ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

Σχεδίαση και υλοποίηση αναπτυξιακού κυκλώματος-ως-εμπορικού-προϊόντος για τον μικροελεγκτή της σειράς PIC 16F877 της εταιρίας Microchip

Πτυχιακή εργασία

του Αθανάσιου Δημούδη (2449)

Επιβλέπων: Δρ. Ιωάννης Α. Καλόμοιρος, Επίκουρος Καθηγητής

ΣΕΡΡΕΣ, ΜΑΙΟΣ 2015

Υπεύθυνη Δήλωση : Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής Τ.Ε. του Τ.Ε.Ι. Κεντρικής Μακεδονίας.

ΠΕΡΙΛΗΨΗ

Οι μικροελεγκτές PIC της εταιρίας Microchip είναι ευρύτατα διαδεδομένοι στη βιομηχανία και την εκπαίδευση. Για την εκμάθηση των μικροελεγκτών και την ανάπτυξη εφαρμογών υπάρχουν στο εμπόριο διάφορες αναπτυξιακές πλακέτες. Πολλές από αυτές χαρακτηρίζονται από πολυπλοκότητα, καθώς προσπαθούν να καλύψουν μεγάλο εύρος εφαρμογών ή/και μικροελεγκτών πάνω στο ίδιο κύκλωμα. Άλλες, πάλι, στηρίζονται σε εργαλεία τρίτων εταιριών, που δεν είναι συμβατά με το ελεύθερο λογισμικό της εταιρίας Microchip. Στο θέμα αυτό πτυχιακής εργασίας προτείνεται ο σχεδιασμός και η υλοποίηση ενός βασικού αναπτυξιακού κυκλώματος για την ανάπτυξη εκπαιδευτικών εφαρμογών για τον μικροελεγκτή PIC16F877. Ο προγραμματισμός των μικροελεγκτών στο κύκλωμα γίνεται με την τεχνική In-Circuit-Programming και χρήση του προγραμματιστή PICkit3, μέσα από το ελεύθερο αναπτυξιακό περιβάλλον MPLAB της Microchip.

Αρχικά τέθηκαν προδιαγραφές και στη συνέχεια υλοποιήθηκε ένα πρωτότυπο σε ράστερ. Κατόπιν, σχεδιάστηκε το κύκλωμα σε κατάλληλο σχεδιαστικό περιβάλλον και δημιουργήθηκαν τις μάσκες για την κατασκευή του τυπωμένου κυκλώματος (PCB). Για το ζητούμενο αυτό αναζητήθηκαν και να μελετήθηκαν εναλλακτικές λύσεις σχεδιαστικών εργαλείων και να εγκαταστάθηκε το φιλικότερο. Το PCB δημιουργήθηκε από εξειδικευμένο εργαστήριο της αγοράς. Τέλος, κολλήθηκαν τα εξαρτήματα πάνω στην πλακέτα και έγιναν οι τελικές δοκιμές του αναπτυξιακού κυκλώματος. Τα αναπτυξιακά κυκλώματα που υλοποιήθηκαν θα ενταχθούν στην εκπαιδευτική διαδικασία του Τμήματος.

Περιεχόμενα

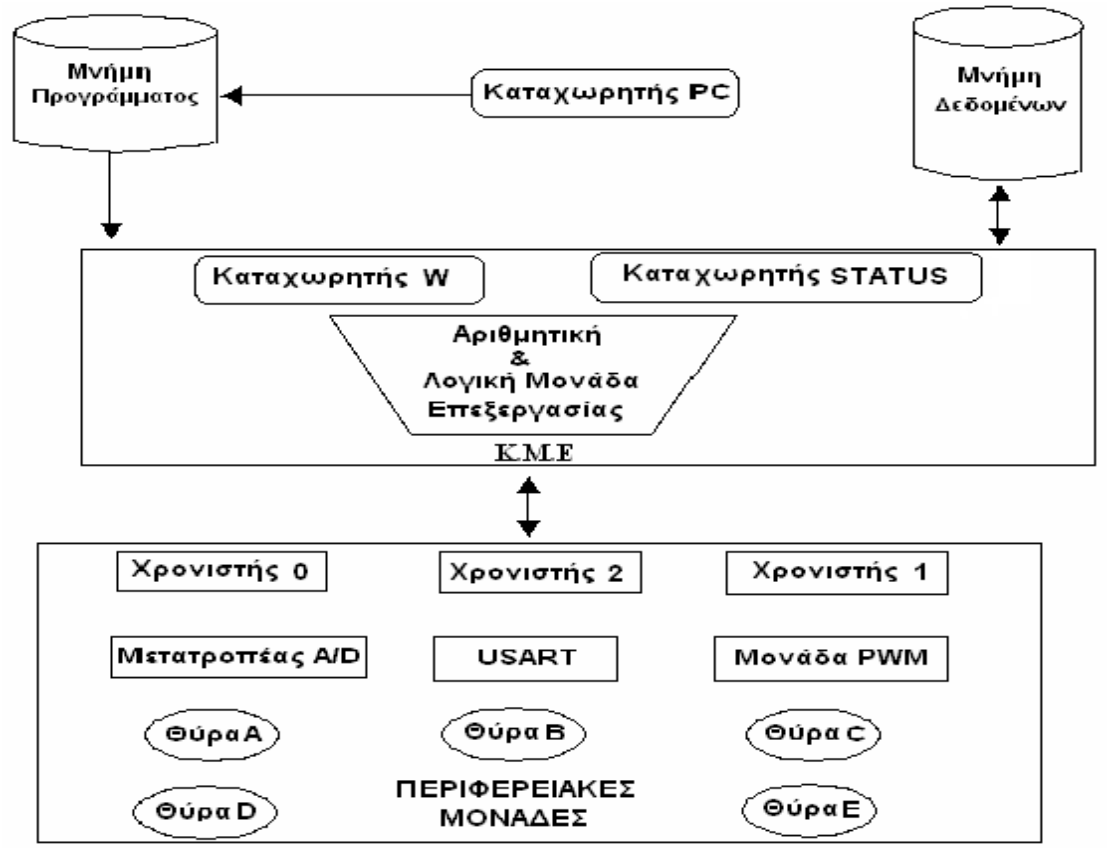
Κεφ. 1 Εισαγωγή στην αρχιτεκτονική και τον προγραμματισμό του PIC16F877

1.1 Αρχιτεκτονική του μικροελεγκτή PIC16F877

Βασικό χαρακτηριστικό της αρχιτεκτονικής πολλών μικροελεγκτών, που δεν απαντάται στους συνηθισμένους μικροεπεξεργαστές, είναι ότι έχουν διαφορετικό διάδρομο για τις εντολές (instructions bus) και διαφορετικό διάδρομο δεδομένων (data bus), για όλα τα υπόλοιπα δεδομένα και αποτελέσματα της επεξεργασίας. Η αρχιτεκτονική αυτή αναφέρεται και ως αρχιτεκτονική Harvard. Ας σημειωθεί εδώ ότι οι συνηθισμένοι μικροϋπολογιστές είναι δομημένοι σύμφωνα με τη λεγόμενη αρχιτεκτονική von Neumann. Σε αυτούς, το πρώτο βήμα για την εκτέλεση μιας λογικής ακολουθίας είναι να καταχωρηθούν στη μνήμη RAM οι εντολές του προγράμματος, μέσω μίας περιφερειακής μονάδας (πληκτρολόγιο ή μαγνητική μνήμη). Το πρόγραμμα, δηλαδή, καταλαμβάνει ένα μέρος στην ίδια μνήμη που διατίθεται επίσης για τα δεδομένα και τα αποτελέσματα της επεξεργασίας. Όταν διακοπεί η τροφοδοσία, τα δεδομένα της μνήμης RAM χάνονται, μαζί με το πρόγραμμα. Την επόμενη φορά ο μικροπεξεργαστής μπορεί να φορτώσει στη μνήμη RAM και να επεξεργαστεί ένα διαφορετικό πρόγραμμα. Οι διάφορες μονάδες του υπολογιστικού συστήματος επικοινωνούν παράλληλα μέσω των ίδιων διαδρόμων δεδομένων, διευθύνσεων και ελέγχου.

Ένα χαρακτηριστικό που κάνει έναν μικροελεγκτή, όπως τον PIC16F877, ιδιαίτερα ελκυστικό και κατάλληλο για απλές εφαρμογές είναι η ιδιότητα της μνήμης του προγράμματος να διαγράφεται και να επαναπρογραμματίζεται μέσω ηλεκτρικών σημάτων. Με άλλα λόγια, η μνήμη προγράμματος είναι *ηλεκτρικά διαγραφόμενη και προγραμματιζόμενη* (electrically erasable – programmable) και γι'αυτό ονομάζεται Flash EEPROM (η λέξη flash δηλώνει ότι η μνήμη λειτουργεί με υψηλή ταχύτητα κατά τη διαγραφή και τον προγραμματισμό). Δεν απαιτείται, δηλαδή, κάποιο ακριβό μηχάνημα διαγραφής των περιεχομένων της μνήμης με τη βοήθεια υπεριώδους ακτινοβολίας, όπως συμβαίνει με άλλες μνήμες. Έτσι, το ίδιο ολοκληρωμένο κύκλωμα μπορεί να χρησιμοποιηθεί για την ανάπτυξη του πρωτότυπου κυκλώματος και του πιλοτικού προγράμματος, καθώς και για την τελική υλοποίηση του

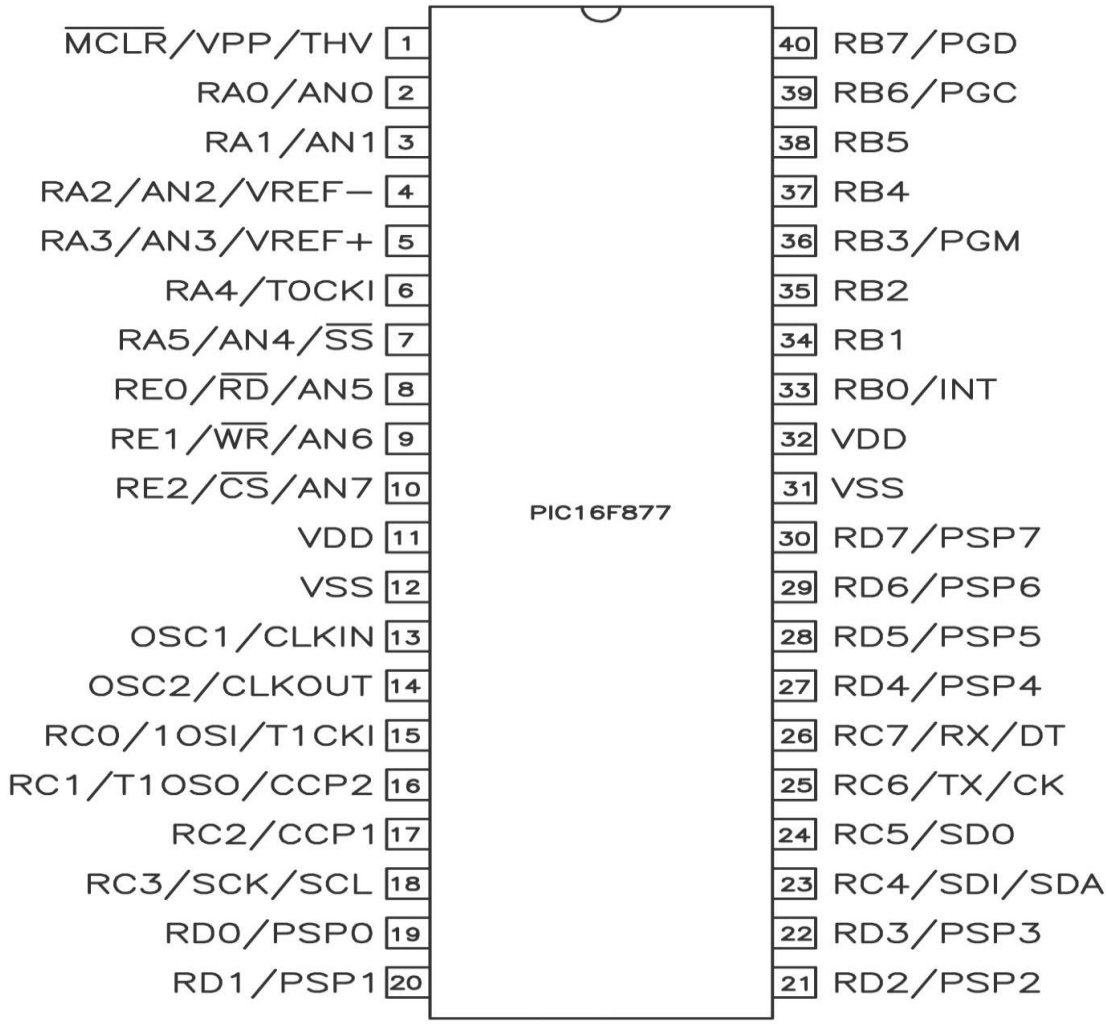
κυκλώματος που θα τεθεί σε χρήση ή και θα παραχθεί μαζικά. Η διαγραφή και ο προγραμματισμός του ολοκληρωμένου μπορούν να γίνουν χιλιάδες φορές. Επιπλέον, ο μικροελεγκτής PIC16F877 έχει τη δυνατότητα να προγραμματίζεται μέσα από εξαιρετικά απλούς προγραμματιστές, δηλαδή κυκλώματα που από τη μια μεριά συνδέονται στη σειριακή ή στη θύρα USB ενός προσωπικού υπολογιστή και από την άλλη συνδέονται σε ορισμένους από τους ακροδέκτες (pins) του μικροελεγκτή, προκειμένου να προγραμματίσουν τη μνήμη EEPROM.



Εικόνα 1.1.1 Βασική Δομή του PIC

Όπως όλοι οι μικροελεγκτές PIC, το μοντέλο 16F877 ενσωματώνει την αρχιτεκτονική Harvard, της οποίας όπως είδαμε, βασικό χαρακτηριστικό είναι ο διαφορετικός διάδρομος εντολών και δεδομένων και η συνακόλουθη διαφορά των μνημών προγράμματος και καταχωρητών/μεταβλητών. Η αρχιτεκτονική των μικροελεγκτών PIC υπακούει στους κανόνες της λεγόμενης αρχιτεκτονικής RISC. Ο διάδρομος εντολών έχει εύρος 14 bits. Κάθε εντολή αποτελείται από μία λέξη 14 bits και συνεπώς εκτελείται σε έναν και μοναδικό κύκλο εκτέλεσης, κάτι που αποτελεί βασικό χαρακτηριστικό της αρχιτεκτονικής RISC. Ο διάδρομος δεδομένων έχει εύρος

8 bits. Ο μικροελεγκτής μπορεί να προγραμματιστεί με 35 συνολικά εντολές και με τη βοήθεια δεκαεπτά καταχωρητών ειδικού σκοπού, που αντιστοιχούν σε συγκεκριμένες θέσεις της μνήμης RAM. Κάθε εντολή εκτελείται σε τέσσερις συνολικά κύκλους του εξωτερικού ωρολογιακού σήματος, που συνιστούν έναν κύκλο εκτέλεσης ή κύκλο εντολής (αποκωδικοποίηση εντολής, ανάγνωση τελεστέου καταχωρητή, επεξεργασία δεδομένων και εγγραφή αποτελεσμάτων στον τελικό προορισμό). Έτσι, εάν ο εξωτερικός ταλαντωτής είναι ένας κρύσταλλος με ιδιοσυχνότητα 4MHz, τότε κάθε εντολή θα εκτελείται σε χρόνο 1 μs (=1/1MHz).



Εικόνα 1.1.2 Ακροδέκτες του PIC16F877

Ο μικροελεγκτής PIC16F877 διαθέτει 368 bytes μνήμης RAM, όπου ο χρήστης μπορεί να ορίσει μεταβλητές και να αποθηκεύσει δεδομένα. Επίσης έχει τριαντατρείς ακροδέκτες εισόδου/εξόδου, μοιρασμένους σε πέντε θύρες, που ονομάζονται PORTA

(6 ακροδέκτες) και PORTB (8 ακροδέκτες), PORTC (8 ακροδέκτες), PORTD (8 ακροδέκτες) και PORTE (3 ακροδέκτες), αντίστοιχα. Διαθέτει τρεις χρονιστές, που λειτουργούν και ως απαριθμητές (Timer0, Timer1, Timer2) κι έναν ακόμη, εσωτερικά ταλαντούμενο χρονιστή, που επιτηρεί και επαναφέρει τον μικροελεγκτή, σε περίπτωση που βρεθεί εκτός ελέγχου. Ο τελευταίος χρονιστής ονομάζεται WDT ή Watchdog Timer.

Η μνήμη προγράμματος (EEPROM) του μικροελεγκτή 16F877 έχει χωρητικότητα 2K ή 2048 bytes. Εκτός από τη μνήμη αυτή, ο 16F877 διαθέτει άλλα 64 bytes μνήμης EEPROM, τα οποία προορίζονται για τη μόνιμη αποθήκευση δεδομένων.

Χαρακτηριστικό	PIC16F877
Μέγιστη συχνότητα λειτουργίας (MHz)	20
Μνήμη προγράμματος Flash (14-bit λέξεις)	8K
Μνήμη δεδομένων (bytes)	368
EEPROM Μνήμη δεδομένων(bytes)	256
Πόρτες Εισόδου/Εξόδου	RA0-5 (6 ακροδέκτες) RB0-7 (8 ακροδέκτες) RC0-7 (8 ακροδέκτες) RD0-7 (8 ακροδέκτες) RE0-2 (3 ακροδέκτες)
Χρονιστές	3
Σειριακή επικοινωνία	USART
Παράλληλη επικοινωνία	PSP (Parallel Slave Port)
10-bit Αναλογική/Ψηφιακή μετατροπή	8 κανάλια
Σύνολο εντολών	35
Ακροδέκτες	40

Πίνακας 1.1.3 Βασικά συστατικά του PIC

1.2 Οι εντολές των μικροελεγκτών PIC

Όπως αναφέρθηκε, οι μικροελεγκτές PIC διαθέτουν ένα περιορισμένο σύνολο από 35 εντολές, που η κάθε μια αποτελείται από 14 bits. Επειδή και ο διάδρομος εντολών έχει μήκος 14 bits κάθε εντολή μπορεί να εκτελεστεί σε έναν μοναδικό κύκλο εκτέλεσης. Ο πίνακας 2.1 παραθέτει τις 35 εντολές που αποτελούν τη γλώσσα Assembly των μικροελεγκτών PIC.

Οι εντολές των ελεγκτών PIC μπορούν να χωριστούν σε τέσσερις κατηγορίες.

1. Αριθμητικές εντολές
2. Ελέγχου εκτέλεσης
3. Ελέγχου του μικροεπεξεργαστή
4. Χειρισμού των bit των καταχωρητών

Η πρώτη κατηγορία αποτελείται από τις «αριθμητικές» εντολές, που περιλαμβάνουν τη πράξη της πρόσθεσης και της αφαίρεσης ανάμεσα στα περιεχόμενα καταχωρητών, καθώς και πράξεις αύξησης ή μείωσης των τιμών τους και πράξεις σε επίπεδο bit.

Στην επόμενη κατηγορία εντολών, ανήκουν οι εντολές «ελέγχου εκτέλεσης». Αυτές, τις αποτελούν οι εντολές άλματος (goto), κλήσης υπορουτίνας (call) και οι εντολές επιστροφής (return) από κάποια υπορουτίνα, καθώς επίσης και οι εντολές διακλάδωσης υπό συνθήκη (btfss, btfsc, decfsz κλπ). Στην επόμενη κατηγορία ανήκουν οι εντολές «ελέγχου του μικροεπεξεργαστή». Οι εντολές αυτές επηρεάζουν βασικά τη λειτουργία του επεξεργαστή και τα κυκλώματα που συσχετίζονται με αυτόν (π.χ. clrwdt, sleep, clrf). Η τελευταία κατηγορία αποτελείται από τις εντολές «χειρισμού των bit των καταχωρητών» (τοποθέτηση ή μηδενισμός bit). Με τις εντολές αυτές ελέγχουμε άμεσα, κάθε ένα ξεχωριστό bit των καταχωρητών. Η πιο προφανής χρήση των εντολών αυτών είναι ο άμεσος έλεγχος επιμέρους κυκλωμάτων και ακροδεκτών του μικροελεγκτή.

Μνημονικό Τελεστής		**Λειτουργία**	**Κώδικας** MSb LSb	**Σημαία - Flag**
Εντολές χειρισμού ψηφιολέξεων - Byte oriented file register operations				
ADDWF	f,	Πρόσθεσε το W και το f	00 0111 dfff ffff	C, DC, Z

	d			
ANDWF	f, d	Κάνε την λογική πράξη AND ανάμεσα στο W και το f	00 0101 dfff ffff	Z
CLRF	f	Μηδένισε το f	00 0001 1fff ffff	Z
CLRW	-	Μηδένισε το W	00 0001 0xxx xxxx	Z
COMF	f, d	Φτιάξε το συμπλήρωμα του f και αποθήκευσέ το στο d	00 1001 dfff ffff	Z
DECF	f, d	Μείωσε την τιμή του f	00 0011 dfff ffff	Z
DECFSZ	f, d	Μείωσε την τιμή του f, παρέκαμψε την επόμενη εντολή αν ο f γίνει 0	00 1011 dfff ffff	
INCF	f, d	Αύξησε την τιμή του f	00 1010 dfff ffff	Z
INCFSZ	f, d	Αύξησε την τιμή του f, παρέκαμψε την επόμενη εντολή αν ο f γίνει 0	00 1111 dfff ffff	
IORWF	f, d	Κάνε την λογική πράξη IOR ανάμεσα στο W και το f	00 0100 dfff ffff	Z
MOVF	f, d	Μετέφερε το περιεχόμενο του f	00 1000 dfff ffff	Z
MOVWF	f	Μετέφερε το περιεχόμενο του W στο f	00 0000 1fff ffff	
NOP	-	Εντολή δίχως λειτουργία (απλή χρονική καθυστέρηση ενός κύκλου μηχανής)	00 0000 0xx0 0000	
RLF	f, d	Μετέφερε προς τα αριστερά το περιεχόμενο του f μέσο του ψηφίου Carry	00 1101 dfff ffff	C
RRF	f, d	Μετέφερε προς τα δεξιά το περιεχόμενο του f μέσο του ψηφίου Carry	00 1100 dfff ffff	C
SUBWF	f, d	Αφαίρεσε το W από το f	00 0010 dfff ffff	C, DC, Z
SWAPF	f, d	Αντιμετάθεσε τα δύο μισά της ψηφιολέξης (Byte) στο f	00 1110 dfff ffff	

XORWF	f, d	Κάνε την λογική πράξη XOR ανάμεσα στο W και το f	00 0110 dfff ffff	Z
Εντολές χειρισμού ψηφίων - Bit oriented file register operations				
BCF	f, b	Μηδένισε το ψηφίο b του καταχωρητή f	01 00bb bfff ffff	
BSF	f, b	Κάνε λογικό 1 το ψηφίο b του καταχωρητή f	01 01bb bfff ffff	
BTFSC	f, b	Εξέτασε το ψηφίο b του καταχωρητή f, παρέκαμψε την επόμενη εντολή αν είναι 0	01 10bb bfff ffff	
BTFSS	f, b	Εξέτασε το ψηφίο b του καταχωρητή f, παρέκαμψε την επόμενη εντολή αν είναι 1	01 11bb bfff ffff	
Εντολές πράξεων με σταθερούς αριθμούς και ελέγχου προγράμματος - Literal and control operations				
ADDLW	k	Πρόσθεσε τον σταθερό αριθμό k με το W	11 111x kkkk kkkk	C, DC, Z
ANDLW	k	Κάνε την λογική πράξη AND ανάμεσα στο k και το W	11 1001 kkkk kkkk	Z
CALL	k	Κάλεσε την υπορουτίνα k	10 0kkk kkkk kkkk	
CLRWDT	-	Μηδένισε τον επιτηρητή Watchdog Timer	00 0000 0110 0100	$\overline{TO}$, $\overline{PD}$
GOTO	k	Πήγαινε και εκτέλεσε την εντολή που υπάρχει στην διεύθυνση k	10 1kkk kkkk kkkk	
IORLW	k	Κάνε την λογική πράξη IOR ανάμεσα στο k και το W	11 1000 kkkk kkkk	Z
MOVLW	k	Μετέφερε το περιεχόμενο του k στο W	11 00xx kkkk kkkk	
RETFIE	-	Επέστρεψε στην διεύθυνση που ήσουν πριν συμβεί η διακοπή (interrupt)	00 0000 0000 1001	
RETLW	k	Επέστρεψε από υπορουτίνα και φόρτωσε τον σταθερό αριθμό k στο W	11 01xx kkkk kkkk	
RETURN	-	Επέστρεψε από υπορουτίνα	00 0000 0000 1000	

SLEEP	-	Ενεργοποίησε την λειτουργία χαμηλής κατανάλωσης (Sleep - κατανάλωση 2μA)	00 0000 0110 0011	$\overline{TO}$, $\overline{PD}$
SUBLW	k	Αφαίρεσε το περιεχόμενο του W από το σταθερό αριθμό k	11 110x kkkk kkkk	C, DC, Z
XORLW	k	Κάνε την λογική πράξη XOR ανάμεσα στο k και το W	11 1010 kkkk kkkk	Z

Πίνακας 1.2.1 Σετ εντολών του PIC

1.3 Οι καταχωρητές ειδικού σκοπού W και STATUS

Ο καταχωρητής μέσω του οποίου πραγματοποιούνται οι περισσότερες πράξεις και λειτουργίες είναι ο λεγόμενος καταχωρητής εργασίας (working register), που αναφέρεται ως καταχωρητής **w**. Αυτός είναι το αντίστοιχο του συσσωρευτή (accumulator), που χρησιμοποιείται σε δημοφιλείς μικροεπεξεργαστές, όπως τους 6502, 8085 και 8051.

Ο καταχωρητής **w** έχει εύρος 8 bits και δεν διευθυνσιοδοτείται. Μπορούμε να φανταζόμαστε ότι ενσωματώνεται στην αριθμητική μονάδα. Περιλαμβάνεται στις περισσότερες εντολές των μικροελεγκτών PIC, διότι μέσω αυτού γίνονται οι μετακινήσεις δεδομένων και οι πράξεις ανάμεσα στους καταχωρητές. Στον καταχωρητή w δεν μπορεί να γίνει προσπέλαση άμεσα, αλλά μετακινούνται τα περιεχόμενά του σε άλλους καταχωρητές, στους οποίους η πρόσβαση είναι άμεση. Κάθε αριθμητική πράξη που επιτελείται στο PIC, χρησιμοποιεί τον καταχωρητή w. Παραδείγματος χάρη αν θέλουμε να προσθέσουμε τα περιεχόμενα δύο καταχωρητών, πρέπει να μεταφέρουμε το περιεχόμενο του πρώτου καταχωρητή στον w και στη συνέχεια να το προσθέσουμε με το περιεχόμενο του δεύτερου καταχωρητή.

Οι ελεγκτές PIC διαθέτουν αρκετά ισχυρή αρχιτεκτονική από την άποψη ότι το αποτέλεσμα μιας αριθμητικής πράξης μπορεί να αποθηκευτεί ή στον καταχωρητή "w", ή στον καταχωρητή προέλευσης των δεδομένων. Αποθηκεύοντας το αποτέλεσμα στον καταχωρητή προέλευσης εξαλείφεται ουσιαστικά η ανάγκη χρήσης πρόσθετων εντολών για την αποθήκευση αυτή.

Με το τρόπο αυτό, η μεταφορά των αποτελεσμάτων απλουστεύεται και γίνεται αποδοτικότερη.

Ο καταχωρητής STATUS (Καταχωρητής Κατάστασης) αποτελεί τον βασικό καταχωρητή που χρησιμοποιείται για τον έλεγχο της εκτέλεσης του προγράμματος. Ο καταχωρητής αυτός χωρίζεται σε τρία τμήματα.

Το πρώτο τμήμα περιέχει τις σημαίες (Flags) ή bits κατάστασης της εκτέλεσης (τις "Z", "dc" και "C"). Τα τρία αυτά bits απεικονίζουν τη κατάσταση της εκτέλεσης του προγράμματος. Το bit "Z", ή η σημαία του μηδενός (Zero Flag), τίθεται σε λογικό "1" όταν το αποτέλεσμα κάποιας πράξης γίνει ίσο με το μηδέν (add, sub, clear, πράξεις λογικής επεξεργασίας). Η σημαία κρατουμένου (Carry Flag) "C", τίθεται σε λογικό "1" όταν το αποτέλεσμα κάποιας πράξης γίνει μεγαλύτερο από 255 (0x0FF), και χρησιμοποιείται για να δηλώσει ότι πρέπει να ενημερωθούν και τα υψηλότερης τάξης bytes που είναι σχετικά με το αποτέλεσμα.

Η σημαία δεκαδικού κρατουμένου (Digit Carry Flag) "dc", τίθεται σε λογικό "1" όταν τα τέσσερα λιγότερο σημαντικά bits (nibble) του αποτελέσματος μιας αριθμητικής πράξης, δώσουν αριθμό μεγαλύτερο από 15.

Αυτά τα bits, που αντιπροσωπεύουν οι σημαίες κατάστασης, μπορούν να διαβαστούν και να εγγραφούν, καθώς και να ενημερώνεται η κατάστασή τους, ανάλογα με την εκτέλεση της κάθε εντολής.

Τα επόμενα δύο bits του καταχωρητή STATUS δείχνουν το τρόπο με τον οποίο ο επεξεργαστής ανταποκρίθηκε κατά την εκτέλεση της διαδικασίας έναρξης του προγράμματος ή κατά την επιστροφή του από έναν ανενεργό κύκλο (sleep). Ο σκοπός για τον οποίο χρησιμοποιούνται τα bits αυτά, καθώς και ο καταχωρητής “PCON” (αν υπάρχει), είναι να μπορεί το πρόγραμμα της εφαρμογής να αντιλαμβάνεται την αιτία για την οποία ο έλεγχος του επεξεργαστή βρέθηκε στην αρχική θέση εκτέλεσης του προγράμματος.

Τα άλλα δύο bits, «RP0» και «RP1» χρησιμοποιούνται αποκλειστικά για τη προσπέλαση των δύο υψηλότερων σελίδων της μνήμης. Είναι δυνατή τόσο η

ανάγνωση όσο και η εγγραφή σε αυτά. Ενώ το bit "irp", χρησιμοποιείται για την επιλογή έμμεσης διευθυνσιοδότησης.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
03h	STATUS	0	RP1	RP0	-	-	Z	DC	C

Πίνακας 1.3.1 Τα bits που αποτελούν τον καταχωρητή STATUS

1.4 Είσοδος/έξοδος και καταχωρητές θυρών

Οι καταχωρητές PORTB και TRISB, PORTD και TRISD, καθώς και οι αντίστοιχοι καταχωρητές των υπόλοιπων θυρών, χρησιμεύουν στη λειτουργία εισόδου/εξόδου, δηλαδή στην ανάγνωση και εγγραφή δεδομένων από και προς τις θύρες. Η εγγραφή σε έναν καταχωρητή TRIS ορίζει ποιοι ακροδέκτες της αντίστοιχης θύρας λειτουργούν σαν είσοδοι και ποιοι σαν έξοδοι. Εγγράφοντας στον καταχωρητή TRISB τον δυαδικό αριθμό 11110000, ορίζουμε τα τέσσερα λιγότερο σημαντικά bits της θύρας B ως εξόδους και τα τέσσερα περισσότερο σημαντικά bits ως εισόδους (0 = έξοδος, 1 = είσοδος):

```
movlw b’11110000’
movwf TRISB
```

1.5 Σήματα διακοπής (Interrupts)

Ο μικροελεγκτής PIC16F877 μπορεί να δεχτεί σήματα διακοπών κυρίως από τις εξής πηγές.

- Εξωτερική διακοπή από τον ακροδέκτη RB0/INT.
- Υπερχείλιση του απαριθμητή-χρονιστή TMR0.
- Κάποια αλλαγή της κατάστασης των ακροδεκτών RB7-RB4 της θύρας B.

Ο καταχωρητής που ρυθμίζει την ενεργοποίηση των διακοπών και καταγράφει ποιες διακοπές σημειώθηκαν είναι ο καταχωρητής INTCON (Interrupt Control).

Το σημαντικότερο bit του καταχωρητή INTCON, το bit b7, είναι το bit της γενικής ενεργοποίησης των διακοπών (General Interrupt Enable, GIE) και όταν είναι μηδέν δεν επιτρέπει να συμβεί καμία διακοπή. Τα υπόλοιπα bits του καταχωρητή

διακρίνονται σε δύο κατηγορίες, αυτά που ενεργοποιούν τις επιμέρους διακοπές (IE) και αυτά που δηλώνουν ότι σημειώθηκε κάποια διακοπή (σημαίες διακοπών, IF). Καθένας από τους βασικούς τύπους διακοπών, που αναφέραμε προηγουμένως έχει το δικό του ζευγάρι IE και IF. Έτσι, για να ενεργοποιηθεί η δυνατότητα να εμφανιστεί διακοπή εξαιτίας της υπερχείλισης του χρονιστή TMR0 πρέπει πρώτα να τεθεί σε λογικό 1 το bit GIE και κατόπι να τεθεί σε λογικό 1 το bit 5 (T0IE). Μόλις εμφανιστεί η υπερχείλιση, η αντίστοιχη σημαία διακοπής T0IF τίθεται σε λογικό 1 και ακολουθεί η διαδικασία που περιγράψαμε: η τιμή του απαριθμητή προγράμματος, που περιέχει τη διεύθυνση της επόμενης εντολής αποθηκεύεται στη *στοίβα* (stack) και το πρόγραμμα μεταβαίνει στη διεύθυνση 0x004 της μνήμης του προγράμματος, προκειμένου να εξυπηρετηθεί η διακοπή.

Το bit που ενεργοποιεί τις διακοπές που προέρχονται από τα τέσσερα ανώτερα bits της θύρας B είναι το bit RBIE και η αντίστοιχη σημαία είναι το bit RBIF. Παρομοίως, οι διακοπές που προέρχονται από σήματα στον ακροδέκτη RB0/INT ενεργοποιούνται και σημειώνονται από το bit 4 (INTE) και το bit 1 (INTF) αντίστοιχα, του καταχωρητή INTCON. Το είδος του μετώπου (θετικό ή αρνητικό) που προκαλεί τη διακοπή στον ακροδέκτη RB0/INT ορίζεται από το bit INTEDG του καταχωρητή OPTION_REG.

Όταν καλείται η υπορουτίνα εξυπηρέτησης της διακοπής, απενεργοποιούνται οι επιπλέον διακοπές και έτσι το σύστημα δεν μπορεί να δεχτεί άλλη διακοπή μέχρι να επιστρέψει από αυτήν που ήδη άρχισε να εξυπηρετεί. Στο τέλος της υπορουτίνας της διακοπής ο χρήστης οφείλει να μηδενίσει εκ νέου τη σημαία της διακοπής και να ενεργοποιήσει ξανά τις διακοπές, τοποθετώντας σε λογικό 1 το bit GIE. Αυτό επιτυγχάνεται με χρήση της εντολής RETFIE στο τέλος της υπορουτίνας της διακοπής. Η εντολή αυτή είναι μια παραλλαγή της εντολής RETURN, και με αυτήν αφενός επιτυγχάνεται η επιστροφή στο κυρίως πρόγραμμα, στη διεύθυνση της εντολής που αποθηκεύτηκε στη στοίβα, και αφετέρου ενεργοποιούνται ξανά οι διακοπές, με τοποθέτηση του GIE σε λογικό 1.

Address	**Name**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

Πίνακας 1.5.1 Τα bits που αποτελούν τον καταχωρητή INTCON

Ένα τελευταίο σημείο προσοχής, που πρέπει να λαμβάνεται υπόψη όταν συμβαίνει μία διακοπή είναι ότι κατά τη διάρκεια της διακοπής πιθανότητα θα μεταβληθούν οι τιμές σημαντικών καταχωρητών, γεγονός που μπορεί να δημιουργήσει προβλήματα μετά τη επιστροφή στο κυρίως πρόγραμμα. Για το λόγο αυτό είναι σκόπιμο οι τιμές των σημαντικών καταχωρητών να αποθηκεύονται κατά την εκκίνηση της υπορουτίνας της διακοπής, ώστε να ανακτώνται μετά την εκτέλεση της υπορουτίνας. Οι σημαντικότεροι καταχωρητές που είναι σκόπιμο να αποθηκεύονται και να ανακτώνται είναι ο καταχωρητής εργασίας W και ο καταχωρητής κατάστασης STATUS.

1.6 Κύκλωμα ADC στον μικροελεγκτή PIC16F877

Όπως ολα τα περιφερειακά του PIC, έτσι και ο ADC ρυθμίζεται ώστε να επιτελεί τις επιθυμητές λειτουργίες, με τη βοήθεια ορισμένων καταχωρητών ειδικού σκοπού. Οι βασικοί καταχωρητές είναι ο ADCON0 και ADCON1. Όπως αναφέρθηκε, οι καταχωρητές ADRESL και ADRESH αποθηκεύουν το αποτέλεσμα της μετατροπής. Με αριστερή στοίχηση, ο ADRESH αποθηκεύει τα 8 πιο σημαντικά bits της μετατροπής, άρα μπορεί να χρησιμοποιηθεί αν επιθυμούμε να απλουστεύσουμε τον μετατροπέα και να τον χρησιμοποιήσουμε σαν να είχε ανάλυση 8-bits.

Επιλογή συχνότητας ρολογιού ADC: Ο μετατροπέας χρονίζεται με τη βοήθεια γεννήτριας παλμών, που λαμβάνεται με υποδιαίρεση της συχνότητας του εξωτερικού κρυστάλλου. Η υποδιαίρεση ρυθμίζεται με τα bits ADCS2, ADCS1, ADCS0, από τα οποία τα δύο λιγότερα σημαντικά ανήκουν στον καταχωρητή ADCON0, ενώ το πιο σημαντικό (ADCS2) ανήκει στον ADCON1 (βλέπε σχήματα 8.4 και 8.5). Μια κατάλληλη υποδιαίρεση για το ρολόϊ του μετατροπέα είναι Fosc/16, που αντιστοιχεί σε bits επιλογής 101.

Τα bits CHS2:CHS0 του ADCON0 επιλέγουν κανάλι εισόδου, μέσω του πολυπλέκτη εισόδου. Έτσι, για να γίνει η μετατροπή του καναλιού Ch0 (RA0) αυτά πρέπει να λάβουν την τιμή 000.

Ο καταχωρητής ADCON1 διαθέτει τέσσερα bits PCFG3:PCFG0 που ρυθμίζουν τη διαμόρφωση των εισόδων AN0 έως AN7, δηλαδή ποιές θα λειτουργούν ως αναλογικές και ποιες ως ψηφιακές. Επίσης, κάποιες μπορούν να διαμορφωθούν ώστε

να δεχτούν είσοδο αναφοράς. Το bit ADFM του καταχωρητή ADCON1 ρυθμίζει τη στοίχηση της λέξης 10-bit που παράγει ο ADC.

Τέλος, για να λειτουργήσει ο ADC πρέπει να θέσουμε το bit ADON του ADCON0 σε λογικό 1 (power up). Για να γίνει η δειγματοληψία μιας νέας αναλογικής τάσης, η σημαία ADIF πρέπει να μηδενιστεί. Είναι απαραίτητο να προβλέψουμε μια μικρή περίοδο αδράνειας, ώστε να σταθεροποιηθεί η τάση στην είσοδο του κυκλώματος δειγματοληψίας και συγκράτησης. Αυτό επιτυγχάνεται με δύο ή τρεις εντολές nop.
Για να ξεκινήσει μια νέα μετατροπή θέτουμε σε λογικό 1 το bit GO του ADCON0.
Όταν ολοκληρωθεί η μετατροπή η σημαία ADIF γίνεται 1. Η σημαία αυτή βρίσκεται στον καταχωρητή PIR1, που ρυθμίζει τα σήματα διακοπής περιφερειακών συσκευών. Όταν η σημαία ADIF γίνει 1, το αποτέλεσμα βρίσκεται στους καταχωρητές ADRESH, ADRESL, και μπορεί να διαβαστεί.

Address	**Name**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON

Πίνακας 1.6. Τα bits που αποτελούν τον καταχωρητή ADCON0

1.7 Καταχωρητές της ασύγχρονης σειριακής θύρας

Η Σύγχρονη-Ασύγχρονη μετάδοση (USART) είναι μία από τις δύο σειριακές μονάδες εισόδου – εξόδου που διαθέτει ο μικροελεγκτής PIC16F877.Για να επικοινωνήσει ο PIC μέσω της σειριακής θύρας, με περιφερειακές συσκευές ή υπολογιστές χρησιμοποιεί τους εξής καταχωρητές ειδικού σκοπού:

ο TXSTA: Transmit Status and Control Register- Καταχωρητής κατάστασης και ελέγχου της αποστολής των δεδομένων.

ο RCSTA: Receive Status and Control Register- Καταχωρητής κατάστασης καιελέγχου της λήψης των δεδομένων.

ο SPBRG: Baud Rate Generation Register – Καταχωρητής παραγωγής του ρυθμού μετάδοσης baud rate.

ο TXREG: Καταχωρητής αποστολής δεδομένων.

ο RCREG: Καταχωρητής λήψης δεδομένων.

Οι πρώτοι τρεις καταχωρητές ρυθμίζουν την λειτουργία της μονάδας USART, ενώ τους άλλους δύο τους χρησιμοποιούμε ως καταχωρητές αποθήκευσης, για να μεταφέρουμε δεδομένα από και προς την USART.

Για την ενεργοποίηση της σειριακής θύρας, το bit SPEN (RCSTA<7>) πρέπει να τεθεί σε λογικό '1'. Για την ενεργοποίηση της ασύγχρονης επικοινωνίας το bit SYNC του καταχωρητή TXSTA (TXSTA<4>) τίθεται σε λογικό 1. Τα bits TRISC < 7:6 > πρέπει να τεθούν σε κατάσταση "10", για να διαμορφωθούν τα pins RC6/TX/CX και RC7/RX/DT της USART ως σειριακή έξοδος και είσοδος αντίστοιχα. Το πρώτο (RC6) εκπέμπει δεδομένα, ενώ το δεύτερο (RC7) λαμβάνει δεδομένα. Για τον καθορισμό του baud rate εγγράφουμε τον καταχωρητή SPBRG με μια τιμή που προκύπτει από τις παρακάτω σχέσεις. Σημειώνεται ότι σε περίπτωση που επιθυμούμε χαμηλούς ρυθμούς μετάδοσης, το bit BRGH (TXSTA<2>) είναι μηδέν, οπότε χρησιμοποιούμε την πρώτη σχέση, ενώ αν επιθυμούμε υψηλούς ρυθμούς μετάδοσης το bit αυτό τίθεται σε λογικό 1 και η τιμή SPBRG προκύπτει από τη δεύτερη σχέση: Η Ασύγχρονη επικοινωνία χρησιμοποιεί τη μορφή μετάδοσης που περιγράψαμε στην παράγραφο 7.1: ένα bit START, οχτώ ή εννιά bit δεδομένων και ένα STOP bit. Η USART στέλνει και λαμβάνει το LSB (Least Significant Bit) πρώτα. Η αποστολή και η λήψη είναι λειτουργικά ανεξάρτητες, όμως χρησιμοποιούν την ίδια μορφή δεδομένων και τον ίδιο ρυθμό μετάδοσης.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
0Bh, 8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	R0IF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
18h	RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
1Ah	RCREG	USART Receive Register								0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
99h	SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Πίνακας 1.7.1 Καταχωρητές που σχετίζονται με την ασύγχρονη επικοινωνία

Ασύγχρονη αποστολή

Ο πυρήνας της μετάδοσης είναι ένας εσωτερικός καταχωρητής, ο καταχωρητής TSR. Ο TSR καταχωρητής περιέχει τα δεδομένα του buffer μετάδοσης TXREG (όπου ο TXREG περιέχει τα δεδομένα αποστολής που φορτώνουμε εμείς από το Software).

Ο TSR δεν φορτώνεται με δεδομένα έως ότου έρθει το STOP bit από την προηγούμενη φόρτωση. Μόλις έρθει το STOP bit τα δεδομένα (εάν υπάρχουν) μεταφέρονται από τον TXREG στον TSR. Μόλις ο TXREG στείλει τα δεδομένα στον TSR, υψώνεται σημαία ότι ο TXREG είναι άδειος. Αυτό δηλώνεται από το bit TXIF (PIR1<4>) με την ένδειξη ‘1’. Αυτή η διακοπή μπορεί να ενεργοποιηθεί θέτοντας ‘1’ το bit TXIE (PIE1<4>). Αν το TXIE είναι ‘0’ το TXIF θα γίνει ‘1’,αλλά δεν θα παραχθεί σήμα διακοπής. Ο TXIF γίνεται ‘0’ μόλις φορτωθούν καινούργια δεδομένα στον TXREG. Η μετάδοση μπορεί να ενεργοποιηθεί θέτοντας ‘1’ το bit TXEN (TXSTA<5>). Η μετάδοση στην πραγματικότητα δεν θα πραγματοποιηθεί μέχρι ο TXREG να φορτωθεί με δεδομένα και ο Γεννήτορας Ρυθμού Μετάδοσης να παράγει τον χρονισμό.

Για να καθορίσουμε μια Ασύγχρονη Αποστολή ακολουθούμε τα εξής βήματα:

1. Αρχικοποιούμε τον καταχωρητή SPBRG με το κατάλληλο ρυθμό μετάδοσης. Εάν επιθυμούμε υψηλούς ρυθμούς μετάδοσης, θέτουμε ‘1’ το bit BRGH

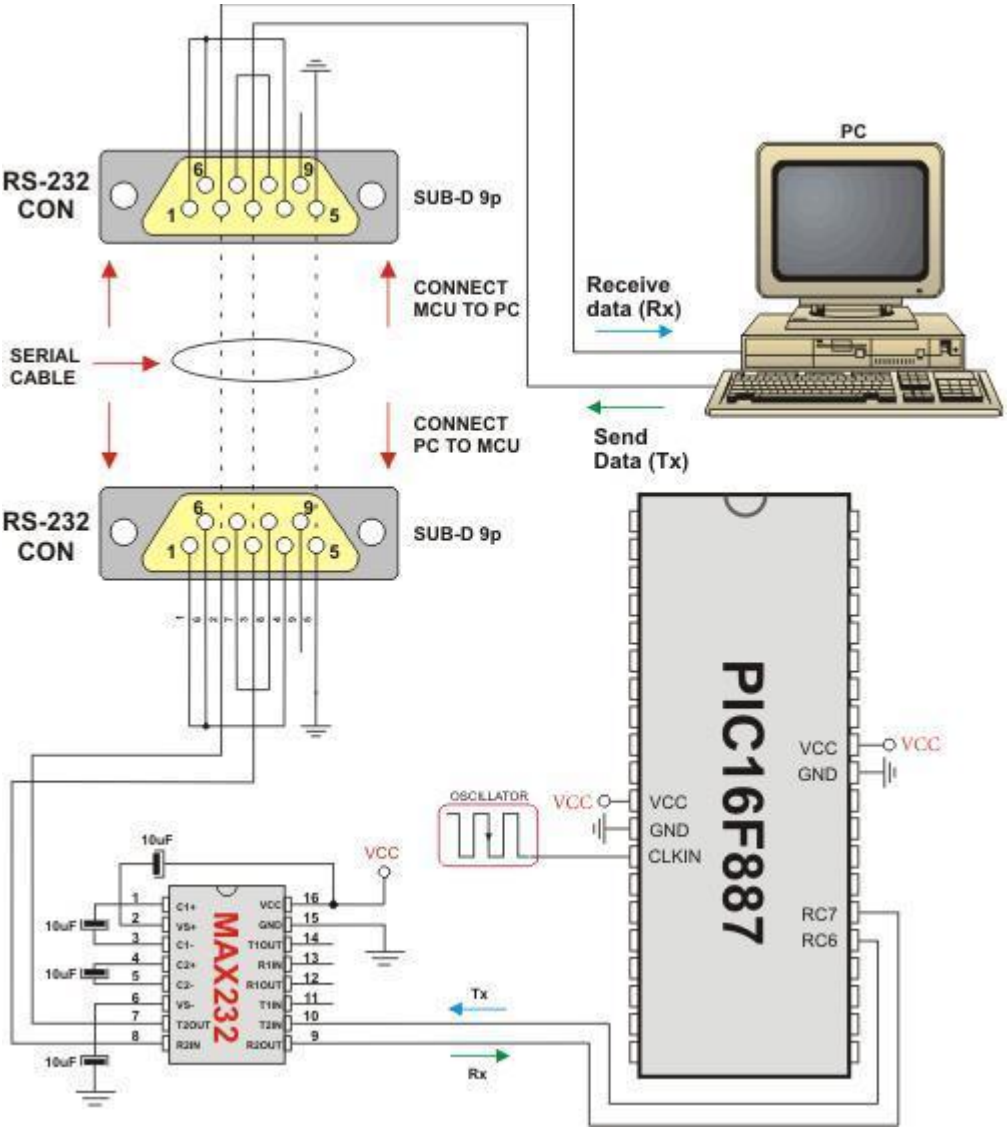
2. Ενεργοποιούμε την Ασύγχρονη Σειριακή πόρτα θέτοντας ‘0’ το bit SYNC (TXSTA<4>) και ‘1’ το bit SPEN (RCSTA<7>)

3. Εάν επιθυμούμε διακοπές, τότε ενεργοποιούμε το bit TXIE (PIE1<4>)

4. Ενεργοποιούμε την μετάδοση θέτοντας ‘1’ το bit TXEN (TXSTA<5>), το οποίο θα θέσει με την σειρά του ‘1’ το bit TXIF (PIR1<4>)

5. Φορτώνουμε στον TXREG τα δεδομένα (ξεκινάει η μετάδοση)

6. Εάν χρησιμοποιούμε διακοπές, σιγουρευόμαστε ότι τα bits (6 και 7) του καταχωρητή INTCON είναι ‘1’



Εικόνα 1.7.1 Διάγραμμα κυκλώματος σειριακής επικοινωνίας

Ασύγχρονη λήψη

Αντίστοιχα με όσα συμβαίνουν στην ασύγχρονη σειριακή εκπομπή συμβαίνουν και κατά την σειριακή λήψη. Τα δεδομένα λαμβάνονται από το pin RC7/RX/DT και μεταφέρονται στον Data Recovery. Ο πυρήνας της λήψης είναι ο εσωτερικός καταχωρητής RSR. Μετά τη λήψη του STOP bit τα δεδομένα λήψης τα οποία βρίσκονται στον RSR, μεταφέρονται στον καταχωρητή RCREG (αν αυτός είναι άδειος). Αν η μεταφορά ολοκληρωθεί η σημαία του καταχωρητή RCIF (PIR1<5>), γίνεται ‘1’. Η πραγματική διακοπή μπορεί να ενεργοποιηθεί θέτοντας ‘1’ το bit RCIE, του καταχωρητή PIE<5>.

Για να καθορίσουμε μια Ασύγχρονη Λήψη ακολουθούμε τα εξής βήματα:

1.Φόρτωση του SPBRG για το κατάλληλο ρυθμό μετάδοσης. Αν θέλουμε υψηλό ρυθμό μετάδοσης τότε το bit BRGH πρέπει να τεθεί ‘1’

2. Ενεργοποιούμε την ασύγχρονη σειριακή πόρτα με το μηδενισμό του SYNC bit και θέτοντας ‘1’ το bit SPEN

3. Αν η διακοπή είναι επιθυμητή θέτουμε ‘1’ το bit RCIE

4.Ενεργοποιούμε την λήψη θέτοντας ‘1’ το bit CREN

5. Η σημαία του bit RCIF θα είναι ‘1’ όταν η λήψη είναι ολοκληρωμένη και μια διακοπή θα λάβει χώρα εφόσον το RCIE bit θα είναι ‘1’

6. Διαβάζουμε τα 8-bit δεδομένων που λήφθηκαν διαβάζοντας τον καταχωρητή RCREG

7. Εάν οποιοδήποτε σφάλμα λάβει χώρα, μηδενίζουμε το σφάλμα, μηδενίζοντας το bit CREN

8. Αν χρησιμοποιούμε διακοπή σιγουρευόμαστε ότι το GIE και το PEIE (7ο και 6ο bit αντίστοιχα) του καταχωρητή INTCON είναι ‘1’

Κεφ. 2 Σχεδίαση και υλοποίηση ενός αναπτυξιακού κυκλώματος

2.1 Σχεδίαση και υλοποίηση στο ράστερ

Το πρώτο βήμα που έγινε ήταν να προκαθοριστούν κάποιες προδιαγραφές για το κύκλωμα. Αρχικό ζητούμενο ήταν να καλυφθούν όλες οι βασικές λειτουργίες του μικροελεγκτή όπως οι παράλληλες θύρες εισόδου εξόδου, διάφοροι ειδικοί καταχωρητές όπως ο ADCON, INTCON και η θύρα USART. Επίσης θεωρήθηκε σημαντικό να είναι διαθέσιμες όλες οι θύρες στο χρήστη για να μπορεί με τη χρήση jumper να συνδέσει τα υλικά εισόδου εξόδου καθώς και τις θύρες του PIC σε ένα ράστερ ή κάποιο άλλο τυπωμένο για να πειραματιστεί και να αναπτύξει περαιτέρω το κύκλωμα και τις γνώσεις του.

Ο PIC16F877 απαιτεί τάση 5V για να λειτουργήσει, οπότε η αρχή της σχεδίασης του κυκλώματος στο ράστερ έγινε από την τροφοδοσία. Το κύκλωμα χρησιμοποιεί ένα τροφοδοτικό τάσης 9V και με την βοήθεια ενός ολοκληρωμένου κύκλωματος τριών ακίδων, το 7805, το οποίο είναι ένας ρυθμιστής τάσης, επιτεύχθηκε 5V τάση στην έξοδο του ώστε να τροφοδοτηθεί το κύκλωμα.

Στη συνέχεια σημαντικό βήμα ήταν η σύνδεση του PICkit3 στους σωστούς ακροδέκτες του μικροελεγκτή ώστε να κατασταθεί δυνατός ο προγραμματισμός του. Από τη στιγμή που έγινε αυτό και δόθηκε τροφοδοσία στο κύκλωμα το MPLAB αναγνώρισε απευθείας με ποιον μικροελεγκτή ήταν συνδεμένος το PICkit3.

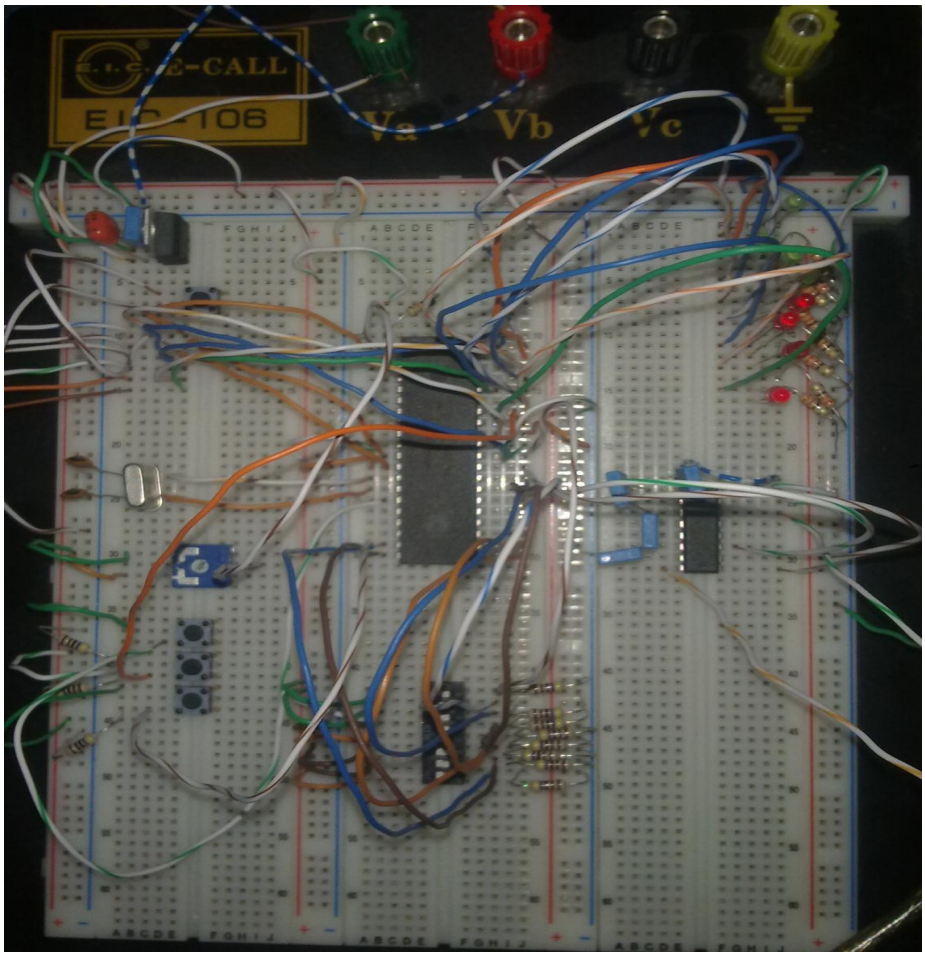
Έχοντας κάνει τις παραπάνω ενέργειες άρχισε ο πειραματισμός με τις θύρες του PIC, αρχικά τοποθετήθηκαν τα leds στη θύρα B και στη συνέχεια ένα DIP switch στη θύρα D. Τα 3 push buttons συνδέθηκαν με το πρώτα pins των θυρών B, C και D. Στο Pin 0 της θύρας A τοποθετήθηκε ένα trimmer, το οποίο χρησιμοποιείτε στη μετατροπή ενός σήματος από αναλογικό σε ψηφιακό. Επίσης προσαρμόστηκε και ένας ταλαντωτής.

Εφόσον τοποθετήθηκαν όλα τα παραπάνω το τελικό στάδιο του δοκιμαστικού κυκλώματος ήταν να χρησιμοποιηθεί ένα ακόμα ολοκληρωμένο, ένα MAX 232, με τη βοήθεια του οποίου μετατρέπονται σήματα RS232, σειριακής θύρας σε επίπεδα TTL για να μπορέσει ο PIC να αναγνωρίσει εισερχόμενα σήματα που βρίσκονται στα

επίπεδα του RS232 αλλά και να μετατραπούν τα σήματα που αποστέλονται στη σειριακή θύρα που χρησιμοποιείτε για την σειριακή επικοινωνία.

Φωτογραφίες από όλα τα εξαρτήματα αλλά και σχετικοί πίνακες υπάρχουν στο παράρτημα Γ.

Το τελικό κύκλωμα στο ράστερ είχε την παρακάτω μορφή.



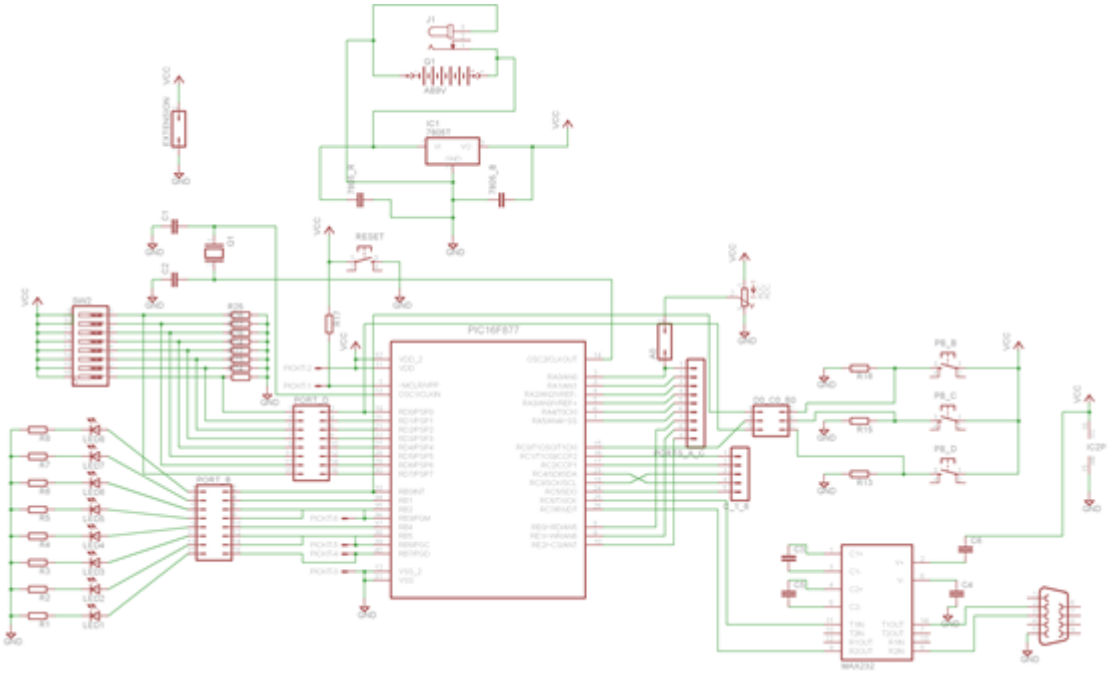
Εικόνα 2.1.1 Τελικό κύκλωμα σε ράστερ

2.2 Σχεδίαση στο EAGLE

Σειρά είχε πλέον η δημιουργία της μάσκας του τυπωμένου κυκλώματος σε ηλεκτρονικό υπολογιστή με τη χρήση κάποιου σχεδιαστικού προγράμματος. Μετά από μία σχετική ερεύνα αποφασίστηκε η χρησιμοποίηση του EAGLE καθώς, πέρα

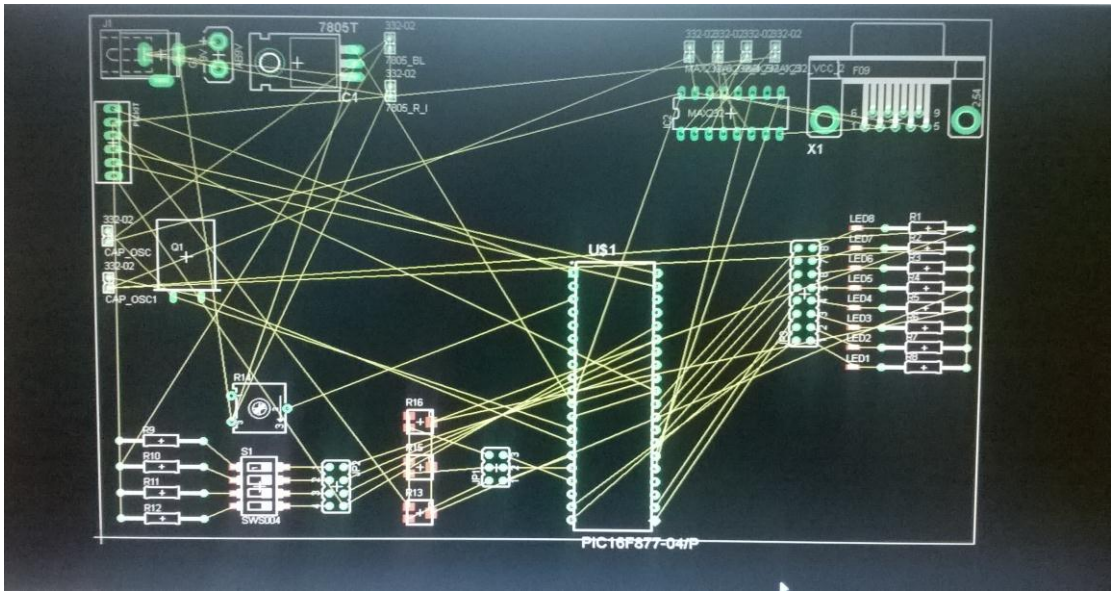
από το οτι η χρήση του είναι εύκολη για κάποιο αρχάριο, αποτελεί ένα πολύ δυνατό εργαλείο με πολλές δυνατότητες και χρησιμοποιείτε πάρα πού αυτή τη στιγμή στην αγορά. Υπάρχει μια δωρεάν, περιορισμένων δυνατοτήτων έκδοση διαθέσιμη στο http://www.cadsoftusa.com/download-eagle/freeware/ .

Το EAGLE χωρίζει τη διαδικασία σχεδίασης σε δύο μέρη. Αρχικά ο χρήστης καλείται να δημιουργήσει το σχηματικό του κυκλώματος επιλέγοντας μέσα από βιβλιοθήκες όλα τα εξαρτήματα που χρησιμοποιεί και να δημιουργήσει όλες τις απαραίτητες ενώσεις μεταξύ αυτών. Αφού ακολουθήθηκε πιστά το λειτουργικό κύκλωμα του ράστερ δημιουργήθηκε το παρακάτω σχέδιο.



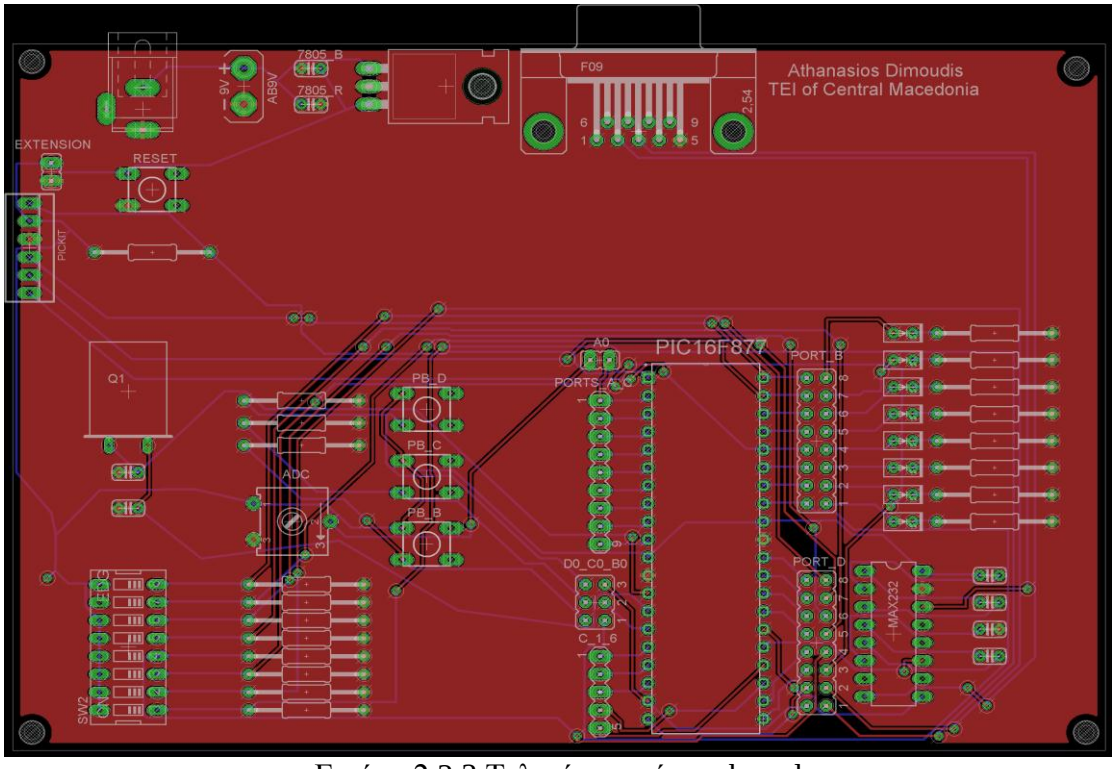
Εικόνα 2.2.1 Το σχηματικό του κυκλώματος στο EAGLE

Αφού έχει ολοκληρωθεί το σχηματικό του κυκλώματος γίνεται εναλλαγή στο περιβάλλον του EAGLE και αρχίζει ο σχεδιασμός του board το οποίο στην αρχή έχει την παρακάτω μορφή:



Εικόνα 2.2.2 Αρχική μορφή του board

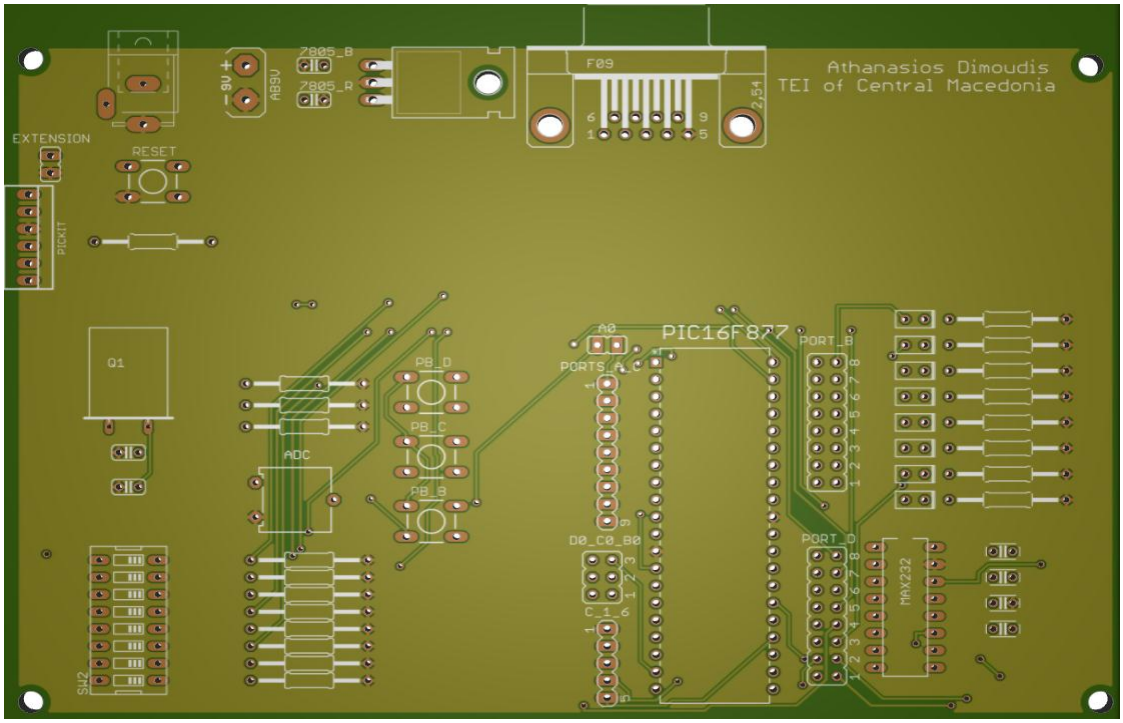
Από εκεί και έπειτα ο χρήστης μετακινεί τα εξαρτήματα στις θέσεις που επιθυμεί και ξεκινάει τη διαδικασία του routing με την οποία καθορίζονται οι διαδρομές που θα πάρουν όλες οι ενώσεις προκειμένου να αποφευχθούν τυχόν βραχυκυκλώματα και δυσλειτουργίες. Το αποτέλεσμα είναι αυτό:



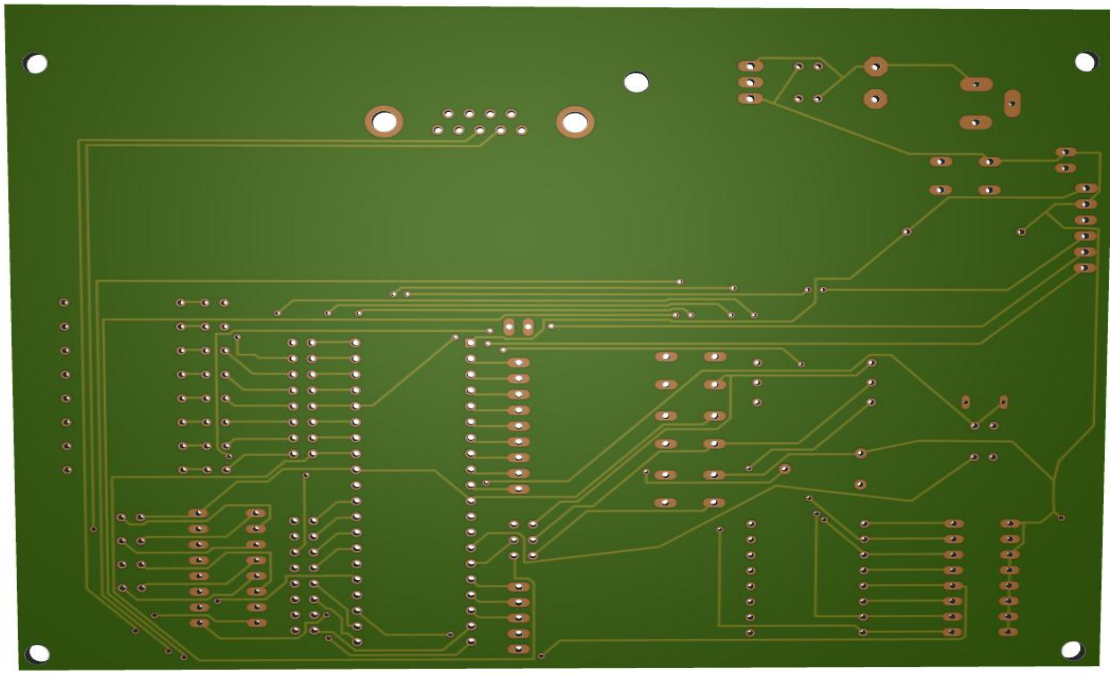
Εικόνα 2.2.3 Τελική μορφή του board

2.3 Εξαγωγή αρχείων προς παραγωγή

Εφόσον παρήχθη ικανοποιητικό αποτέλεσμα άρχισε η ερεύνα εντοπισμού κάποιου εργαστηρίου που να μπορεί να κατασκευάσει το τυπωμένο. Δεδομένου ότι χρειαζόμασταν αρκετά αντίγραφα διότι προορίζονται για να ενταχθούν στην εκπαιδευτική διαδικασία του Τμήματος ζητήθηκαν προσφορές από διάφορα εργαστήρια και επιλέχθηκε η πιο συμφέρουσα. Αυτό που είχαμε όμως στα χέρια μας μέχρι στιγμής ήταν κάποια αρχεία που παρήγαγε το EAGLE τα οποία ήταν σε συγκεκριμένη μορφή που αναγνωρίζει μόνο αυτό το πρόγραμμα. Λόγω της ποικιλίας προγραμμάτων σχεδίασης υπάρχουν κάποιοι τύποι αρχείων οι οποίοι χρησιμοποιούνται ευρέως στην αγορά για να αποφεύγονται τυχόν αναντιστοιχίες. Τέτοιοι τύποι αρχείων είναι τα αρχεία Gerber, PostScript και Bitmap. Αναφέρω τους συγκεκριμένους που ζητήθηκαν από τα εργαστήρια που ήρθαμε σε επικοινωνία. Ο τύπος αρχείων που ήταν πιο χρήσιμος ήταν τα Gerber Files τα οποία εξήχθησαν από το EAGLE χρησιμοποιώντας έναν CAM processor. Υπάρχουν κάποιες σελίδες, όπως η http://mayhewlabs.com/webGerber/ , οι οποίες επιτρέπουν να ελέγξει κάποιος αν έχει κάνει σωστά την εξαγωγή των αρχείων και να πάρει μια ιδέα για το πώς θα είναι το τελικό προϊόν.

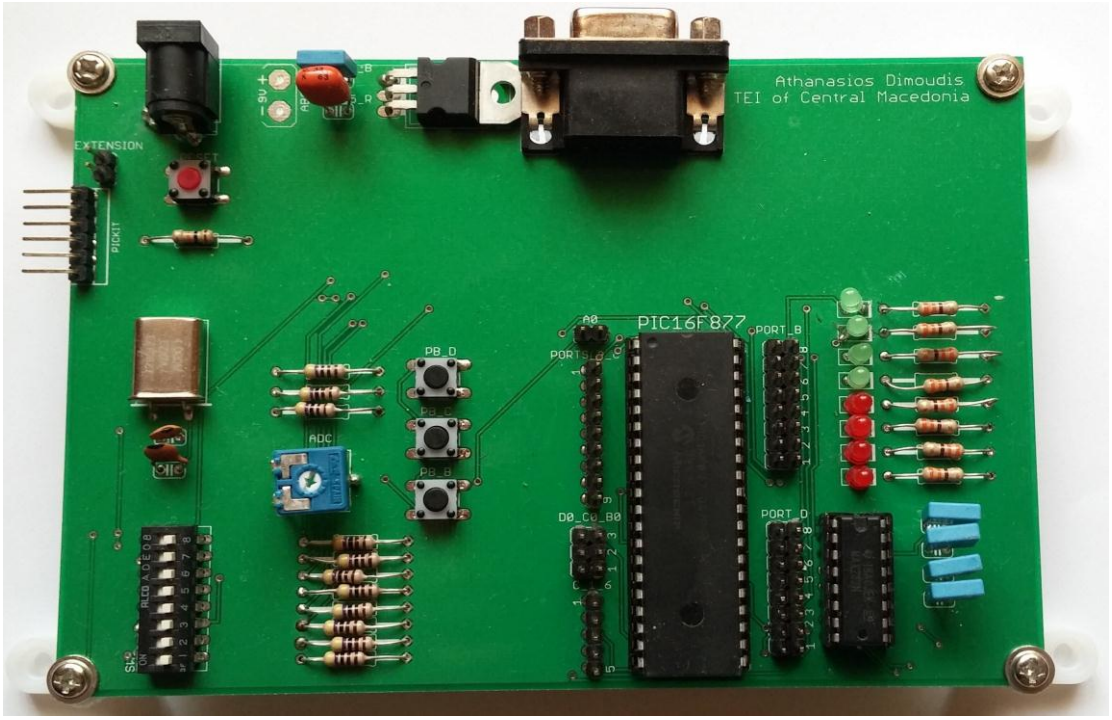


Εικόνα 2.3.1 Μπροστινή άποψη του board από τα αρχεία gerber



Εικόνα 2.3.2 Πίσω άποψη του board από τα αρχεία gerber

Οπότε ήταν όλα έτοιμα για να αποσταλεί η παραγγελία και αφού παρελήφθησαν τα τυπωμένα και τοποθετήθηκαν πάνω όλα τα εξαρτήματα το τελικό αποτέλεσμα ήταν το εξής:



Εικόνα 2.3.3 Τελικό προϊόν, αναπτυξιακό κύκλωμα

Κεφ. 3 Εκπαιδευτικές εφαρμογές

3.1 Προγραμματισμός των θυρών PIO

Το παρακάτω πρόγραμμα επιδεικνύει τον τρόπο προγραμματισμού των θυρών Parallel I/O (PIO) του PIC για είσοδο και για έξοδο. Με την χρήση του DIP switch δίνουμε λογικό 1 στον ακροδέκτη 0 της θύρας D και παρατηρούμε την εναλλαγή στην έξοδο, δηλαδή στα leds, και στην συνέχεια δίνοντας λογικό 1 και στον ακροδέκτη 1 παρατηρούμε την δεύτερη εναλλαγή.

```
#include "p16f877.inc"

;Initialization

__CONFIG _CP_OFF & _WDT_OFF & _HS_OSC & _PWRTE_OFF & _CPD_OFF
& _WRT_ENABLE_ON & _BODEN_ON & _LVP_OFF & _DEBUG_ON

        Org 0   ;Τοποθέτησε την επόμενη εντολή στη θέση 4 του προγράμματος

        bsf STATUS, RP0      ;Πήγαινε στο bank1

        movlw b'00000000'    ;Κάνε όλα τα pin της θύρας B έξοδο

        movwf TRISB

        movlw b'11111111'    ;Κάνε όλα τα pin της θύρας D εισόδο

        movwf TRISD

        bcf STATUS, RP0      ;Επέστρεψε στο bank0

main

        movlw b'01010101'    ;Μετέφερε στον καταχωρητή w

        movwf PORTB          ;Μετέφερε στην θύρα B

        btfss PORTD, 0       ;Περίμενε μέχρι το D0 λάβει 1

        goto $-1
```

```
        movlw b'10101010'    ;Αν λάβει το D0 1 μετέφερε στην θύρα B
        movwf PORTB
        btfss PORTD, 1       ;Περίμενε μέχρι το D0 λάβει 1
        goto $-1
        goto main
end
```

3.2 Διακοπές

Το παρακάτω πρόγραμμα επιδεικνύει την χρήση διακοπών. Αρχικά μηδενίζει η έξοδος, δηλαδή βλεπουμε το μοτίβο 00000000 στα leds, και κάθε φορά που αντιλαμβάνεται ο μικροελεγκτής μια διακοπή στον ακροδέκτη RB0, με τη χρήση του αντίστοιχου push button, αυξάνει κατά 1 την έξοδο στη θύρα B.

```
#include "p16f877.inc"
;Initialization
__CONFIG _CP_OFF & _WDT_OFF & _HS_OSC & _PWRTE_OFF & _CPD_OFF & _WRT_ENABLE_ON & _BODEN_ON & _LVP_OFF & _DEBUG_ON
TEMP equ 20h

        Org 0
        goto start           ;Πήγαινε στο κυρίως πρόγραμμα
        Org 4                ;Τοποθέτησε την επόμενη εντολή στη θέση 4 του προγράμματος
        incf TEMP, F         ;Αύξησε τον καταχωρητή TEMP κατά 1
        movf TEMP,W
```

```
        movwf PORTB         ;Μετέφερε τον TEMP στην θύρα B
        bcf INTCON, INTF    ;Μηδένισε την σημαία της διακοπής
        retfie              ; Επέστρεψε από την διακοπή

start
        clrf TEMP           ;Μηδένισε τον TEMP
        bsf STATUS, RP0     ; Πήγαινε στην σελίδα μνήμης 1
        movlw b'00000001'
        movwf TRISB         ; Κάνε την θύρα C έξοδο
        bcf STATUS, RP0     ; Επέστρεψε στη σελίδα μνήμης 0
        bsf INTCON, INTE    ; Ενεργοποίησε την διακοπή RB0
        bcf INTCON, INTF    ;Μηδένισε την σημαία της διακοπής RB0
        bsf INTCON, GIE     ;Ενεργοποίησε τις διακοπές
loop    goto loop           ;Περίμενε για διακοπή

        END
```

3.3 Μετατροπή αναλογικού σήματος σε ψηφιακό

Στο συγκεκριμένο παράδειγμα χρησιμοποιείτε το trimmer του αναπτυξιακού από το οποίο λαμβάνει σήμα ο μικροελεγκτής και στη συνέχεια το ψηφιοποιεί δίνοντας μας στα leds 256 καταστάσεις αριθμώντας από το 0 ως το 255 τα οποία αντιστοιχούν στα 0-5 Volt που παίρνει ως είσοδο το trimmer.

```
#include "p16f877.inc"

;Initialization

__CONFIG _CP_OFF & _WDT_OFF & _HS_OSC & _PWRTE_OFF & _CPD_OFF
& _WRT_ENABLE_ON & _BODEN_ON & _LVP_OFF & _DEBUG_ON

                Org 00

                ;initialize ADC

                bsf STATUS, RP0

                movlw b'00011111'    ;Κάνε τα πρώτα 5 pin της θύρας A είσοδο

                movwf TRISA

                movlw b'00000000'    ;Κάνε την θύρα B έξοδο

                movwf TRISB

                movlw b'01000010'    ;left justified, ADCS2=1, 3 Dig 5 Analog ch

                movwf ADCON1

                bcf STATUS, RP0

                movlw b'01000001'    ;101 Fosc/16, ch0, ADON

                movwf ADCON0

                clrf PORTB



                ;conversion

loop1   bcf PIR1, ADIF

                nop      ; περίμενε να σταθεροποιηθεί η έξοδος του κυκλώματος S&H
```

```
        nop
        nop
        bsf ADCON0, GO
wait    btfss PIR1, ADIF
        goto wait

;read ADC
        movf ADRESH, w
        movwf PORTB
        goto loop1

        end
```

3.4 Ασύγχρονη σειριακή επικοινωνία

Στην ασύγχρονη σειριακή επικοινωνία πρέπει να προκαθοριστεί το baud rate. Στην περίπτωση μας είναι στα 2404 bps. Για να πραγματοποιηθεί η σύνδεση χρειάζεται η ύπαρξη ενός τερματικού στον υπολογιστή καθώς και ένα καλώδιο usb σε RS232 το οποίο θα προσαρμοστεί στην θύρα DB9 του αναπτυξιακού. Εγώ χρησιμοποίησα το Terminal v1.91b το οποίο διανέμεται δωρεάν και μπορείτε να το βρείτε στο https://sites.google.com/site/terminalbpp/ . Στο τερματικό πρέπει να οριστούν baud rate 2400, data bits 8 και parity bit κανένα.

```
#include "p16f877.inc"

;Initialization

__CONFIG _CP_OFF & _WDT_OFF & _XT_OSC & _PWRTE_OFF & _CPD_OFF
& _WRT_ENABLE_ON & _BODEN_ON & _LVP_OFF & _DEBUG_ON
```

```
Org 0
call initialize
movlw b'00110011' ;δοκιμαστικό μοτίβο στην θύρα εξόδου
movwf PORTB
main
movlw d'30'
call send
call receive
movwf PORTB
loop goto loop

initialize
bsf STATUS, RP0
movlw 0x00
movwf TRISB          ;H Port B γίνεται έξοδος
movlw b'10000001'    ; Τα RC7 (RX) και RC0 γίνονται είσοδοι
movwf TRISC
movlw b'00100100'    ;Ενεργοποίηση της αποστολής, high speed baud rate
movwf TXSTA
movlw d'103'         ;2404 bps, εφόσον ο κρύσταλλος  είναι 4MHZ
movwf SPBRG
bcf STATUS,RP0
```

```
movlw b'10010000'    ;η θύρα είναι ενεργή, μεταφορά 8-bit
movwf RCSTA          ;συνεχής λήψη ενεργή
return

send
btfss PIR1,TXIF      ; Περίμενε μέχρι το TXIF γίνει  '1'
goto $-1             ; πριν φορτωθεί ο TXREG
movwf TXREG
return

receive
btfss PIR1,RCIF      ; όταν λαμβάνεται νέος χαρακτήρας RCIF=1
goto receive         ; περίμενε να λάβεις
movf RCREG,w         ; διάβασε τον RCREG
return
END
```

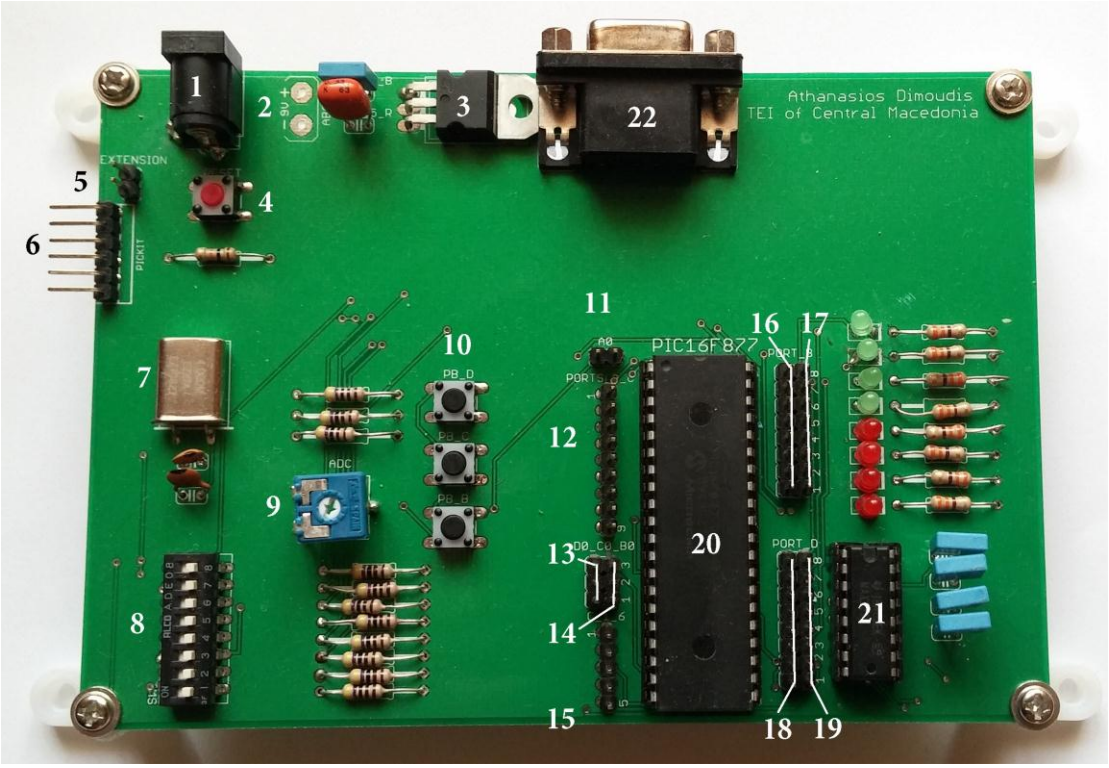
Κεφ. 4 Συμπεράσματα και μελλοντική επέκταση

Το προϊόν της πτυχιακής εργασίας είναι ένα αναπτυξιακό κύκλωμα το οποίο καλύπτει τις βασικές λειτουργίες του μικροελεγκτή και είναι ένα χρήσιμο εργαλείο για κάποιον αρχάριο ουτοσώστε να τον βοηθήσει να γνωρίσει τον PIC16F877, να μάθει να τον προγραμματίζει και εφόσον αποκτήσει μια σχετική εμπειρία χρησιμοποιώντας τα jumper που έχουν τοποθετηθεί σε όλες τις θύρες του PIC να πειραματιστεί με αυτές και να προσθέσει και άλλες λειτουργίες σε ένα ράστερ. Οι προσδοκίες μας έχουν καλυφθεί στο maximum και πιστεύω πως η ένταξη του στο εκπαιδευτικό πρόγραμμα του τμήματος και συγκεκριμένα στο εργαστήριο του μαθήματος ‘‘Προγραμματισμός συστημάτων σε πραγματικό χρόνο’’ , θα ωφελήσει τους φοιτητές δίνοντας τους μια ρεαλιστική εικόνα τέτοιων συστημάτων. Επίσης πιστεύω πως μπορεί να αποτελέσει μια πολύ καλή βάση για περαιτέρω εξέλιξη. Υπάρχει μια μεγάλη ποικιλία εφαρμογών που μπορούν να προσαρμοστούν στο συγκεκριμένο μικροελεγκτή. Αυτό που θα ήθελα να δω είναι η σύνδεση μια LCD οθόνης η οποία μπορεί να χρησιμοποιηθεί τόσο για προβολή αποτελεσμάτων αλλά γιατί όχι και για κάποιο παιχνίδι. Επίσης κάτι άλλο που βρίσκω ενδιαφέρον είναι η προσάρτηση διάφορων αισθητήρων όπως ήχου ή και θερμοκρασίας.

Βιβλιογραφία

1. Καλόμοιρος, Ι. ‘‘Αρχές Προγραμματισμού Πραγματικού Χρόνου: Εφαρμογές σε μικρά ενσωματωμένα συστήματα’’, 2012
2. PIC16F87X Data Sheet, 2001 Microchip Technology Inc.
3. PICkit™ 3 Programmer/Debugger User’s Guide, 2009 Microchip Technology Inc.
4. Πεκμεστζή, Κ. , ‘‘Συστήματα Μικροϋπολογιστών’’, Μικροελεγκτές AVR και PIC, 2009, Κεφάλαιο 5
5. Whilmsurt, T. ‘‘Designing Embedded Systems With PIC Microcontrollers’’, Principles And Applications, 2010
6. Blum, J. ‘‘Tutorial For Eagle’’, 1,2,3 Ιούνιος 2012, ανακτήθηκε από https://www.youtube.com/user/sciguy14 και http://www.jeremyblum.com/2012/06/09/tutorial-1-for-cadsoft-eagle-schematic-design/
7. Πανταζόπουλος, Π. ‘‘Οδηγός χρήσης Pic’’, http://www.electronics-lab.com/pic-in-greek/guide/Guide_to_use_the_PIC.htm
8. Καλόμοιρος, Ι ‘‘Embedded Systems Programming and Architectures’’, Department of Post Graduate studies in Communications and Informatics, TEI Κεντρικής Μακεδονίας
9. Device Configuration Overview, and Design Tips for the PICmicro® Microcontroller Configuration, 2001 Microchip Technology Inc
10. MPLAB® IDEUser’s Guide with MPLAB Editor and MPLAB SIM Simulator, 2009 Microchip Technology Inc.

Παράρτημα Α



Επεξήγηση αναπτυξιακού κυκλώματος

1. Υποδοχή τροφοδοτικού
2. Επαφές για σύνδεση μπαταρίας 9V
3. Ολοκληρωμένο 7805
4. Reset button
5. Ακροδέκτες για τροφοδοσία σε ράστερ, +5V το πάνω, γείωση το κάτω
6. Ακροδέκτες για σύνδεση PICkit3
7. Ταλαντωτής 4MHz
8. DIP switch 8pin, οι ακροδέκτες οδηγούνται στα pins του στοιχείου 19
9. Trimmer
10. 3 push buttons, οι ακροδέκτες οδηγούνται στα σε αυτούς του στοιχείου 13

11. Αριστερό Pin ο ακροδέκτης του Trimmer (9), δεξί Pin ακροδέκτης 0 θύρας A

12. Ακροδέκτες 0,1,2,3,4,5, θύρας A και 0,1,2 θύρας E (από πάνω προς τα κάτω)

13. Ακροδέκτες των Push buttons (10)

14. Ακροδέκτες των RD0, RC0 και RB0 (από πάνω προς τα κάτω)

15. Ακροδέκτες 1,2,3,4,5 θύρας C (από πάνω προς τα κάτω)

16. Ακροδέκτες 7,6,5,4,3,2,1,0 θύρας B (από πάνω προς τα κάτω)

17. Ακροδέκτες των LEDS (από πάνω προς τα κάτω)

18. Ακροδέκτες 7,6,5,4,3,2,1,0 θύρας D (από πάνω προς τα κάτω)

19. Ακροδέκτες του DIP switch (8) 7,6,5,4,3,2,1,0 (από πάνω προς τα κάτω)

20. PIC16F877

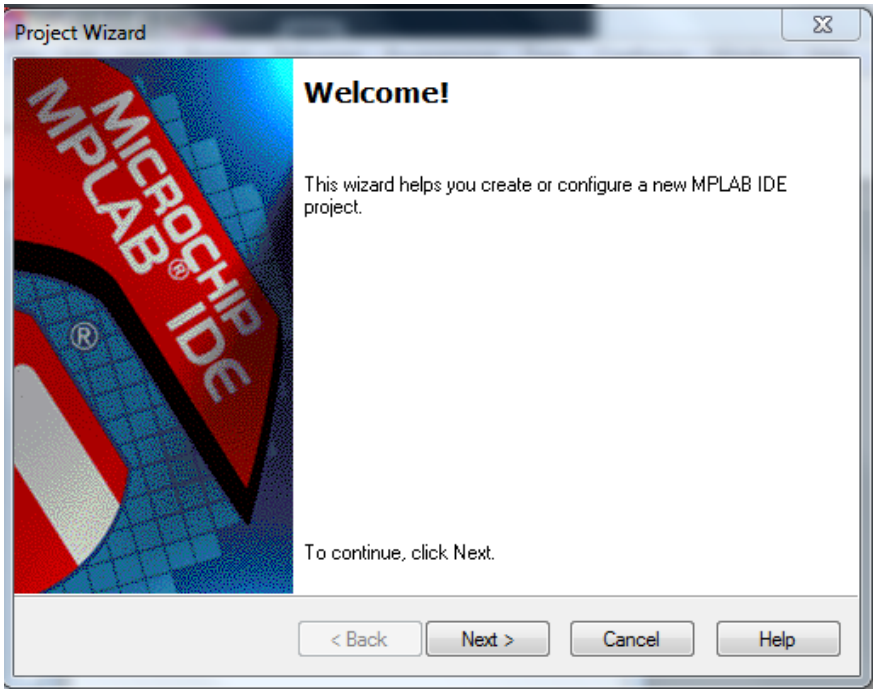
21. Max232

22. DB9 female

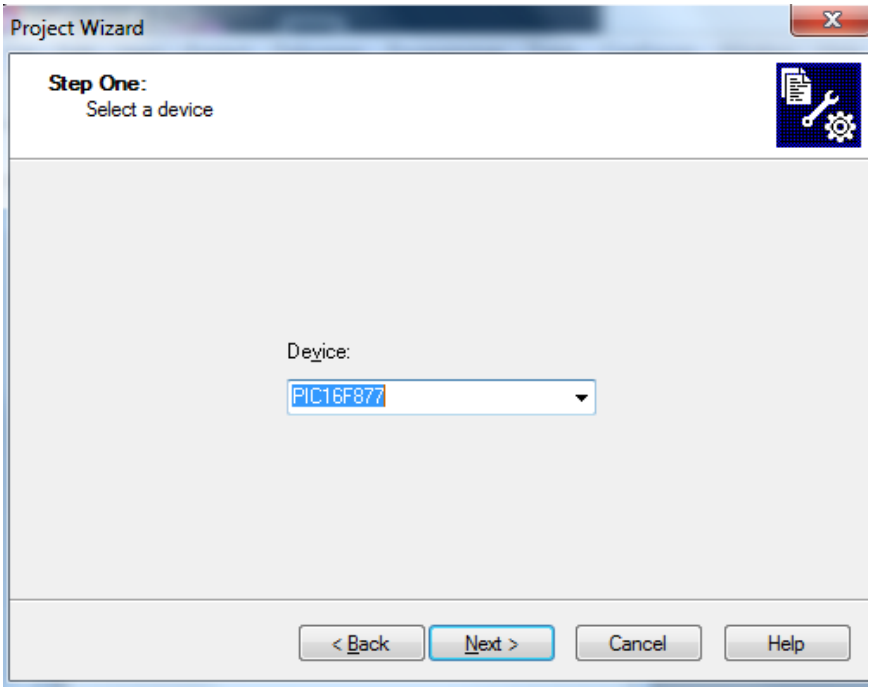
Παράρτημα Β

Αυτό το παράρτημα θα σας βοηθήσει να τρέξετε τις εκπαιδευτικές εφαρμογές στο MPLAB IDE. Η έκδοση που χρησιμοποιήθηκε είναι η 8.92 η οποία είναι ακόμη διαθέσιμη στο site της Microchip, αν και έχει κυκλοφορήσει και νεότερη έκδοση, και μπορείτε να τη βρείτε στο παρακάτω σύνδεσμο http://www.microchip.com/pagehandler/en-us/devtools/dev-tools-parts.html

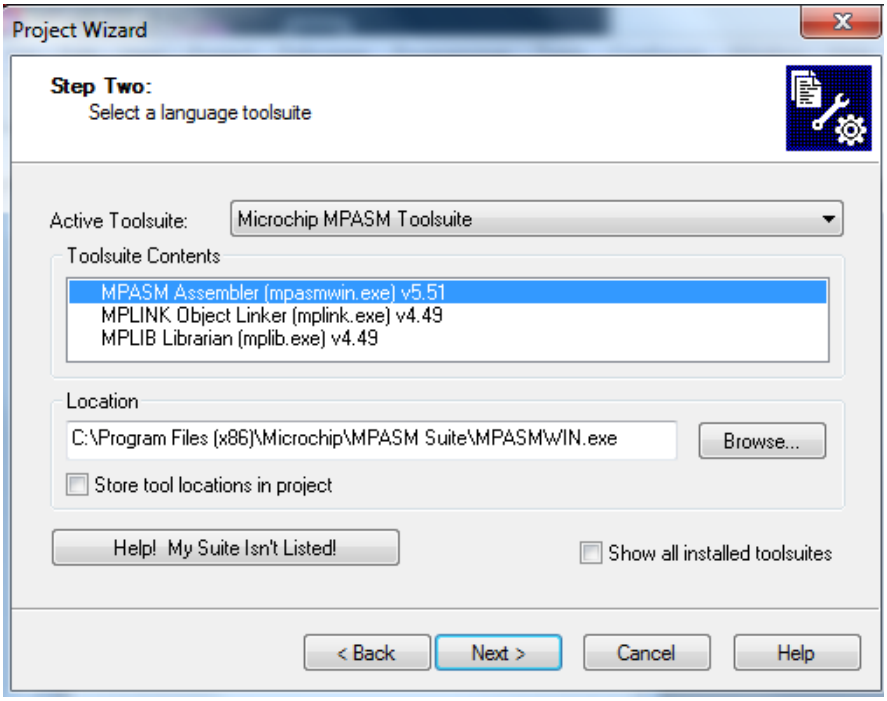
Αφού κατεβάσετε και εγκαταστήσετε το MPLAB,συνδέστε το PICkit3 στον υπολογιστή με το USB καλώδιο. Μόλις το MPLAB το εντοπίσει θα κατεβάσει και εγκαταστήσει αυτόματα το πιο πρόσφατο firmware για το PICkit3. Εφόσον γίνει αυτό επιλέγετε Project->Project Wizard και ανοίγει ένα παράθυρο που θα σας βοηθήσει να δημιουργήσετε το project.



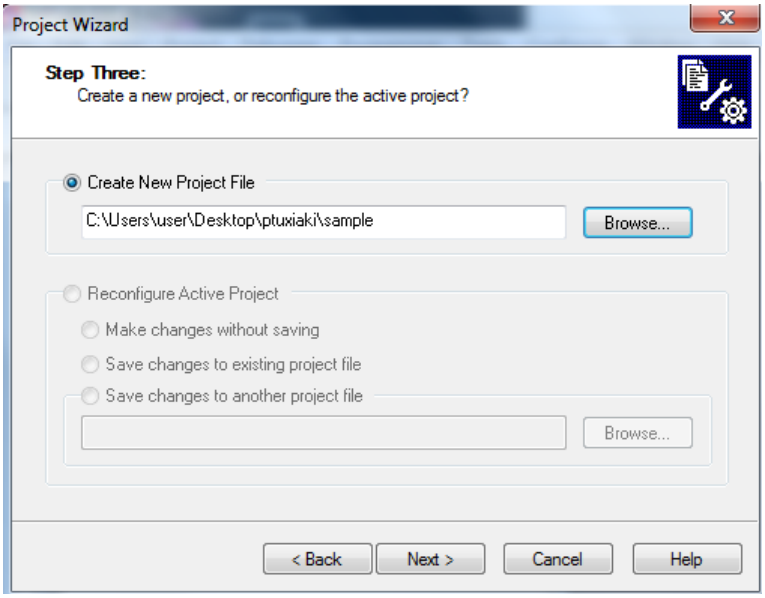
Αρχικά πατάτε next και μετά επιλέγετε από τη λίστα των μικροελεγκτών τον PIC16F877 και πατάτε Next



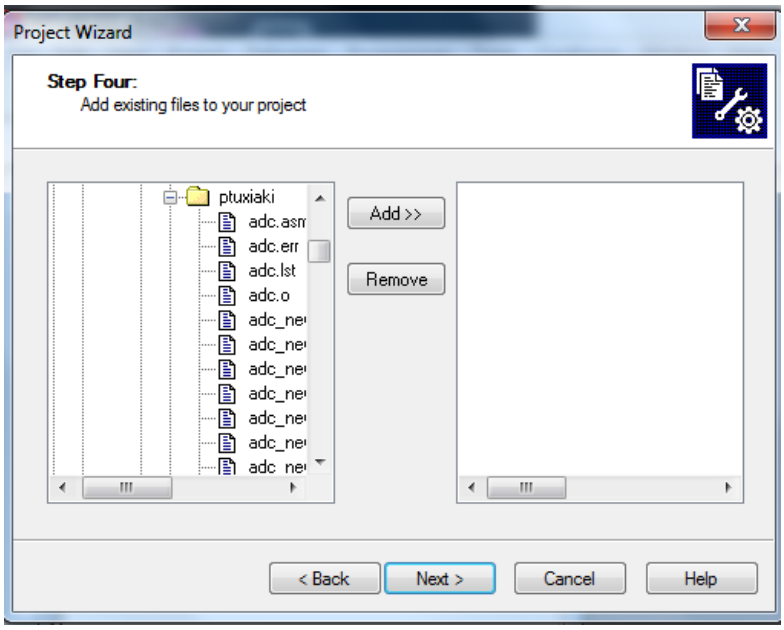
Next και στην επόμενη καρτέλα



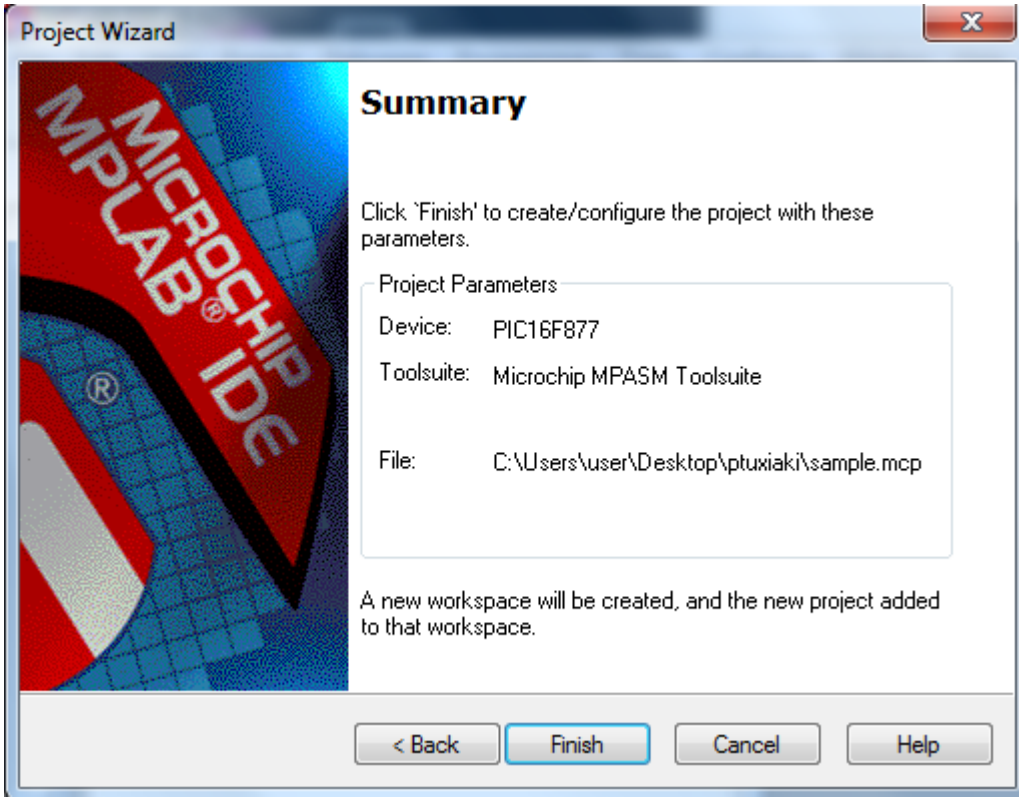
έπειτα δίνεται το όνομα του Project και επιλέγετε από το κουμπί Browse τον φάκελο που επιθυμείτε να το αποθηκεύσετε και πατάτε Next



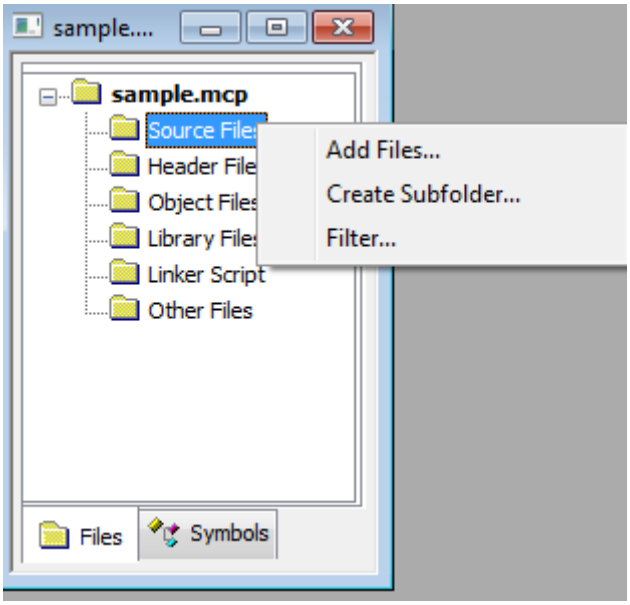
Next στην επόμενη εκτός και αν έχετε κάποιο έτοιμο αρχείο με κώδικα να προσθέσετε στο project



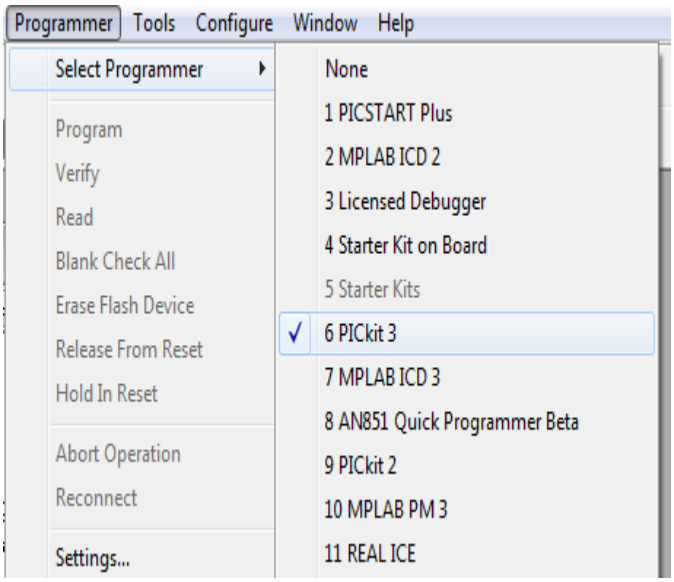
Και τέλος Finish.

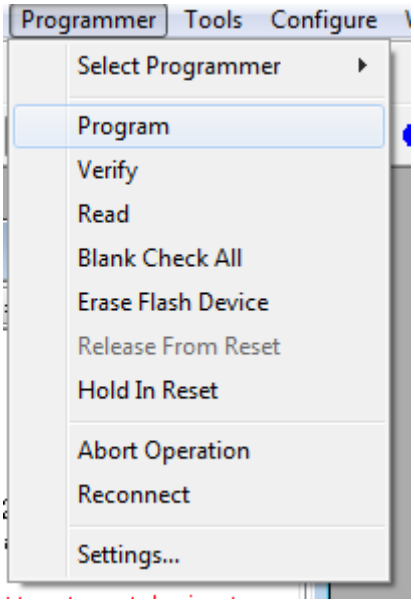


Στη συνέχεια επιλέγετε File->New και στο παράθυρο που ανοίγει γράφεται τον κώδικα που επιθυμείτε και κάνετε save as δίνοντας όποιο όνομα θέλετε και προσέχετε να το αποθηκεύσετε σε μορφή assembly. Το δημιουργηθέν αρχείο πρέπει να το συνδέσετε με το project σας και για να το κάνετε αυτό επιλέγετε και προσέχετε να είναι τσεκαρισμένη η επιλογή Project, και εφόσον είναι, στο παράθυρο που έχει το όνομα του project σας με την κατάληξη .mcw στην καρτέλα Source Files κάνετε δεξί κλικ Add Files και προσθέτετε έτσι το αρχείο με τον κώδικα που αποθηκεύσατε προηγουμένως.



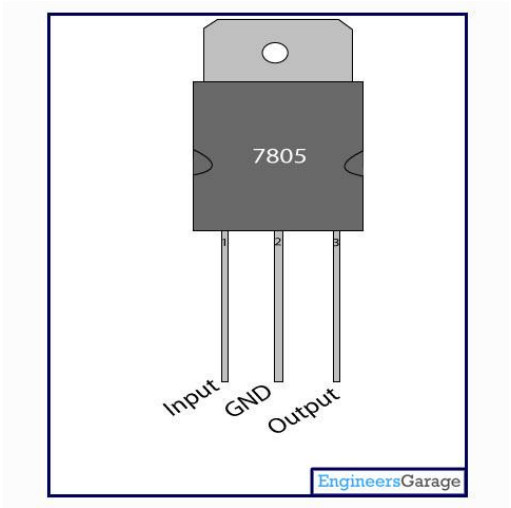
Στην συνέχεια επιλέγετε Project->Build All και τέλος στην καρτέλα Programer επιλέγετε το PICkit3 και πατάτε Program και έτσι ο μικροελεγκτής έχει πλέον προγραμματιστεί.



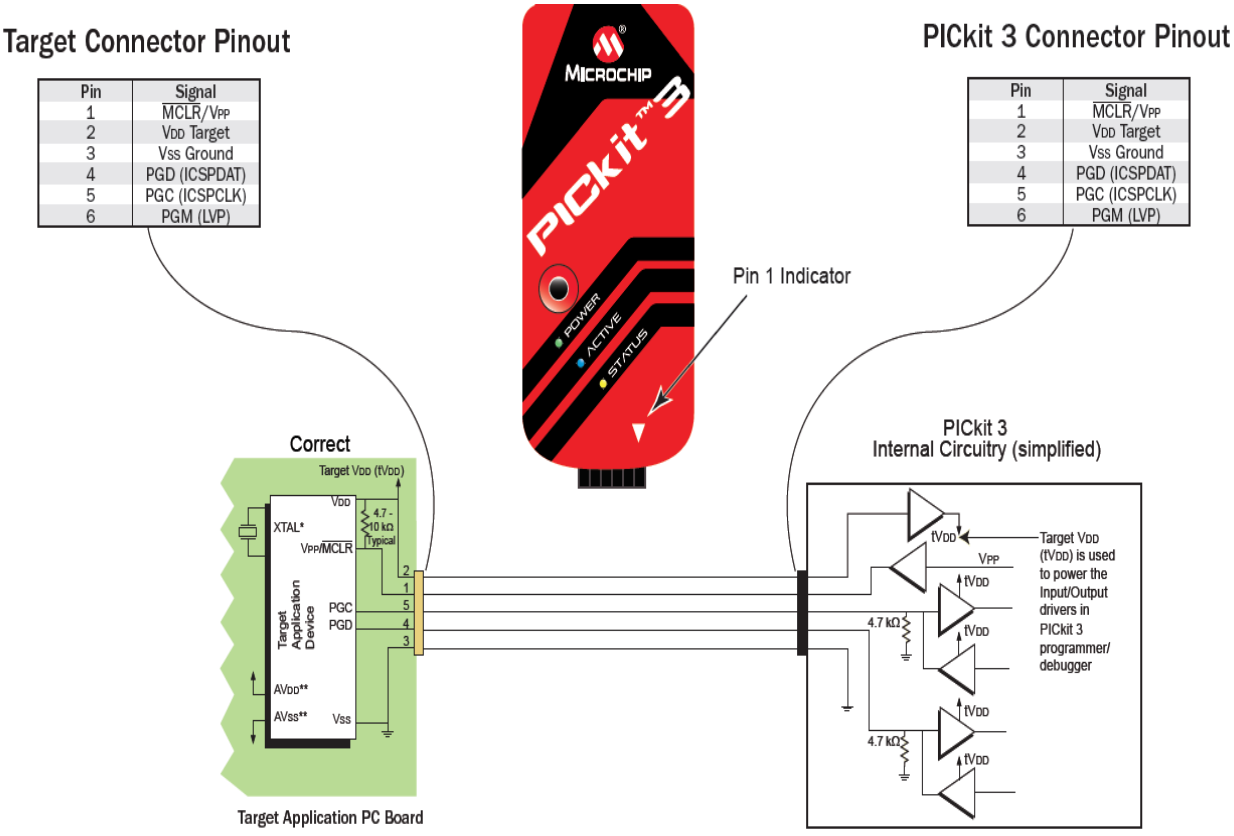


Προτείνεται να αποσπάτε από το κύκλωμα το PICkit3 μετά τον προγραμματισμό του μικροελεγκτή διότι σε εφαρμογές όπως η σειριακή επικοινωνία ενδέχεται να επηρεάσει τα αποτελέσματα κατά το reset .

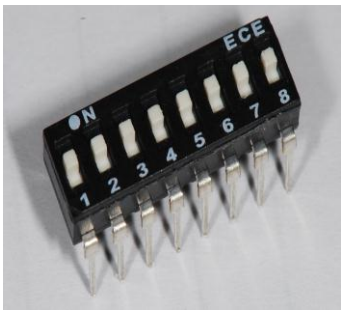
Παράρτημα Γ



PIC16F877 Ολοκληρωμένο 7805



PICkit3 και πίνακας σύνδεσης του



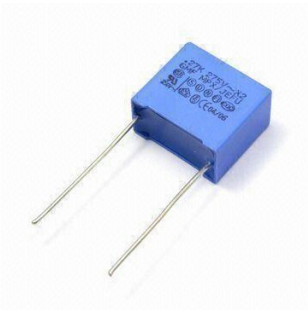
DIP switch 8bit

LED

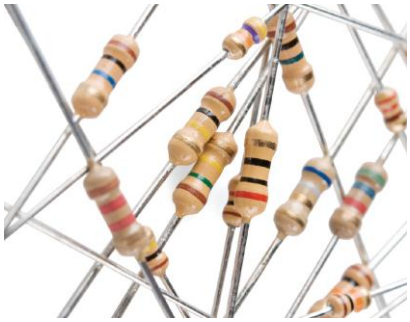
Push button

Κεραμικός πυκνωτής 22μF

Trimmer



Πυκνωτής πολυπροπενίου

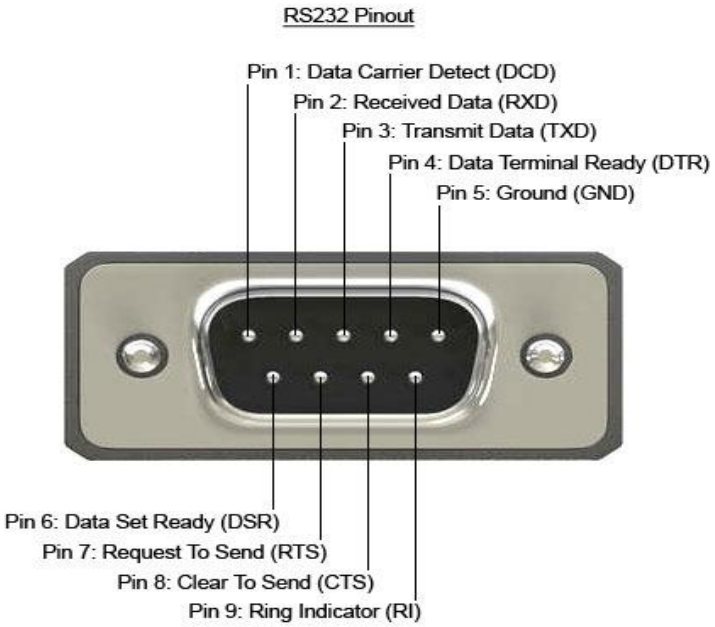


Αντιστάσεις

Ταλαντωτής

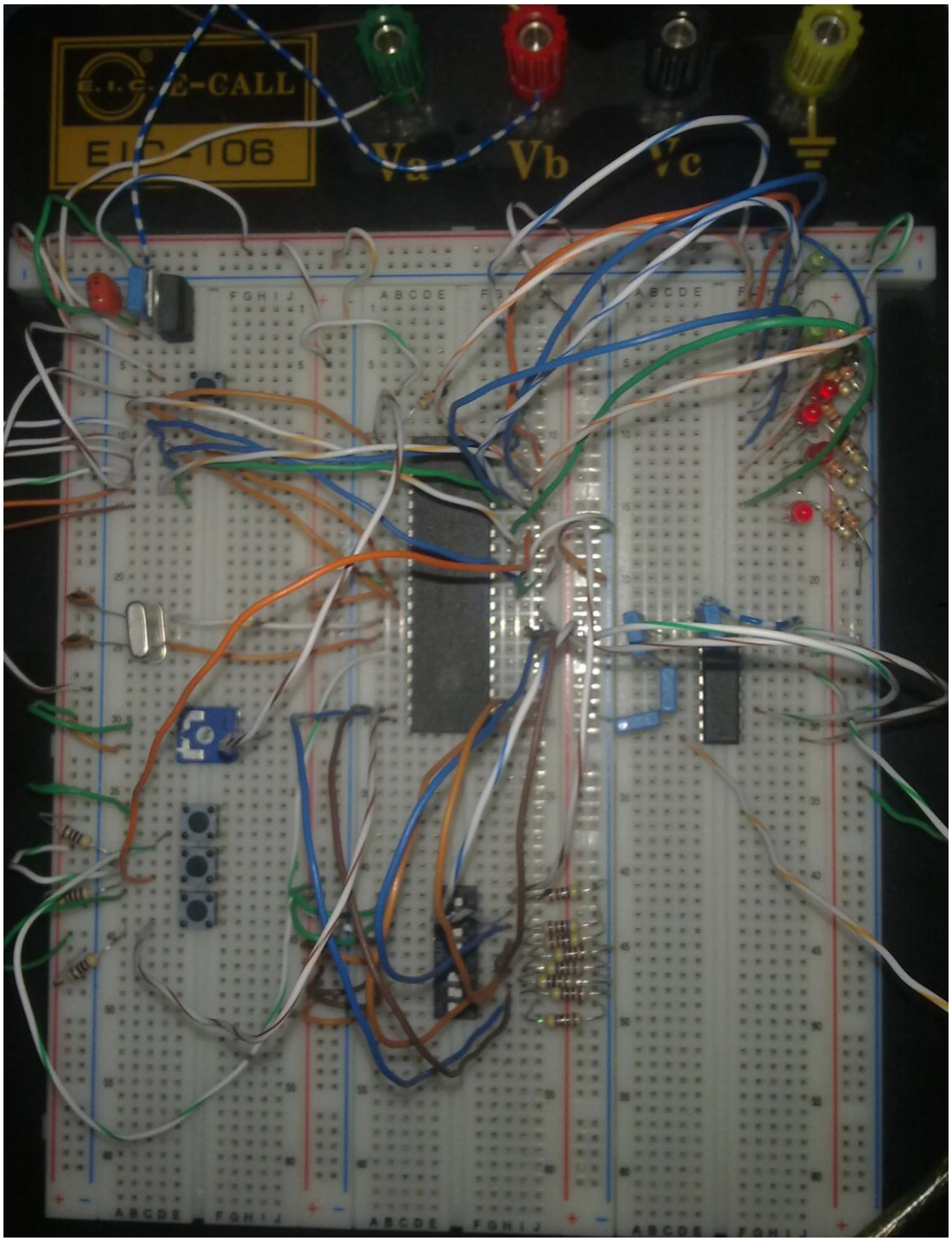
RS232 line type and logic level	RS232 voltage	TTL voltage to/from MAX232
Data transmission (Rx/Tx) logic 0	+3 V to +15 V	0 V
Data transmission (Rx/Tx) logic 1	-3 V to -15 V	5 V
Control signals (RTS/CTS/DTR/DSR) logic 0	-3 V to -15 V	5 V
Control signals (RTS/CTS/DTR/DSR) logic 1	+3 V to +15 V	0 V

Αντιστοίχιση τάσεων προτύπου RS232 σε TTL

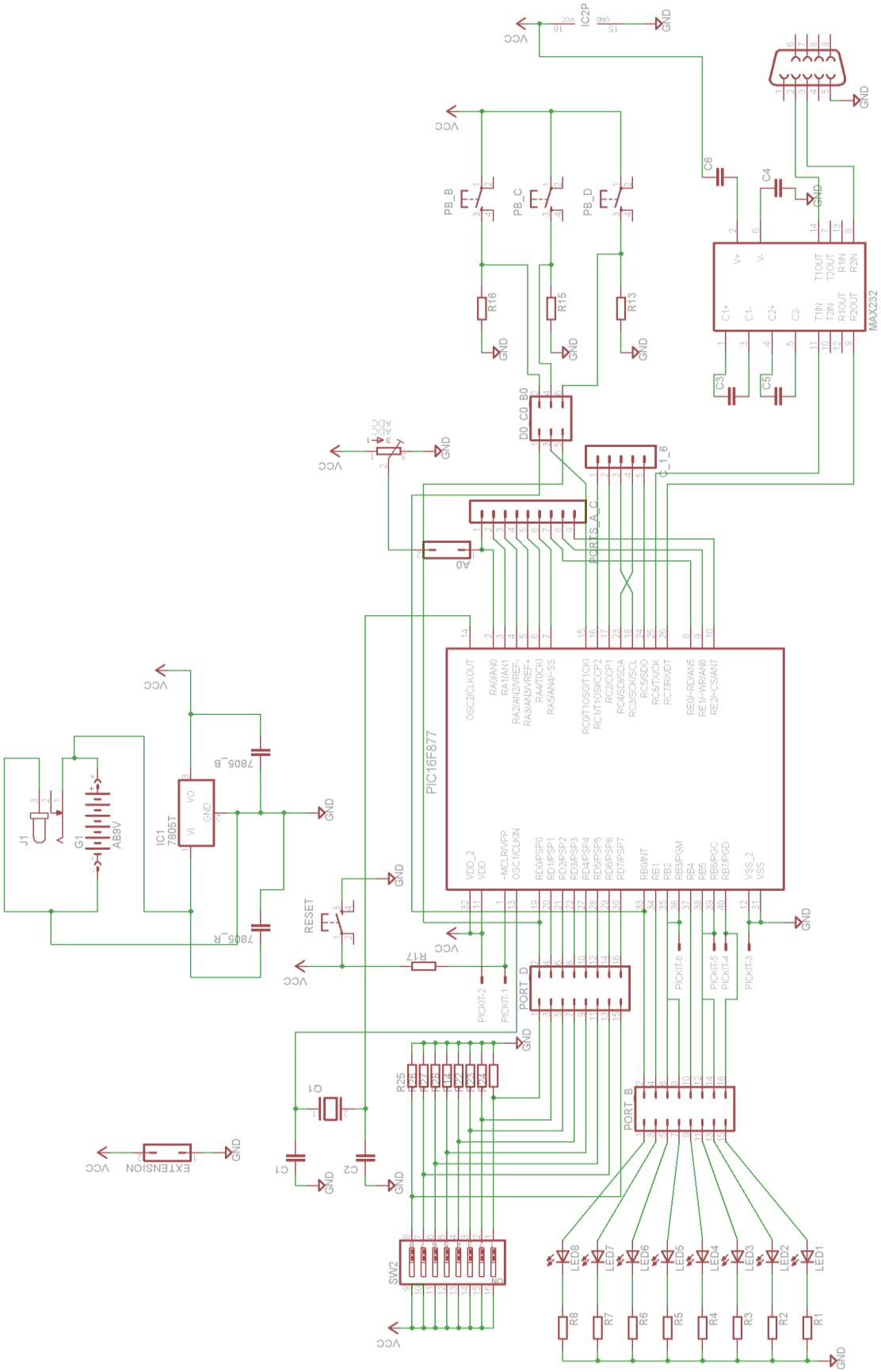


Ανάλυση ακροδεκτών θύρας RS232

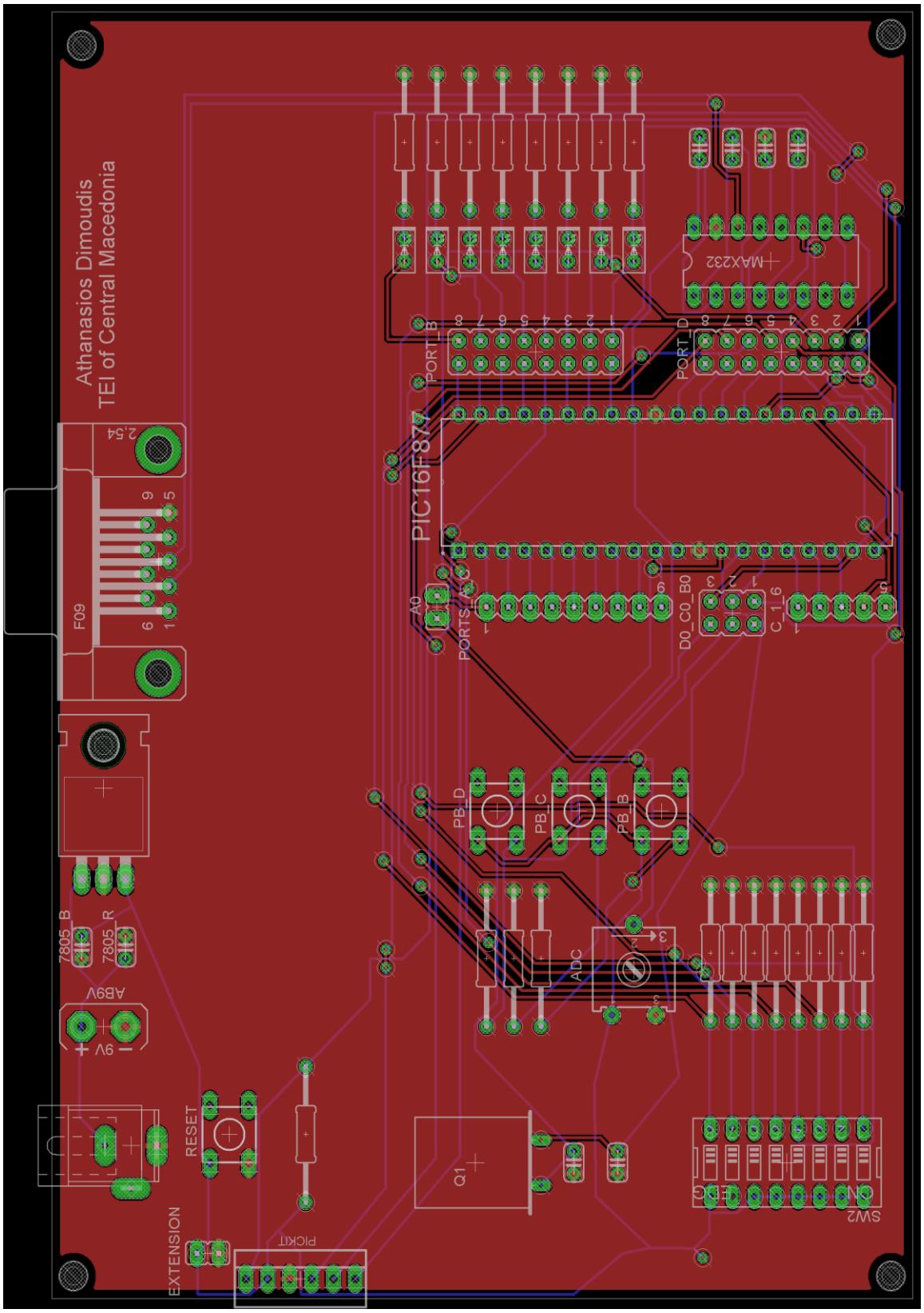
Αντάπτορας usb σε RS232



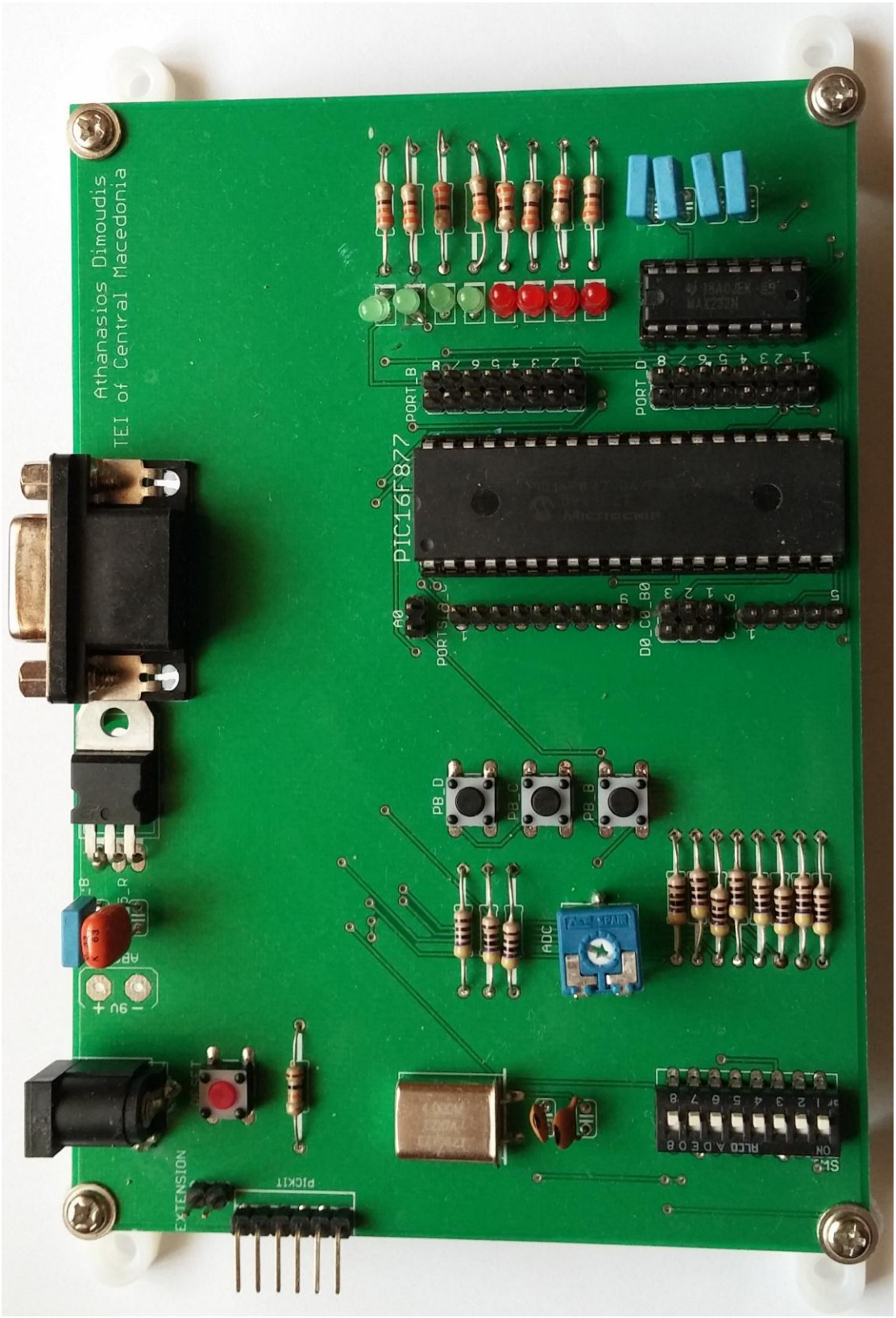
Το δοκιμαστικό κύκλωμα στο ράστερ σε πιο ευκρινές μέγεθος



Το σχηματικό σε πιο ευκρινές μέγεθος



Το τελικό σχέδιο του board σε πιο ευκρινές μέγεθος



Το προϊόν της πτυχιακής σε πιο ευκρινές μέγεθος