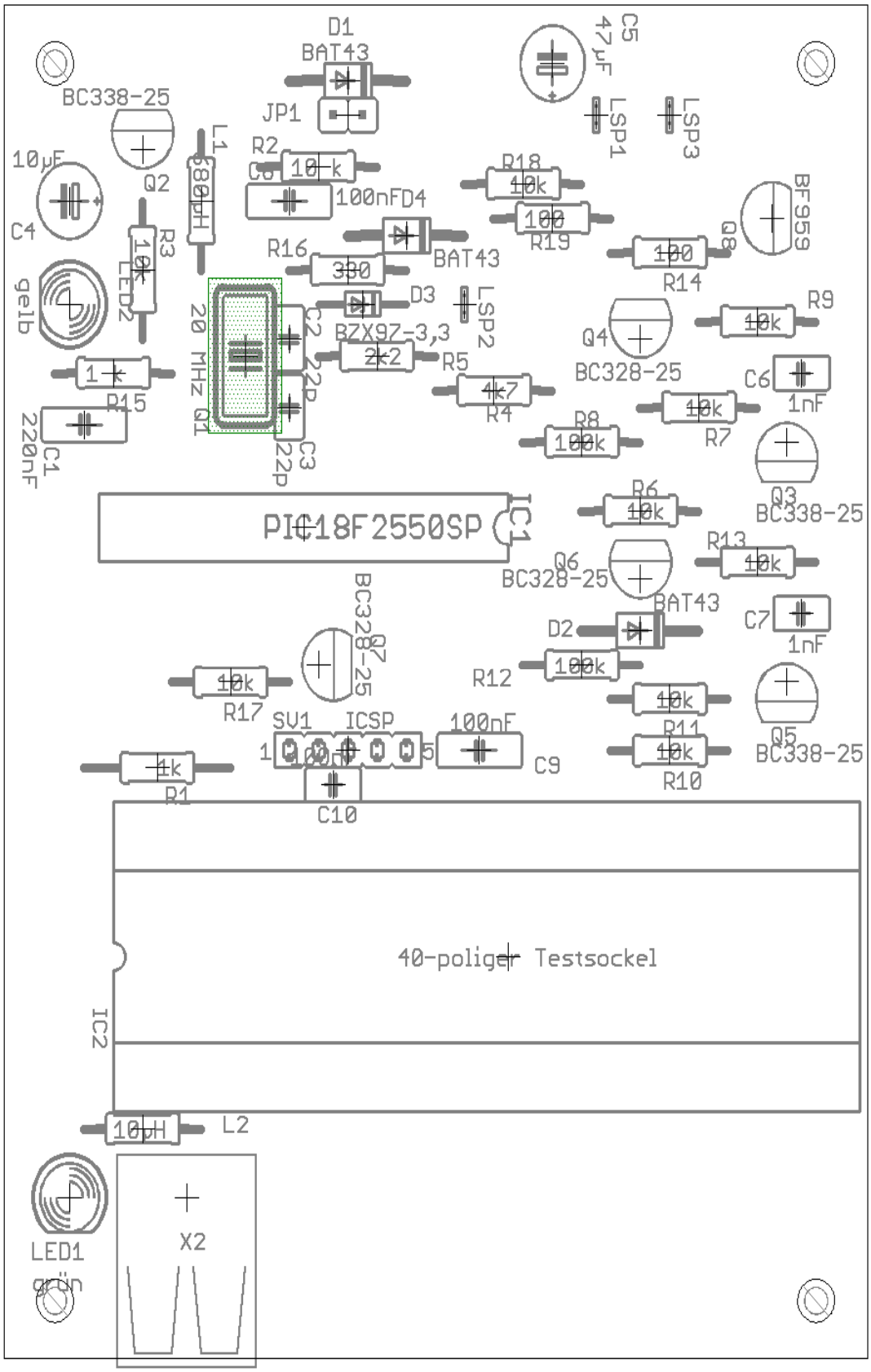
Μια pull-up αντίσταση και ένα jumper για το pin 1 του 18F2550 παραγωγή υψηλού ή
χαμηλού επίπεδου. Αναγνωρίζει τη 18F2550 κατά την εναλλαγή χαμηλού επίπεδο, και τότε αρχίζει ο bootloader, αλλιώς ξεκινά η firmware του Programmer.

5.3 Κύκλωμα.

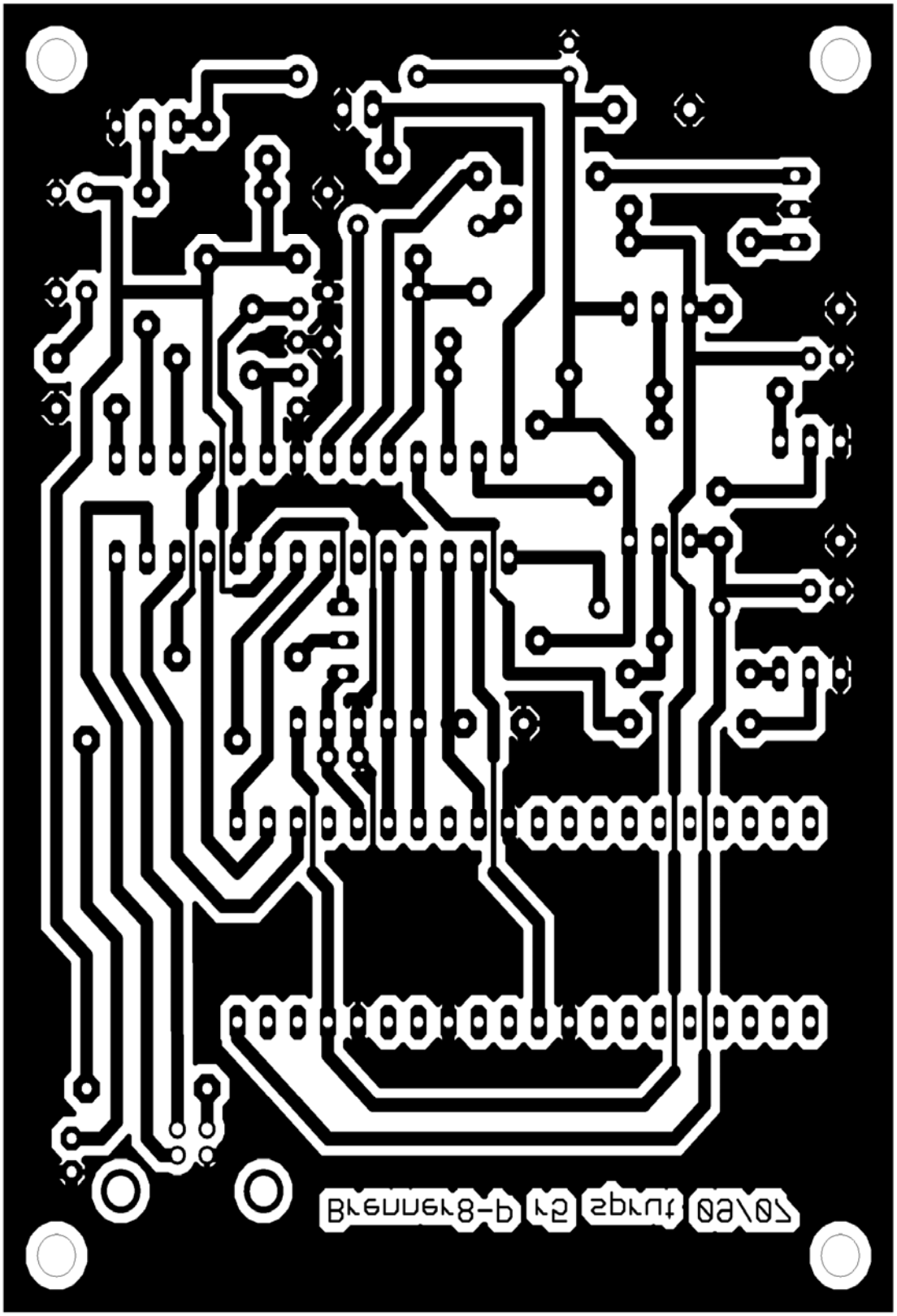
5.3.1 Διάγραμμα Κυκλώματος.



5.3.2 Σχέδιο Τοποθέτησης.



5.3.3 PCB Layout.



5.3.4 Λίστα Εξαρτημάτων του Programmer.

Part	Value	Reichelt Conrad
C1	220nF	Z5U-5 220n
C2, C3	22p	Kerko 22p
C4	10˝F	rad10/100
C5	47˝F	rad47/35
C6, C7	1nF	X7R-2,5 1,0n
C8	100nF	Z5U-5 100n
C9	100nF	Z5U-5 100n
C10	100nF	Z5U-2,5 100n
D1, D2	BAT43	BAT 43
D3	BZX97-3,3	ZF 3,3
D4	BAT43	BAT 43
IC1	PIC18F2550SP	PIC18F2550-I/SP
	28-polige IC-Fassung	GS 28P-S
IC2	40-poliger Testsockel	
JP1	Jumper	Jumper 2,54 RT
L1	680˝H	SMCC 680˝
L2	10˝H	SMCC 10˝
LED1	grón	LED5mm2MAgn
LED2	gelb	LED5mm2MAge
LSP1	Lφtstift	
LSP2	Lφtstift	
LSP3	Lφtstift	
Q1	20 MHz	20,0000-HC49U-S
Q2, Q3	BC338-25	BC338-25
Q4	BC328-25	BC328-25
Q5	BC338-25	BC338-25
Q6, Q7	BC328-25	BC328-25
Q8	BF959	BF959
R1	1k	1/4W 1K
R2, R3	10k	1/4W 10K
R4	4k7	1/4W 4,7K
R5	2k2	1/4W 2,2K
R6, R7	10k	1/4W 10K
R8	100k	1/4W 100K
R9..R11	10k	1/4W 10K
R12	100k	1/4W 100K
R13	10k	1/4W 10K
R14	100	1/4W 100
R15	1 k	1/4W 1K
R16	330	1/4W 330
R17, R18	10k	1/4W 10K
R19	100	1/4W 100
SV1	ICSP	BL 1X10G 2,54
X2	USB-B-H	USB BW

5.4 Εγκατάσταση Drivers του US Burn (μόνο για Windows).[3,4]

Αφού η κατασκευή Brenner8 συναρμολογηθεί πλήρως, καθώς και ο έλεγχος του PIC είναι σωστός, όπως και το Firmware (τουλάχιστον το bootloader) ,επόμενο βήμα είναι η εγκατάσταση των Windows. Η Brenner8 απαιτεί από την Microchip ειδικό πρόγραμμα εγκατάστασης (mpusbapi.dll),το οποίο βρίσκεται στην ιστοσελίδα Microchip, ή στο λογισμικό πακέτο του US Burn στην θέση της αρχικής σελίδας. Αφού γίνει η εγκατάσταση του driver, ο Brenner8 μπορεί να συνδεθεί με τον υπολογιστή των Windows. Τα Windows βρίσκουν αυτόματα σε αυτόν την άγνωστη συσκευή USB PIC Brenner8 σε μία από τις εξόδους. Στη συνέχεια, τα Windows ζητούν την εγκατάσταση του προγράμματος οδήγησης.

Εικόνα 5.4: Hardware Οδηγός 1

Αφού κάνουμε κλικ, στη συνέχεια είμαστε στον οδηγό εγκατάστασης υλικού.

Εικόνα 2: Hardware Οδηγός 2

Ύστερα, μπορούμε να επιλέξουμε τη μικρότερη από τις δύο πιθανές επιλογές για τον οδηγό για χειροκίνητη εγκατάσταση. Αφού κάνουμε αυτό κάνουμε κλικ

στο Next. Τώρα θα πρέπει να τοποθετήσουμε τον Us Burn σε μια κατηγορία συσκευών. Δεδομένου ότι δεν ήταν και δεν ταιριάζει πραγματικά με μία από τις συνήθεις ομάδες, επιλέγουμε Άλλες συσκευές και κάνουμε κλικ στο Next.

Εικόνα 5.5: Hardware Οδηγός 3

Τώρα μπορούμε να το πάρουμε για να επιλέξουμε το πρόγραμμα οδήγησης συσκευής. Σε αυτό το σημείο κάνουμε κλικ στο κουμπί Disk.

Εικόνα 5.6: Επιλέγουμε ένα πρόγραμμα οδήγησης συσκευής.

Στο παράθυρο αυτό τώρα πρέπει να ορίσουμε την διαδρομή στον κατάλογο που περιέχει το πρόγραμμα οδήγησης.

Εικόνα 5.7: Ρύθμιση της διαδρομής του οδηγού.

Τα Windows ψάχνουν σε αυτόν τον κατάλογο για να βρούνε ένα κατάλληλο πρόγραμμα οδήγησης για έναν PIC Brenner8.

Εικόνα 5.8: Επιλέγουμε ένα πρόγραμμα οδήγησης συσκευής 2.

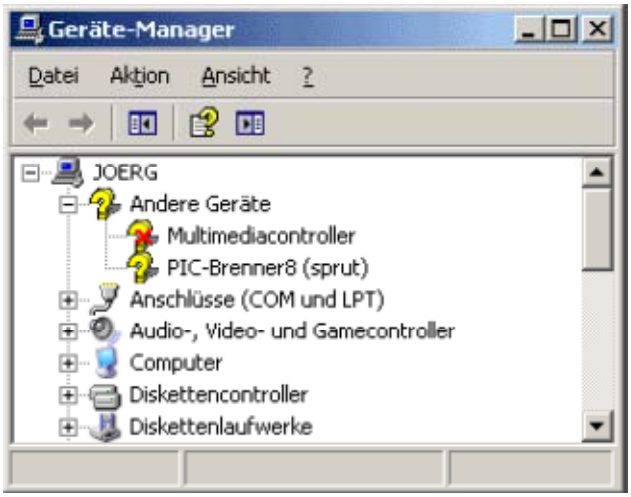
Στο παράθυρο που επιβεβαιώνει την επιλογή πατάμε το Next(εικόνα 9).

Εικόνα 5.9: Ο οδηγός μπορεί να εγκατασταθεί.

Τα Windows θα εγκαταστήσουν το πρόγραμμα οδήγησης. Τελειώνουμε με ένα κλικ και η εγκατάσταση ολοκληρώθηκε.

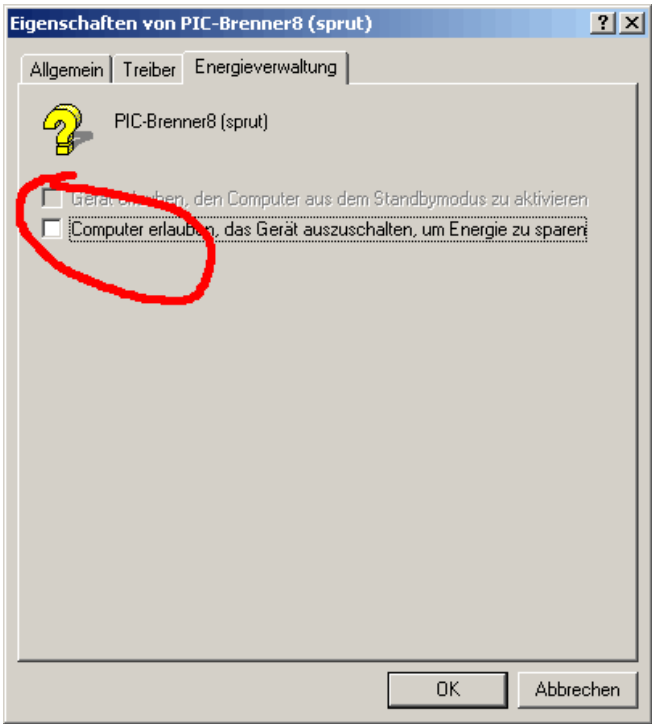
Εικόνα 5.10: Ο οδηγός είναι έτοιμος προς χρήση.

Από τώρα και στο εξής μπορούμε να βρούμε το Brenner8 στο Device Manager.



Εικόνα 5.11: Διαχείριση Συσκευών.

Στη Διαχείριση συσκευών, θα πρέπει να απαγορευθεί η λειτουργεία στο λειτουργικό σύστημα του Brenner8 για εξοικονόμηση ενέργειας. Διαφορετικά, υπάρχει περίπτωση η
Brenner8 να λειτουργεί εξαιρετικά αργά.



Εικόνα 5.12: Διαχείριση συσκευών – Επιλογή εξοικονόμησης ενέργειας.

Έτσι, η Brenner8 είναι έτοιμη για λειτουργεία.

5.5 Βαθμονόμηση του PIC Programmer.[4]

Αφού η Brenner8 έχει συναρμολογηθεί πλήρως, ο PIC microcontroller έχει ελεγχθεί και εγκατασταθεί με το σωστό firmware και το πρόγραμμα οδήγησης USB έχει ρυθμιστεί, πρέπει να προγραμματιστεί και να βαθμονομηθεί η τάση του Brenner8 μέσω της US Burn. Μια μη βαθμονομημένη Brenner8 είναι σε θέση να καταστρέψει κάθε PIC σε πολύ σύντομο χρονικό διάστημα! Μπορούμε εύκολα να δημιουργήσει ένα προγραμματισμό τάσης Vpp = 25V. Και σε αυτην την περίπτωση δεν σώζεται κανένα PIC! Η Brenner8 παράγει τον προγραμματισμό τάσης Vpp χρησιμοποιώντας ένα μικρό μεταγωγικό ρυθμιστή. Από το λογισμικό, το ποσό της τάσης μπορεί να ποικίλει. Είναι συνεπώς σίγουρο ότι κάθε PIC παίρνει την καλύτερη από αυτόν τάση προγραμματισμού. Αυτό λειτουργεί μόνο όταν το βέλτιστο σημείο ελέγχου του PIC της τάσης προγραμματισμού μπορεί να μετρηθεί με ακρίβεια. Η τάση μετράται μέσω ενός διαιρέτη τάσης στον Brenner8, και συγκρίνεται με μια τάση αναφοράς (μια δίοδος Zener). Για τη βαθμονόμηση, μπορεί να καθοριστεί η τάση Zener και του διαιρέτη τάσης ανάλογα.

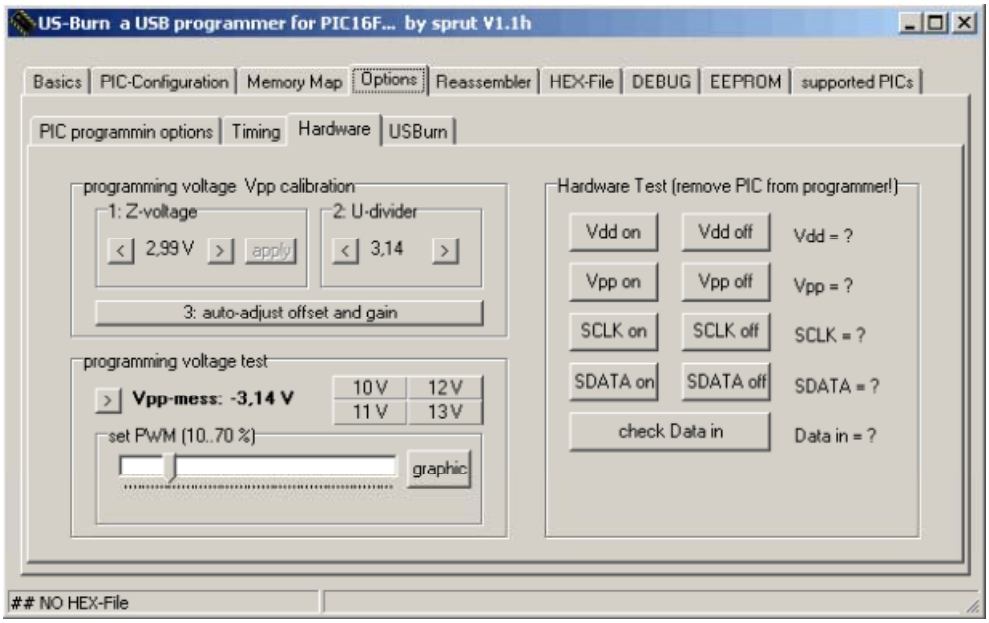
5.5.1Βαθμονόμηση του PIC Programmer στα Windows.

Απαιτούνται:
· Brenner8.
· US Burn λογισμικό.
· Βολτόμετρο.

Προετοιμασία
Συνδέουμε τον Brenner8 στον υπολογιστή.
Εκκίνηση του US Burn στον υπολογιστή
Επιλέγουμε την καρτέλα Options - hardware αλλαγή.
Οι απαραίτητες ρυθμίσεις στο "κουτί" Τάσης προγραμματισμού Vpp που εκτελεί την βαθμονόμηση, και πραγματοποιείται σε τρία βήματα.
1 Ρύθμιση της τάσης Zener
2 Ρύθμιση της τάσης δείκτη διαιρέτη
3 Αυτόματη ρύθμιση του ελεγκτή.

Βήμα 1: Τάση Zener.
Στο διάγραμμα του κυκλώματος παρέχεται μια 3.3V δίοδος Zener, αλλά οι τυπικοί διόδοι Zener έχουν μια ανοχή 10%. Μπορείτε να ανατρέξουμε στην τάση, ώστε να έχουμε κέρδος. Η τάση στα άκρα της διόδου D3 Zener μετριέται χρησιμοποιώντας το βολτόμετρο. (Π.χ. για Brenner8 μεταξύ των κολλήσεις πινέζες LSP2 και LSP3.) Η τιμή της τάσης στη συνέχεια, ορίζει την Z-τάση πεδίου. Τα δύο πλήκτρα βέλους, η τάση μεταξύ 2V και 4V μπορεί να ρυθμιστεί σε 0,01 V προσαυξήσεις. Έτσι, η νέα τιμή θα πρέπει να είναι αποτελεσματική, και ύστερα κάνουμε κλικ στο κουμπί Εφαρμογή.

Στη συνέχεια της βαθμονόμησης, η οποία είναι ελαφρώς μικρότερη από 1 εμφανίζεται συνήθως στο παράθυρο κειμένου στην καρτέλα με τις βασικές ρυθμίσεις.



Εικόνα 5.13: USBurn - Επιλογές Hardware.

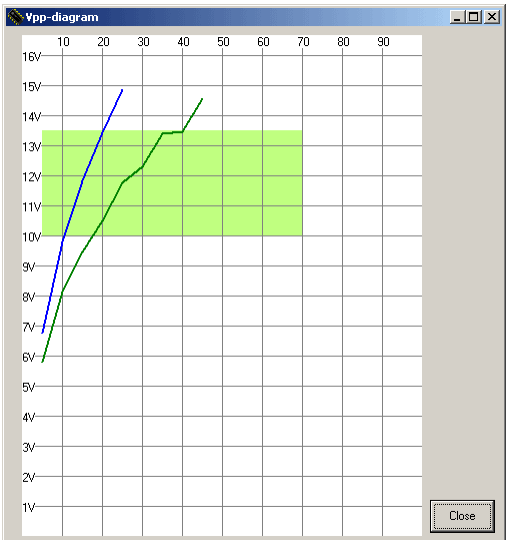
Βήμα 2: διαιρέτη τάσης

Η διαίρεση τάσης για τη μέτρηση της τάσης Vpp έχει τις αντιστάσεις R4 και R5. Ο δείκτης στον διαιρέτη τάσης του είναι θεωρητικά 3.14. Στην πράξη βέβαια, η τιμή διαφέρει. Η ρύθμιση γίνεται στο διαχωριστικό U-box. Για την προετοιμασία μπορούμε να κλείσουμε το βολτόμετρο μεταξύ της καθόδου των D1 και Vss (εναλλακτικά C5). (Π.χ. για Brenner8 μεταξύ LSP1 LSP3.) Χρησιμοποιούμε τον ρυθμιστή για να ορίσούμε την PWM, που παρέχει τάση περίπου 13V (για το πολύμετρο). Η τάση πρέπει να είναι όσο το δυνατόν υψηλότερη, αλλά πρέπει να μην υπερβαίνει τα 14V. Το USBurn Σε ορισμένες εκδόσεις του slider μπορεί και να μην λειτουργεί, τότε η λύση μας είναι να χρησιμοποιηθεί όπως είναι. Η Brenner8 μετρά επίσης τάση και εμφανίζει ανάγνωση όπως μετράται η Vpp. Με την αλλαγή του V-διαιρέτη τώρα μετράται από τον Brenner8 η τάση, αξία στην μετρούμενη τιμή του πολύμετρου, καθώς και κατά προσέγγιση.

Βήμα 3: Ελεγκτής ρυθμίσεις
Τέλος, θα πρέπει να καθορίσουμε την συμπεριφορά του ελεγκτή US Burn για τις επόμενες λειτουργείς του για να ορίσουμε ακριβείς τάσεις. Για να γίνει αυτό απλά κάνουμε κλικστο κουμπί αυτόματη ρύθμιση offset. OUS Burn κάνει τώρα αυτόματα όλες τις απαραίτητες μετρήσεις, η οποίες διαρκούν περίπου 6 δευτερόλεπτα. Από εκεί και πέρα οι πολύ υψηλές τάσεις που δημιουργούνται επιτρέπουν κατά τη διάρκεια της δοκιμής, αν στην υποδοχή δοκιμής υπάρχει θύρα ICSP και όχι USB, την λειτουργεία του US Burn χωρίς να είναι συνδεδεμένος κάποιος PIC.Στη συνέχεια, ορισμένοι αριθμοί εμφανίζονται στο παράθυρο κειμένου καρτέλας Basic που θα είναι χρήσιμοι σε περίπτωση αντιμετώπισης προβλημάτων. Για να ελέγξουμε το αποτέλεσμα, μπορούμε να χρησιμοποιήσουμε τα 10V, 11V, 12V και κάνουμε κλικ 13V. Στη συνέχεια, μία τάση προγραμματισμού θα πρέπει να προσαρμοστεί σε αυτές τις τιμές περίπου. Το τυπικό σφάλμα ρύθμισης είναι περίπου 0,3 V.

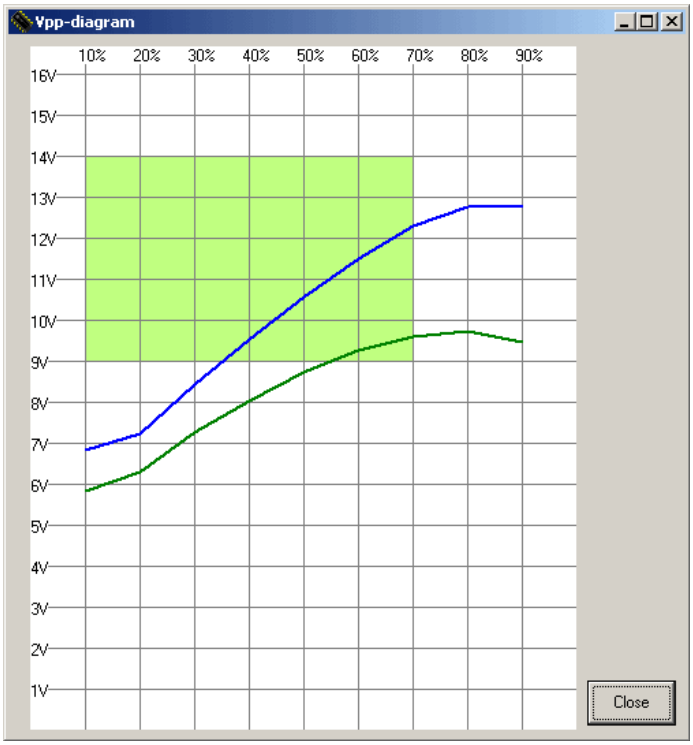
5.5.2 Τέλος Βαθμονόμησης.
Τα δεδομένα βαθμονόμησης αποθηκεύονται μόνιμα στον US Burn. Κατά το κλείσιμο του προγράμματος, αντίγραφα ασφαλείας του προγράμματος με τις τιμές του αρχείου να βρίσκονται στο usburn.ini αποθηκεύονται και είναι διαθέσιμα την επόμενη φορά που το πρόγραμμα θα ξεκινήσει.

5.5.3 Αντιμετώπιση Προβλημάτων
Στο εικόνα 5.14 υπάρχει η τάση προγραμματισμού και θα μας εμφανιστεί αν πατήσουμε την επιλογή test graphic. Στη συνέχεια, δημιουργεί μία γραφική παράσταση των τάσεων του US Burn ρυθμιστή εξόδου. Το γράφημα δείχνει την τάση Vpp για διαφορετικές τιμές ανάλογα με τον χρόνο του ελεγκτή με Vpp-off (μπλε γραμμή) και Vpp-ένα (πράσινη γραμμή). Η δημιουργία του γραφικού διαρκεί λίγα δευτερόλεπτα. Επειδή σε αυτό το σημείο πολύ υψηλές τάσεις δημιουργούνται κατά τη διάρκεια της δοκιμής δεν πρέπει σε καμία περίπτωση να υπάρχει PIC συνδεδεμένος στον Programmer. Δεδομένου ότι ο Brenner8 μπορεί να μετρήσει μόνο τάσεις μέχρι 15V, δεν μετρά υψηλότερες τιμές από αυτές που σχεδιάζονται σε διάγραμμα.



Εικόνα 5.14: Διάγραμμα Vpp – κανονική.

Η περιοχή με το πράσινο φως είναι το φυσιολογικό εύρος λειτουργίας του ελεγκτή. Και τα δύο σύνολα χαρακτηριστικών. Αυτή η περιοχή θα πρέπει να περάσει από κάτω προς τα πάνω. Το παρακάτω γράφημα δείχνει ένα Brenner8 με έναν κακό ρυθμιστή τάσης. Σε αυτήν την περίπτωση, οι δημιουργούμενες τάσεις είναι πάρα πολύ χαμηλές. Η αιτία του σφάλματος μπορεί να μην είναι μόνο πρόβλημα με το υλικό του, πιθανότατα να είναι και κάποια λάθος επιλογή κάποιας διόδου.



Εικόνα 5.15: Διάγραμμα Vpp - τάση πολύ χαμηλή.

5.6 Το Πρόγραμμα US Burn για Windows. [4]

Το US Burn είναι ένα πρόγραμμα των Windows που αναλύει ένα HEX αρχείο, και μετά την ανάλυση μεταδίδει δεδομένα μέσω σύνδεσης USB που στο Brenner8. Πέρα από αυτές τις υπηρεσίες, το US Burn προσφέρει κάποιες επιπλέον επιλογές όπως:
· Ελεύθερη διαμόρφωση του PIC
· HEX προβολή αρχείων
· Reassembler
· Τα δεδομένα EEPROM σε οθόνη
· Βαθμονόμηση Brenner8
· Τοποθέτηση νέου firmware στο PIC (με bootloader)

5.6.1 Απαιτήσεις για τη χρήση του US Burn.

Το Us Burn λειτουργεί υπό Win98/ME/NT/2000/XP/Vista καθώς και Win7. Μετατρέπει τα προγράμματα PIC αποκλειστικά στην Intel Hex8 μορφή. Ένα περιβάλλον ανάπτυξης PIC που χρησιμοποιείται είναι το MPLAB, το οποίο πρόγραμμα είναι και η προεπιλογή του.

5.6.2 Εγκατάσταση.

Το πρόγραμμα είναι διαθέσιμο ως ένα αρχείο ZIP για λήψη μέσα από την επίσημη σελίδα. Το αρχείο του usburnxx.ZIP όνομα περιέχει τα ακόλουθα αρχεία.
· Usburnxx.exe Το κύριο πρόγραμμα στην τρέχουσα έκδοση
· Readme.txt Σύντομος Οδηγός
· Usburn.hlp το αρχείο βοήθειας
· Dll αρχείο Mpusbapi.dll για την πρόσβαση USB (από την Microchip)
· Dll αρχείο Picdef3.dll για τους τύπους PIC που διαχειρίζεται
· Picdef03.dat PIC αρχείου δεδομένων
· Setdef03.dat PIC αρχείου δεδομένων
· Fildef03.dat PIC αρχείου δεδομένων
· Cfgdef03.dat PIC αρχείου δεδομένων
· Cekdef03.dat PIC αρχείου δεδομένων
· Texdef03.dat PIC αρχείου δεδομένων
και το φάκελο:
· Με οδηγό USB (από την Microchip) που περιέχει 6 αρχεία δεδομένων PIC μαζί με Database.

5.6.3 Quick Start.

· Αντιγράφουμε όλα τα αρχεία σε έναν κατάλογο filename.zip
· Σύνδεση με PC Brenner8
· Εγκατάσταση του προγράμματος οδήγησης
· Κάνουμε Run το πρόγραμμα
· Τοποθετήστε τον PIC στην 40πινη υποδοχή
· Επιλέγουμε το παράθυρο του προγράμματος PIC μεγέθους της υποδοχής και της οικογένειας PIC.
· Προσδιορισμός κλειδιού στο PIC Προγραμματιστή
· Επιλέγουμε το HEX αρχείο ως πηγή για το US Burn.
· Εάν είναι απαραίτητο, κάνουμε config ρυθμίσεις
· Γράψτε το αρχείο HEX στο κουμπί PIC.
· Ολοκληρώθηκε

5.6.4 Βασικά Στοιχεία για Καύση του PIC.

Υπεύθυνο για τον προγραμματισμό του PIC είναι μία σειριακή ζεύξη δεδομένων (ICSP - στο κύκλωμα Serial Programming) που αποτελείται από μια γραμμή clock και μιας γραμμής δεδομένων. Επιπλέον, ο PIC απαιτεί μια 5V τάση τροφοδοσίας και μια (περίπου) 12V τάση προγραμματισμού, και φυσικά, την σύνδεση γείωσης. Σε Brenner8 η γραμμή ρολογιού, η γραμμή δεδομένων και η γραμμή απευθείας από 5V δημιουργεί τον έλεγχο του PIC. Η γραμμή τάσης προγραμματισμού ενεργοποιείται με το τρανζίστορ. Ο Burner έχει ένα 5-pin connector ICSP, όλων των 5-burn σημάτων (12V, 5V, η γείωση, τα δεδομένα, ο χρονισμός). Εδώ, ένας προσαρμογέας με υποδοχή για ένα PIC ή ένα καλώδιο που έχουν ήδη κατασκευαστεί σε ένα κύκλωμα για PIC Burning μπορεί να συνδεθεί.
Το Brenner8 έχει μια 40-pin υποδοχή δοκιμής, κατά την οποία όλα τα 14 - και 16-bit πακέτα
μπορούν να προγραμματιστούν με DIL πακέτο.

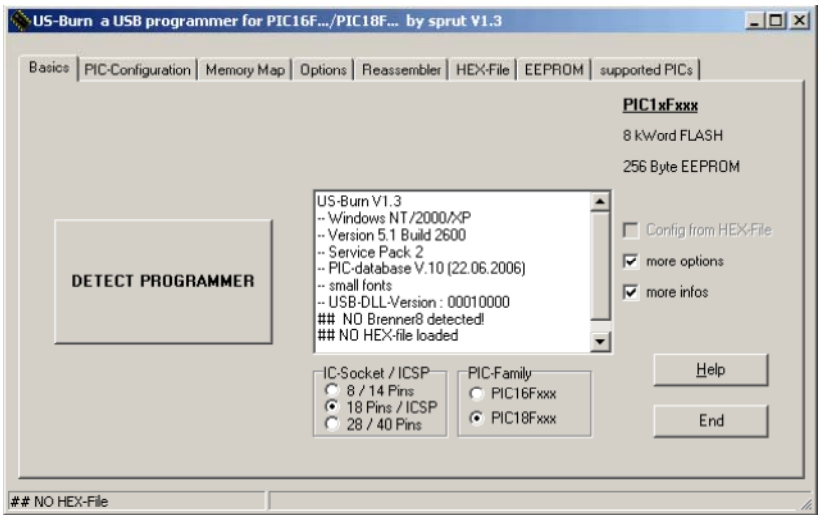
5.6.5 Λειτουργία του προγράμματος.

Όταν ξεκινήσει το πρόγραμμα, ανοίγει τα εγκατεστημένα προγράμματα οδήγησης USB, αναζητά και βρίσκει την Brenner8 και στη συνέχεια συνδέεται με το παράθυρο του προγράμματος του. Αλλά αν υπάρχει κάτι που δεν λειτουργεί, μπορεί να υπάρχει ένα μήνυμα σφάλματος,όπως αναλύουμε παρακάτω:

Έλλειψη USB Drivers

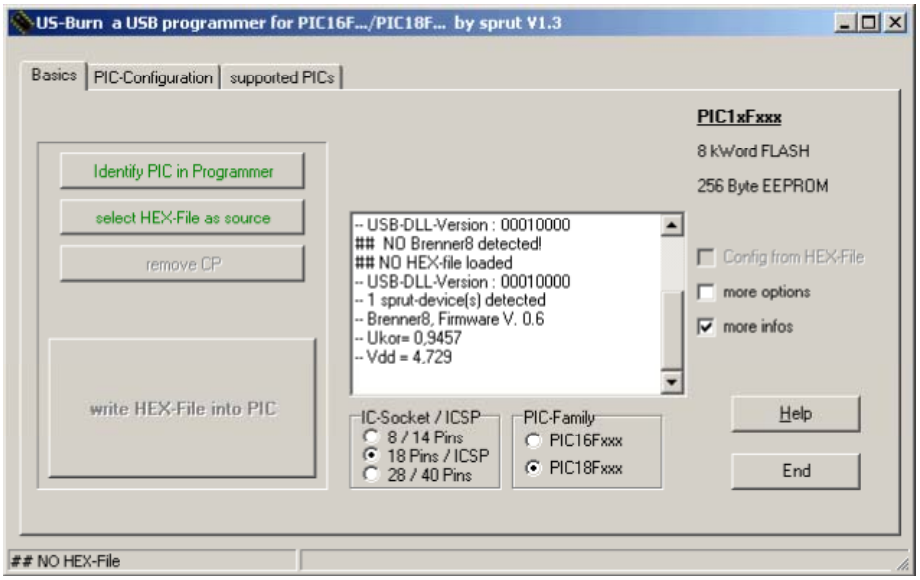
Το πρόγραμμα απαιτεί τον US Burner Κάψτε από την Microchip, στο οποίο περιλαμβάνεται αρχείο ZIP. Αν το πρόγραμμα δεν βρεί το πρόγραμμα οδήγησης, αναφέρει πως
Έχει ξεκινήσει λάθος. Μια εργασία με το πρόγραμμα δεν είναι δυνατή έως ότου ο οδηγός έχει εγκατασταθεί.

Έλλειψη Burner.



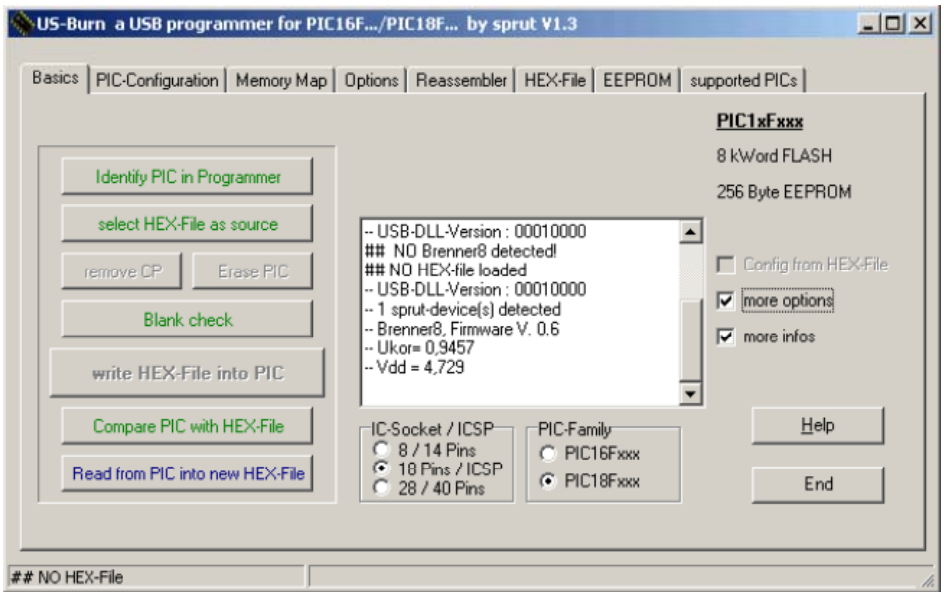
Εικόνα 5.16: Μήνυμα λάθους όταν δεν υπάρχει Burner.

Όταν ξεκινάμε το πρόγραμμα ελέγχου US Burn εμφανίζονται οι θύρες USB σε μια συνδεδεμένη Brenner8 από πίσω. Ως εκ τούτου, ο καυστήρας θα πρέπει να είναι αυτή τη στιγμή η θύρα USB συνδεδεμένη. Αν το πρόγραμμα δεν βρει έναν καυστήρα, αυτός ανταποκρίνεται με παραπάνω παράθυρο: Αφού κάνετε κλικ στο "PROGRAMMER ανίχνευση", το πρόγραμμα επαναλαμβάνει την Αναζήτηση. Είναι τώρα ένας γγραφέας που βρέθηκε (γιατί συνδέθηκε αργότερα) και καταγράφει το κανονικό παράθυρο του προγράμματος. Υπάρχει ένα απλοποιημένο και ένα πλήρες παράθυρο του προγράμματος. Η απλουστευμένη εμφάνιση της εφαρμογής, μόνο στα πραγματικά απαραίτητα χαρακτηριστικά. Αυτό απλοποιεί την λειτουργία του προγράμματος.



Εικόνα 5.17: Απλοποιημένο κύριο παράθυρο του προγράμματος.

Κάνοντας κλικ στο πεδίο περισσότερες επιλογές είναι όλα τα άλλα Ενεργός χαρακτηριστικά του US Burn.



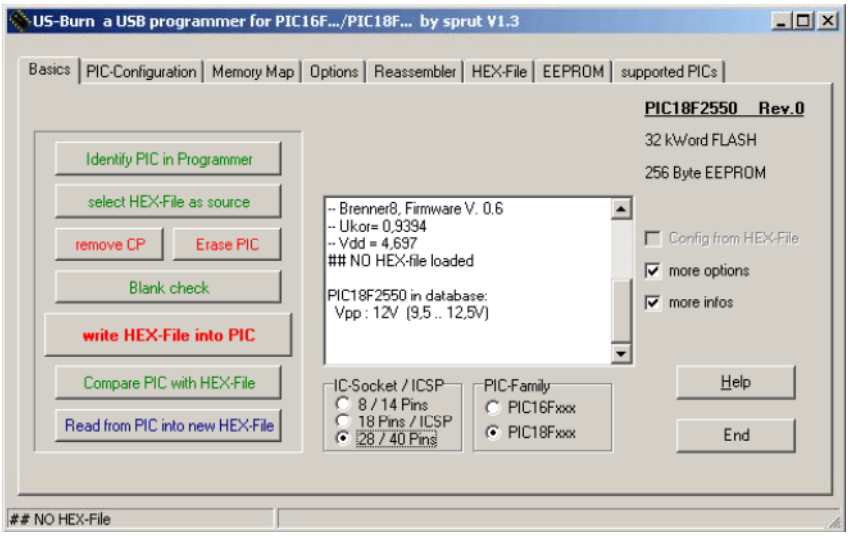
Εικόνα 5.18: Πλήρες κύριο παράθυρο του προγράμματος - NO PIC ακόμη να αναγνωριστεί

Στη σελίδα με τα Βασικά υπάρχει ένα παράθυρο καταγραφής, στην καύση του US σχετικά με τις διαδικασίες και τις Εκδηλώσει που αναφέρθηκαν. Όλες οι γραμμές που ξεκινούν σε αυτό το παράθυρο με το «# #» και αντιπροσωπεύουν προειδοποιήσεις ή μηνύματα λάθους, και θα πρέπει να τηρούνται αυστηρά. Το πρόγραμμα είναι αρκετά εύκολο,σχεδόν σαν να μας μιλάει. Παίρνουμε τις πληροφορίες, στη συνέχεια, απενεργοποιεί το "more info" επιλογή. Το χρώμα του φόντου του παραθύρου αλλάζει Log σε έντονο κόκκινο με λάθη και
επιτυχή εκτόξευση πειράματα σε ανοιχτό πράσινο. Εφ 'όσον ο ακριβής τύπος του επεξεργαστή PIC δεν είναι ακόμη γνωστή, αρνήθηκε το Πρόγραμμα της πρόσβασης εγγραφής στο PIC. Γιατί υπάρχουν μερικά κουμπιά ακόμα με ειδικές επιλογές.

5.6.6 Ρύθμιση του Τύπου PIC.

Όλες οι πληροφορίες σχετικά με τις στρατηγικές προγραμματισμού και διαμορφώσεις από τις φωτογραφίες περιλαμβάνονται σε 6 PIC αρχεία δεδομένων (PIC Database) που είναι αποθηκευμένα στο πακέτο ZIP του US. Το US πιθανών να μην έχει βρει για να καταγράψει τα αρχεία, με αποτέλεσμα να υπάρχει ένα μήνυμα σφάλματος όπως π.χ. από:
missing PIC-database: picdef03.dat
Όσο το US Burn δεν βρίσκει αυτά τα αρχεία, δεν είναι λειτουργικό. Τώρα, η τελευταία κίνηση είναι να προγραμματιστεί το PIC στην υποδοχή χρήσης καυστήρα, ή να συνδεθούμε μέσω του καλωδίου ICSP.

Τότε πρέπει να παρατάξουμε το IC-Socket/ICSP, οπότε το μέγεθος του PIC να είναι συνδεδεμένο στην υποδοχή ICSP. Εάν ένα PIC συνδέεται με ένα καλώδιο ICSP στον καυστήρα, τότε θα πρέπει η 18pin/ICSP να επιλεγεί, δεν έχει σημασία πόσο μεγάλη είναι η PIC. Επίσης, πρέπει να έχουμε ορίσει στο πεδίο Family PIC, είτε πρόκειται για ένα 14-bit πυρήνα PIC (PIC16Fxxx/PIC12F6xx), ένα 16-bit πυρήνα (PIC PIC18FXXX) ή 12-bit πυρήνα PIC (PIC10Fxxx/PIC1xF5xx). Για την 14-bit υπεύθυνοι Series PIC12F6xx είναι η σωστή PIC16Fxx ρύθμιση. Για κάθε 12-bit πυρήνα PIC (επίσης π.χ. PIC16F54) είναι η σωστή PIC10Fxx ρύθμιση. Εδώ αν είναι μια λανθασμένη ρύθμιση είναι επιλεγμένη, ο PIC δεν μπορεί να γραφτεί σωστά. Είναι επίσης δυνατό, αν και πολύ απίθανο ότι μια εσφαλμένη ρύθμιση του PIC μπορεί να υποστεί ζημιά. Το US Burn υποστηρίζει περίπου 200 διαφορετικούς τύπους PIC. Πατώντας το κουμπί προσδιορισμού Προγραμματιστή PIC στο Burn προσδιορίζεται το είδος του PIC στον καυστήρα που χρησιμοποιείται, ή συνδέθηκε μέσω καλωδίου ICSP.
Αυτό ελέγχει επίσης τη λειτουργία του καυστήρα. Αν η PIC δεν ανιχνευθεί, τότε υπάρχει μια δυσλειτουργία, και το PIC μπορεί για λόγους ασφάλειας να διαγραφούν τα δεδομένα του. Μόνο στην οικογένεια PIC "PIC10Fxxx» (δηλαδή το 12-bit πυρήνα PIC) δεν λειτουργεί, και έτσι θα πρέπει πατώντας το κουμπί "Προσδιορισμός PIC προγραμματιστή" σε ένα
Πλαίσιο ελέγχου μπορεί να επιλεγεί χειροκίνητα.



Σχήμα 5.19: Αναγνωρισμένος PIC- πλήρη κύριο παράθυρο του προγράμματος.

Προσδιορίζοντας το PIC προγραμματιστή η λειτουργία εντοπίζει, επίσης, αν η μνήμη προγράμματος (FLASH) και η μνήμη δεδομένων (EEPROM) του αντιγράφου προστατεύεται (προστασία Κώδικα). Ένα αντίγραφο που προστατεύεται δεν μπορεί ούτε να «καεί» ούτε να διαβαστεί. Σε αυτές τις περιπτώσεις, η Συγκρίνετε τα χαρακτηριστικά και την καύση να αποτύχει. Η λειτουργία Ανάγνωσης διαβάζει μια σειρά από προστασίας με κωδικό μόνο μηδενικά. Εδώ, ο PIC θα διαγραφεί ταυτόχρονα.

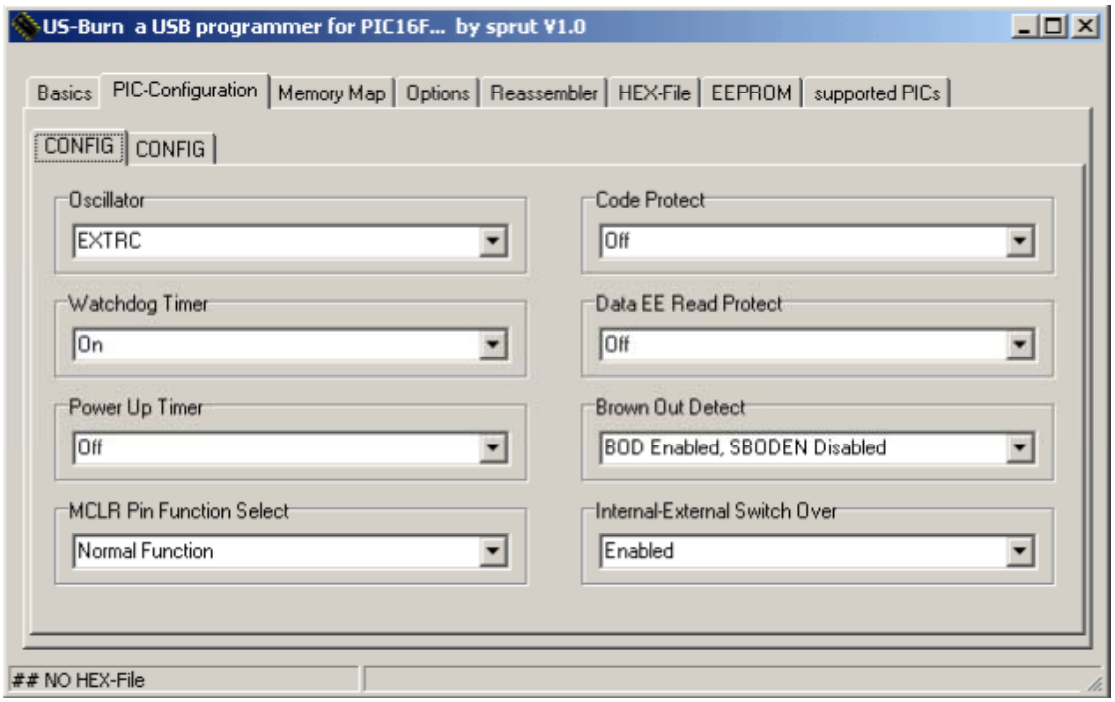
Αν ο PIC αναγνωρίζεται, τότε το παράθυρο καταγραφής και η αναγκαία προγραμματιστική τάση να κάψει το PIC εμφανίζεται. Το πρόγραμμα είναι ρυθμισμένο αυτόματα έτσι ώστε να δημιουργεί μια σωστή τάση προγραμματισμού.

5.6.7 Μεταμόρφωση αρχείου Hex.

Η επιλογή αρχείου HEX ως πλήκτρο πηγής (δεύτερος από πάνω) εμφανίζει ένα παράθυρο διαλόγου και ανοίγει, με την οποία κάποιος επιλέγει το αντικείμενο που πρέπει να καεί στο αρχείο HEX PIC. Εκεί το US ανοίγει και καίει τον κατάλογο που χρησιμοποιήθηκε τελευταία. Συχνά, το HEX αρχείο περιέχει ρυθμίσεις για την PIC. Η US Burn βρίσκει πληροφορίες αυτές στην HEX αρχείο, και η φόρτωση του αρχείου υιοθετείται αυτόματα. Δεν θα πρέπει να χρησιμοποιούν αυτά τα δεδομένα διαμόρφωσης, η ρύθμιση μπορεί να αλλάξει χειροκίνητα εάν το αντίστοιχο σήμα ελέγχου (Config από το αρχείο HEX) αφαιρεθεί κάνοντας κλικ. Το US Burn ελέγχει τις ρυθμίσεις που περιέχονται στο HEX αρχείο αληθοφάνειας. Αν υπάρχουν σφάλματα που διαπιστώθηκαν, η προειδοποίηση είναι επίσης λέξη διαμόρφωσης πάνω στο σφάλμα. Προσπαθούμε να ρυθμίσουμε το σφάλμα στην αυτόματη διόρθωση, και στη συνέχεια επιλέγουμε Config από HEX χρήστη deaktiviert.Der αρχείου, σε αυτή την περίπτωση, η ρύθμιση παραμέτρων χρησιμοποιώντας το PICConfiguration ελέγχουμε το παράθυρο πριν γίνει η καύση του PIC.

5.6.8 Διαμόρφωση του PIC Programmer.

Εάν η διαμόρφωση του καυστήρα δεν φορτώθηκε με το hex αρχείο, ή αν πρέπει να τροποποιήσουμε το φορτίο, κάτι που μπορεί να ρυθμιστεί με το χέρι. Το Config πεδίο από HEX αρχείο δεν πρέπει να είναι ενεργό (δεν υπάρχει σημάδι ελέγχου). Όλες οι κάθε PIC τύπου πιθανές ρυθμίσεις ρυθμίζονται στη PIC σελίδα διαμόρφωσης που είναι διαθέσιμη και μπορεί να αλλάξει. Η εμφάνιση του παραθύρου ρύθμισης παραμέτρων προσδιορίζεται με τον τύπο του PIC. Ενδέχεται νααποτελούνται από πολλά παράθυρα. Εφ 'όσον δεν καθορίζεται ο τύπος PIC που έχει ανιχνευθεί, το μήνυμα this page μείνει σκόπιμα κενό.



Εικόνα 5.20: Παράθυρο Διαμόρφωσης.

Το κάψιμο ενός PIC χωρίς ρυθμίσεις σπάνια οδηγεί σε ένα λειτουργικό PIC. Οι προεπιλεγμένες ρυθμίσεις συνήθως επιλέγουν έναν εξωτερική RC ταλαντωτή και την ενεργοποίηση του Watchdog Timer.

5.6.9 Καύση του PIC.

Πατώντας το αρχείο HEX γράφουμε στο κουμπί PIC για να ξεκινήσουμε τη διαδικασία εγγραφής, και στην συνέχεια μας δείχνει την πρόοδο με τη γραμμή προόδου. Αν έχει διαβάσει ένα HEX αρχείο προηγουμένως θα πρέπει να σβηστεί, και ύστερα τοποθετείται ένα νέο αρχείο, φρέσκο »μπορεί να διαβαστεί, και στη συνέχεια να επιλέξουμε καύση στο PIC. Μετά την καύση, το αποτέλεσμα ελέγχεται, και το αποτέλεσμα της δοκιμής στο παράθυρο εξόδου.

5.6.10 Σύγκριση του PIC με το HEX Αρχείο.

Αφού πατήσουμε το συγκρίνουμε με το κουμπί HEX PIC αρχείο, το περιεχόμενο να είναι ίδιο με του PIC σε σύγκριση με το αρχείο που έχει φορτωθεί HEX. Οι εναλλακτικές ρυθμίσεων της διάταξης που έχουν επιλεχθεί θα χρησιμοποιηθούν, και όχι από το HEX αρχείο. Αυτό βέβαια δεν ισχύει για τις ρυθμίσεις OSCCAL. Αφού η σύγκριση είναι στο παράθυρο καταγραφής, ο αριθμός των σφαλμάτων που διαπιστώθηκαν εμφανίζεται. Αυτό δεν λειτουργεί με φωτογραφίες με κωδικό προστασίας.

5.6.11 Διαγραφή του PIC.

Αφού πατήσουμε το κουμπί Διαγραφή, το περιεχόμενο του PIC θα διαγραφεί. Ανάλογα με τον τύπο του PIC, συνήθως δεν διαγράφονται τα πάντα, αλλά σίγουρα σε κάθε περίπτωση η Μνήμη προγράμματος. Αν πραγματικά θέλετε να διαγράψετε ό, τι κατόρθωσε να αφαιρέσει από το CP. Το US Burn θα καταγράψει το hex πρόγραμμα του μικροελεγκτή στη θέση που είναι σβησμένη από την προηγούμενη φορά.

.

5.6.12 Λευκός Έλεγχος του PIC.

Αφού πατήσουμε το πλήκτρο Check Blank ελέγχεται κατά πόσο η PIC έχει καθαριστεί σωστά. Το US Burn σε κάθε περίπτωση, στις περιοχές μνήμης που πρόκειται να προγραμματιστεί πριν κάνει την καύση και πάλι, καθαρίζει αυτό το τεστ για την καύση από τις φωτογραφίες δεν είναι απαραίτητες. Η εξέταση είναι απαραίτητη μόνο αν έχουμε χρησιμοποιήσει τη γνωστοποίηση σε τρίτους και έχει διαγραφεί έτσι ώστε να μην περάσει τον κωδικό μας λάθος.

5.6.13 Διαβάζοντας το PIC.

Αφού πατήσουμε το Read από το αρχείο HEX pic σε νέο κουμπί, το περιεχόμενο του PIC διαβάζεται αυτόματα. Ολόκληρη η μνήμη του προγράμματος καθώς και τα δεδομένα EEPROM της περιοχής και η διαμόρφωση της ταυτότητας θα διαβαστεί. Οι μόνες αναγνωρίσιμες και προς ανάγνωση τιμές μπορούν να αποθηκευτούν ως αρχείο HEX, Το πρόγραμμα προσπαθεί να ανιχνεύσει κενές περιοχές αποθήκευσης και ο PIC θα τις αγνοήσει. Αν θέλουμε να αντιγράψουμε τα περιεχόμενα του PIC σε άλλες PIC, θα πρέπει το HEX αρχείο να αποθηκευτεί χειροκίνητα την δεύτερη φορά.

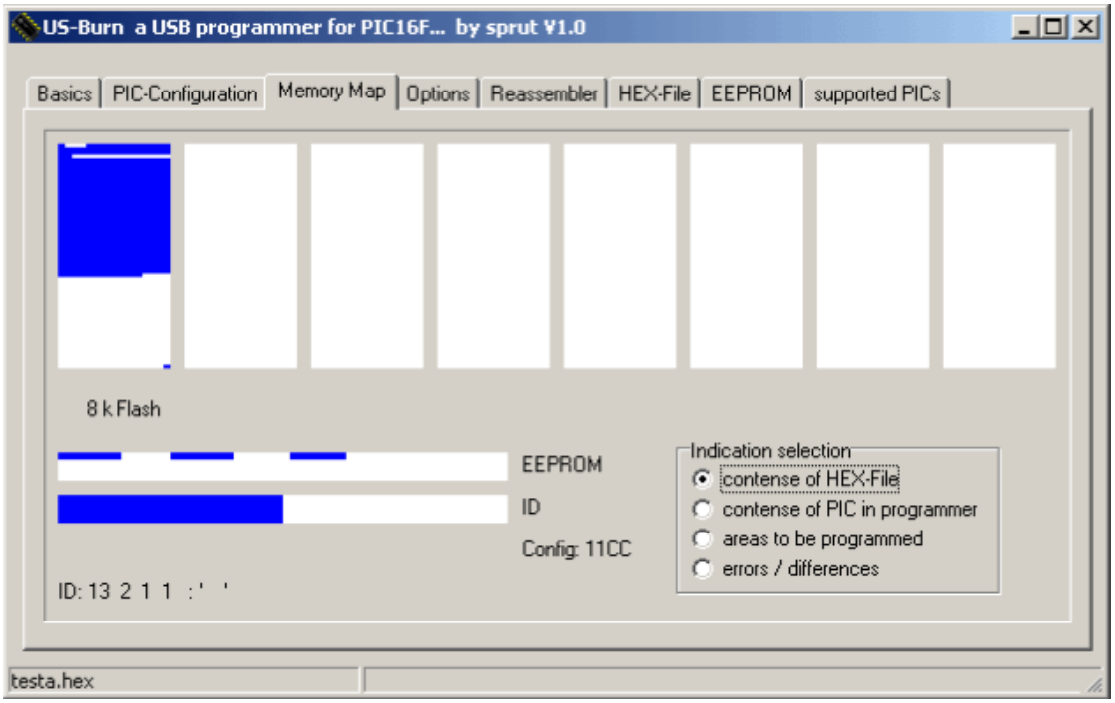
5.6.14 Κωδικός για προστασία Κατάργησης.

Αφού πατήσουμε το κουμπί Κατάργησης για την CP μνήμη διαβάζουμε την προστασία του PIC και καταργείται. Την ίδια στιγμή, το σύνολο του PIC θα διαγραφεί. Εάν ένας PIC προστατεύεται από κωδικό προστασίας, μπορεί να εξηγηθεί από την ταυτότητα και να καθορίσει τη λειτουργία του προγραμματιστή.

5.7 Επιπλέον Χαρακτηριστικά. [4]

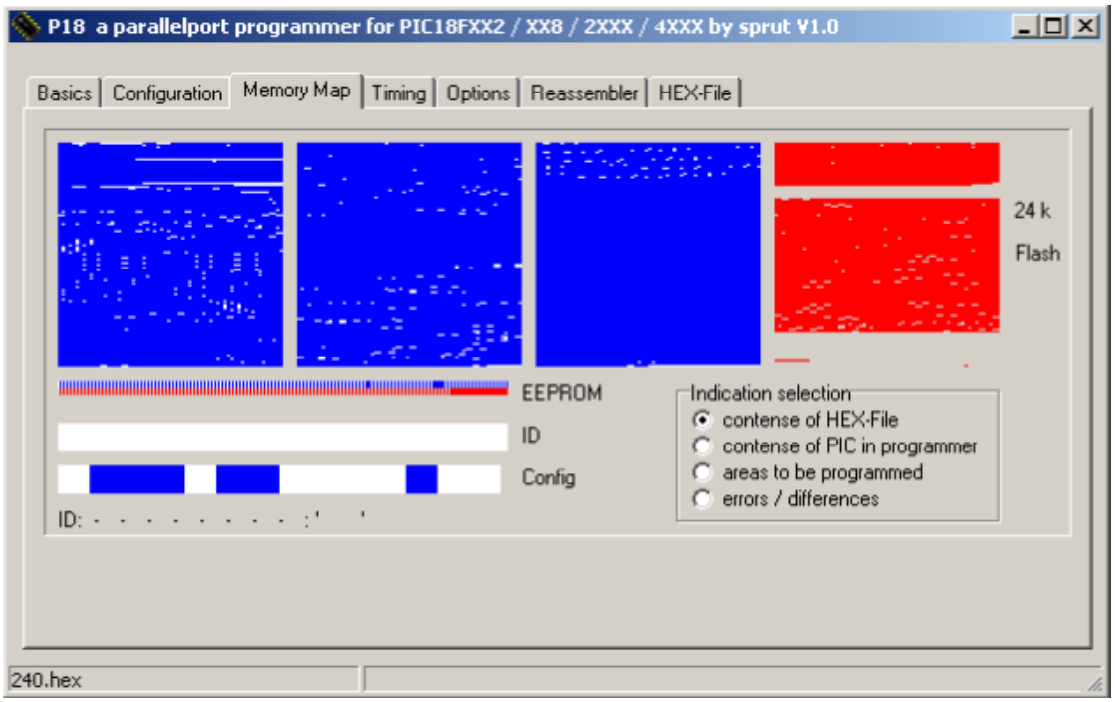
5.7.1 Γραφική απεικόνιση μνήμης.

Στο US Burn υπάρχει μία γραφική απεικόνιση της χρήσης της μνήμης του PIC
Η οθόνη μπορεί να εναλλάσσεται μεταξύ:
1 Το περιεχόμενο του αρχείου HEX
2 Περιεχόμενο του PIC
3 Τις περιοχές του PIC για την καύση
4 Οι διαφορές μεταξύ των PIC και HEX αρχείο
Τα σημεία 2 και 4 μας εμφανίζονται μόνο μετά την ανάγνωση του PIC για να διορθώσει τις πληροφορίες. Η οθόνη είναι όμως μέχρι 8 μπλοκ. Ανάλογα με τον τύπο που χρησιμοποιείται από φωτογραφίες αυτά τα κομμάτια είναι διαφορετικά. Αν ο δείκτης του ποντικιού σε ένα μπλοκ μνήμης μετατοπίζεται, στη συνέχεια, εκτός από την περιοχή διευθύνσεων του δρομέα του ποντικιού το Μπλοκ μνήμης εμφανίζεται.



Εικόνα 5.21 : Γραφική οθόνη με μία φορτωμένη Μνήμη προγράμματος.

Η γραφική οθόνη απεικονίζει τη μνήμη που χρησιμοποιείται και την διαθέσιμη μνήμη PIC. Οι περιοχές που εμφανίζονται της μνήμης αντιστοιχούν πάντα στην FLASH και EEPROM μέγεθος των εντοπισμών. Οι αχρησιμοποίητες περιοχές μνήμης εμφανίζεται στα λευκά. Ο κώδικας του προγράμματος είναι φορτωμένος και εμφανίζεται σε μπλε χρώμα.

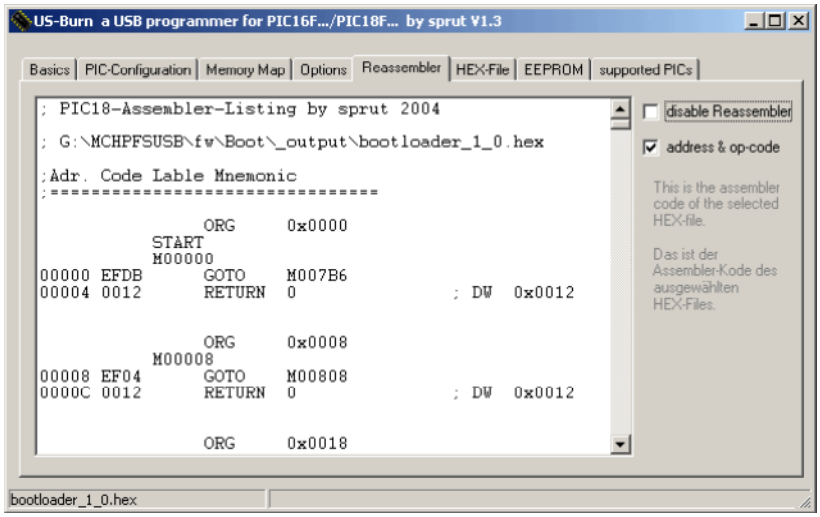


Εικόνα 5.22: Γραφική απεικόνιση μνήμης - το πρόγραμμα και τα δεδομένα EEPROM.

Αν τμήματα του κώδικα που περιέχονται στο HEX αρχείο, ή των δεδομένων δεν είναι η
FLASH και EEPROM της τρέχουσας που ταιριάζει στον PIC, οι μη αντιστοίχιση δεδομένων
εμφανίζεται με κόκκινο χρώμα. Επίσης, διαπιστώνουμε ότι κατά τη διάρκεια των διαφορών καίγονται ή να συγκρίνει μεταξύ hexfile και PIC περιεχομένου (π.χ. σφάλματα καύσης) εμφανίζονται με κόκκινο χρώμα.

5.7.2 Reassembler.

Μόλις ένα HEX αρχείο έχει επιλεγεί στο κεντρικό παράθυρο του προγράμματος,
αποσυναρμολογούμε το περιεχόμενο του προγράμματος του HEX αρχείου.
Το αποτέλεσμα μπορεί θα εμφανιστεί στο παράθυρο reassembler.
Βρείσκουμε το reassembler το οποίο θα είναι με προφανές σφάλμα κώδικα (άγνωστος εντολές, μεταβαίνει στο πουθενά), ο αριθμός των λαθών στο κύριο παράθυρο εμφανίζεται. Δεν είναι πάντα εύκολο για τον κώδικα του προγράμματος και των δεδομένων του reassembler να διακρίνει το ένα το άλλο, ειδικά όταν το άλμα μέσα από τους υπολογισμούς που εκτελούνται στο μητρώο PC. Οι περιοχές μνήμης που απαιτούνται για την Reassembler διατηρούν δεδομένα να συγκεντρωθούν, αλλά εδώ αγνοούνται σφάλματα κωδικού ένα "DW 0x ....» και επισυνάπτεται ως ένα σχόλιο για κάθε γραμμή.

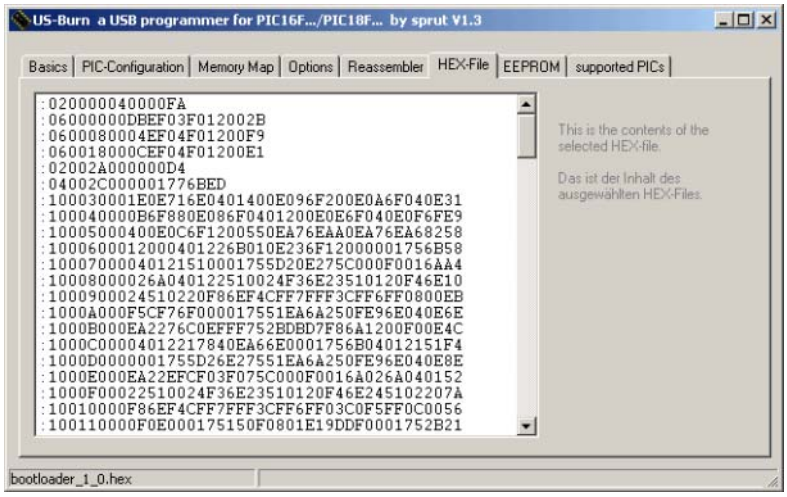


Εικόνα 5.23: Παράθυρο Reassembler.

Η ολοκληρωμένη reassembler μπορεί να απενεργοποιηθεί (απενεργοποίηση reassembler). δεδομένου ότι η Reassembler κατά τη φόρτωση ενός αρχείου HEX καθώς και το κάψιμο
επανασυνδέονται σε όλο τον κώδικα, μπορεί να είναι μια καθυστέρηση για τους αργούς υπολογιστές που έχουν εκτελέσει το έργο. Η reassembler μπορεί να οδηγήσει σε μηνύματα λάθους προγράμματος, εάν ο κώδικας που χρησιμοποιείται, λειτουργεί ακόμα στην US Burn
Δεδομένου ότι η reassembler μπορεί να επιβραδύνει το έργο. Κάτω από 16-bit των Windows (Windows 98/ME) είναι το μέγεθος της προβολής κειμένου για 64kbyte
περιορισμένο. Ως εκ τούτου, σύμφωνα με αυτά τα λειτουργικά συστήματα για μεγάλα αρχεία, ο ανακατασκευασμένα κωδικός δεν εμφανίζεται.

5.7.3 Ανάλυση Αρχείων HEX.

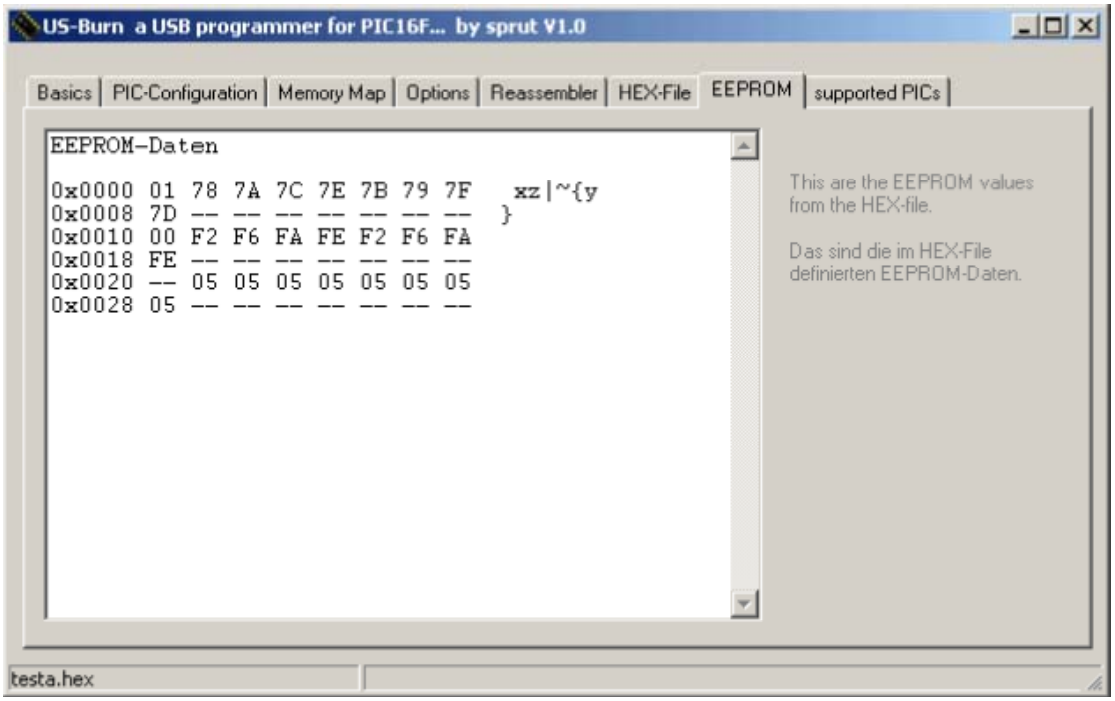
Μόλις ένα HEX αρχείο έχει επιλεγεί στο κεντρικό παράθυρο του προγράμματος, μπορούμε να προβάλουμε αυτό το HEX αρχείο στο παράθυρο της προβολής αρχείων. Η επεξεργασία του HEX αρχείου δεν είναι δυνατή Κάτω από 16-bit των Windows (Windows 98/ME) είναι το μέγεθος της προβολής κειμένου για 64kbyte περιορισμένο. Ως εκ τούτου, σύμφωνα με αυτά τα λειτουργικά συστήματα ένα εξαιρετικά μεγάλο αρχείο HEX δεν εμφανίζεται.



Εικόνα 5.24: Παράθυρο του αρχείου HEX.

5.7.4 Λειτουργεία EEPROM Παραθύρου.

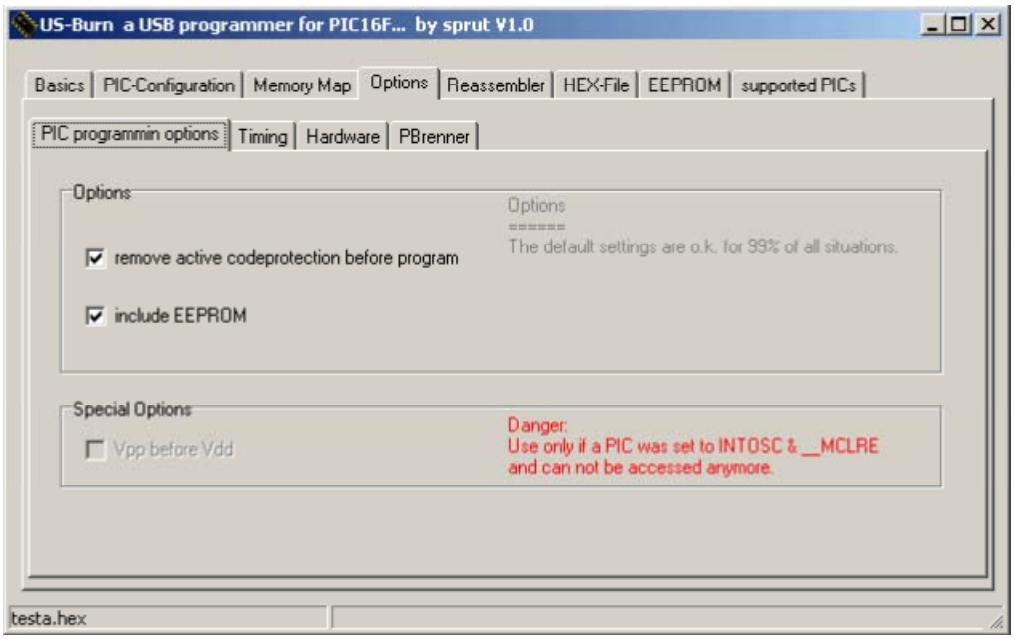
Μόλις ένα HEX αρχείο έχει επιλεγεί στο κεντρικό παράθυρο του προγράμματος, μπορούμε να δούμε το αρχείο HEX που περιέχονται στα δεδομένα EEPROM στο παράθυρο EEPROM. Οι διευθύνσεις EEPROM εμφανίζονται στο αρχείο HEX, και περιλαμβάνονται εκτός από τις δεκαεξαδικές τιμές και τα δεδομένα EEPROM. Επίσης, είναι δυνατή η παραγωγή χαρακτήρες ASCII.



Εικόνα 5.25: Παράθυρο EEPROM.

5.7.5 Επιλογές-Options.

Μερικές επιλογές είναι πάντα ενεργοποιημένες κατά την εκκίνηση. Στο παράθυρο Επιλογές μπορούν να είναι είτε on ή off και αυτό δεν είναι συνήθως απαραίτητο. Άρα αποθηκεύονται οι επιλεγμένες ρυθμίσεις.



Εικόνα 5.26: Παράθυρο Options.

5.7.5.1 Απομάκρυνση του Κωδικού Προστασίας πριν από το Πρόγραμμα.

Αν αυτή η επιλογή είναι ενεργοποιημένη, τότε ελέγχεται κατά την διαδικασία της καύσης, αν ο PIC κώδικας είναι προστατευμένος. Αν ισχύει αυτή η περίπτωση, η προστασία κωδικού διαγράφεται ολόκληρη στο PIC πριν από την καύση. Στις περισσότερες περιπτώσεις, αυτό έχει νόημα. Αλλά το US Burn δεν ελέγχει κατά πόσο απλά οι περιοχές του κώδικα PIC προστατεύονται. Εάν κάποιος έχει για παράδειγμα ένα PIC στην οποία τον κωδικό της περιοχής ή δεδομένων είναι προστατευόμενος και θέλουμε να αποκτήσουμε αυτά τα δεδομένα και τους πρόσθετους κωδικούς / στοιχεία για να κάψει σε άλλους τομείς.

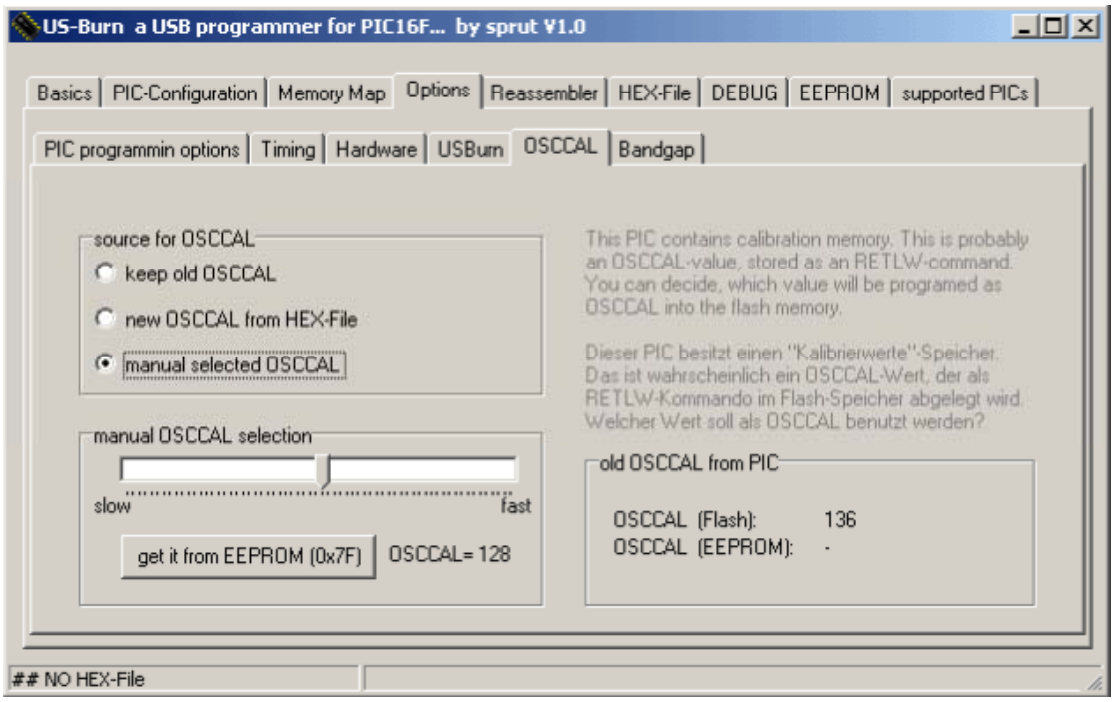
5.7.5.2 Vpp πριν την Vdd.

Αυτή η επιλογή αντιστρέφει την ακολουθία ενεργοποίησης της λειτουργίας και τάσης προγραμματισμού του PIC. Αυτό είναι σε αντίθεση με την περιγραφή του κατασκευαστή και έχει ως εκ τούτου δυνητικούς κινδύνους για το PIC. Εάν ένα PIC προγραμματιστεί με τις επιλογές και τις INTOSC χωρίς MCLR pin ενεργοποιημένη, τότε μπορεί να συμβεί ότι αυτός δεν μπορεί πλέον να προγραμματιστούν. Τότε, και μόνο τότε, μπορούμε να επιλέξετε αυτή την επιλογή και στη συνέχεια δοκιμάζουμε το chip για διαγραφή. Αν η διαγραφή λειτουργεί, θα πρέπει να απενεργοποιήσουμε την επιλογή. Κατά την εκκίνηση

του προγράμματος, αυτή η επιλογή είναι πάντα απενεργοποιημένη για λόγους ασφαλείας.

5.8 OSCCAL-Editor. [4]

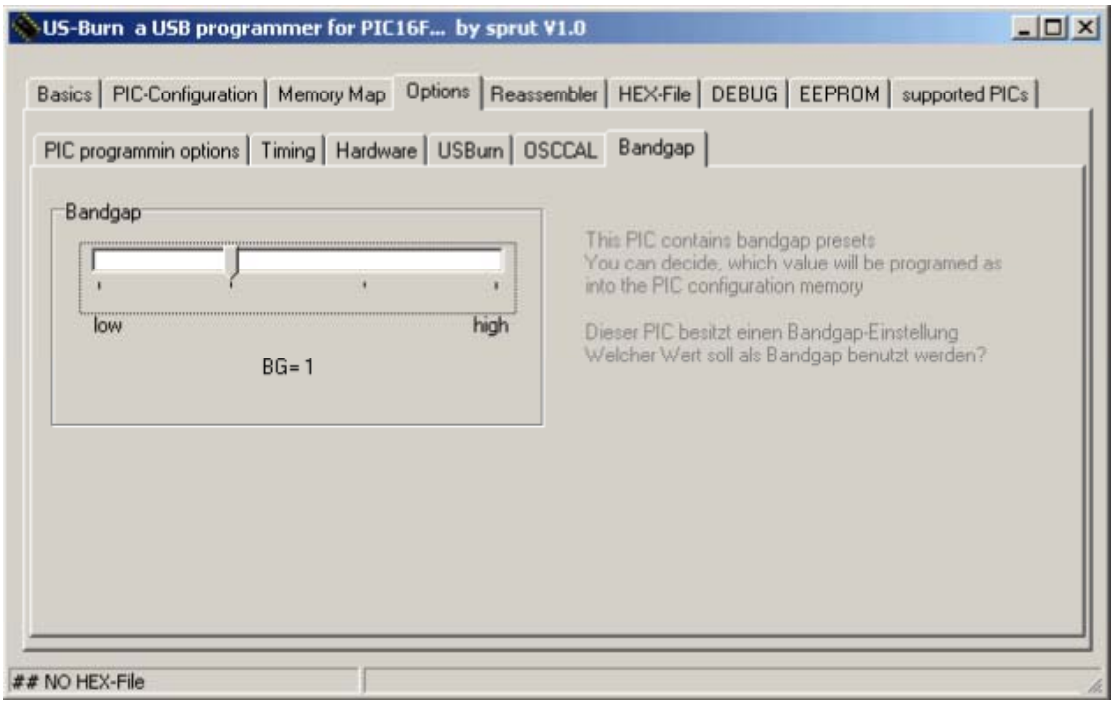
Κάποιοι programmers έχουν μια τιμή βαθμονόμησης για τον εσωτερικό ταλαντωτή. Αυτό το OSCCALWert είναι στην flash μνήμη, π.χ. 0x3ff στη διεύθυνση ως μέρος ενός RETLWBefehls που αποθηκεύονται. Τέτοιος επαναπρογραμματισμός του PIC OSCCAL έχει τιμή στη μνήμη flash και παραμένει. Αλλά δεν μπορούμε ποτέ να προσδιορίσουμε την αξία του προγραμματισμού, αλλά τη στιγμή που θα χαθεί. Στη συνέχεια, με τον OSCCAL OSCCAL επεξεργαστή μια νέα τιμή πρέπει να καθοριστεί. Η OSCCAL είναι μια αριθμητική τιμή μεταξύ 0 και 255 και κανονικά είναι περίπου στο κέντρο, δηλαδή στα 128. Στην περίπτωση αυτή, η σωστή τιμή είναι συνήθως στην Flash, αλλά ορισμένα προγράμματα είναι σε άλλη θέση σε ένα νέο OSCCAL το οποίο τέθηκε από την EEPROM. Αλλά αυτό είναι απλώς ένα αντίγραφο ασφαλείας. Στον κάτω μέρος του κώδικα η εικόνα, για παράδειγμα, έχει βρεθεί στην 136 του μνήμη flash, ένα OSCCAL. Τίποτα δεν στέκεται στην EEPROM. Στην πηγή για το παράθυρο OSCCAL μπορούμε να αποφασίσουμε ποια από τις OSCCAL αξίες θα πρέπει να χρησιμοποιείται. Συνήθως, διατηρούν παλιά OSCCAL το δικαίωμα ρύθμισης. Στην παρούσα περίπτωση, ένα PIC φλας χρησιμοποιείται στη συνέχεια. Εάν αυτό χάνεται (όπως στο παράδειγμα), τότε μπορούμε να πάρουμε μια OSCCAL από τις χρησιμοποιούμενες (αν κάποιος είναι στο HEX αρχείο), ή η OSCCAL το HEX σε ένα αρχείο.



Εικόνα 5.27: Σύνταξη OSCCAL.

5.9 Διάκενο Ζώνης Έκδοσης. [4]

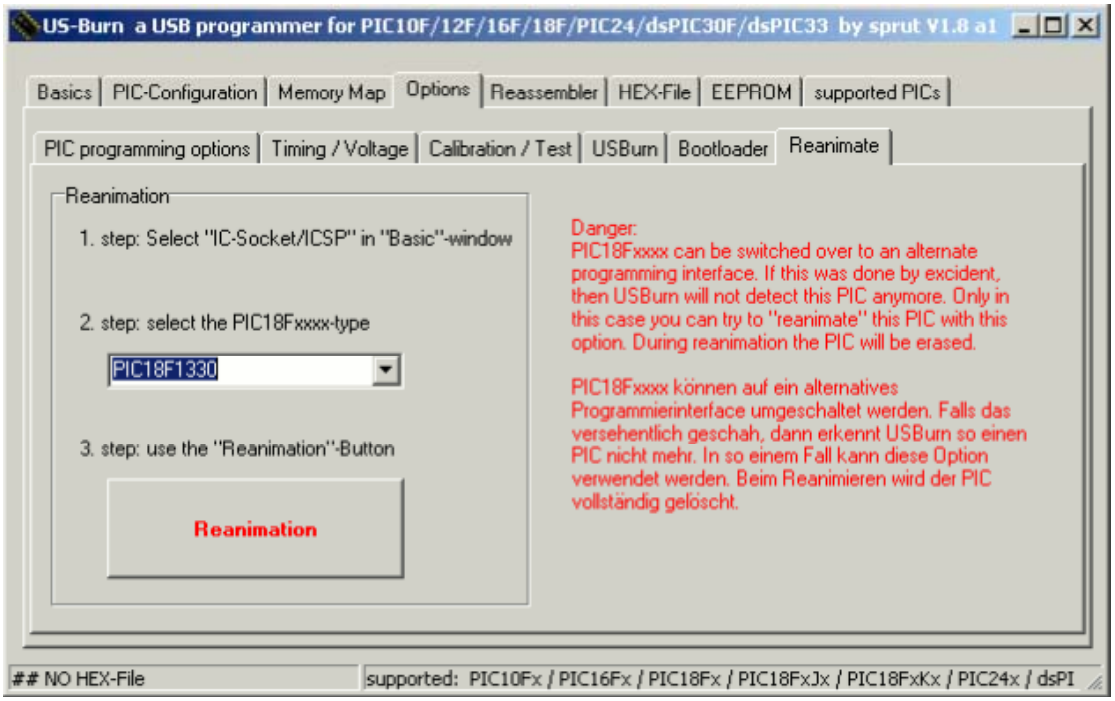
Κάποιοι κώδικες έχουν μια τιμή βαθμονόμησης για την εσωτερική διάκενο ζώνης της πηγή τάσης. Αυτή η τιμή χάσματος είναι αποθηκεμένη στη διαμόρφωση του PIC. Ο επαναπρογραμματισμός των εν λόγω υπεύθυνων διάκενων ζώνης αξίας διατηρείται. Αλλά δεν μπορούμε να αποκλείσουμε ότι η αξία χάνεται κατά τον προγραμματισμό. Στη συνέχεια, μπορεί να ρυθμιστεί μία νέα τιμή διάκενου ζώνης editor. Όταν πατήσουμε «εντοπίσε PIC Προγραμματιστής από την τρέχουσα τιμή διάκενου ζώνης»
Διαβάζει το πρόγραμμα τον PIC, και το ρυθμιστικό βρίσκεται σε αυτήν την τιμή. προς την
επόμενη περίοδο προγραμματισμού (να γράψουμε το αρχείο HEX σε PIC) έχει διαφορετική αξία από pic σε pic. Επιτρέπονται τα κατοικίδια, μεταξύ του εκτοπίσματος του slider και τον προγραμματισμό του PIC. Εάν μετά τον προγραμματισμό μια φορά κανουμε κλικ στο Συγκρίνετε με το αρχείο HEX PIC, τότε φυσικά, ένα σφάλμα στη διαμόρφωση αναφέρεται ως η μεταβολή στην BGWert στη διαμόρφωση του PIC και η αξία του αρχείου HEX συμπίπτουν.



Εικόνα 5.28:Διάκενο Ζώνης Έκδοσης.

5.10 Ανάκληση. [4]

Μέσα από μια ιδιαιτερότητα ορισμένων τύπων PIC18Fxxxx, είναι δυνατόν ότι ένα PIC ενεργοποιείται με την καύση με ένα λανθασμένη διαμόρφωση σε μια λειτουργία κατά την οποία δεν είναι πλέον ανταποκρίσιμη προς τον καυστήρα. ("ICD / ICSP Port Enable bit"), η Brenner αναφέρει στη συνέχεια, κάνοντας κλικ στον PIC προγραμματιστή εντοπίσει ένα νέο Αυτό το αναγνωριστικό δεν περιλαμβάνεται στη βάση δεδομένων. Εάν τα φαινόμενα αυτά με PIC συμβούν, ο οποίος αρχικά δουλεύει μια χαρά (και μόνο σε αυτή την περίπτωση, μπορεί να αντισταθμίσει την PIC12/16/18 με την επιλογή ανάνηψης επανέλθει σε κανονική λειτουργία. Εδώ, το PIC θα διαγραφεί.



Εικόνα 5.29: Παράθυρο Ανάκλησης.

Για να γίνει ανάκληση πρέπει πρώτα το παράθυρο να είναι στη βασική επιλογή για ICSocket/ICSP και επιλεγμένο. Τότε είναι η επιλογή - παράθυρο animation-και επιλέγουμε το σωστό τύπο PIC. Στη συνέχεια το κουμπί ανάκλησης μπορεί να πατηθεί. Το USBurn τώρα προσπαθεί να επαναφέρει στη ζωή τους διακόπτες του PIC στο βασικό παράθυρο και ξεκινά αυτόματα Προσδιορισμό προγραμματιστή PIC. Αν λειτουργεί σωστά, τότε η PIC αναγνωρίζεται τώρα και πάλι. Αν όχι, τότε έχουμε το πρόβλημα της μη αναγνώρισης για άλλη αιτία. Το US Burn εμποδίζεται από την έκδοση V1.8 στην οποία ένα PIC με μια τέτοια εσφαλμένη διαμόρφωση είναι προγραμματισμένος έτσι ώστε αυτό το πρόβλημα ελπίζουμε στο μέλλον Χάνει το νόημα.

5.11 Παράμετροι της Γραμμής Εντολών. [4]

Το US Burn κατά την εκκίνηση εκτελεί επιχειρήματα της γραμμής εντολών διαδοχικά, τότε η σωστή σειρά παρατηρείται. Σε όλες τις επιλογές προηγείται το "/". Μέσα σε μια επιλογή δεν είναι επιτρεπόμενο το Space. Τουλάχιστον ένα κενό διάστημα μεταξύ των επιλογών πρέπει να σταθεί με άνω και κάτω τελεία. Αν μια επιλογή έχει μια παράμετρο, τότε
ακολουθεί τις εντολές χωρίς κενά Μια τυπική κλήση του προγράμματος θα μπορούσε να μοιάζει με αυτό:
usb19.exe /S18 /F18 /P /Ic:\testfiles\test.hex /p /w /x
Οι επιλογές στη γραμμή εντολών μπορούν να χωριστούν σε τέσσερις διαδοχικές ομάδες:
· Με / S / F / P τύπου PIC και τη σύνδεσή του με τον καυστήρα να καθοριστεί.
· Με / I είναι το όνομα του hex αρχείο καθορίζεται (μόνο αν είναι απαραίτητο ή / και w / c που ακολουθούν).
· Με / p, / e, / b / w / c η πραγματική δράση προσδιορίζεται.
· Με / x, το πρόγραμμα τερματίζεται.

Επιλογές με παραμέτρους:
/ S. .. (PIN)-pin count (μόνο Brenner8) ή ICSP υποδοχή
/ F. .. (ΟΙΚΟΓΕΝΕΙΑ) οικογένεια PIC
/ I. .. (ΣΕ) το όνομα του αρχείου HEX (μεγάλες I) (για Write & σύγκριση)
Επιλογές χωρίς παραμέτρους
/ P (PIC) αναγνωρίζει τον τύπο PIC (πρωτεύουσα P)
/ P (αφαίρεση cp) αφαιρείται από την προστασία κωδικό PIC (μικρό p)
/ E (διαγραφή) delete PIC
/ B (κενό) ελέγχει αν η PIC είναι άδειο
/ W (write), κάψιμο HEX αρχείο στο PIC
/ C (σύγκριση) συγκρίνετε PIC ικανοποιημένοι με το HEX αρχείο
/ X (end) τερματίζει το πρόγραμμα και δημιουργεί το αρχείο καταγραφής...

/ S (Πρίζα) Pin Count (μόνο Brenner8) ή ICSP σύνδεση
Έτσι ώστε ο καυστήρας μπορεί να ελέγξει το PIC στην υποδοχή δοκιμής του Brenner8 σωστά θα πρέπει να γνωρίζει πόσες πινέζες που έχει ή αν είναι συνδεδεμένο στην υποδοχή ICSP. Πιθανές παράμετροι στην υποδοχή ελέγχου είναι 8, 14, 18, 28 και 40. Αυτός είναι ο αριθμός των ακίδων του pic DIL πακέτου. Είναι το PIC, αλλά συνδέεται με την υποδοχή ICSP είναι να επιλέξετε πάντα τις ICSP παραμέτρους. Σημαντικό είναι ότι μόνο PICs με 14-bit πυρήνα (PIC16F. .. και κάποια PIC12F ...) και 16-bit Πυρήνας (PIC18F. ..) μπορούν να καούν στην υποδοχή δοκιμής. Απαιτείται από όλες τις άλλες η υποδοχή ICSP. Στη συνέχεια, με την επιλογή / F, μια οικογένεια PIC θα επιλεγεί, η οποία καίει αποκλειστικά μέσω του συνδετήρα ICSP (PIC18FxxJxx, PIC24, dsPIC30, dsPIC33), τότε ο καυστήρας έχει οριστεί σε ICSP, ακόμη και αν υπάρχει άλλη επιλογή που η πρίζα χρησιμοποιείται.

Παράδειγμα:
usb19.exe /SICSP .....
usb19.exe /S28 .....

/ F Οικογένεια PIC. Υπάρχουν μια σειρά από ριζικά διαφορετικές οικογένειες επεξεργαστών PIC. Ο US Burn απαιτεί οι πληροφορίες να ανήκουν σε κάποια οικογένεια ο PIC. Αυτό Ρυθμίζεται με τις παραμέτρους της επιλογής / F.
Παράδειγμα:
usb19.exe /SICSP /F30 .....
usb19.exe /S28 /F16 .....

/ P καθορισμός τύπου PIC. Μετά την υποδοχή (/ S) και την οικογένεια που έχει οριστεί (/ F), τότε αυτή η επιλογή επιλέγει το αυτόματο του προσδιορισμού. Μόνο μετά το σωστό εντοπισμό του τύπου μπορεί να έχει επιπλέον λειτουργίες (διαγραφή, καύση ...) και οι οποίες χρησιμοποιούνται. Για το λόγο αυτό, πρέπει ο / P να έχει δυνατότητα σωστής ανάκλασης στις / S / F που ακολουθούν. Παράδειγμα:
usb19.exe / SICSP / F30 / P .....
usb19.exe / S28 / F16 / P .....

/ I (IN) Το όνομα του αρχείου HEX. Το US Burn μπορεί να βάλει το HEX αρχείο ναδιαβάσει (για το / και w / c). Αυτή η επιλογή καθορίζει ένα όνομα αρχείου που είναι αποκλειστικά και μόνο για το αρχείο εισόδου (για Όπτηση είναι με / W και χρησιμοποιείται για τη σύγκριση με το / c). Το HEX αρχείο δεν είναι στον ίδιο κατάλογο με USBurn, τότε το εξάγωνο όνομα αρχείου για την αναζήτηση ακολουθεί ολόκληρη τη διαδρομή.Παράδειγμα:
usb19.exe / SICSP / F30 / P / Ic: test.hex .....
usb19.exe / S28 / F16 / P / Ic: test.hex .....
/ P (αφαίρεση cp). Αφαιρείται από την προστασία κωδικού PIC. Ένας ενεργοποιημένος κωδικός προστασίας PIC είναι απενεργοποιημένος, ενώ το φλας Μνήμης προγράμματος και τα δεδομένα της μνήμης EEPROM διαγράφονται. Η διαγραφή της Διαμόρφωση και το user ID δεν είναι όλοι οι τύποι Επίτευξη PIC. Μία πλήρης διαγραφή ωστόσο εξασφαλίζεται από το συνδυασμό / και p / s. Παράδειγμα:
usb19.exe / SICSP / F30 / P / Ic: test.hex / p .....
usb19.exe / S28 / F16 / P / Ic: test.hex / p .....

/ E (διαγραφή) delete PIC. Το PIC θα διαγραφεί. Αυτή η επιλογή δεν λειτουργεί για όσο διάστημα η προστασίας κωδικού είναι ενεργοποιημένη. Σε αυτή την περίπτωση θα πρέπει επίσης να χρησιμοποιηθεί πριν από τις / e / p. Πριν από την πυροδότηση, η επιλογή αυτή δεν είναι απαραίτητη, επειδή πριν από την καύση του PIC διαγράφεται αυτόματα ούτως ή άλλως. Παράδειγμα:
usb19.exe / SICSP / F30 / P / Ic: test.hex / e ……..
usb19.exe / S28 / F16 / P / Ic: test.hex / p / e ……..

/ W (write) κάψετε το HEX αρχείο στο PIC. Τα περιεχόμενα ενός αρχείου HEX (το οποίο ορίστηκε σε / I πριν) περιέχονται σε καύση στο PIC. Η επιλογή / w προκαλεί αυτόματα διαγραφή του PIC, και την απομάκρυνση των Κωδικών προστασίας πριν από την καύση και μια δοκιμή Burning. Καθορισμός της / p / e και / c δεν είναι επομένως απαραίτητη. Παράδειγμα:
usb19.exe / SICSP / F30 / P / Ic: test.hex / w .....
usb19.exe / S28 / F16 / P / Ic: test.hex / w .....

/ C (σύγκριση) συγκρίνει τη PIC με το HEX αρχείο. Το περιεχόμενο του PIC με τα HEX περιεχόμενα του αρχείου ενός αρχείου (με το οποίο / I στο παρελθόν έχει οριστεί) σε σύγκριση και εμφανίζεται σφάλμα. Παράδειγμα:
usb19.exe / SICSP / F30 / P / Ic: test.hex / c .....
usb19.exe / S28 / F16 / P / Ic: test.hex / c .....

/ X (end) Τερματίζει το πρόγραμμα και δημιουργεί το αρχείο καταγραφής. Αυτή η επιλογή διασφαλίζει ότι USBurn τερματίζεται. Για να εξασφαλιστεί ότι όλα τα ζητήματα της καταγραφής στο παράθυρο μετά είναι ακόμη διαθέσιμα, ένα αρχείο καταγραφής (usburnlog.txt) είναι που παράγει τα περιεχόμενα του παραθύρου καταγραφής.
Προκειμένου να διευκολύνει την αυτόματη ανάλυση των αρχείων καταγραφής, ξεκινά η
usburn.log αρχείο με μία γραμμή είτε σε "OK" ή "FAIL". Εάν υπάρχει "FAIL" τότε έχει μία από τις λειτουργίες του καυστήρα (αναγνωρίζοντας τις PICTypes,Burning, συγκρίσεις) είχε ως αποτέλεσμα ένα μήνυμα σφάλματος. Αν η USBurn κάλεσε χωρίς την τελική επιλογή /X παραμένει ,το Πρόγραμμα είναι ενεργό και μπορεί να λειτουργεί κανονικά. Ένα αρχείο καταγραφής δεν θα είναι έτοιμο προς εργασία επειδή ο χρήστης μπορεί να δει ακόμη και το παράθυρο καταγραφής των USBurn. Παράδειγμα:
usb19.exe / SICSP / F30 / P / Ic: test.hex / w / x
usb19.exe / S28 / F16 / P / Ic: test.hex / w / x.

5.12 Πιθανά Προβλήματα και Λύσεις. [4]

5.12.1 Άγνωστη συσκευή.

Σύμπτωμα:
Τα Windows αναγνωρίζουν αυτόματα το βύσμα του καυστήρα σε μια συσκευή στον υπολογιστή, αλλά δεν μπορεί να προσδιοριστεί. Κατά συνέπεια, έχουν εγκατασταθεί και δεν έχει κατάλληλο προγράμμα οδήγησης.
Αιτία:
Ίσως ο χρόνος ανάδρασης του PIC δεν είναι αλήθεια. Είδος χαλαζία ή / και Config ρύθμιση του ελέγχου PIC θα πρέπει να ξαναελεγχθούν.

5.12.2 Αναξιόπιστη λειτουργία.

Σύμπτωμα 1:
Ο US Burn ανιχνεύεται στην πρώτη σύνδεση με τον υπολογιστή και το πρόγραμμα οδήγησης να έχει εγκατασταθεί. Ο καυστήρας έχει επανασυνδεθεί αργότερα, και δεν αναγνωρίζεται.
Σύμπτωμα 2:
Ο καυστήρας ανιχνεύεται. Για μεγαλύτερες ενέργειες (ανάγνωση, εγγραφή) διακόπτεται η
Καύση λογισμικού. Αν δεν ανταποκρίνεται με την επανεκκίνηση λογισμικού, το σφάλμα είναι στο είδος του μηνύματος «997^{η}»
Αιτία:
Ο πυκνωτής 220nF για Vusb pin του φόρου PIC είναι αποσυνδεδεμένος. Πρέπει να
ελέγξουμε τον πυκνωτή σύνδεσης.
Σύμπτωμα 3:
Το πράσινο LED ανάβει. Στη συνέχεια, μπορεί να δει κανείς πλέον καύση του PIC.
Εάν ο καυστήρας έχει αποσυνδεθεί από το καλώδιο USB και, στη συνέχεια επανασυνδέεται
πάλι μετά από κάποιο χρονικό διάστημα η πράσινη λυχνία LED αναβοσβήνει.
Αιτία:
Ο ελεγκτής Vpp δημιουργεί μερικές φορές μια (φαινομενική) Κατά τη διάρκεια τάσης, ένα
Κύκλωμα ασφαλείας ενεργοποιείται. Ο καυστήρας πρέπει να επαναρυθμιστεί.
Σύμπτωμα 4:
Το Brenner8 ανιχνεύεται, USBurn αναγνωρίζει τον καυστήρα. Connected PICTyp σωστά. Αλλά κατά την έναρξη της πυροδότησης 997 έρχεται ένα λάθος
Αιτία:
Ο έλεγχος Vpp της Brenner8 έχει αποτύχει, το PIC για αυτό το είδος προγραμματίζεται μέσα σε εύλογο χρονικό διάστημα (περίπου 1 s), και με αρκετή ακρίβεια. Ο καυστήρας πρέπει να επαναρυθμιστεί στηνΣυνήθως αρκεί να επαναλάβετε το βήμα 3 της βαθμονόμησης.

5.12.3 Απουσία Μεμονωμένων Σημάτων.

Σύμπτωμα 1:
Το σήμα ρολογιού ή δεδομένων, ή Vdd του πρωτοκόλλου ICSP λείπει.
Αιτία:
Ο έλεγχος PIC δεν πιέστηκε σε IC υποδοχή του.

5.12.4 Λανθασμένη Z Τάσης.

Σύμπτωμα 1:
Η τάση Zener είναι μόνο 0.7V
Αιτία:
Η δίοδος Zener είναι κολλημένη ανάποδα. Η Zener είναι σειρά διόδων.
Σύμπτωμα 2:
Η τάση Zener είναι πολύ κάτω από την αναμενόμενη τάση Z-διόδου.
Αιτία:
Η αντίσταση σειράς της διόδου Zener έχει μια λανθασμένη (πολύ υψηλή) τιμή. Ελέγξτε την αντίσταση.

5.12.5 Ανεπαρκής Τάση Προγραμματισμού.

Σύμπτωμα:
Ο προγραμματισμός τάσης Vpp είναι πολύ χαμηλός. Μόλις φτάσει στο μέγιστο 12V, η βαθμονόμηση δεν είναι δυνατή.
Αιτία:
Οι δίοδος D1 χρησιμοποιεί μια κανονική δίοδο Si. Αν η D1 είναι μια δίοδο Schottky προς
(BAT43) πρέπει να αντικατασταθεί.

5.12.6 USB Error SE: 100

Την στιγμή κατά την καύση εμφανίζεται ένα μικρό παράθυρο σφάλματος USB και στο παράθυρο καταγραφής βλέπουμε να είναι "USB Σφάλμα SE: 100". Στην περίπτωση αυτή, ο καυστήρας δεν έχει εντός προκαθορισμένου χρόνου για να αποκρίνεται σε μια εντολή. Μπορεί να υπάρξει μια κατέρρευση σύνδεσης USB που οφείλεται σε ένα πρόβλημα υλικού, αλλά συχνά η αιτία είναι στη βαθμονόμηση. Ειδικότερα, εάν το μήνυμα στην αρχή της επιστολής του PIC ή λαμβάνει χώρα κατά την έναρξη της ανάγνωσης από το PIC, τότε η αιτία είναι πολύ αργή στον έλεγχο της τάσης προγραμματισμού Vpp. Σε αυτή την περίπτωση, επαναλαμβάνουμε το στάδιο 3 της βαθμονόμησης. Στη συνέχεια, το πρόβλημα θα πρέπει να να μην εμφανίζεται.

10.12 Απεγκατάσταση US Burn.
Για να απεγκατασταθεί εντελώς το πρόγραμμα και να αφαιρεθεί από τον υπολογιστή θα πρέπει να αφαιρέσουμε τα εξής αρχεία:
· Xxx \ usburn *. Exe
· Xxx \ mpusbapi.dll
· Xxx \ mpusbapi.dll
· Xxx \ picdef3.dll
· Xxx \ * 03.dat
· Xxx \ usbrn.hlp
· Xxx \ usburn.ini
και το πρόγραμμα οδήγησης USB επίσης για να απεγκατασταθεί .
Είναι, xxx \ 'για τον κατάλογο εγκατάστασης του US Burn.

Βιβλιογραφία κεφαλαίου

[1]Μικροελεγκτές PIC" Σταμάτης Αλατσαθιανός, Β.Γκιούρδας Εκδοτική
[2]Wakerly J. F., Digital Design: Principles and Practices, 2nd Edition, (Chap. 11),
[3]http://www.pi-schools.gr/lessons/tee/electronic/biblia.php
[4] http://www.sprut.de/electronic/pic/projekte/brenner8/index.htm#smd8p9
[5]http://www.electronics-lab.com/pic-in-greek/