ΣΧΕΔΙΑΣΗ ΨΗΦΙΑΚΟΥ ΠΑΛΜΟΓΡΑΦΟΥ 8 ΚΑΝΑΛΙΩΝ ΧΑΜΗΛΩΝ ΣΥΧΝΟΤΗΤΩΝ

Εκπόνηση πτυχιακής εργασίας

Κοντάκος Κυριάκος 2967
Ευθυμιάτος Κωνσταντίνος 4126

Εποπτεύων καθηγητής: Κ. Νομικός

Περιεχόμενα

Κεφάλαιο 1. Εισαγωγή

Στην σημερινή εποχή της τεχνολογίας και της πληροφορίας η ύπαρξη ενός οργάνου παρατήρησης σημάτων σε πραγματικό χρόνο και καταγραφής αυτών είναι απαραίτητη. Για την παρατήρηση έχουμε τους παλμογράφους, αναλογικούς και ψηφιακούς, οι οποίοι μπορεί να έχουν πολλές δυνατότητες και επιλογές αλλά έχουν το μειονέκτημα του ότι είναι ογκώδεις. Υπάρχουν βέβαια και μικροί φορητοί αλλά το ήδη υψηλό κόστος ενός παλμογράφου ανεβαίνει ακόμα υψηλότερα. Επιπλέον δεν έχουν πάντα την δυνατότητα καταγραφής των σημάτων. Μια εναλλακτική λύση θα μπορούσε να είναι ένας καταγραφέας δεδομένων (data logger). Και αυτή η λύση όμως έχει κάποια αρνητικά καθώς η χρήση της συσκευής αυτής είναι η καταγραφή των δεδομένων και όχι η παρατήρηση σημάτων σε πραγματικό χρόνο.

Η δικιά μας προσέγγιση βρίσκεται κάπου στο ενδιάμεσο των δύο προαναφερθέντων. Δηλαδή φτιάξαμε μία συσκευή η οποία μπορεί να μετατρέψει τον ηλεκτρονικό υπολογιστή σε έναν ψηφιακό παλμογράφο μικρής, σχετικά, δειγματοληψίας και παράλληλα μπορεί να αποθηκεύει τα σήματα που δείχνει στην οθόνη, στον σκληρό δίσκο, εκτελώντας χρέη data logger. Επιπροσθέτως η συσκευή μας μπορεί να καταγράφει ταυτόχρονα οκτώ (8) κανάλια εισόδου, κάτι που μπορούν να κάνουν μόνο εξειδικευμένες και επομένως ακριβές συσκευές της αγοράς.

Κεφάλαιο 2. Σχεδίαση του δικού μας συστήματος

Το σύστημα μας βασίζεται κυρίως σε δύο ολοκληρωμένα, το AT90S2313 της εταιρίας ATMEL και το MAX197 της εταιρίας MAXIM.

Το πρώτο είναι ένας μικροελεγκτής του οποίου οι δυνατότητες επιγραμματικά είναι οι παρακάτω.

- Αρχιτεκτονική AVR® RISC
- AVR υψηλής απόδοσης(σχέση ταχύτητας εκτέλεσης εντολών προς κατανάλωση), αρχιτεκτονικής RISC
  - 118 εντολές – οι περισσότερες ενός κύκλου
  - 32X8 καταχωρητές γενικής χρήσης
  - Εκτέλεση 10 MIPS στα 10MHz
- Μνήμη δεδομένων και προγράμματος
  - 2K bytes προγραμματιζόμενης μνήμης συστήματος Flash με αντοχή 1000 κύκλους εγγραφής διαγραφής
  - 128 bytes SRAM
  - 128 bytes EEPROM
  - κλείδωμα της μνήμης FLASH και EEPROM για ασφάλεια δεδομένων.
- Περιφερειακά
  - 1 μετρητής/χρονιστής 8-bit με ξεχωριστό Prescaler
  - 1 μετρητής/χρονιστής 16-bit με ξεχωριστό Prescaler, λειτουργίες capture, compare και PWM 8,9 ή 10 bit
  - εσωτερικό αναλογικό συγκριτή
  - προγραμματιζόμενος χρονιστής Watchdog
  - SPI σειριακό ενδιάμεσο για τον προγραμματισμό του μικροελεγκτή μόνο με 4 καλώδια
  - Full duplex UART

Το δεύτερο ολοκληρωμένο είναι ένας A/D converter διακριτικής ικανότητας 12-bit. Επιγραμματικά οι δυνατότητες του είναι οι παρακάτω.

- Ανάλυση 12-bit
- Τροφοδοσία +5V
- Περιοχή τιμών εισόδου επιλέξιμη μέσω λογισμικού: ±10V,±5V,0-10V,0-5V
- Πολυπλέκτης εισόδου με προστασία υπερτάσεων (±16.5V)
- 8 αναλογικά κανάλια εισόδου

- 6μs χρόνος μετατροπής, 100ksps ρυθμός δειγματοληψίας
- εσωτερικός ή εξωτερικός έλεγχος καταγραφής
- εσωτερική 4.096V ή εξωτερική τάση αναφοράς
- δύο καταστάσεις χαμηλής κατανάλωσης
- εσωτερικό ή εξωτερικό ρολόι με κρύσταλλο

2.1 Τεχνικά χαρακτηριστικά

- Τροφοδοσία: 240V AC/ 5VA
- Εύρος τάσεων εισόδου: 0..5V, 0..10V, ±5V, ±10V (Προγραμματιζόμενο)
- A/D Κανάλια: 8 (Προγραμματιζόμενο)
- Περίοδος δειγματοληψίας: 1..255msec (Προγραμματιζόμενο)
- Έξοδος δεδομένων: Σειριακά μέσω RS232 .
- Επικοινωνία: RS232, 115200 bps, 8, N, 1

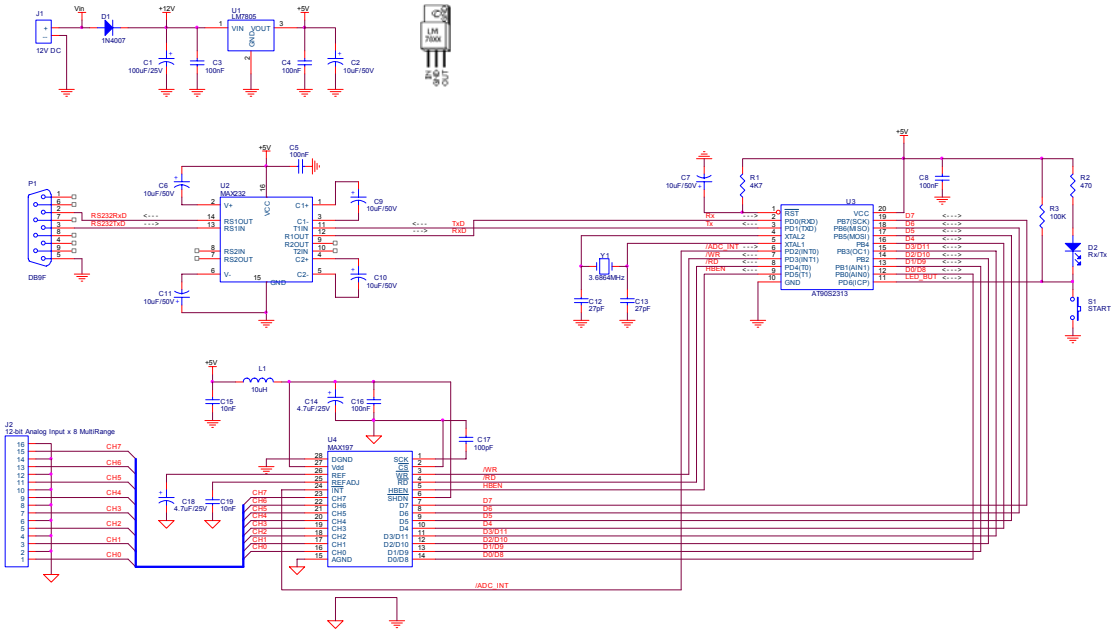
2.2 Σχεδίαση κυκλώματος

2.2.1 Σχηματικά

α) Κυρίως κύκλωμα

Στην εικόνα 1 βλέπουμε το σχηματικό κύκλωμα της κύριας πλακέτας της συσκευής μας. Η σχεδίαση έχει γίνει με το πρόγραμμα OrCAD για Windows®.

Εικόνα 1 – Σχηματικό κυρίως κυκλώματος.



Στις εικόνες που ακολουθούν θα δούμε τα επιμέρους τμήματα του κυκλώματος μας.

Καταρχήν στην εικόνα 2 βλέπουμε το κύκλωμα τροφοδοσίας το οποίο έχει σαν κύριο στοιχείο του ένα σταθεροποιητή τάσης LM7805.

Τα 12 Volt DC(18V) τα παίρνουμε από κατάλληλο κύκλωμα ανόρθωσης με χρήση μετασχηματιστή από την δεύτερη πλακέτα που περιέχει και το γενικό τροφοδοτικό.

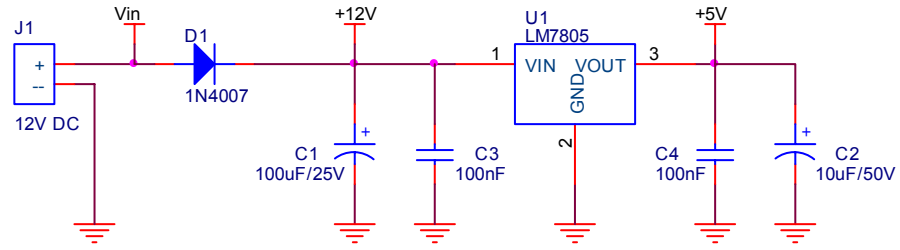
Για προστασία από τυχόν εσφαλμένη ανάστροφη συνδεσμολογία, έχουμε προσθέσει την δίοδο D1(1N4007) που είναι μια δίοδος γενικής χρήσης με ονομαστικό επιτρεπόμενο ρεύμα διέλευσης 1A, υπεραρκετό για το κύκλωμα μας.

Ο πυκνωτής C1(100μF/25V) φροντίζει ώστε να υπάρχει πάντα διαθέσιμο αρκετό φορτίο, ώστε να καλύπτει τις ανάγκες ρεύματος του κυκλώματος κατά τις απότομες αλλαγές κατάστασης, δεδομένου ότι πρόκειται να τροφοδοτήσει ψηφιακό κύκλωμα.

Οι πυκνωτές C1 και C4(100nF) βρίσκονται κοντά στους ακροδέκτες του ολοκληρωμένου σταθεροποίησης U1(7805) αποτρέποντας τυχόν ταλαντώσεις στις οποίες είναι δυνατό να εισέλθει το ολοκληρωμένο κατά την προσπάθεια του να σταθεροποιήσει την τάση εξόδου του στα 5V DC.

Ο πυκνωτής C2(10μF/50V) φροντίζει και αυτός να δίνει το φορτίο του όταν χρειάζεται ώστε να ελαχιστοποιούνται οι βυθίσεις της τάσης στην γραμμή των +5V και να λειτουργεί ομαλότερα και ο σταθεροποιητής, αλλά κυρίως ο μικροελεγκτής.

Εικόνα 2 – Σχηματικό τμήματος τροφοδοσίας κυρίου κυκλώματος.



Στην εικόνα 3 έχουμε το «μυαλό» του συστήματος μας ένα μικροελεγκτή AT90S2313 της οικογένειας AVR της ATMEL. Ο χρονισμός του βρίσκεται στα 3.6864 MHz για λόγους ασφαλούς ασύγχρονης μετάδοσης των δεδομένων μέσω του UART στον Η/Υ. Η συχνότητα αυτή του κρυστάλλου την έχουμε επιλέξει έτσι ώστε σε οποιοδήποτε ρυθμό Baud να έχουμε μηδενικό σφάλμα. Είναι δηλαδή μια «UART friendly» συχνότητα.

Ο πυκνωτής C8(100nF) συμβάλει στην μείωση του θορύβου υψηλής συχνότητας πάνω στην γραμμή των 5V που προέρχεται από τις γρήγορες αλλαγές κατάστασης στο εσωτερικό του μικροελεγκτή.

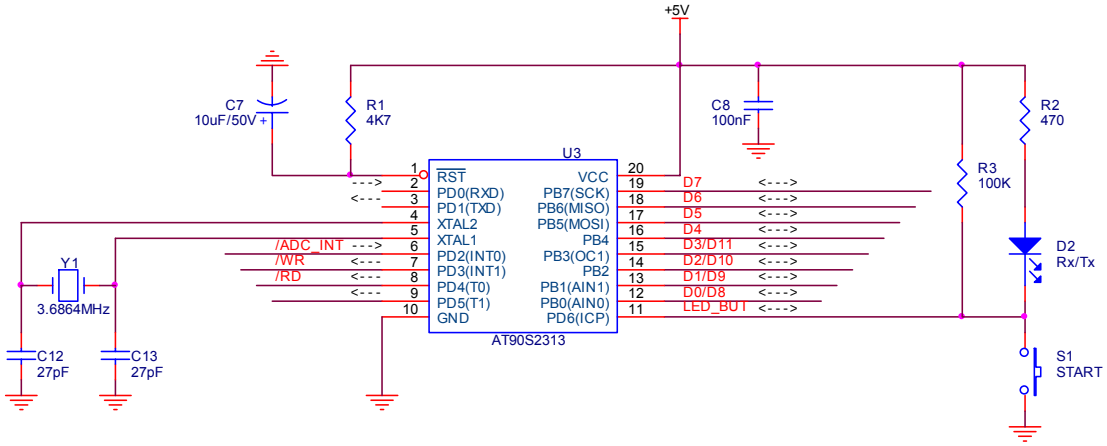
Το δικτύωμα R1(4K7), C7(10μF/50V) παράγει τον παλμό RESET χαμηλού δυναμικού που απαιτείται για να ξεκινήσει σωστά την λειτουργία του ο AVR, όταν εφαρμόζεται η τάση τροφοδοσίας.

Το εσωτερικό UART του AVR καταλήγει στους ακροδέκτες 2(RxD) και 3(TxD) όπου είναι συνδεδεμένο το κατάλληλο ενδιάμεσο κύκλωμα(MAX232) για την επικοινωνία με τον υπολογιστή.

Στον ακροδέκτη 11 έχουμε συνδέσει ένα LED(D2) με την αντίστοιχη αντίσταση περιορισμού ρεύματος(R2, 470Ω), αλλά και ένα διακόπτης πίεσης(S1). Το LED και ο διακόπτης αποτελούν το MMI(Man Machine Interface) της κατασκευής μας καθώς μέσω του LED έχουμε ένδειξη της στιγμής που γίνεται η μέτρηση στον ADC και μέσω του διακόπτη μπορούμε να ξεκινήσουμε ή να σταματήσουμε χειροκίνητα μετρήσεις.

Όλοι οι υπόλοιποι I/O ακροδέκτες του μικροελεγκτή είναι αφιερωμένοι στην επικοινωνία με το ολοκληρωμένο του αναλογικού σε ψηφιακό μετατροπέα(ADC).

Εικόνα 3 – Σχηματικό τμήματος μικροελεγκτή.



Στην συνέχεια βλέπουμε τον A/D converter του συστήματος μας, το MAX197 της εταιρίας Maxim/Dallas (εικόνα 4).

Η τροφοδοσία του A/D από τα 5V γίνεται μέσω του πηνίου L1(10μH) το οποίο μαζί με τους πυκνωτές C15(10nF) και C14(4,7μF/25V), C16(100nF) αποτελούν ένα φίλτρο χαμηλών συχνοτήτων τύπου “Π” που σκοπό έχει την απομάκρυνση τυχόν παρασιτικών συχνοτήτων και άλλων θορύβων που έχουν επικαθίσει πάνω στην γραμμή των +5V και προέρχονται από το υπόλοιπο κύκλωμα που τροφοδοτείται από τα +5V, συγκεκριμένα από τα U2(MAX232) και U3(AT90S2313).

Ο πυκνωτής C17(100pF) που συνδέεται στον ακροδέκτη 1 του U4(MAX197) χρησιμεύει στην παραγωγή του σήματος χρονισμού του μετατροπέα και θέτει αυτόν τον χρονισμό στα 1.56MHz περίπου, όταν έχουμε επιλέξει εσωτερικό χρονισμό με κατάλληλες ρυθμίσεις που γίνονται στον καταχωρητή του MAX197 από τον AVR.

Η τάση αναφοράς που χρειάζεται ο A/D παράγεται από το ίδιο το ολοκληρωμένο MAX197 και έχει τιμή 4,096V. Για το φιλτράρισμα και την σταθεροποίηση της τάσης αυτής που

εμφανίζεται στον ακροδέκτη 26, συνδέσαμε σε αυτόν, τον πυκνωτή C18(4,7μF/25V) και στον ακροδέκτη 25 τον πυκνωτή C19(10nF).

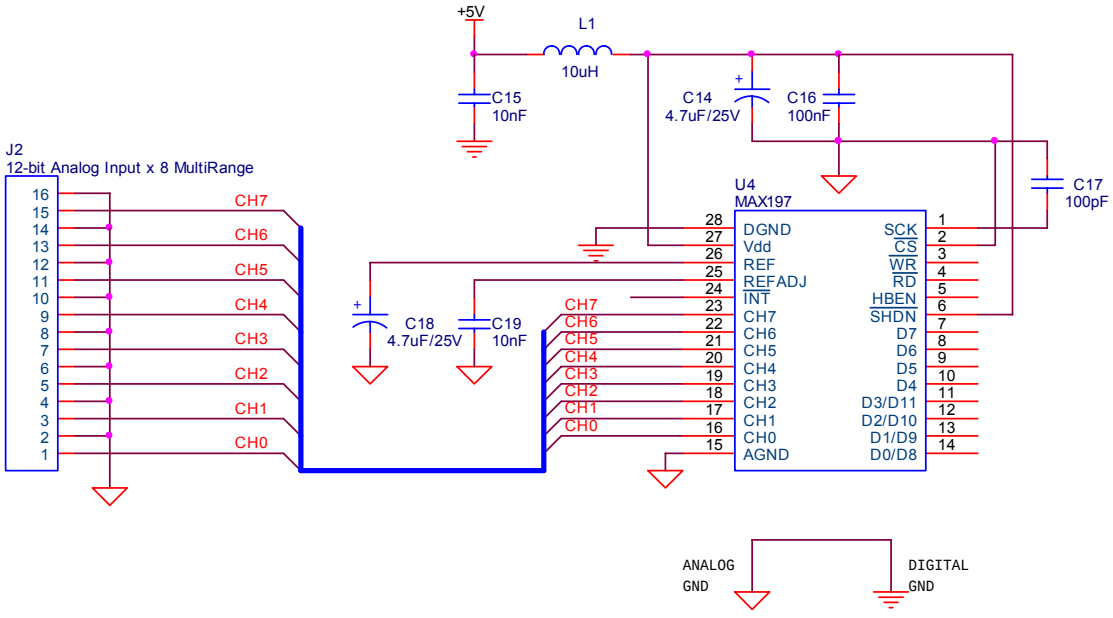
Στους ακροδέκτες 16 ως 23 βρίσκονται τα οκτώ αναλογικά κανάλια εισόδου του A/D και τα οποία καταλήγουν σε ένα συνδετήρα(κλέμα) αντίστοιχων θέσεων.

Τα σήματα /CS(Chip Select) και /SHDN(Shutdown) επιλέξαμε να τα έχουμε μόνιμα συνδεδεμένα σε κατάλληλο δυναμικό (/CS=0V) ώστε να είναι μόνιμα επιλεγμένο το ολοκληρωμένο και να μην μπαίνει ποτέ σε λειτουργία χαμηλής κατανάλωσης (/SHDN=5V) καθώς κάτι τέτοιο δεν είναι απαραίτητο.

Το σήματα /INT, HBEN, /WR, /RD και το data bus(pin 7 ως 14) συνδέονται με τον μικροελεγκτή AVR και αποτελούν τον δίαυλο ελέγχου και μεταφοράς δεδομένων. Η επικοινωνία γίνεται με αυτό το παράλληλο interface ανά οκτώ bit δεδομένων κάθε φορά κατά τρόπο που θα αναλυθεί στο κεφάλαιο 2.2.3 όπου αναλύεται το λογισμικό του μικροελεγκτή.

Ένα σημαντικό σημείο του κυκλώματος είναι το σημείο όπου ενώνονται οι δύο γραμμές γείωσης του κυκλώματος, η αναλογική και η ψηφιακή γείωση. Οι δύο αυτές γραμμές, αν και αποτελούν και οι δύο το σημείο επιστροφής του ρεύματος, είναι ουσιαστικό να αποτελούν χωριστές γραμμές στην πλακέτα και να ενωθούν σε ένα μόνο συγκεκριμένο σημείο ώστε να περιοριστεί και να αποτραπεί η είσοδος του ψηφιακού θορύβου στο αναλογικό μέρος του κυκλώματος.

Εικόνα 4 – Σχηματικό τμήματος ADC.



Το τελευταίο κομμάτι του κυρίως κυκλώματος είναι το MAX232 της εταιρίας Maxim/Dallas που επιτρέπει την επικοινωνία του μικροελεγκτή μας με τον Η/Υ μέσω της σειριακής πόρτας (εικόνα 5).

Η επικοινωνία γίνεται σύμφωνα με το πρωτόκολλο RS232/EIA232 το οποίο προβλέπει τρία καλώδια σύνδεσης ως εντελώς απαραίτητα για μια αμφίδρομη επικοινωνία. Ένα καλώδιο για εκπομπή(Tx), ένα για λήψη(Rx) και ένα ως κοινό σημείο αναφοράς (GND). Τα ηλεκτρικά σήματα στις γραμμές Rx, Tx είναι της μορφής +12V και -12V. Τα +12V αντιστοιχούν σε λογικό “0” (space) και τα -12V σε λογικό “1” (mark) ενώ οι στάθμες από -3V σε +3V είναι μη αναγνωρίσιμες σαν αποδεκτή λογική κατάσταση. *(Στην πράξη οι περισσότερες συσκευές με σειριακή θύρα RS232 αναγνωρίζουν σαν λογικό “0” οποιαδήποτε τάση από +3V ως +12V και σαν λογικό “1” οποιαδήποτε τάση μικρότερη από +3V).*

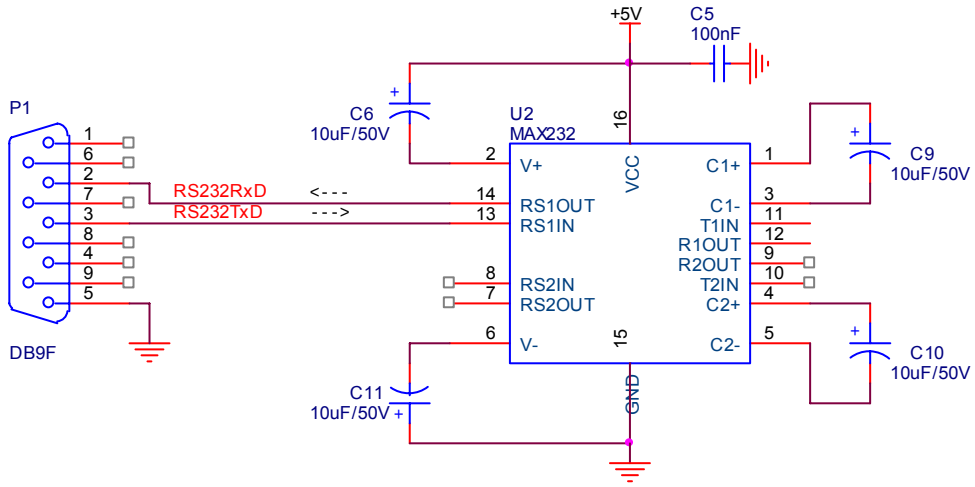
Ο μικροελεγκτής παράγει και δέχεται σήματα των 0V και +5V οπότε χρειαζόμαστε τον μετατροπέα που φαίνεται στην εικόνα 5.

Ο μετατροπέας(MAX232) δέχεται σήματα TTL/CMOS της τάξης των 5V στον ακροδέκτη 11 και τα μετατρέπει σε -12V στον ακροδέκτη 14 ενώ τα 0V τα μετατρέπει σε +12V. Αντίστοιχα, δέχεται σήματα της τάξης των -12V στον ακροδέκτη 13 και τα μετατρέπει σε 0V στον ακροδέκτη 12 ενώ τα +12V τα μετατρέπει σε 0V. Για να το καταφέρει αυτό χρησιμοποιεί ένα εσωτερικό κύκλωμα παραγωγής συμμετρικής τροφοδοσίας ±12V. Το κύκλωμα αυτό λέγεται αντλία φόρτισης και βασίζεται σε δύο πυκνωτές, έναν για κάθε πολικότητα, τους C9 και C10(10μF/50V). Στην σταθεροποίηση και την μείωση της κυμάτωσης στις εξόδους των ±12V(pin 2 & 6) συμβάλλουν οι πυκνωτές C6 και C11(10μF/50V).

Όπως και στα άλλα ολοκληρωμένα, έτσι και εδώ έχουμε συνδέσει τον πυκνωτή C5(100nF) όσο γίνεται πιο κοντά στους ακροδέκτες τροφοδοσίας του MAX232 με σκοπό την μείωση του θορύβου που παράγει το ίδιο το ολοκληρωμένο στην γραμμή των +5V λόγω των εσωτερικών κυκλωμάτων ανύψωσης τάσης μέσω διακοπτικών συστημάτων.

Στο ίδιο ολοκληρωμένο υπάρχουν άλλες δύο ίδιες γραμμές μετατροπής από TTL/CMOS στάθμες σε RS232 και αντίστροφα τις οποίες όμως δεν χρησιμοποιούμε.

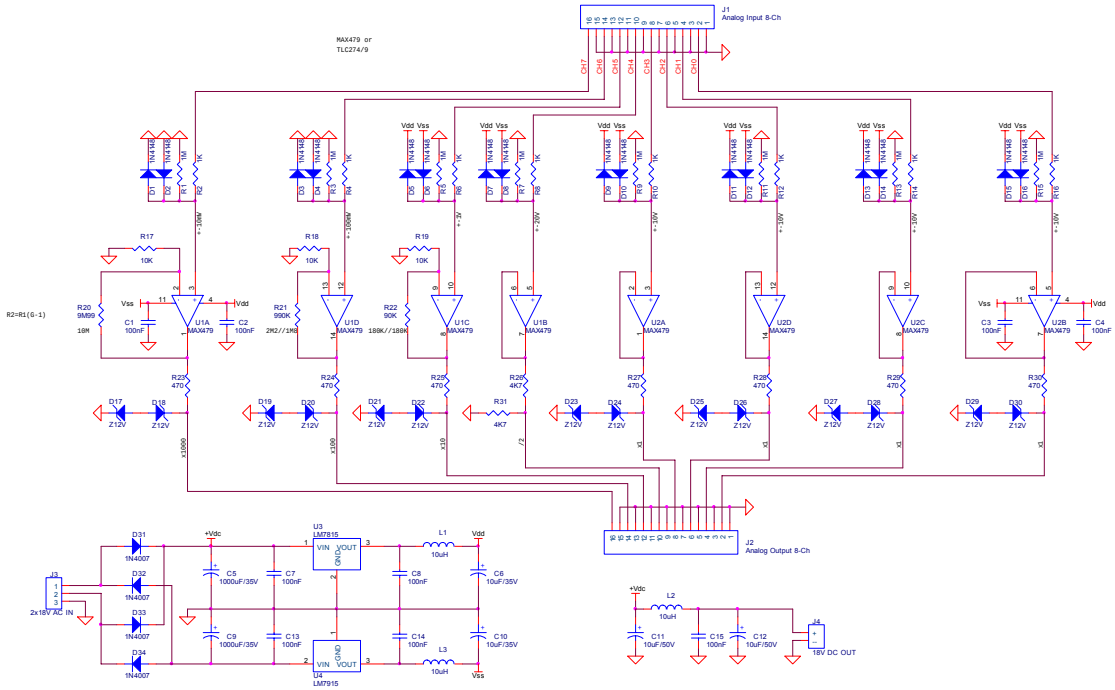
Εικόνα 5 – Σχηματικό τμήματος επικοινωνίας (RS232).



β) Κύκλωμα απομόνωσης εισόδων(Buffer).

Επιπροσθέτως υλοποιήσαμε μια διάταξη για την ενίσχυση των σημάτων εισόδου και την προστασία του συστήματος μας από τυχών υπερτάσεις που θα μπορούσαν να προκύψουν. Το κύκλωμα το βλέπουμε στην εικόνα 6.

Εικόνα 6 – Σχηματικό Buffer



Στο κύκλωμα έχουν χρησιμοποιηθεί δύο τετραπλοί τελεστικοί ενισχυτές (MAX479) και οι σταθεροποιητές τάσεως LM7815 και

LM7915 για δημιουργία συμμετρικής τάσεως ±15V που χρειάζεται ο παραπάνω τελεστικός ενισχυτής για την τροφοδοσία του.

Το γενικό τροφοδοτικό αποτελείται από μια γέφυρα ανόρθωσης(D31, D32, D33, D34), με τους κατάλληλους πυκνωτές C5, C9(1000μF/35V) για το φιλτράρισμα της τάσης στα ±18V DC.

Από την γραμμή των +18V θέλουμε να τροφοδοτήσουμε την κυρίως πλακέτα, οπότε την πηγαίνουμε σε ένα συνδετήρα(κλέμα) αφού πρώτα την περάσουμε μέσα από ένα φίλτρο τύπου “Π” με τα στοιχεία C11, L2, C15 και C12.

Τα ολοκληρωμένα U3(7815) και U4(7915) δέχονται την συμμετρική τάση των ±18V και την μετατρέπουν σε ±15V. Κοντά στους ακροδέκτες των ολοκληρωμένων αυτών υπάρχουν οι απαραίτητοι πυκνωτές των 100nF που χρησιμεύουν για την αποφυγή των ταλαντώσεων των ολοκληρωμένων, όπως εξηγήσαμε και στο κυρίως κύκλωμα.

Στις δύο γραμμές των +15V και -15V υπάρχουν φίλτρα τύπου “Π” με πηνία και πυκνωτές για να παρέχουν φιλτραρισμένη την συμμετρική τάση των ±15V στους τελεστικούς για να έχουμε όσο γίνεται λιγότερη παρεμβολή θορύβου στα σήματα μέτρησης από την τροφοδοσία.

Οι τελεστικοί ενισχυτές είναι συνδεμένοι ως ενισχυτές με ενίσχυση από +1 ως +1000. Συγκεκριμένα, έχουμε επιλέξει τα τέσσερα πρώτα κανάλια να λειτουργούν ως απλοί απομονωτές(x1) του σήματος εισόδου χωρίς να κάνουν ενίσχυση, με την αναστρέφουσα είσοδο συνδεδεμένη με την έξοδο. Το πέμπτο κανάλι κάνει μεν απομόνωση χωρίς ενίσχυση όπως τα προηγούμενα αλλά στην έξοδο του βρίσκεται ένας διαιρέτης τάσης με δύο όμοιες αντιστάσεις(R26=R31=4,7KΩ) ώστε τελικά να γίνεται διαίρεση του σήματος εισόδου κατά 2. Στα υπόλοιπα κανάλια, οι τελεστικοί ενισχυτές λειτουργούν ως ενισχυτές με ενίσχυση x10, x100 και x1000 αντίστοιχα. Οι τιμές στο δικτύωμα αντιστάσεων έχουν επιλεγεί σύμφωνα με την γνωστή σχέση:

$$R1 = R2 \bullet (G - 1)$$

Σε κάθε είσοδο καναλιού έχουν τοποθετηθεί δίοδοι πρόσδεσης στις τάσεις τροφοδοσίας ±15V ή στο κοινό σημείο(GND) με σκοπό την προστασία των εισόδων των τελεστικών ενισχυτών από υπερτάσεις που θα μπορούσαν να καταστρέψουν τα ολοκληρωμένα. Βοήθεια στον σκοπό αυτό προσφέρουν και οι σε σειρά αντιστάσεις του 1KΩ που υπάρχουν σε κάθε κανάλι και περιορίζουν το ρεύμα εισόδου σε αποδεκτά επίπεδα για τις διόδους προστασίας. Οι αντιστάσεις προς την γη

του 1ΜΩ “γειώνουν” τις μη αναστρέφουσες εισόδους των τελεστικών ώστε να “βλέπουν” ένα σταθερό δυναμικό γης όταν δεν εφαρμόζεται κάποιο σήμα μέτρησης σε αυτούς.

Οι έξοδοι των τελεστικών δεν είναι άμεσα συνδεδεμένοι με την κυρίως πλακέτα όπου βρίσκεται ο A/D, αλλά παρεμβάλλονται κατάλληλοι ψαλιδιστές τάσης με διόδους zener των 12V, τοποθετημένοι back to back έτσι ώστε με την βοήθεια των αντιστάσεων περιορισμού ρεύματος των 470Ω να περιορίζουν τις θετικές και τις αρνητικές κορυφές του σήματος στα ±12V, επίπεδα αποδεκτά από το ολοκληρωμένο του A/D(MAX197) το οποίο εγγυάται ότι δεν πρόκειται να καταστραφεί αρκεί τα όρια των σημάτων που εμφανίζονται στις εισόδους του να μην υπερβαίνουν τα ±16V, σύμφωνα με το φυλλάδιο χαρακτηριστικών του κατασκευαστή(MAXIM). Η έξοδος του καναλιού 5 είναι διαφορετική από τις υπόλοιπες καθώς σε αυτό το κανάλι γίνεται διαίρεση του σήματος κατά δύο αλλά ο τιμές των αντιστάσεων(4,7ΚΩ) του διαιρέτη τάσης έχουν επιλεγεί έτσι ώστε να συμβαδίζουν με την αντίσταση εξόδου των τελεστικών αλλά και την αντίσταση εισόδου του A/D.

Συγκεντρωτικά, τα χαρακτηριστικά των καναλιών της πλακέτας του απομονωτή έχουν ως φαίνονται στον παρακάτω πίνακα.

Πίνακας 1 – Χαρακτηριστικά καναλιών Buffer

ΚΑΝΑΛΙ	ΣΗΜΑ ΕΙΣΟΔΟΥ	ΕΝΙΣΧΥΣΗ	ΑΝΤΙΣΤΑΣΗ ΕΙΣΟΔΟΥ
1	±10V	x1	1ΜΩ
2	±10V	x1	1ΜΩ
3	±10V	x1	1ΜΩ
4	±10V	x1	1ΜΩ
5	±15V	/2	1ΜΩ
6	±1V	x10	1ΜΩ
7	±100mV	x100	1ΜΩ
8	±10mV	x1000	1ΜΩ

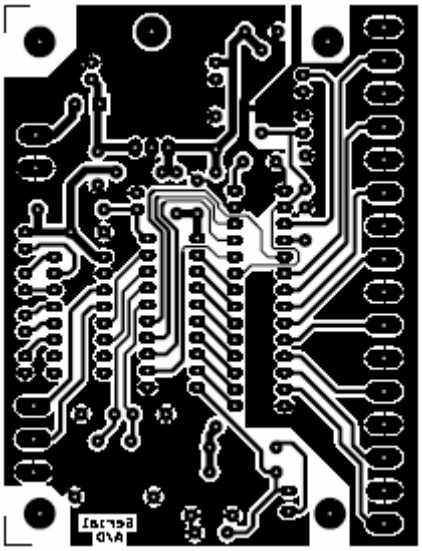
Δεδομένου ότι η δειγματοληψία του σήματος εισόδου θα γίνεται με το πολύ 1000Hz, το σήμα εισόδου δεν επιτρέπεται να ξεπεράσει τα 500Hz σύμφωνα με τον κανόνα του Nyquist. Η συχνότητα των 500Hz είναι αρκετά μικρή και μέσα στα όρια

λειτουργίας των τελεστικών ενισχυτών που χρησιμοποιήσαμε, ακόμα και με την ενίσχυση του x1000.

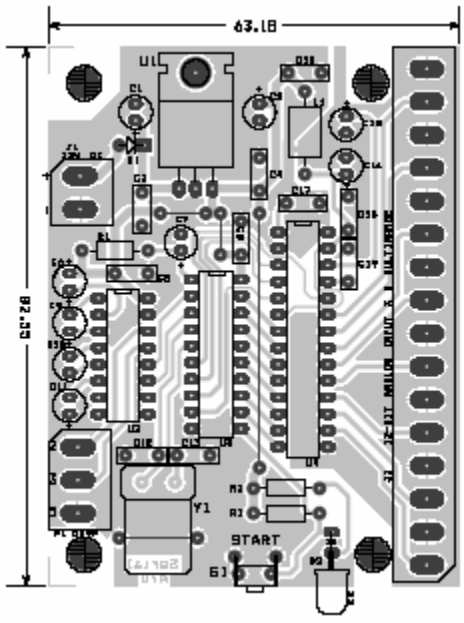
2.2.2 Τυπωμένο κύκλωμα

Στην εικόνα 7 βλέπουμε το τυπωμένο κύκλωμα, μίας όψης, της κατασκευής μας. Πρόκειται για την κάτω όψη της πλακέτας, του κυρίως κυκλώματος. Στην εικόνα 8 βλέπουμε την πλακέτα μας με τις διαστάσεις της καθώς και με τα εξαρτήματα αυτής. Ο κατάλογος υλικών βρίσκεται στο παράρτημα Α, στον πίνακα 3.

Εικόνα 7 – Όψη χαλκού πλακέτας κυρίως κυκλώματος.



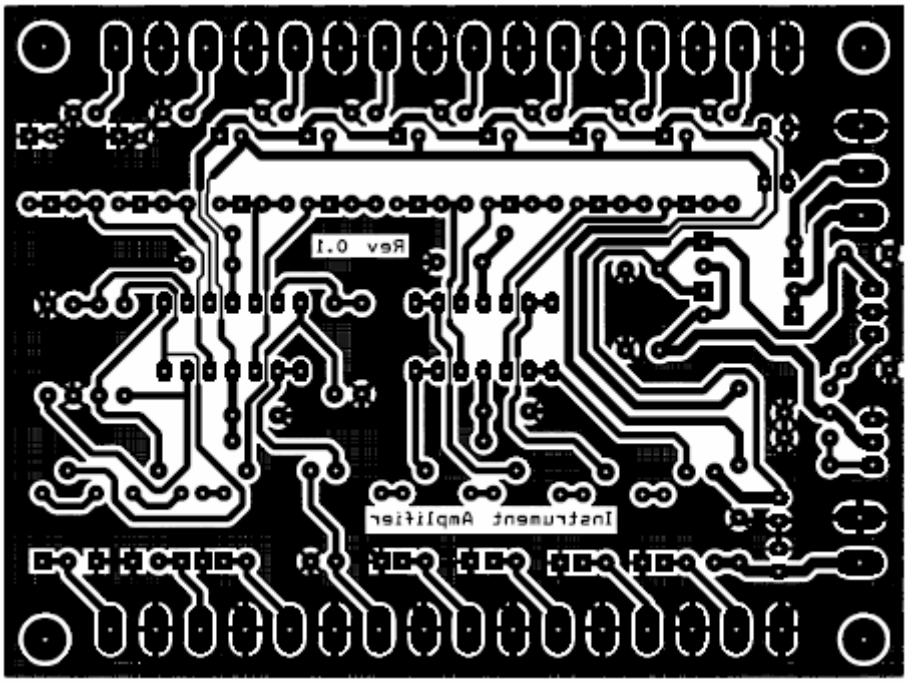
Εικόνα 8 – Τοπογραφικό πλακέτας κυρίως κυκλώματος. Κάτω από το πηνίο L1 φαίνεται το μοναδικό σημείο στο οποίο ενώνονται η αναλογική και η ψηφιακή γείωση.



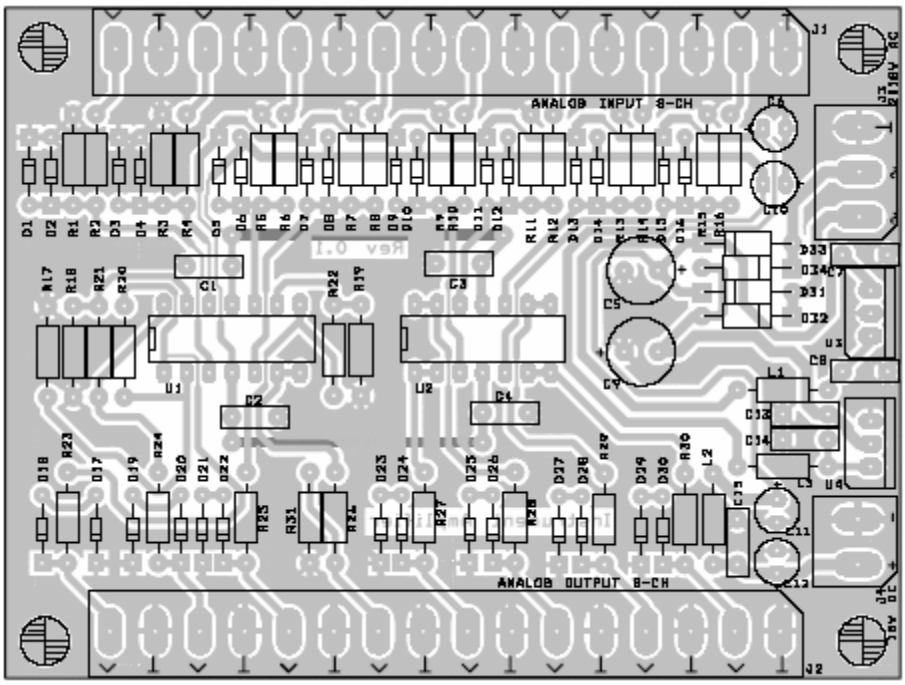
Φυσικά δεν θα μπορούσαμε να παραλείψουμε το αντίστοιχο τυπωμένο κύκλωμα και για το Buffer/Instrument Amplifier του

κυκλώματος. Οι εικόνες 9 και 10 δείχνουν αντίστοιχα την κάτω όψη της πλακέτας και την τοποθέτηση των εξαρτημάτων. Ο κατάλογος υλικών βρίσκεται στο παράρτημα Α, πίνακας 4.

Εικόνα 9 – Όψη χαλκού πλακέτας Buffer.



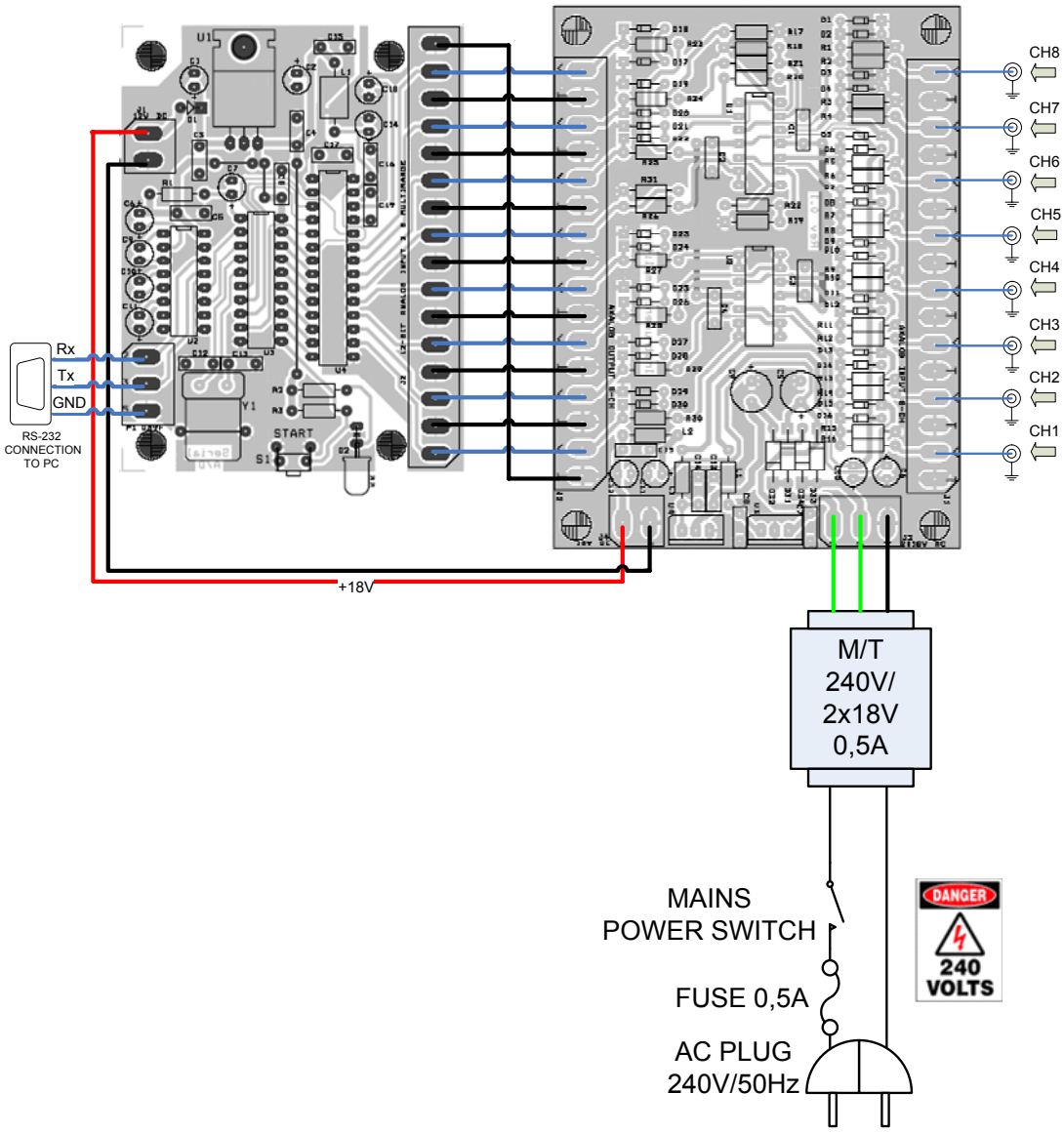
Εικόνα 10 – Τοπογραφικό πλακέτας Buffer.



2.2.3 Διάγραμμα καλωδιώσεων

Στην εικόνα 11 βλέπουμε το διάγραμμα καλωδιώσεων που μας δείχνει την συνδεσμολογία των επιμέρους τμημάτων της συσκευής μας.

Εικόνα 11 – Διάγραμμα καλωδιώσεων.



2.3 Σχεδίαση λογισμικού

Στο κεφάλαιο που ακολουθεί θα αναλύσουμε το πρόγραμμα που γράψαμε για τον έλεγχο της συσκευής μας, τόσο αυτό της ίδιας της συσκευής όσο και αυτό του υπολογιστή. Κατ' αρχήν ας δούμε το πρόγραμμα του μικροελεγκτή.

2.3.1 Λογισμικό μικροελεγκτή

Το να βάζαμε τον κώδικα εδώ και να τον αναλύαμε γραμμή-γραμμή, θα ήταν κάτι το κουραστικό, καθώς είναι αρκετά μεγάλος. Γι' αυτό το λόγο παραθέτουμε ένα διάγραμμα ροής για το πως δουλεύει το σύστημα μας. Ο πλήρης κώδικας μαζί με τις υπορουτίνες βρίσκεται στο παράρτημα Β. Στην εικόνα 12 βλέπουμε το διάγραμμα ροής του βασικού προγράμματος του μικροελεγκτή μας.

Το πρόγραμμα αποτελείται από το κυρίως πρόγραμμα, ρουτίνα διακοπής(interrupt) που έχει να κάνει με την λήψη δεδομένων από το εσωτερικό UART του μικροελεγκτή και μερικές υπορουτίνες.

Αναλυτικά η λειτουργία του προγράμματος μας σύμφωνα με το διάγραμμα ροής της εικόνας 12 είναι η εξής:

- Ξεκινώντας το πρόγραμμα, γίνονται κάποιες βασικές εργασίες αρχικοποίησης:
  - Αρχικοποίηση του WatchDog Timer στα 1,9sec ο οποίος προφυλάσσει το πρόγραμμα από "κολλήματα" που είναι στατιστικά επιβεβαιωμένο ότι προκύπτουν σε όλα τα συστήματα όπου τρέχει κάποιος κώδικας. Αν μέσα στα 1,9sec δεν έχουμε κάνει reset στον WatchDog, ο μικροελεγκτής θα υποστεί επανεκκίνηση.
  - Απενεργοποίηση του Analog Comparator.
  - Αρχικοποίηση του Stack Pointer στην τελευταία διεύθυνση της μνήμης SRAM ώστε να έχουμε αρκετή μνήμη.
  - Αρχικοποίηση των θυρών B & D του μικροελεγκτή στις επιθυμητές τιμές, ανάλογα με τα εξαρτήματα που συνδέονται στα I/O.

- Περιμένουμε 280ms περίπου ώστε να σταθεροποιηθεί η κατάσταση των υπόλοιπων κυκλωμάτων.
- Αρχικοποιούμε τους πιο σημαντικούς καταχωρητές/ μεταβλητές σε κάποιες default τιμές.
- Καθαρίζουμε την μνήμη SRAM γεμίζοντας την με λογικό μηδέν “0”.
- Απενεργοποιούμε την διακοπή υπερχείλισης (Overflow Interrupt) του Timer1. Στην συνέχεια του προγράμματος θα χρησιμοποιήσουμε αυτή την διακοπή για να παράγουμε το χρόνο δειγματοληψίας.
- Ενεργοποιούμε την εκπομπή(Tx) και λήψη(Rx) μέσω UART για 115200,8,N,1. Φροντίζουμε ώστε να ενεργοποιήσουμε την διακοπή(Interrupt) του Rx ώστε να είμαστε σίγουροι ότι δεν θα χαθεί ποτέ κάποιο από τα δεδομένα των εντολών μέσω UART.
- Στέλνουμε μέσω UART το μήνυμα καλωσορίσματος, τις default ρυθμίσεις και την λίστα εντολών, πληροφορίες χρήσιμες για να ξέρουμε την έκδοση του προγράμματος(firmware) και πως δουλεύει το σύστημα.
- Μετά από τις αρχικοποιήσεις, το σύστημα εισέρχεται στο κυρίως πρόγραμμα(Main Loop)…

- Στο κυρίως πρόγραμμα(Main Loop) πραγματοποιούμε τα παρακάτω βήματα:
  - Φροντίζουμε σε κάθε επιστροφή στην αρχή του Main Loop να κάνουμε Reset στον WatchDog Timer. Αυτό το σημείο είναι ουσιαστικά και το μοναδικό σε όλο το πρόγραμμα που γίνεται το Reset και αυτό προσθέτει βαθμούς στο επίπεδο αξιοπιστίας του κώδικα, όσον αφορά τα “κολλήματα”.
  - Εισερχόμαστε σε ένα βρόχο ελέγχου όπου έχουμε απενεργοποιήσει τοπικά όλες τις διακοπές (interrupts) για να μην επηρεαστούν ορισμένοι καταχωρητές/μεταβλητές που χρησιμοποιούνται και από την ρουτίνα διακοπής του UART Rx. Ο χρόνος που διαρκεί αυτή η απενεργοποίηση είναι πολύ μικρός, τόσο που δεν υπερκαλύπτεται καμιά διακοπή.

Μέσα στον βρόχο ελέγχου, ελέγχουμε από τρεις πηγές αν υπάρχει λόγος να πάρουμε μια μέτρηση. Οι τρεις αυτές πηγές είναι οι εξής:

1) Πίεση διακόπτη έναρξης/λήξης μετρήσεων,
2) Έναρξη/λήξη μιας μέτρησης λόγω εντολής από το πρόγραμμα ελέγχου στον υπολογιστή και
3) Συνεχής εκτέλεση μετρήσεων αν είμαστε σε αυτόν τον τρόπο(mode) λειτουργίας.

Αν έχει οριστεί λήξη των μετρήσεων, το Main Loop επαναλαμβάνεται έως ότου έστω και μια από αυτές να σημαίνει έναρξη μετρήσεων. Τότε το πρόγραμμα προχωράει να πάρει μετρήσεις.

Α. Ξεκινάει ενεργοποιώντας το ενδεικτικό LED για να δείξει ότι ξεκινάει μετρήσεις.

Β. Ελέγχει αν στις ρυθμίσεις έχει επιλεγεί ένα μόνο συγκεκριμένο κανάλι(Γ.1.) ή όλα τα κανάλια μαζί(Δ.1.).

Γ.1. Αν πρόκειται για ένα κανάλι, ενεργοποιεί την λειτουργία υπερχείλισης του Timer/Counter1 για χρόνο 1ms.

Γ.2. Ορίζει το κανάλι του A/D και πραγματοποιεί μια ψεύτικη μέτρηση για να σταθεροποιηθούν τα εσωτερικά κυκλώματα πολυπλεξίας εισόδου του A/D.

Γ.3. Πραγματοποιεί μια μέτρηση στο ίδιο κανάλι με τον A/D. Αν για κάποιο λόγο ο A/D δεν αποκρίνεται σωστά, ένα μήνυμα σφάλματος επιστρέφει στον υπολογιστή μέσω του UART Tx.

Γ.4. Τα 12 bit δεδομένων που επέστρεψε ο A/D, μετατρέπονται από δυαδική μορφή σε ASCII δεκαεξαδικά, πακετάρονται ως εξής: Vxxx<CR> , MSB πρώτο και στέλνονται στον υπολογιστή με το UART Tx για περαιτέρω επεξεργασία.

Π.χ. η δυαδική τιμή 0010 1001 1101 αποστέλλεται ως: V29D<CR> όπου <CR> = $0D.

Γ.5. Αφού αποσταλεί η μέτρηση, το πρόγραμμα χρησιμοποιεί τον Timer/Counter1 που έχει βήμα υπερχείλισης 1ms για να καθυστερήσει τόσα βήματα(1ms) όσα είναι η περίοδος δειγματοληψίας(1..255ms).

Γ.6. Με το πέρας της περιόδου δειγματοληψίας, απενεργοποιεί το ενδεικτικό LED και επιστρέφει στην αρχή του κυρίως προγράμματος(Main Loop).

Δ.1. Αν όλα τα κανάλια έχουν επιλεγεί, το πρόγραμμα ακολουθεί τα βήματα Γ.1. ως Γ.6. με την μόνη διαφορά ότι επαναλαμβάνει τα βήματα Γ.2., Γ.3., Γ.4. ακολουθιακά για όλα τα οκτώ(8) κανάλια του A/D και επιστρέφει στον υπολογιστή τις μετρήσεις με την μορφή: Vaaabbbcccdddeeefffgggghhh<CR> , MSB πρώτο και στέλνονται στον υπολογιστή με το UART Tx για περαιτέρω επεξεργασία.

- Στην ρουτίνα διακοπής UART Rx πραγματοποιείται η λήψη και αποκωδικοποίηση των εντολών που αποστέλλει το πρόγραμμα ελέγχου του υπολογιστή.
  - Στην ρουτίνα αυτή χρησιμοποιείται ένας buffer δεδομένων βάθους 10 bytes όπου αποθηκεύονται με την σειρά όσα δεδομένα διαβάζονται από τον καταχωρητή UDR του AVR, ανά ένα byte σε κάθε διακοπή(interrupt).
  - Σε κάθε διακοπή(interrupt) ελέγχει αν το byte που διάβασε είναι το <CR> (=$0D) ή έχει γεμίσει ο buffer. Αν έστω και μια από τις δυο συνθήκες είναι αληθής, καλεί την ρουτίνα αποκωδικοποίησης εντολών, η οποία ελέγχει αν υπάρχει μια εντολή αναγνωρίσιμη και αν αυτή η εντολή έχει ληφθεί χωρίς σφάλματα, προχωρεί στην εκτέλεση της και επιστρέφει <CR> σαν acknowledge. Παρακάτω δίνεται η λίστα των εντολών τις οποίες αναγνωρίζει η κατασκευή μας στην τρέχουσα έκδοση(v1.3d):

Command List:

#Hx**<CR>** : *sets ADC's c**H**annel. <x> must be in the range 0 to 8. From 0 to 7 it represents the desired channel. The value of 8 means: report ALL channels in one packet every time(see below).*

#Lx**<CR>** : *sets ADC's sca**L**e input. <x> must be in the range 0..3.*
0 : 0..5V
1 : 0..10V
2 : ±5V
3 : ±10V

#Txxx**<CR>** : *sets sampling **T**ime period in 1msec units when in continuous measure mode. <xxx> can be in the range 001..255 when channel is 0..7 but if you chose reporting ALL channels, the range is reduced to 003..255.*

#C**<CR>** : *sets **C**ontinuously measure mode and starts measurements.*

#S**<CR>** : *sets **S**ingle shot measure mode.*

#G**<CR>** : *It means **G**o... It starts a new measure in single shot measure mode.*

*Caution! **<CR>** = 0D hex*

After every correctly received command, it acknowledges with a <CR>

- Οι υπορουτίνες που χρησιμοποιούνται στον πρόγραμμα αφορούν τις εξής λειτουργίες:
  - Παραγωγή χρόνου καθυστέρησης σύμφωνα με την εξίσωση:

    t(sec)=[6*2+4+3+3*(x^3+x^2+x)]/3,6864MHz

    όπου το “x” είναι μια 8-bit τιμή(1..255) που εισάγεται στην ρουτίνα με την βοήθεια του καταχωρητή r16(temp). Η ρουτίνα αυτή μπορεί να παράγει χρόνους καθυστέρησης από 7,6μs(r16=1) μέχρι 13,5sec(r16=255).
  - Μετατροπές δεδομένων από ASCII decimal σε binary, από ASCII hex σε binary και από ASCII hex σε binary. Αυτές οι υπορουτίνες χρησιμοποιούνται στην αποστολή των δεδομένων μετρήσεων στον υπολογιστή και στην λήψη-αποκωδικοποίηση των εντολών από αυτόν.
  - Αποστολή δεδομένων μέσω του UART Tx σε polled mode(χωρίς διακοπές), ανά ένα byte ή ομάδα bytes αποθηκευμένα στην μνήμη FLASH του AVR.
  - Επικοινωνία με τον A/D μέσω του 8+4 bit παράλληλου interface. Αυτές οι ρουτίνες γράφουν μια λέξη 8-bit στον A/D, διαβάζουν μια λέξη 16-bit

και πραγματοποιούν μια μέτρηση σε προγραμματιζόμενο κανάλι εισόδου, με την βοήθεια των δυο προηγούμενων ρουτινών.

Το γράψιμο μιας 8-bit λέξης στον εσωτερικό καταχωρητή ελέγχου(Control Register) του A/D γίνεται τοποθετώντας την έξη στην πόρτα του A/D και μετάγοντας σε χαμηλό δυναμικό για σύντομο διάστημα τον ακροδέκτη /WR του A/D.

Το διάβασμα μιας 16-bit λέξης από τον A/D γίνεται σε δύο βήματα. Πρώτα, διατηρώντας σε χαμηλό δυναμικό τον ακροδέκτη HBEN του A/D διαβάζουμε το πρώτο low byte μετάγοντας τον ακροδέκτη /RD σε χαμηλό δυναμικό. Μετά υψώνουμε τον ακροδέκτη HBEN(High Byte Enable) του A/D σε υψηλό δυναμικό και διαβάζουμε το high byte μετάγοντας τον ακροδέκτη /RD σε χαμηλό δυναμικό. Συνδυάζοντας τα δύο byte που διαβάζαμε έχουμε μια 16-bit λέξη, από την οποία τα χρήσιμα δεδομένα είναι τα 12 λιγότερο σημαντικά στοιχεία.

Για την πραγματοποίηση μιας μέτρησης, αρχικά γράφουμε το byte ελέγχου στον καταχωρητή ελέγχου του A/D, θέτοντας το κανάλι μέτρησης(1..8), την κλίμακα μέτρησης(5 ή 10V) και το αν το σήμα μέτρησης είναι μονής ή διπλής πολικότητας. Μετά από ένα μικρό delay, παρακολουθούμε τον ακροδέκτη /INT του A/D και όταν αυτός μεταχθεί σε χαμηλό δυναμικό, η μέτρηση έχει ολοκληρωθεί, οπότε διαβάζουμε τα 16-bit δεδομένων που αντιπροσωπεύουν την τιμή του σήματος εισόδου.

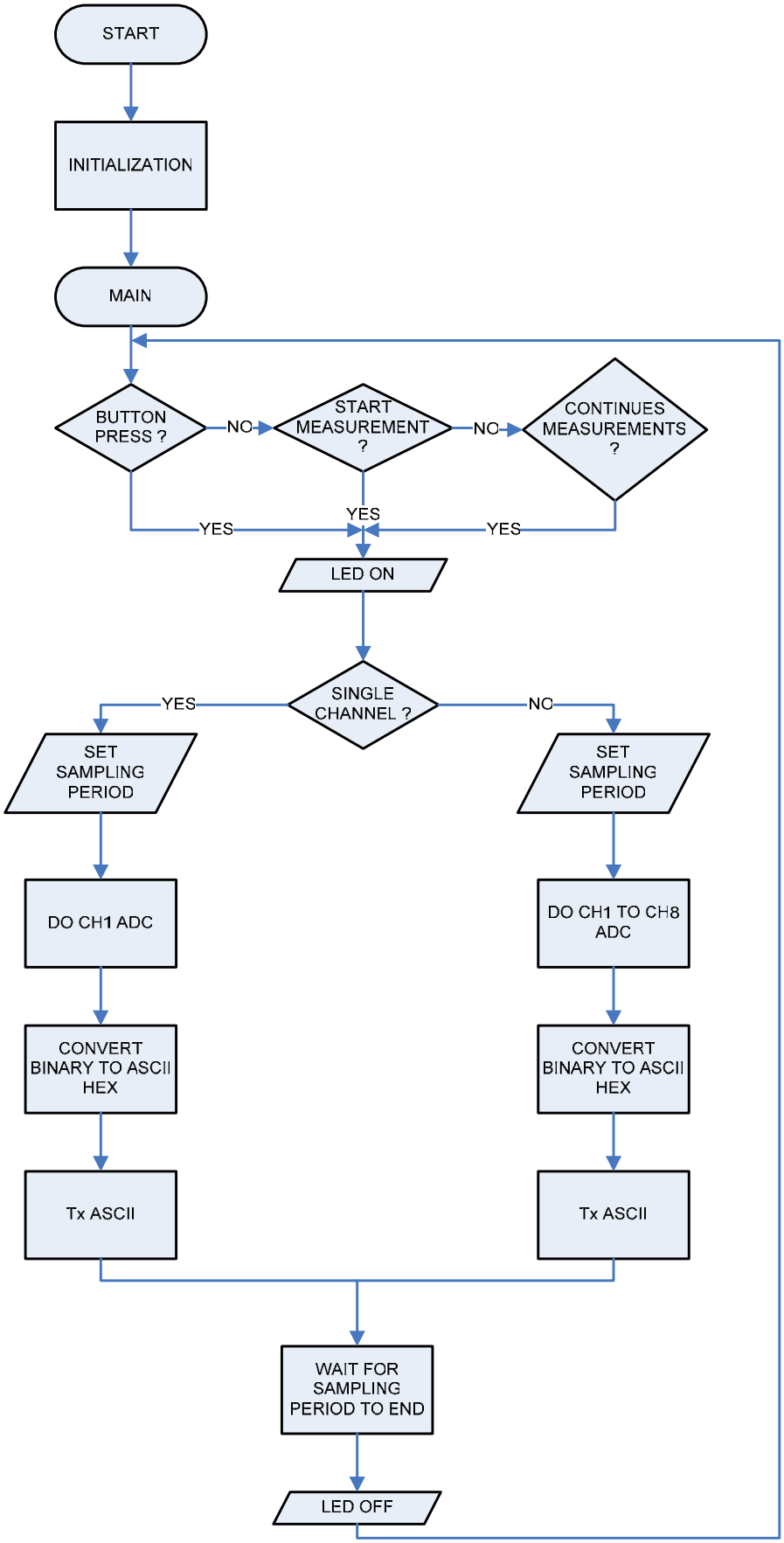
Στις εικόνες 14 έως 17 μπορούμε να δούμε τον καταχωρητή του MAX197, την κβάντιση του αναλογικού σήματος και την μετατροπή του σε ψηφιακό, το χρονοδιάγραμμα της μετατροπής του AD και της επικοινωνίας με τον μικροελεγκτή, και το λειτουργικό διάγραμμα του MAX197, αντίστοιχα.

Η ανάλυση του MAX197 είναι 12-bit. Αυτό μας δίνει διακριτική ικανότητα ανάλογη με κλίμακα μέτρησης. Η τιμές της διακριτικής ικανότητας φαίνονται στον πίνακα 2.

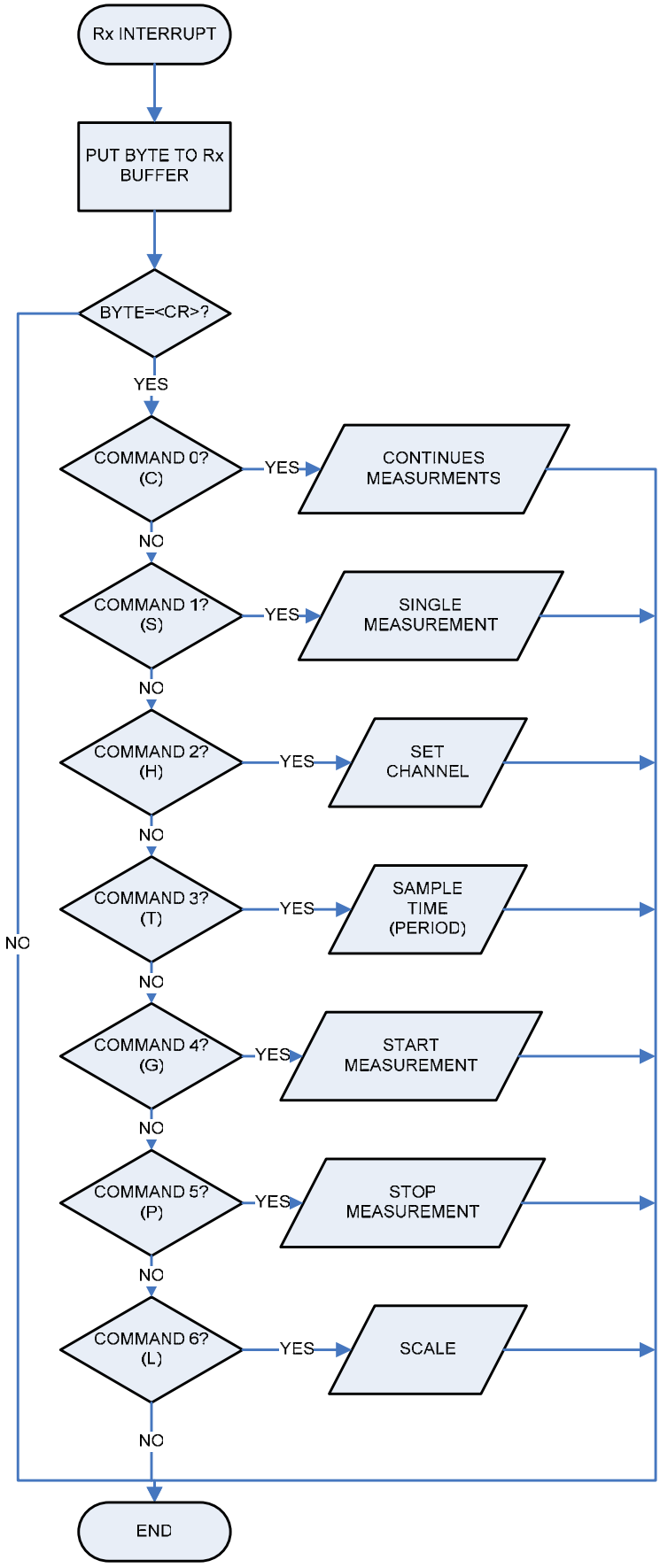
Πίνακας 2

Κλίμακα (Volt)	Δυαδική έξοδος (Hex)	Διακριτική ικανότητα 12 bit (mVolt)
0..5	000..FFF	1,2207
0..10	000..FFF	2,4414
-5..+5	800..7FF	2,4414
-10..+10	800..7FF	4,8828

Εικόνα 12 – Διάγραμμα ροής του κυρίως προγράμματος.



Εικόνα 13 – Διάγραμμα ροής ρουτίνας UART Rx Interrupt.



Εικόνα 14 – Εσωτερικός καταχωρητής MAX197

Table 2. Control-Byte Format

D7 (MSB)	D6	D5	D4	D3	D2	D1	D0 (LSB)
PD1	PD0	ACQMOD	RNG	BIP	A2	A1	A0

BIT	NAME	DESCRIPTION
7, 6	PD1, PD0	These two bits select the clock and power-down modes (Table 4).
5	ACQMOD	0 = internally controlled acquisition (6 clock cycles), 1 = externally controlled acquisition
4	RNG	Selects the full-scale voltage magnitude at the input (Table 3).
3	BIP	Selects unipolar or bipolar conversion mode (Table 3).
2, 1, 0	A2, A1, A0	These are address bits for the input mux to select the "on" channel (Table 5).

Table 3. Range and Polarity Selection

BIP	RNG	INPUT RANGE (V)
0	0	0 to 5
0	1	0 to 10
1	0	±5
1	1	±10

Table 4. Clock and Power-Down Selection

PD1	PD0	DEVICE MODE
0	0	Normal Operation / External Clock Mode
0	1	Normal Operation / Internal Clock Mode
1	0	Standby Power-Down (STBYPD); clock mode is unaffected
1	1	Full Power-Down (FULLPD); clock mode is unaffected

Table 5. Channel Selection

A2	A1	A0	CH0	CH1	CH2	CH3	CH4	CH5	CH6	CH7
0	0	0	*							
0	0	1		*						
0	1	0			*					
0	1	1				*				
1	0	0					*			
1	0	1						*		
1	1	0							*	
1	1	1								*

Εικόνα 15 – Μετατροπή Αναλογικό σε Ψηφιακό.

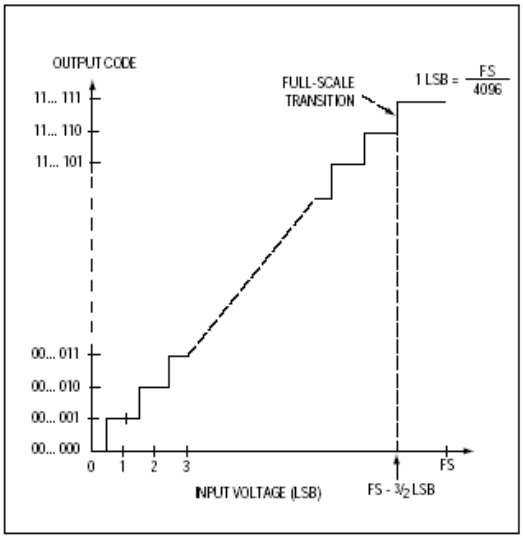


Figure 10. Unipolar Transfer Function

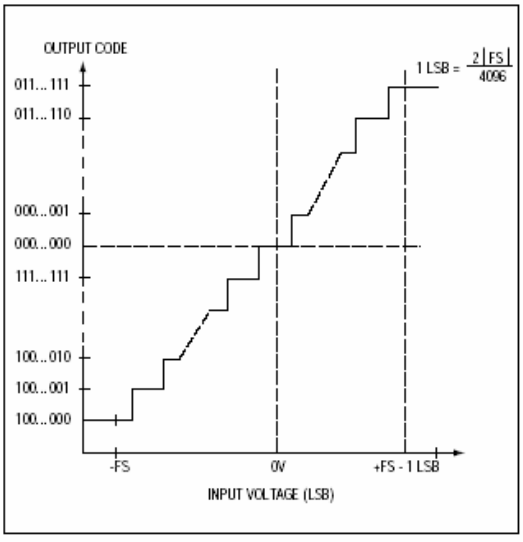
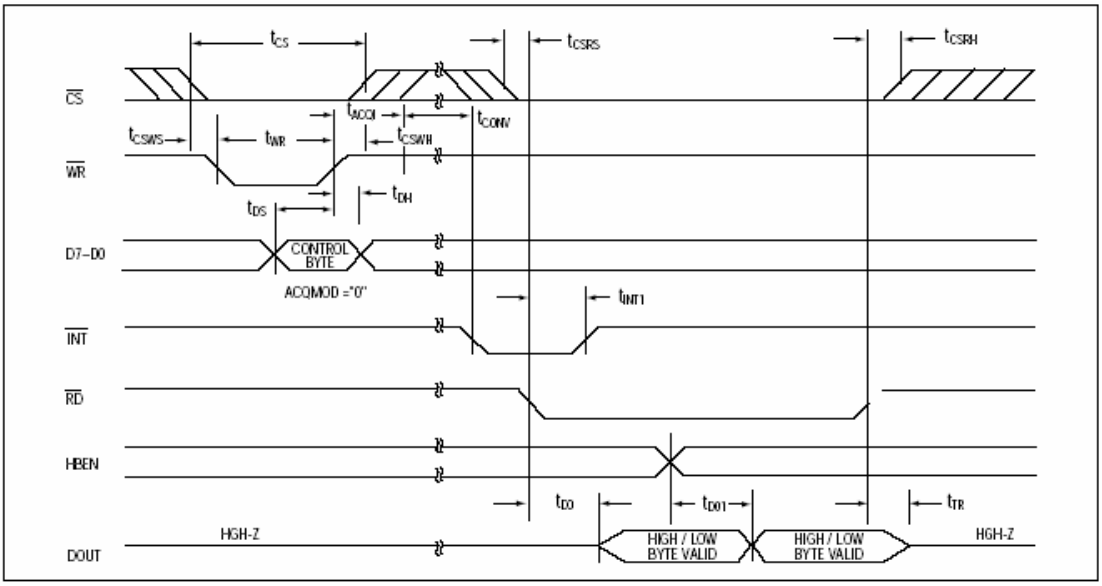
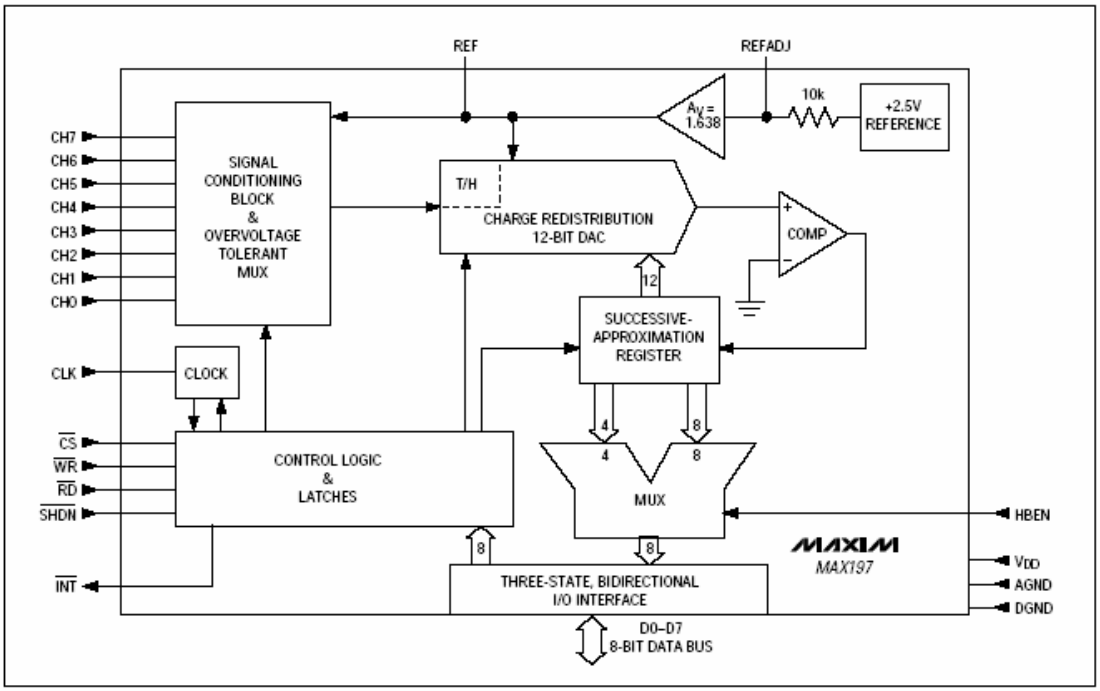


Figure 11. Bipolar Transfer Function

Εικόνα 16 – Χρονοδιάγραμμα μετατροπής AD.



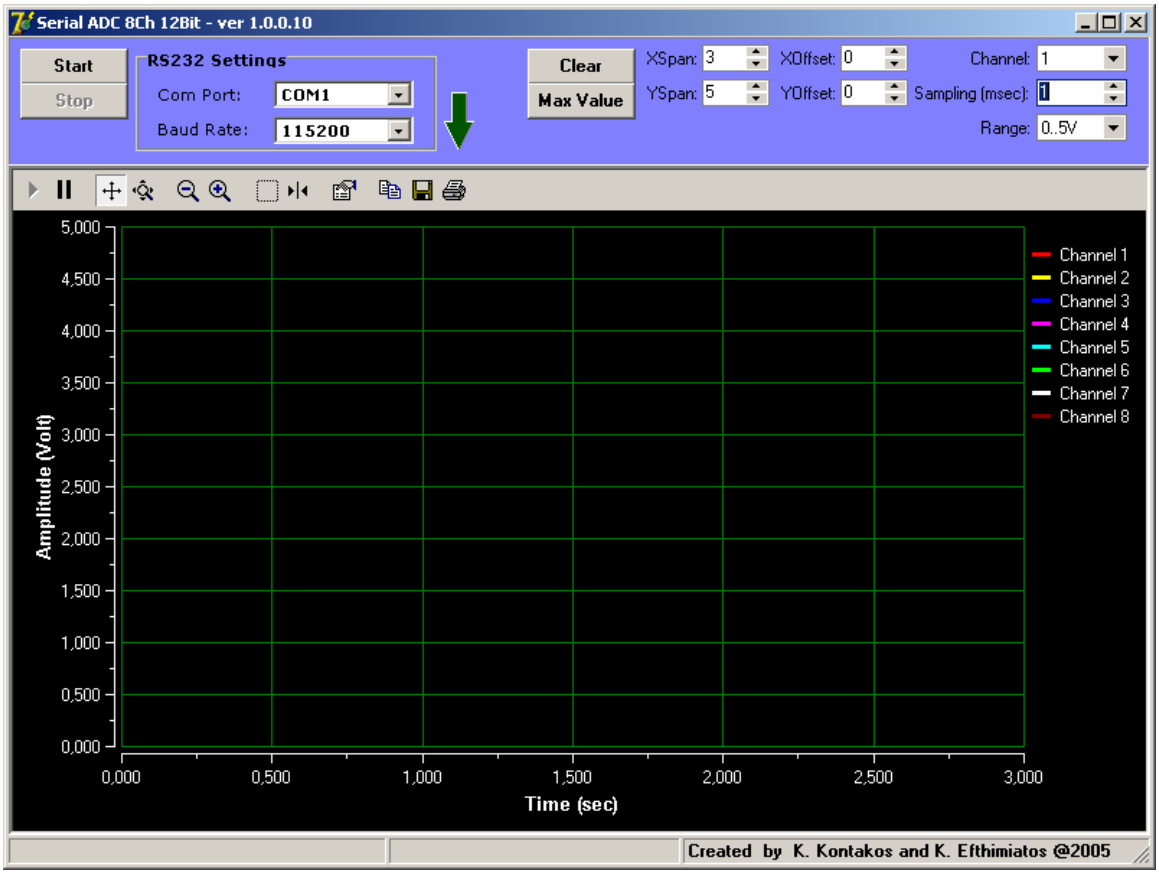
Εικόνα 17 – Λειτουργικό διάγραμμα MAX197



2.3.2 Λογισμικό Υπολογιστή

Στην εικόνα 18 βλέπουμε το πρόγραμμα που γράψαμε στην Borland Delphi 7™ για να μετράμε με την συσκευή μας και να βλέπουμε ζωντανά τις μετρήσεις στην οθόνη του υπολογιστή μας.

Εικόνα 18 – Κυρίως πρόγραμμα υπολογιστή

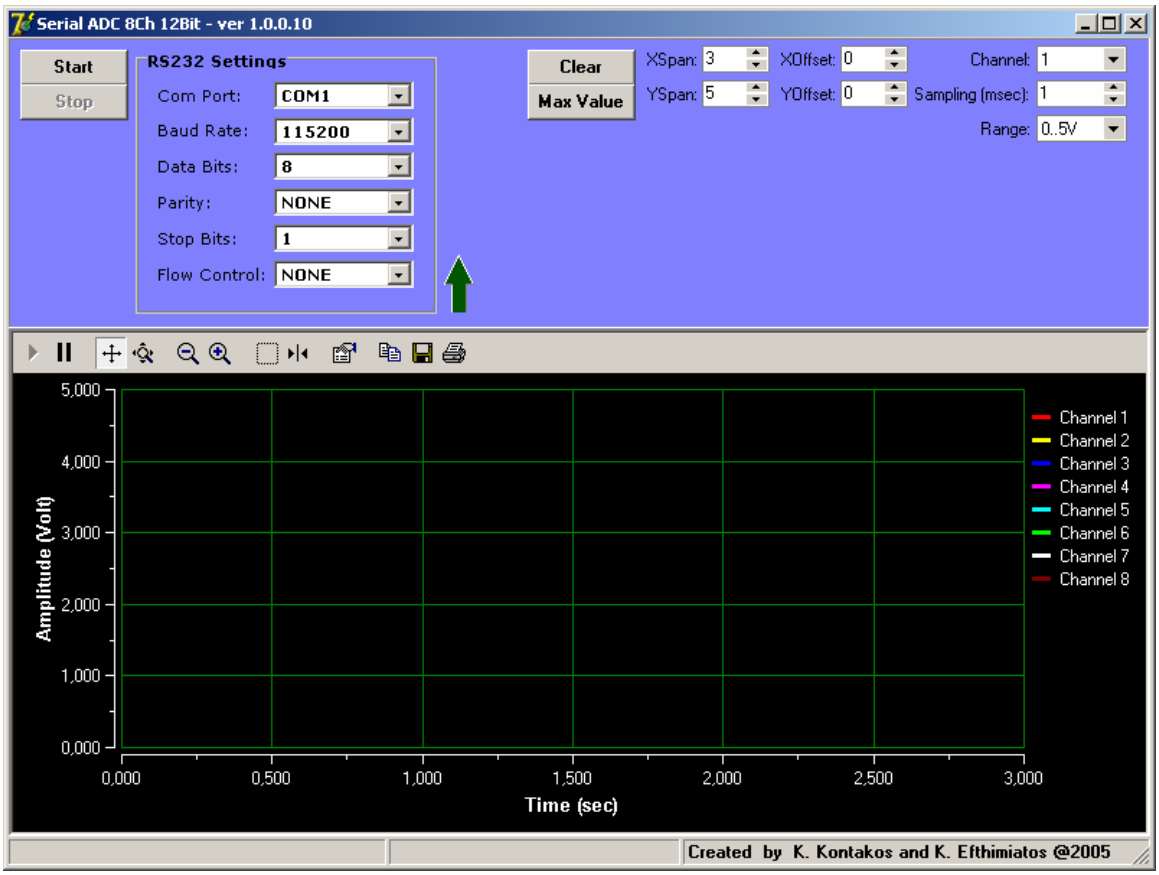


Στο πάνω μέρος παρατηρούμε τα κουμπιά ελέγχου. Ας τα δούμε όμως αναλυτικότερα την λειτουργία τους παρακάτω.

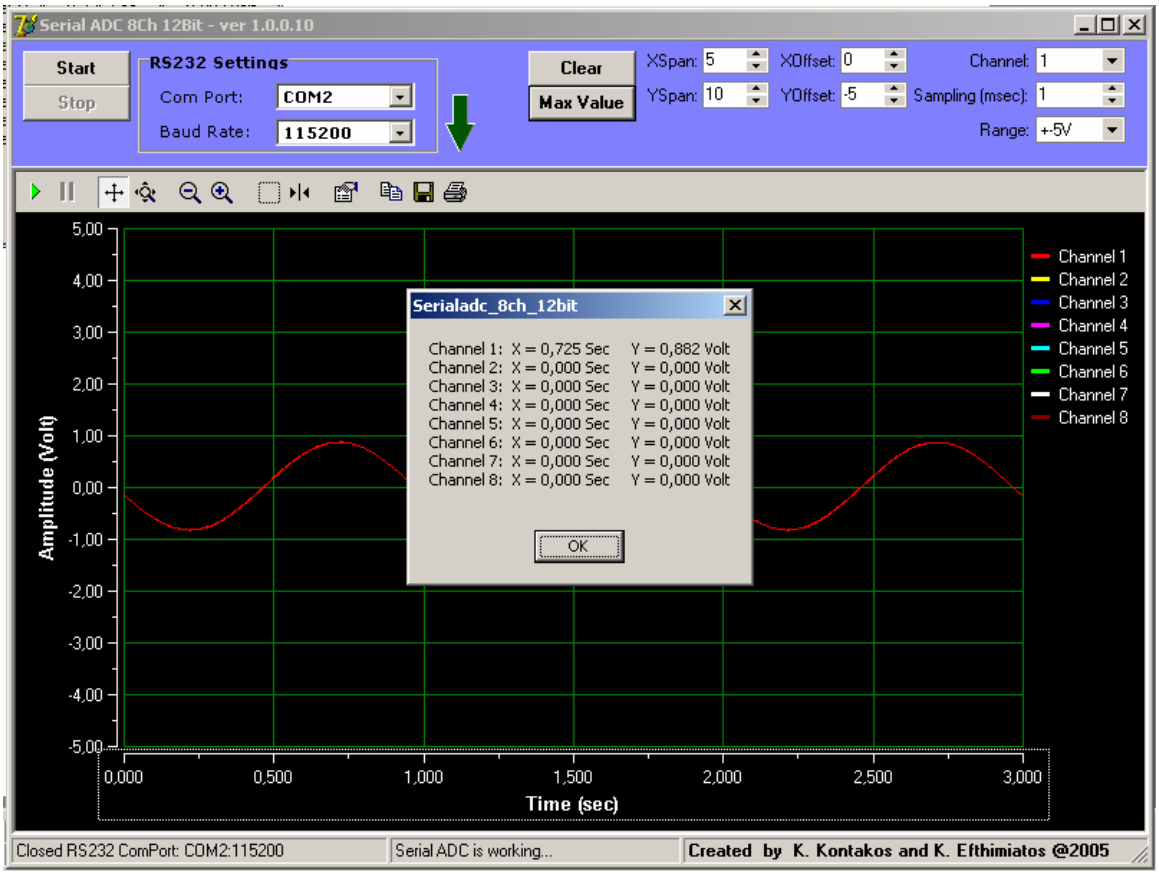
- Start: Με αυτό το κουμπί ξεκινά η συσκευή μας να μετρά και να στέλνει τις μετρήσεις στον υπολογιστή.
- Stop: Το αντίστοιχο κουμπί για να σταματήσει.
- RS232 Settings: Σε αυτή την περιοχή θέτουμε τις ρυθμίσεις για την σειριακή επικοινωνία του υπολογιστή με την συσκευή μας. Οι επιλογές που έχουμε είναι τυπικές για κάθε σειριακή επικοινωνία RS232 όπως βλέπουμε στην εικόνα 18. Αν πατήσουμε όμως το πράσινο βελάκι που βρίσκεται στα δεξιά του RS232 settings, θα εμφανιστούν μερικές πιο προχωρημένες ρυθμίσεις όπως βλέπουμε και στην εικόνα 19.

- Clear: Με το κουμπί αυτό μπορούμε να καθαρίσουμε την περιοχή σχεδίασης όταν θέλουμε να πάρουμε καινούργιες μετρήσεις.
- Max Value: πατώντας αυτό το κουμπί εμφανίζεται ένα παραθυράκι το οποίο μας δείχνει την μέγιστη τιμή κάθε καναλιού και σε ποια χρονική στιγμή έχει εμφανιστεί αυτή η τιμή (Εικόνα 20).
- XSpan, YSpan, XOffset, YOffset: Εδώ μπορούμε να ορίσουμε χειροκίνητα τα όρια της περιοχής σχεδίασης.
- Channel: Εδώ επιλέγουμε το κανάλι από το οποίο θα μετράμε.
- Sampling (msec): Εδώ ορίζουμε την περίοδο δειγματοληψίας σε χιλιοστά του δευτερολέπτου.
- Range: Θέτουμε τα όρια του ADC. Οι δυνατές επιλογές είναι 0-5V, 0-10V, ±5V και ±10V.

Εικόνα 19 – Ρυθμίσεις RS232



Εικόνα 20 – Max Value



Ανάμεσα στα κύρια κουμπιά ελέγχου και την περιοχή σχεδίασης παρατηρούμε κάποια δευτερεύοντα, τα οποία χρησιμεύουν στον έλεγχο αυτής. Αναλυτικότερα και από δεξιά προς τα αριστερά είναι:

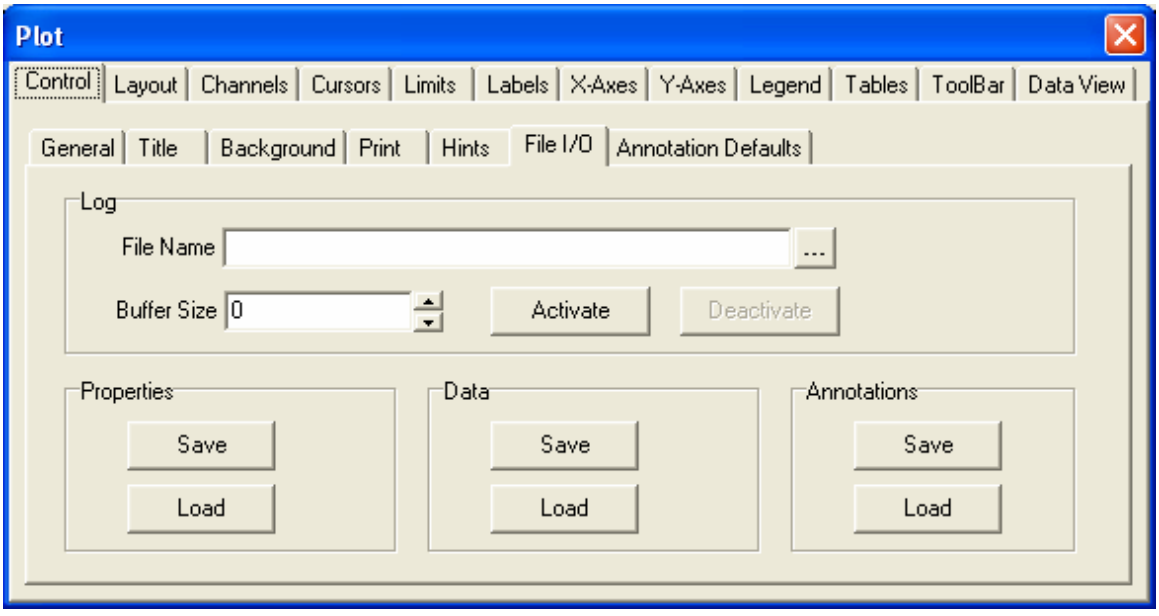
- Resume all (Tracking): Όταν είναι πατημένο αυτό το κουμπί βλέπουμε τον άξονα του χρόνου να προχωρά μαζί με τις μετρήσεις δείχνοντας μας κάθε στιγμή τι μετράμε.
- Pause all (Tracking): Αντίθετα με το προηγούμενο κουμπί, αυτό μας δείχνει μόνο τόσες μετρήσεις όσες χωράνε στο χρονικό όριο που έχουμε θέσει στον οριζόντιο άξονα, ανεξαρτήτως του ότι η συσκευή συνεχίζει να μετράει.
- Scroll (Axes): Επιτρέπει όταν είναι πατημένο την μετακίνηση των αξόνων με την βοήθεια του ποντικιού.
- Zoom (Axes): Επιτρέπει την μεγέθυνση και την σμίκρυνση των αξόνων.
- Zoom Out All Axes: Κάνει σμίκρυνση της περιοχής σχεδίασης.
- Zoom In All Axes: Κάνει μεγέθυνση της περιοχής σχεδίασης.
- Zoom Box: Μεγεθύνει μία περιοχή που θα επιλέξουμε με την βοήθεια του ποντικιού.

- Cursor: Εμφανίζει έναν κέρσορα στην περιοχή σχεδίασης ώστε να μπορούμε να δούμε τι τιμή έχουμε μετρήσει.
- Properties: Εμφανίζει ένα παράθυρο με διάφορες προχωρημένες επιλογές για την παραμετροποίηση της περιοχής σχεδίασης. Αναλυτικότερα θα δούμε τις σημαντικότερες από αυτές στην επόμενη παράγραφο.
- Copy To Clipboard: Αντιγράφει την κυματομορφή που βλέπουμε στην περιοχή σχεδίασης στο πρόχειρο του συστήματος.
- Save To File: Σώζει την κυματομορφή που βλέπουμε στην περιοχή σχεδίασης σε ένα αρχείο εικόνας.
- Print: Εκτυπώνει την κυματομορφή που βλέπουμε.

Ας δούμε όμως αναλυτικότερα τις προχωρημένες επιλογές που έχουμε σχετικά με την περιοχή σχεδίασης.

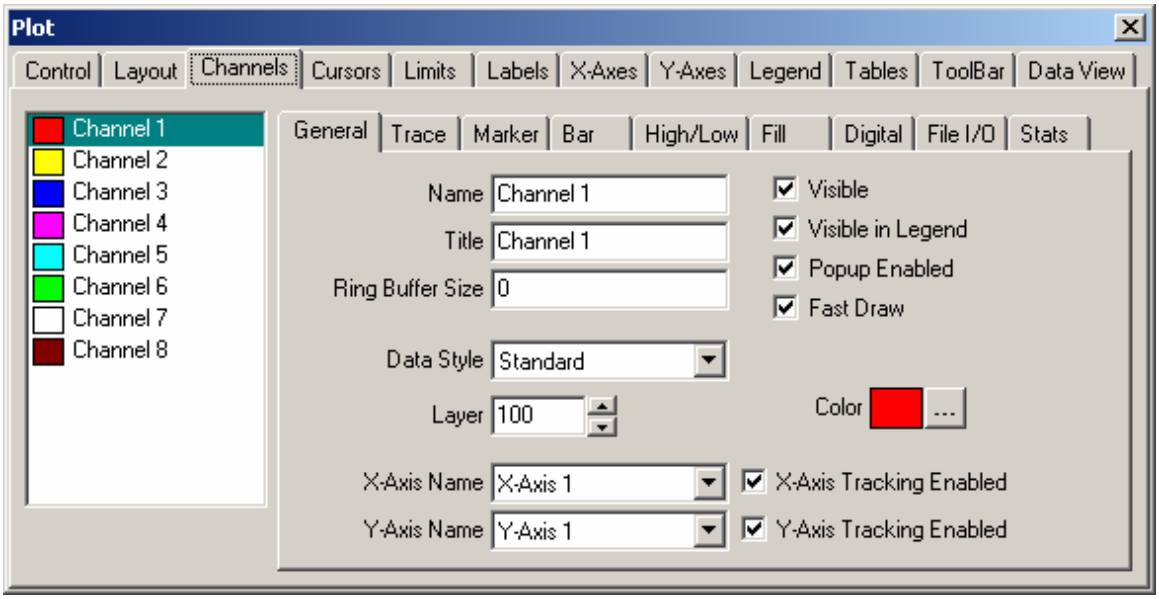
Στην εικόνα 21 βλέπουμε την καρτέλα Control απ' όπου και μπορούμε να ρυθμίσουμε κάποιες γενικές επιλογές. Μία πολύ σημαντική από αυτές, είναι αυτή σχετικά με την αποθήκευση των δεδομένων που μετράμε σε αρχείο, για περαιτέρω επεξεργασία με κάποιο σχετικό πρόγραμμα.

Εικόνα 21 – Καρτέλα Control



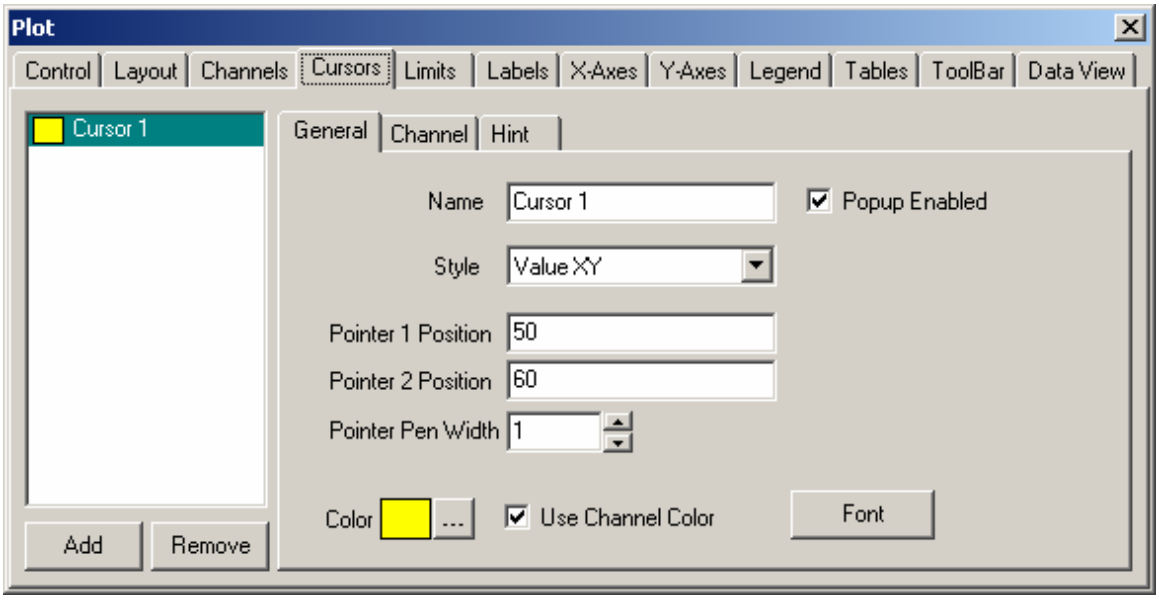
Στην καρτέλα Channels μπορούμε να ρυθμίσουμε ότι αφορά τα κανάλια που μετράμε (εικόνα 22). Ρυθμίσεις όπως το χρώμα σχεδίασης ή το όνομα του καναλιού, μεταξύ των άλλων, γίνονται από εδώ.

Εικόνα 22 - channels



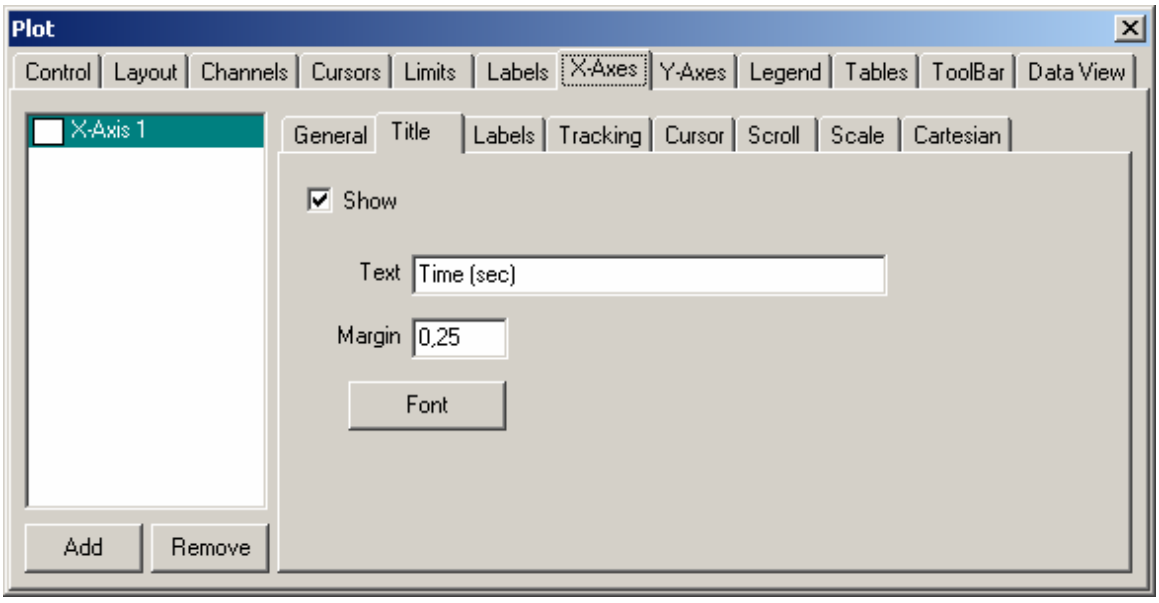
Από την καρτέλα με την ονομασία Cursors αλλάζουμε τις επιλογές σχετικά με τους κέρσορες εμφάνισης τιμών (εικόνα 23).

Εικόνα 23 - Cursors

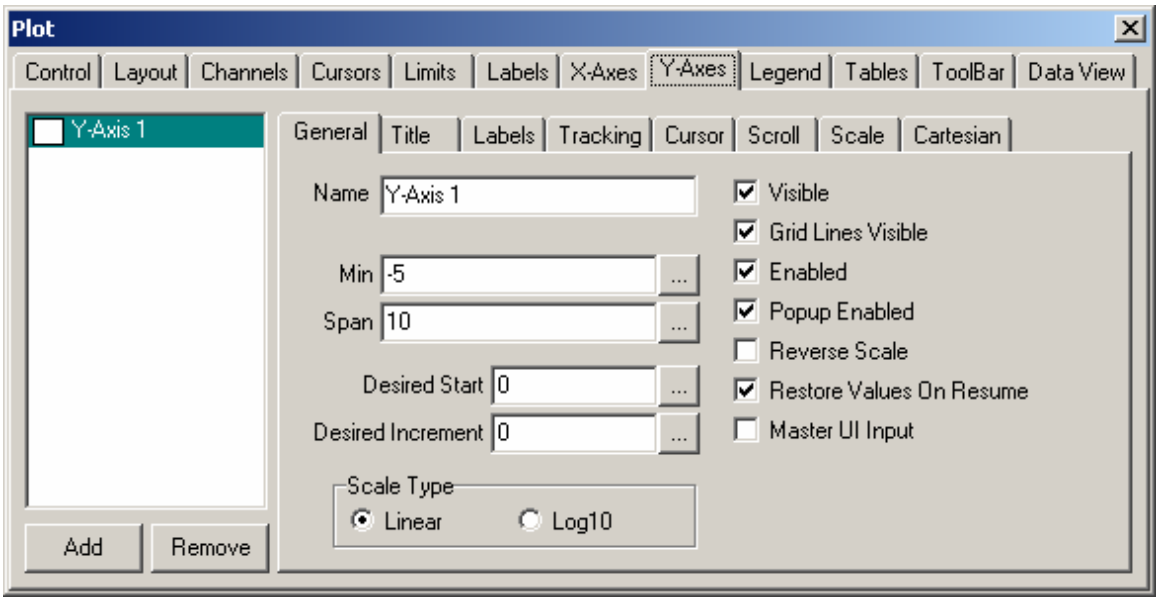


Στις εικόνες 24 και 25 βλέπουμε τις καρτέλες που αφορούν τις ρυθμίσεις σχετικά με τους άξονες X και Y. Ρυθμίσεις όπως η ονομασία του άξονα, το αν θα είναι γραμμικός ή λογαριθμικός, και άλλες γίνονται από αυτές τις δύο καρτέλες.

Εικόνα 24 – X-Axes

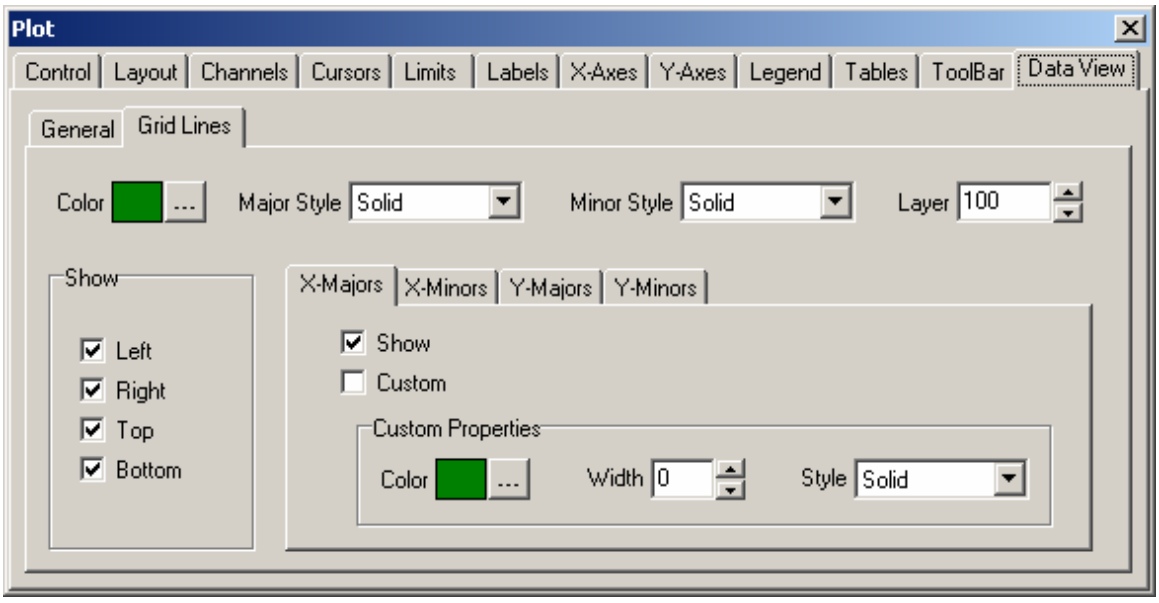


Εικόνα 25 – Y-Axes



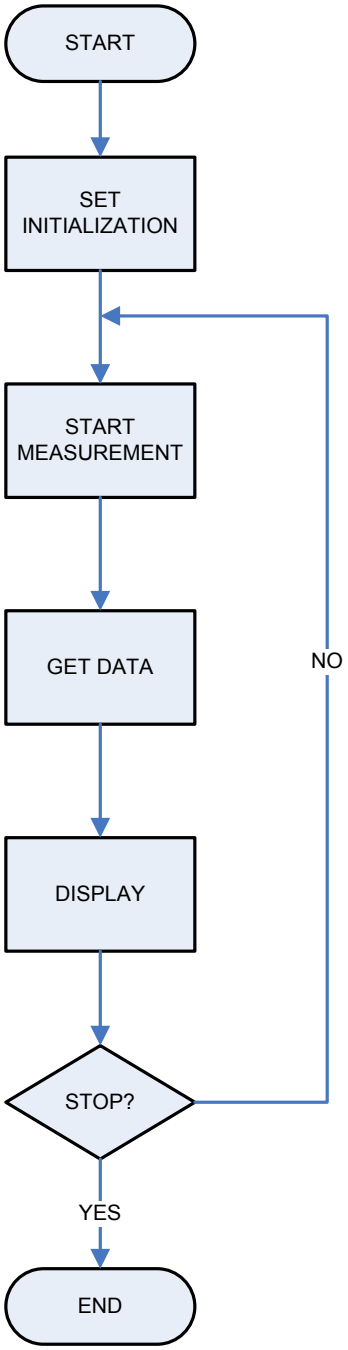
Στην καρτέλα με την ονομασία Data View έχουμε επιλογές περί των βοηθητικών γραμμών που υπάρχουν στην περιοχή σχεδίασης (Εικόνα 26).

Εικόνα 26 – Data View



Στην εικόνα 27 βλέπουμε ένα διάγραμμα ροής το οποίο μας δείχνει συνοπτικά την λειτουργία του προγράμματος που έχουμε γράψει για την χρήση της συσκευής μας από τον ηλεκτρονικό μας υπολογιστή.

Εικόνα 27 – Διάγραμμα ροής προγράμματος Delphi



Ας δούμε όμως λίγο πιο αναλυτικά την λειτουργία του προγράμματος μας. Με το που ανοίξουμε το πρόγραμμα, αυτό έρχεται σε κατάσταση αναμονής εντολών από τον χρήστη (εικόνα 14). Από την στιγμή που εμείς θα πατήσουμε το πλήκτρο Start, το πρόγραμμα μας παίρνει τις ρυθμίσεις που του έχουμε ορίσει (Set Initialization). Στην συνέχεια στέλνει τις κατάλληλες εντολές μέσω της σειριακής πόρτας στην συσκευή μας ώστε αυτή να ξεκινήσει τις μετρήσεις (Start measurement). Κάθε μέτρηση που τελειώνει

αποστέλλεται, μέσω της σειριακής πόρτας στον υπολογιστή (Get Data). Το επόμενο βήμα είναι να εμφανιστεί η κάθε μέτρηση στην οθόνη του υπολογιστή μας (Display). Αφού εμφανιστεί η εκάστοτε μέτρηση, γίνεται ένας έλεγχος εάν έχει πατηθεί το πλήκτρο Stop. Αν δεν έχει πατηθεί, τότε επαναλαμβάνεται η διαδικασία από το βήμα Start Measurement και κάτω. Σε αντίθετη περίπτωση η όλη διαδικασία σταματάει και το πρόγραμμα επανέρχεται σε κατάσταση αναμονής εντολών, έχοντας όμως κρατήσει τις μετρήσεις που έχει ήδη πάρει έτσι ώστε να μπορέσουμε να τις μελετήσουμε ή και να τις αποθηκεύσουμε.

2.4 Πρωτόκολλο επικοινωνίας.

Command List:

#Hx**<CR>** : *sets ADC's c**H**annel. <x> must be in the range 0 to 8. From 0 to 7 it represents the desired channel. The value of 8 means: report ALL channels in one packet every time(see below).*

#Lx**<CR>** : *sets ADC's sca**L**e input. <x> must be in the range 0..3.*
0 : 0..5V
1 : 0..10V
2 : ±5V
3 : ±10V

#Txxx**<CR>** : *sets sampling **T**ime period in 1msec units when in continuous measure mode. <xxx> can be in the range 001..255 when channel is 0..7 but if you chose reporting ALL channels, the range is reduced to 003..255.*

#C**<CR>** : *sets **C**ontinuously measure mode and starts measurements.*

#S**<CR>** : *sets **S**ingle shot measure mode.*

#G**<CR>** : *It means **G**o... It starts a new measure in single shot measure mode.*

Caution! ***<CR>*** *= 0D hex*

After every correctly received command, it acknowledges with a <CR>.

Returned digitized value is formatted as:

Vxxx**<CR>** : *if single channel is chosen.*

Vaaabbbcccdddeeefffgggghhh**<CR>** :
if ALL channels are chosen. <aaa> represents channel 0, and <hhh> represents channel 7.

<xxx>, <aaa>, <bbb>, <ccc>, <ddd>, <eee>, <fff>, <ggg>, <hhh>, is a 12-bit HEX value in ASCII. It shall be in the range 000..FFF.

If chosen Scale is 0..5V:	*000 = 0V*
	FFF = 5V
If chosen Scale is 0..10V:	*000 = 0V*
	FFF = 10V
If chosen Scale is ±5V:	*800 = -5V*
	000 = 0V
	7FF = +5V
If chosen Scale is ±10V:	*800 = -10V*
	000 = 0V
	7FF = +10V

Caution! ***<CR>*** *= 0D hex*

Default Settings:

A/D Input Channel = 0

A/D Input Scale = 0..5V

Sample Period Time = 10msec

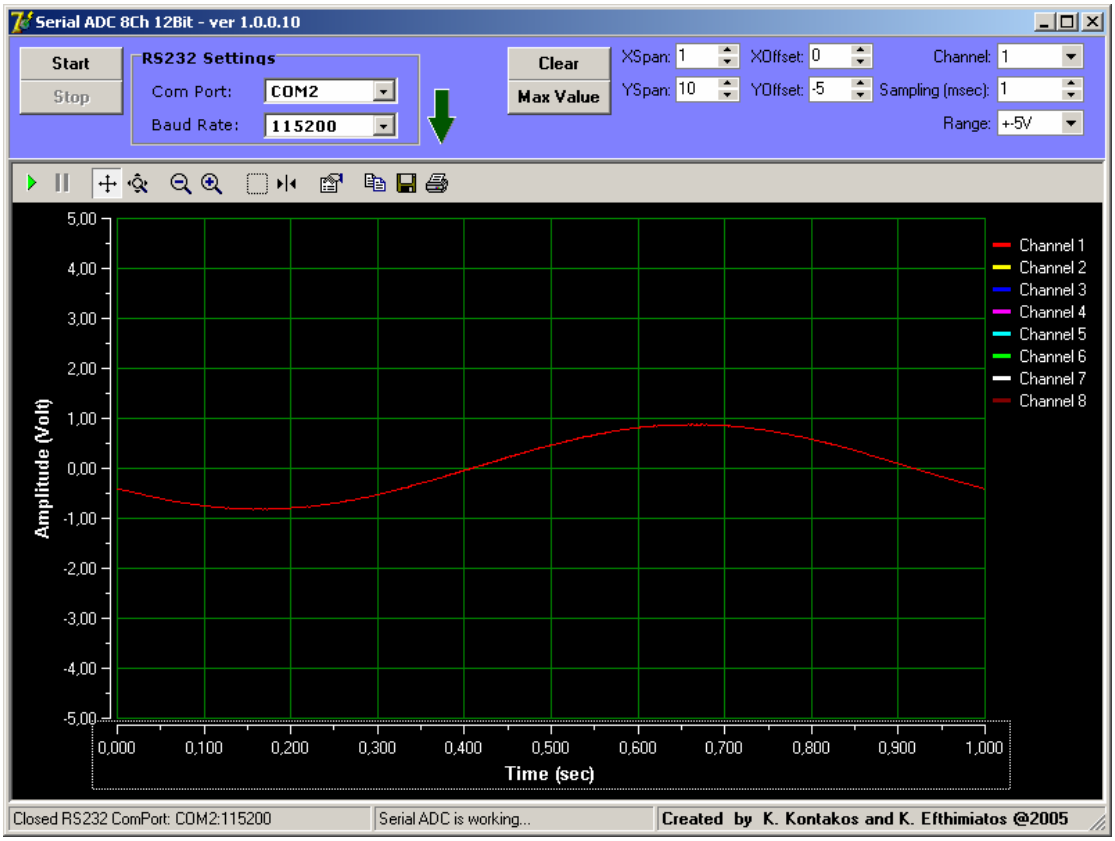
Single Shot Measure Mode

Κεφάλαιο 3 - Παραδείγματα μετρήσεων.

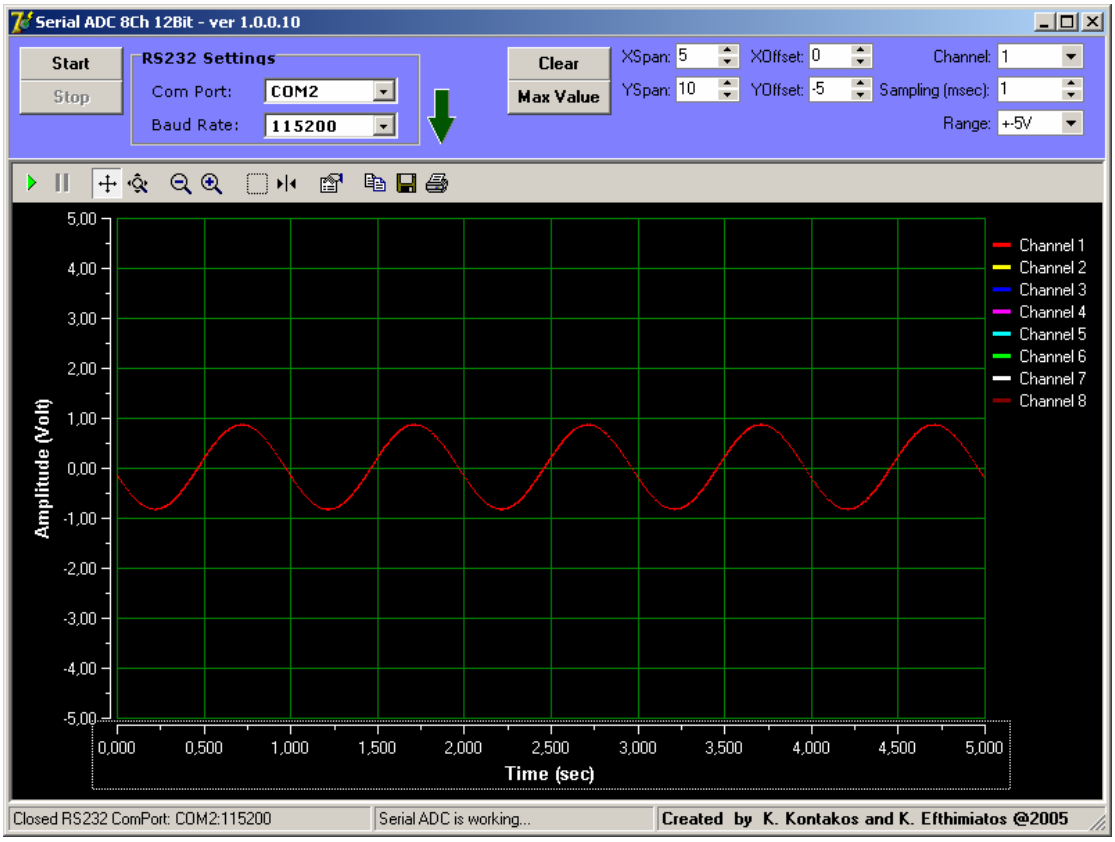
Στις εικόνες που ακολουθούν παραθέτουμε κάποια παραδείγματα μετρήσεων σε διάφορες συχνότητες, με ένα σήμα εισόδου εξ' αιτίας της έλλειψης δεύτερης παροχής σήματος αναφοράς. Στην εικόνα 26 φαίνεται το σήμα που πήραμε από την επαφή του ακροδέκτη του καναλιού 7 με το σώμα μας, ενισχυμένο X100.

Τα σήματα που μετρήσαμε προέρχονται από την συσκευή του εργαστηρίου ερευνών του Τ.Ε.Ι. Αθηνών, Programmable Synthesizer HM8134 1Hz-1GHz της HAMEG.

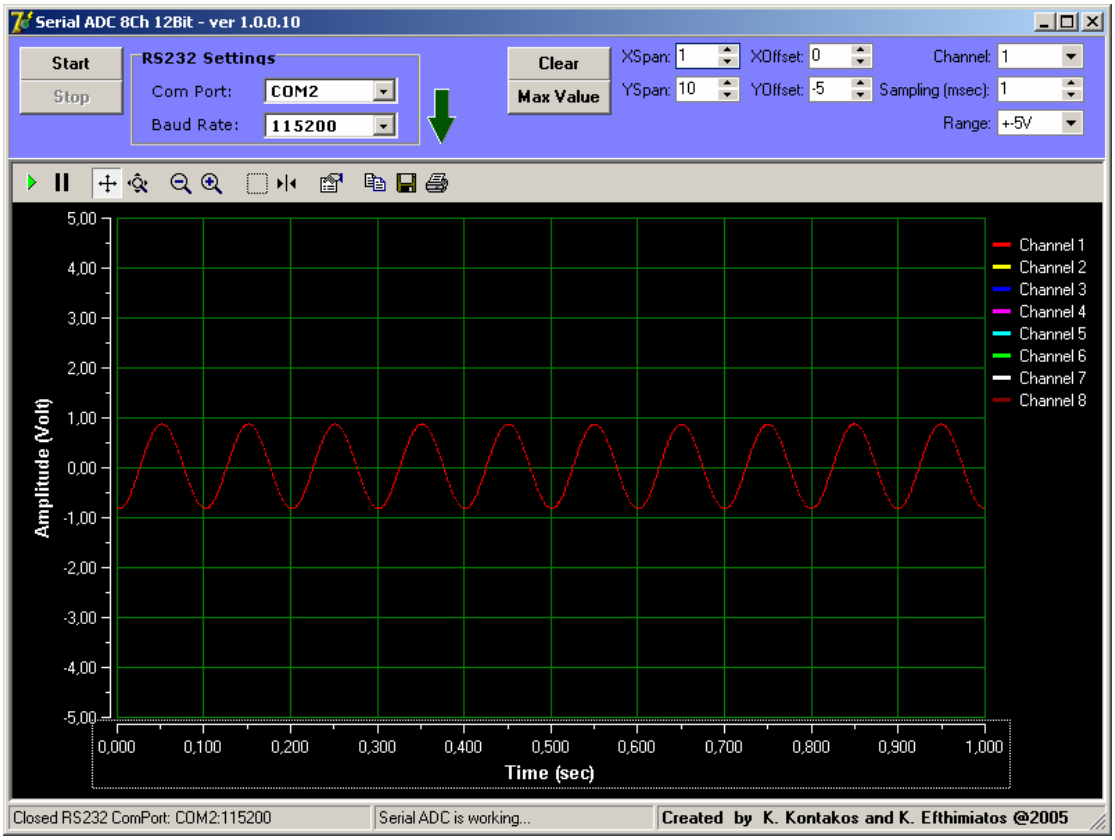
Εικόνα 19 – 1Hz, 600mVolt rms



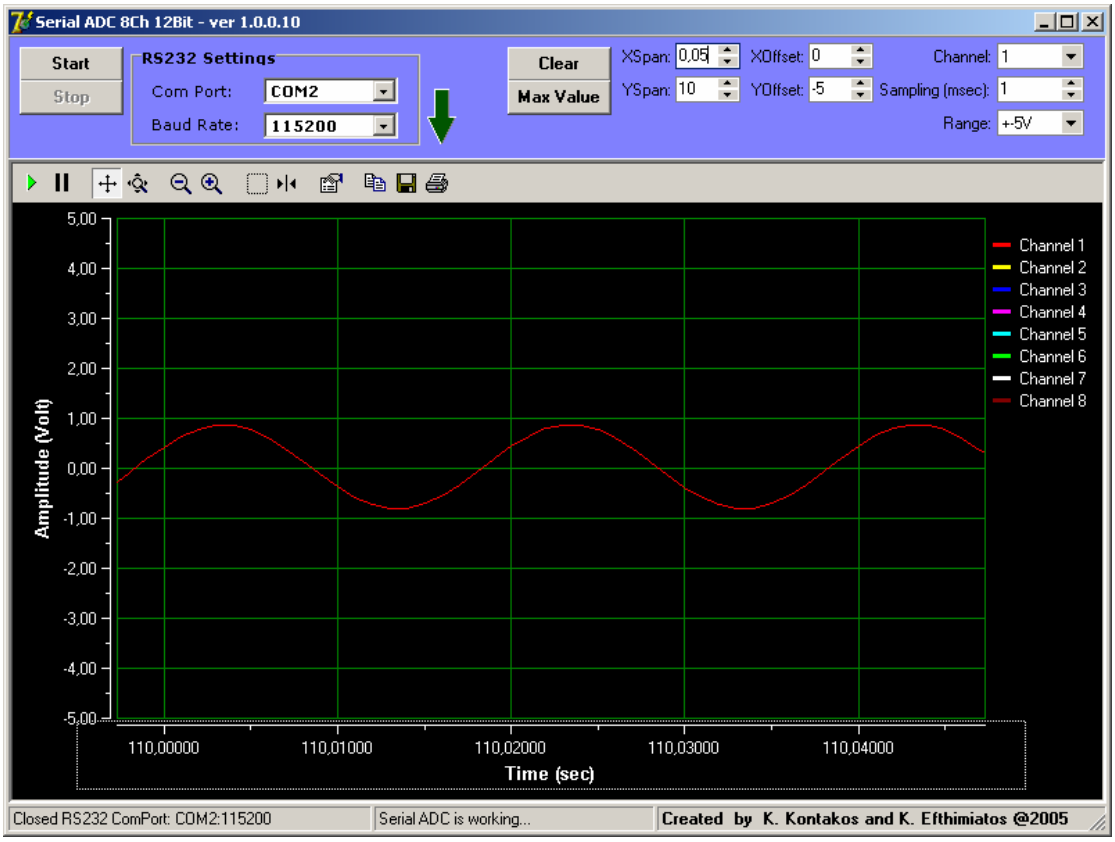
Εικόνα 20 – 1Hz, 600mVolt rms



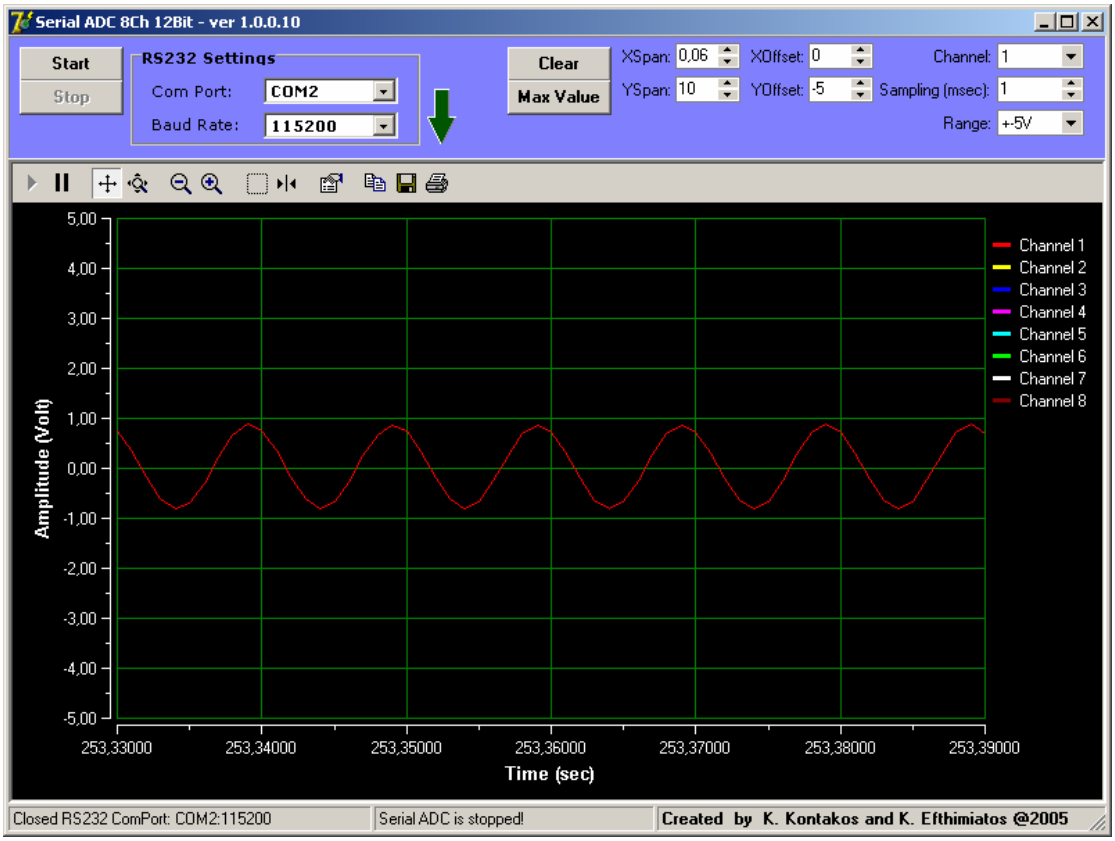
Εικόνα 21 – 10Hz, 600mVolt rms



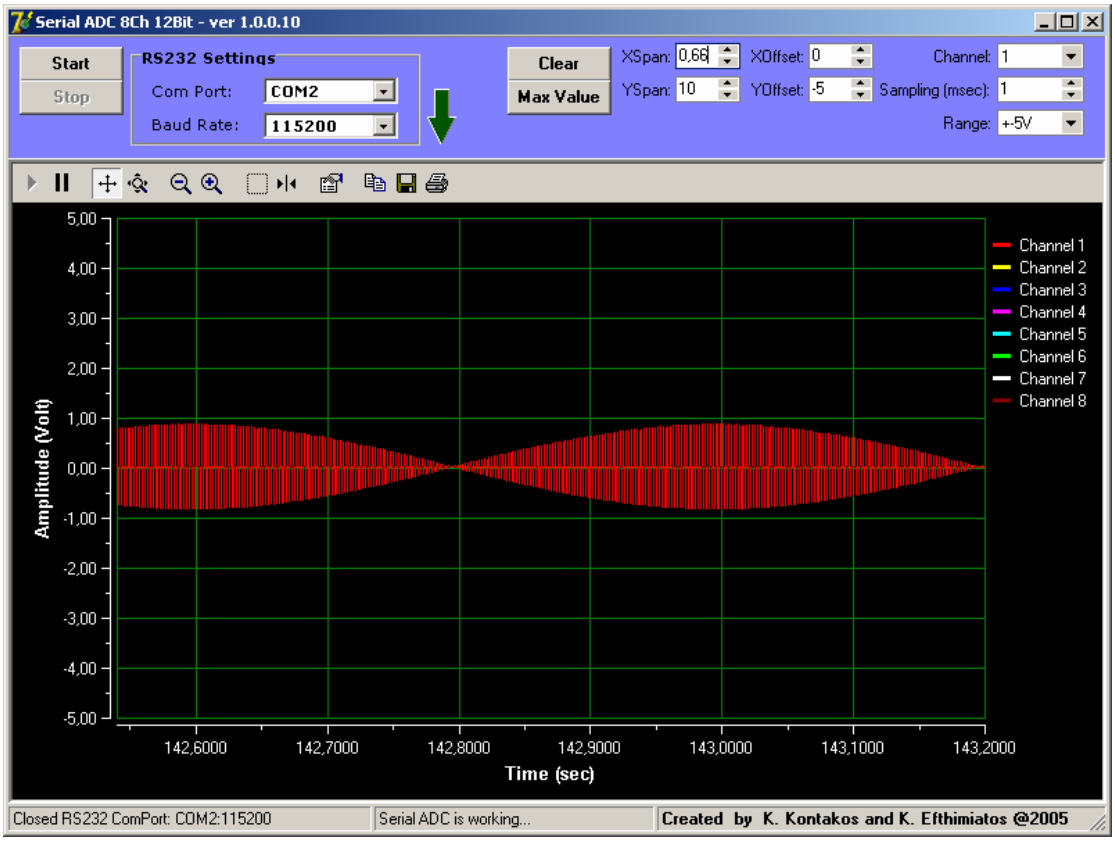
Εικόνα 22 – 50 Hz, 600mVolt rms



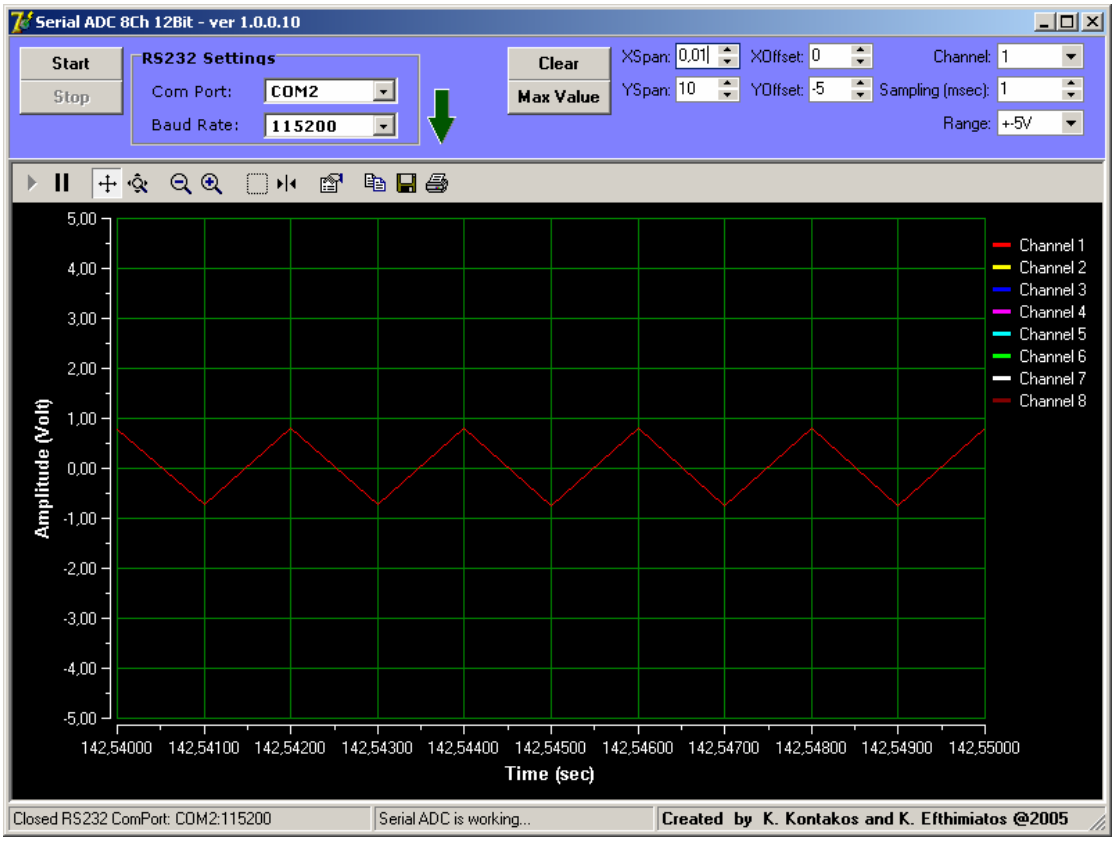
Εικόνα 23 – 100 Hz, 600mVolt rms



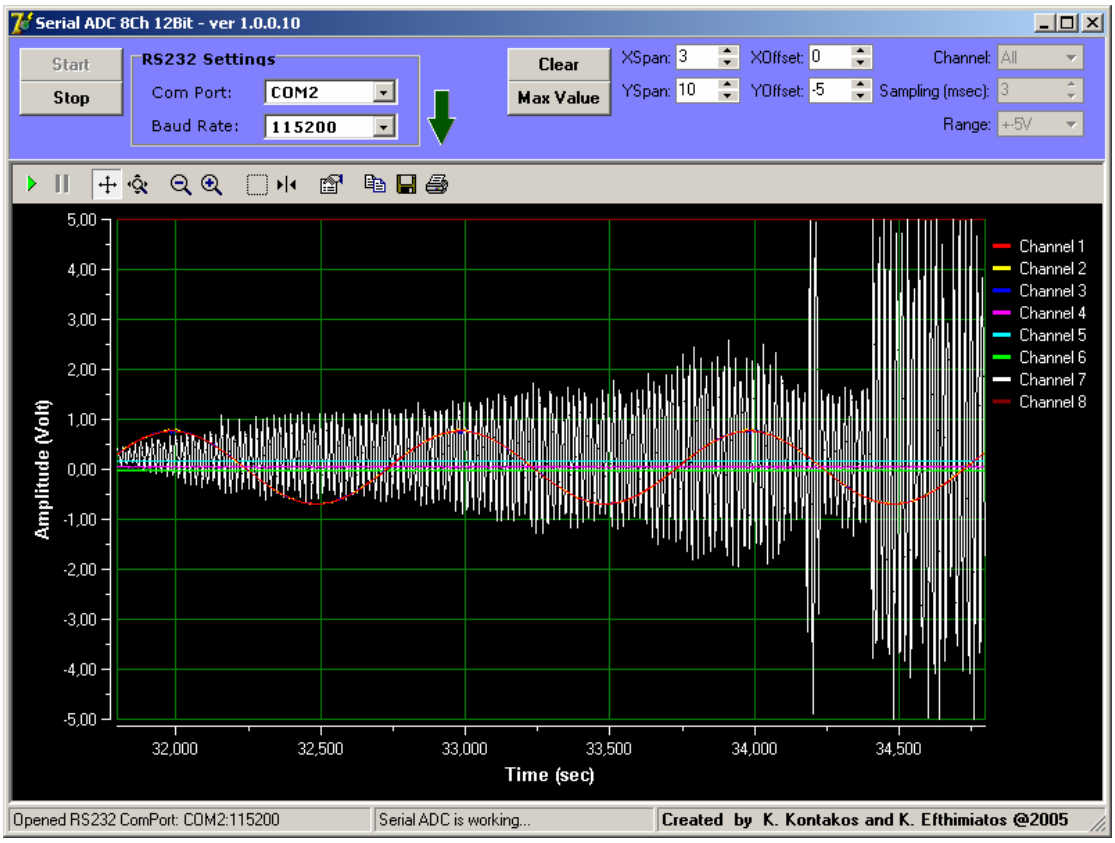
Εικόνα 24 – 500Hz, 600mVolt rms. Φαινόμενο Aliasing λόγω κανόνα Nyquist.



Εικόνα 25 – 500Hz, 600mVolt rms Zoomed



Εικόνα 26 – Ch1: 1Hz, 520 mVolt rms / Ch2: Human noise X100.



Κεφάλαιο 4 – Επίλογος, Συμπεράσματα.

Εν κατακλείδι, αναπτύξαμε μία αρκετά χρήσιμη και εύχρηστη συσκευή, η οποία είναι ικανή να καταγράψει μεγάλης διάρκειας σήματα, περιοριζόμενο μόνο από την αποθηκευτική ικανότητα του σκληρού δίσκου του υπολογιστή που την ελέγχει. Ο αριθμός των καναλιών μας επιτρέπει να καταγράφουμε μέχρι και οχτώ κανάλια ταυτόχρονα με μέγιστη συχνότητα δειγματοληψίας στο 1 KHz (μέγιστη συχνότητα σήματος εισόδου 500 Hz λόγω κανόνα Nyquist). Η διακριτική ικανότητα των 12 bit μας επιτρέπει να έχουμε μια πολύ ικανοποιητική ευκρίνεια των υπό μέτρηση σημάτων.

Η συσκευή μας χρησιμοποιήθηκε επιτυχώς σε εφαρμογές όπου χρειαζόταν καταγραφή δεδομένων σε βάθος χρόνου. Μία από αυτές τις εφαρμογές ήταν σε πτυχιακές εργασίες σπουδαστών του τμήματος Τ.Ι.Ο. του Τ.Ε.Ι. Αθηνών, όπου κατεγράφησαν αποκρίσεις οπτικών διατάξεων. Μία δεύτερη εφαρμογή είναι η βαθμονόμηση δεκτών προσεισμικών σημάτων, που χρησιμοποιούνται στο ερευνητικό πρόγραμμα του κ. Κ. Νομικού σχετικό με την πρόγνωση σεισμών. Η βαθμονόμηση αυτή γίνεται με ειδικό πρόγραμμα ελέγχου που αναπτύχθηκε από τον κ. Γρ. Κουλούρα (Ph.D. Candidate - Brunel University - Electrical Engineering & Electronics Research – England).

Κλείνοντας θα θέλαμε να ευχαριστήσουμε τον κ. Κ. Νομικό για την συμβολή του και την υποστήριξη, που μας παρείχε κατά την εκπόνηση αυτής της πτυχιακής εργασίας.

ΠΑΡΑΡΤΗΜΑ Α

Πίνακες Υλικών.

Πίνακας 3 - Κυρίως πλακέτα.

Α/Α	ΠΟΣΟΤΗΤΑ	ΣΥΜΒΟΛΟ	ΤΙΜΗ
1	1	C1	100μF/25V electrolytic
2	6	C2,C6,C7,C9,C10,C11	10μF/50V electrolytic
3	5	C3,C4,C5,C8,C16	100nF polyester
4	2	C13,C12	27pF ceramic
5	2	C18,C14	4.7μF/25V electrolytic
6	2	C19,C15	10nF polyester
7	1	C17	100pF ceramic
8	1	D1	1N4007
9	1	D2	RED LED (Rx/Tx)
10	1	J1	2-pin κλέμα (12V DC)
11	1	J2	16-pin κλέμα (12-bit Analog Input x 8 MultiRange)
12	1	L1	10μH
13	1	P1	3-pin κλέμα (to DB9 Female)
14	1	R1	4,7KΩ 1/4W
15	1	R2	470Ω 1/4W
16	1	R3	100KΩ 1/4W
17	1	S1	Push Button for PCB 90° (START)
18	1	U1	LM7805 – TO220
19	1	U2	MAX232 – DIP16
20	1	U3	AT90S2313-10PI – DIP20
21	1	U4	MAX197ACNI – DIP28
22	1	Y1	XTAL 3.6864MHz

Πίνακας 4 - Πλακέτα Buffer.

Α/Α	ΠΟΣΟΤΗΤΑ	ΣΥΜΒΟΛΟ	ΤΙΜΗ
1	9	C1,C2,C3,C4,C7,C8, C13,C14,C15	100nF polyester
2	2	C9,C5	1000uF/35V electrolytic
3	4	C6,C10,C11,C12	10uF/50V electrolytic
4	16	D1,D2,D3,D4,D5,D6,D7, D8,D9,D10,D11,D12, D13,D14,D15,D16	1N4148
5	14	D17,D18,D19,D20,D21, D22,D23,D24,D25,D26, D27,D28,D29,D30	Zener 12V
6	4	D31,D32,D33,D3	1N4007
7	1	J1	16-pin κλέμα (Analog Input 8-Ch)
8	1	J2	16-pin κλέμα (Analog Output 8-Ch)
9	1	J3	3-pin κλέμα (2x18V AC IN)
10	1	J4	2-pin κλέμα (18V DC OUT)
11	3	L1,L2,L3	10uH
12	8	R1,R3,R5,R7,R9,	1MΩ 1/4W

		R11,R13,R15	
13	8	R2,R4,R6,R8,R10, R12,R14, R16	1KΩ 1/4W
14	3	R17,R18,R19	10KΩ 1/4W
15	1	R20	9M99(10MΩ) 1/4W
16	1	R21	990K(2,2MΩ//1,8MΩ 1/4W)
17	1	R22	90K(180KΩ//180KΩ) 1/4W
18	7	R23,R24,R25,R27, R28,R29,R30	470Ω 1/4W
19	2	R31,R26	4,7KΩ 1/4W
20	2	U1,U2	MAX479 - DIP14
21	1	U3	LM7815 - TO220
22	1	U4	LM7915 - TO220

ΠΑΡΑΡΤΗΜΑ Β

Α) Κώδικας μικροελεγκτή σε AVR assembly.

```
;**** A P P L I C A T I O N   N O T E   F O R   A V R  AT90S2313 ***********
;* Title:             AD with MAX197 and serial output.
;* Version:           Ver1.3e - 2/3/2004
;* Last updated:      2/3/2004
;* Target MCU:        AT90S2313(3.6864MHz)
;* Author:            Kyriakos Kontakos – Efthimiatos Konstantinos
;* E-mail:            kkontak@hotmail.com, kostas.efthimiatos@mycosmos.gr
;* DESCRIPTION:   It measures some voltage values with the ADC MAX197
;* and sends them out by using the UART in ASCII-hex format
;* through RS232 to MAIN PROCESSING STATION.
;* Files:             ADwithMAX197.asm        -       Current file.(main)
;*                    ADwithMAX197B.asm       -       Subroutines file.
;*                    2313def.inc   -       AT90S2313 definitions file.
;*****************************************************************************

.include "2313def.inc"       ;include AT90S2313 registers definitions

;*****************************************************************************
;*              Define registers here
;*****************************************************************************

.def        DelayL      = r0    ;\
.def        DelayM      = r1    ; > Delay counters.
.def        DelayH      = r2    ;/
.def        Channel     = r3    ;Channel register.
.def        Volt0       = r4    ;LSB
.def        Volt1       = r5    ;MSB
.def        WaitCounter = r6    ;
.def        ChCntr      = r7    ;
.def        Scale       = r8    ;
;.def       reserved    = r9    ;
;.def       reserved    = r10   ;
;.def       reserved    = r11   ;
;.def       reserved    = r12   ;
;.def       reserved    = r13   ;
;.def       reserved    = r14   ;
;.def       reserved    = r15   ;

.def        temp        = r16   ;Temporary register.
.def        data        = r17   ;Data register..
.def        dataH       = r18   ;Data High register.
.def        MyStat      = r19   ;My Status register.
.def        UARTStat    = r20   ;My UART Status register.
.def        RxPointer   = r21   ;Rx Buffer pointer.
```

```
.def        Time        = r22  ;Time register(in 5ms steps)
.def        EEa         = r23  ;EEPROM address register.
.def        LEDcounter  = r24  ;
;.def       DebCntr     = r25  ;
;.def       reserved    = r26  ;
;.def       reserved    = r27  ;
;.def       reserved    = r28  ;
;.def       reserved    = r29  ;
;.def       reserved    = r30  ;
;.def       reserved    = r31  ;

;************** Register bit definitions.      **********************
;------------  UARTStat  ------------
.equ        RxJobPend   = 0    ;Rx Job Pending.
.equ        RxBufOv     = 1    ;Rx Buf Overflow.
.equ        RxBufFull   = 2    ;Rx Buffer Full.On limit.
.equ        RxBufEmp    = 3    ;Rx Buffer Empty.
;.equ       TxJobPend   = 4    ;Tx Job Pending.
;.equ       TxBufOv     = 5    ;Tx Buf Overflow.
;.equ       TxBufFull   = 6    ;Tx Buffer Full.On limit.
;.equ       TxBufEmp    = 7    ;Tx Buffer Empty.


;------------  MyStat  ------------(All flags are active High)
.equ        ContMeasure= 0     ;Execute Measurements Continuously.
.equ        EnMeasure   = 1    ;Enable Measurements.
.equ        NewMeasure= 2      ;Start New Voltage Measurement.
.equ        ButHold     = 3    ;Button Hold
;.equ       LEDOn       = 4    ;Leave LED ON for some time.
;.equ       Reserved    = 5    ;
;.equ       Reserved    = 6    ;
;.equ       Reserved    = 7    ;

;*****************************************************************************
;
;*                       Data to SET here.
;*****************************************************************************
;

;************** Constants.       ************************************

.set        Fxtal       = 3686400   ;Xtal's frequency in Hz.
                                    ;Baudrate has zero error with this
                                    ;xtal.
.set        BAUDRATE    = 115200    ;UART baudrate in bps.
.set        WaitTime    = 10        ;Time to wait between each A/D
                                    ; conversion in 1ms steps.
.set        LEDTTL      = 10        ;LED Time To Live(Main Loop's
                                    ;cycles).
;.set       Vref        = 4096000   ;ADC's Vref in µV.

;************** Leave these unchanged.   **************************
```

```
.equ        RxBufSize   = 10        ;Rx buffer size.
;.equ       TxBufSize   = 10        ;Tx buffer size.
;.set       Delay2sec   = 172       ;
;.set       delay1sec   = 138       ;
;.set       delay50ms   = 40        ;Goto clearthis(;) with SEARCH.
;.set       delay7ms    = 1         ;Also clear these(;) on the left.
;.set       divider     = 1;0       ;

.set        BAUDRATE  =(Fxtal/(16*BAUDRATE))-1
;.set       Delay2sec   = Delay2sec/divider
;.set       Delay1sec   = Delay1sec/divider
;.set       Delay50ms   = Delay50ms/divider
;.set       Vref        = Vref/10

;***** PORT B PINS *****
;           Name:           Pin: Direction:     Meaning:
.equ        D0D8            = 0  ;<I/O>        ADC's D0/D8.
.equ        D1D9            = 1  ;<I/O>        ADC's D1/D9.
.equ        D2D10           = 2  ;<I/O>        ADC's D2/D10.
.equ        D3D11           = 3  ;<I/O>        ADC's D3/D11.
.equ        D4              = 4  ;<I/O>        ADC's D4.
.equ        D5              = 5  ;<I/O>        ADC's D5.
.equ        D6              = 6  ;<I/O>        ADC's D6.
.equ        D7              = 7  ;<I/O>        ADC's D7.

;***** PORT D PINS *****
;           Name:           Pin:  Direction:    Meaning:
.equ        RXD             = 0   ;<I>         RS232 Input.
.equ        TXD             = 1   ;<O>         RS232 Output.
.equ        ADCINT          = 2   ;<I>         INT from ADC.
.equ        WR              = 3   ;<O>         WR to ADC.
.equ        RD              = 4   ;<O>         RD to ADC.
.equ        HBEN            = 5   ;<O>         HBEN to ADC.
.equ        LEDBUT          = 6   ;<I/O>       LED & BUTTON.

;***** "Scale" register bits *****
;           Name:           Pin:  Direction:    Meaning:
.equ        RNG             = 0   ;<I>         ADC's Range
                                  ;            5V(0)/10V(1) bit.
.equ        BIP             = 1   ;<O>         ADC's
                                  ;            Unipolar(0)/Bipolar(1)
                                  ;            bit.


;*****************************************************************************
;*                      EEPROM Segment.
;*****************************************************************************
.eseg                               ;EEPROM SEGMENT
.org 1                              ;START of EEPROM.
```

```
;.db            0b00000000          ;dummy byte.


;*********************************************************************
;*                      SRAM Segment.
;*********************************************************************
.dseg                                           ;SRAM SEGMENT
.org RAMEND-127                                 ;START of SRAM.

RxBufAddr:   .byte   RxBufSize    ;Rx Buffer.


;*********************************************************************
;*                      FLASH code starts here.
;*********************************************************************
.cseg                                   ;CODE SEGMENT
.org 0                                  ;START CODE
                     rjmp   RESET       ;Jump to RESET interrupt routine.

.org INT0addr                           ;External Interrupt0 Vector Address
                     reti   ;INT0       ;No External Interrupt(INT0).

.org INT1addr                           ;External Interrupt1 Vector Address
                     reti   ;INT1       ;No External Interrupt(INT1).

.org ICP1addr                           ;Input Capture1 Interrupt Vector
                                        ;Address
                     reti   ;ICP1       ;Input Capture1 Interrupt.

.org OC1addr                            ;Output Compare1 Interrupt Vector
                                        ;Address
                     reti   ;OC1        ;Output Compare1 Interrupt.

.org OVF1addr                           ;Overflow1 Interrupt Vector Address
                     reti   ;OVF1       ;Overflow1 Interrupt.

.org OVF0addr                           ;Overflow0 Interrupt Vector Address
                     reti   ;OVF0       ;Overflow0 Interrupt.

.org URXCaddr                           ;UART Receive Complete Interrupt
                                        ;Vector Address
                     rjmp   UARTRxC     ;UART Receive Complete Interrupt.

.org UDREaddr                           ;UART Data Register Empty Interrupt
                                        ;Vector Address
                     reti   ;UDRE      ;UART Data Register Empty Interrupt.

.org UTXCaddr                           ;UART Transmit Complete Interrupt
                                        Vector Address
```

```
                reti    ;UTXC       ;UART Transmit Complete Interrupt.

.org ACIaddr                        ;Analog Comparator Interrupt Vector
                                    ;Address
                reti    ;ACI        ;Analog Comparator Interrupt.



;***************************************************************************
;*          |Pin:       7     6     5     4     3     2     1     0
;*PORT |Name:     D7    D6    D5    D4D12 D3D11 D2D10 D1D9 D0D8
;*  B      |Direction: IO    IO    IO    IO    IO    IO    IO    IO
;*-------------------------------------------------------------------------
;*          |Pin:       7     6     5     4  3     2     1     0
;*PORT |Name:     x     LEDBUT HBEN RD WR    ADCINT TXD RXD
;*  D      |Direction: x     IO    O     O  O     I     O     I
;***************************************************************************

RESET:
        wdr                         ;\
        ldi     temp,0b00001111     ; > Set WatchDog for 1.9s.
        out     WDTCR,temp          ;/
        wdr

        ldi     temp,1<<ACD
        out     ACSR,temp           ;Disable Analog Comparator.

        ldi     temp,RAMEND         ;Initialize StackPointer
        out     SPL,temp            ; to end of internal RAM.

        ldi     temp,0b00000000     ;
        out     DDRB,temp           ;Initialise
        ldi     temp,0b01111010     ;direction
        out     DDRD,temp           ;of ports.

        ldi     temp,0b11111111     ;
        out     portb,temp          ;Portb=0s & 1s
        ldi     temp,0b00011001     ;
        out     portd,temp          ;Portd=0s & 1s

        ldi     temp,70                 ;
        rcall   Delay                   ;Wait for...
                                        ;70->283ms,100->820ms,110-
                                        ;>1.093ms

        ;cbi    ddrd,LEDBUT             ;Turn OFF LED.

        clr     DelayL              ;Initialize some registers.
        clr     DelayM              ;
```

```
                clr     DelayH              ;
                clr     temp                ;
                clr     zl
                clr     zh
                clr     data                ;
                clr     Channel             ;Default ADC channel = 0.
                clr     Scale           ;Default ADC input Scale = 0..5V.
                clr     RxPointer
                ldi     LEDcounter,LEDTTL
                ldi     UARTStat,(1<<RxBufEmp)
                ldi     Time,WaitTime
                ldi MyStat,(0<<ContMeasure)|(0<<EnMeasure)|
                                (0<<NewMeasure)|(0<<ButHold)  ;Default Single
                                ;                             Measurements.
;               ldi     DebCntr,$39

                clr     zh                  ;
                ldi     zl,RAMEND-127       ;Fill all RAM with zeros.
                                            ; ClearRAM:
                st      z+,zh               ;
                cpi     zl,RAMEND+1         ;
                brne    ClearRAM            ;

;---------------------- Initialize Timer1 ----------------------------------

                in      temp,TIMSK
                cbr     temp,1<<TOIE1       ; 1
                out     TIMSK,temp          ; 1 Disable timer interrupt.

;---------------------- Initialize UART interface. -------------------------

                ldi     temp,BAUDRATE       ;
                out     UBRR,TEMP           ; load baudrate

                ldi     temp,(1<<TXEN)|(1<<RXEN)|(0<<TXCIE)|(1<<RXCIE)

                out     UCR,temp            ;enable tx/rx, enable rx int.

                ;out    UDR,temp            ;


;---------------------- Send welcome message through UART. -------------------

                ldi     zl,low(WelcomeMes*2)
                ldi     zh,high(WelcomeMes*2)
                rcall   PutMes

                ldi     zl,low(DefaultSetMes*2)
                ldi     zh,high(DefaultSetMes*2)
                rcall   PutMes
```

```
                ldi     zl,low(CommandList*2)
                ldi     zh,high(CommandList*2)
                rcall   PutMes


                rjmp    MainLoop            ;Goto Main Loop.


;--------------------- Initialize ADC. -----------------------------------

InitADC:
                mov     data,Channel        ;Set Channel
                rcall   ReadADC             ;Just do a dummy read

                tst     temp                ;If ADC doesn't respond,
                brne    InitADC_OK          ; report error.

                ldi     zl,low(ADCErrorMes*2)
                ldi     zh,high(ADCErrorMes*2)
                rcall   PutMes

InitADC_OK:

                rjmp    MainLoop            ;Goto Main Loop.

;--------------------- Measure Voltage from ADC. ------------------------------
StartMeasurement:
                sbi     ddrd,LEDBUT         ;Turn LED On to show a new
                                            ;measurement.

                cbr     MyStat,1<<NewMeasure    ;Kill "NewMeasure" flag, since
                                                ; it is only an one-time flag.

                sei


;               inc     DebCntr
;               cpi     DebCntr,$3A         ;Loop DebCntr in ASCII $30..$39.
;               brne    ContinueMeas        ;  '0'..'9'
;               ldi     DebCntr,$30


        ;       in      DelayM,ddrd
        ;       ldi     temp,1<<LEDBUT      ;Toggle LED every 10
                                            ;measurements.
        ;       eor     DelayM,temp
        ;       out     ddrd,DelayM
ContinueMeas:
```

```
                sbrs    Channel,3
                rjmp    SingleChannel

MeasureAllChannels:

;----- Set timer

                clr     temp
                out     TCCR1b,temp         ; 1 Prescaler=Counter1 off.
                ldi     temp,$D4;D4;F1;F1;B8
                out     TCNT1H,temp         ; 1 Load timer for 3ms;1ms;5ms.
                ldi     temp,$FF;CD;A1;9E;00
                out     TCNT1L,temp         ;
                ldi     temp,1<<TOV1        ; 1
                out     TIFR,temp           ; 1 Clear possible old overflow.
;               in      temp,TIMSK
;               cbr     temp,1<<TOIE1       ; 1
;               out     TIMSK,temp          ; 1 Disable timer interrupt.
                ;sei                        ; 1 Enable global interrupts.
                ldi     temp,0b00000001     ; 1
                out     TCCR1b,temp         ; 1 Prescaler=Clock.

;----- /Set timer

                ldi     data,'V'
                rcall   putchar             ;'V'

                clr     ChCntr

MeasureAllChannels_1:
                mov     data,ChCntr         ;Set Channel
                rcall   ReadADC             ;Do dummy read.

                mov     data,ChCntr         ;Set Channel
                rcall   ReadADC             ;Measure voltage

                mov     Volt0,data
                mov     Volt1,dataH

;               sbrs    Scale,BIP                   ;If Bipolar mode,
;               rjmp    MeasureAllChannels_2

;               lsr     Volt1                       ;throw away LS Bit0.
;               ror     Volt0

;MeasureAllChannels_2:

                mov     data,Volt1
                rcall   BINtoASCIIHex
```

```
                rcall   putchar             ;Voltage's (3)BYTE IN ASCII-HEX.

                mov     data,Volt0
                swap    data
                rcall   BINtoASCIIHex
                rcall   putchar             ;Voltage's (2)BYTE IN ASCII-HEX.

                mov     data,Volt0
                rcall   BINtoASCIIHex
                rcall   putchar             ;Voltage's (1)LS BYTE IN ASCII-HEX.

                inc     ChCntr
                sbrs    ChCntr,3
                rjmp    MeasureAllChannels_1

                ldi     data,$0D            ;Send <CR>
                rcall   putchar

                mov     WaitCounter,Time
                dec     WaitCounter
                dec     WaitCounter

                rjmp    MeasureEnd

SingleChannel:

;----- Set timer

                clr     temp
                out     TCCR1b,temp         ; 1 Prescaler=Counter1 off.
                ldi     temp,$F1;F1;B8
                out     TCNT1H,temp          1 Load timer for 1ms;5ms.
                ldi     temp,$DF;9A;00
                out     TCNT1L,temp         ;
                ldi     temp,1<<TOV1        ; 1
                out     TIFR,temp           ; 1 Clear possible old overflow.
;               in      temp,TIMSK
;               cbr     temp,1<<TOIE1       ; 1
;               out     TIMSK,temp          ; 1 Disable timer interrupt.
                ;sei                        ; 1 Enable global interrupts.
                ldi     temp,0b00000001     ; 1
                out     TCCR1b,temp         ; 1 Prescaler=Clock.

;----- /Set timer

                mov     data,Channel        ;Set Channel.
                rcall   ReadADC             ;Do dummy read.

                mov     data,Channel        ;Set Channel.
                rcall   ReadADC             ;Measure voltage.
```

```
                mov     Volt0,data
                mov     Volt1,dataH


                tst     temp                    ;If ADC doesn't respond,
                brne    SingleChannel_ADC_OK    ; report error.

                ldi     zl,low(ADCErrorMes*2)
                ldi     zh,high(ADCErrorMes*2)
                rcall   PutMes

SingleChannel_ADC_OK:


                ldi     data,'V'
                rcall   putchar                 ;'V'

;               sbrs    Scale,BIP               ;If Bipolar mode,
;               rjmp    SingleChannel_2

;               lsr     Volt1                   ;throw away LS Bit0.
;               ror     Volt0

;SingleChannel_2:

                ;mov    data,Volt1
                ;swap   data
                ;rcall  BINtoASCIIHex
                ;rcall  putchar          ;Voltage's (4)MSBYTE IN ASCII-HEX.

                mov     data,Volt1
                rcall   BINtoASCIIHex
                rcall   putchar          ;Voltage's (3)BYTE IN ASCII-HEX.

                mov     data,Volt0
                swap    data
                rcall   BINtoASCIIHex
                rcall   putchar          ;Voltage's (2)BYTE IN ASCII-HEX.

                mov     data,Volt0
                rcall   BINtoASCIIHex
                rcall   putchar          ;Voltage's (1)LS BYTE IN ASCII-HEX.


;               mov     data,DebCntr     ;Debugging
;               rcall   putchar

                ldi     data,$0D         ;Send <CR>
                rcall   putchar
```

```
        ;       ldi     data,$0A            ;Send <LF> This might be useless.
        ;       rcall   putchar

                mov     WaitCounter,Time

MeasureEnd:

;--------------------- Wait for some ms declared by "Time" register. -----
;                                   [1..255 x 1ms]

                ;mov    WaitCounter,Time

WaitForAWhile_1:
                in      temp,TIFR
                sbrs    temp,TOV1
                rjmp    WaitForAWhile_1

WaitForAWhile:
                dec     WaitCounter
                breq    WaitForAWhile_End

                clr     temp
                out     TCCR1b,temp         ; 1 Prescaler=Counter1 off.
                ldi     temp,$F1;F1;B8
                out     TCNT1H,temp         ; 1 Load timer for 1ms;5ms.
                ldi     temp,$AA;9E;00
                out     TCNT1L,temp         ;
                ldi     temp,1<<TOV1        ; 1
                out     TIFR,temp           ; 1 Clear possible old overflow.
                ;in     temp,TIMSK
                ;cbr    temp,1<<TOIE1       ; 1
                ;out    TIMSK,temp          ; 1 Disable timer interrupt.
                ;sei                        ; 1 Enable global interrupts.
                ldi     temp,0b00000001     ; 1
                out     TCCR1b,temp         ; 1 Prescaler=Clock.

                rjmp    WaitForAWhile_1

WaitForAWhile_End:

        ;       ldi     data,'W'            ;Debugging
        ;       rcall   putchar
        ;       mov     data,WaitCounter
        ;       rcall   putchar


;
;               mov     temp,Time
;WaitForAWhile:
```

```
;               rcall   Delay1ms
;               dec     temp
;               brne    WaitForAWhile
;

;--------------------- Main Loop. -----------------------------------------

MainLoop:
                wdr
                ;sei

                ;nop

                ;sbrc   UARTStat,RxJobPend      ;If anyone from these
                        conditions occurs,
;               rcall   UARTRxDec               ; go to decode the command
                                                ; in RX buffer.
;               sbrc    UARTStat,RxBufFull      ;
;               rcall   UARTRxDec               ;
;               sbrc    UARTStat,RxBufOv        ;
;               rcall   UARTRxDec               ;

                cli

        ;       rjmp    ReleaseButton           ;Enable this line to test time
                                                ; accuraccy.

                cbi     ddrd,LEDBUT     ;If someone presses the button,
                rcall   Delay10us       ; and all conditions are ok,
                sbic    pind,LEDBUT
                rjmp    NoButtonPress

                sbrc    MyStat,ButHold
                rjmp    ReleaseButton

                sbr     MyStat,1<<NewMeasure    ; set flag to start a new
                                                ; measurement.


                ldi     temp,1<<EnMeasure
                eor     MyStat,temp             ;Toggle "EnMeasure" flag.
                sbrs    MyStat,EnMeasure        ;If Measurements are
                                                ; disabled,
                cbr     MyStat,1<<NewMeasure    ; clear "NewMeasure" flag.

                sbr     MyStat,1<<ButHold
                rjmp    ReleaseButton
NoButtonPress:
                cbr     MyStat,1<<ButHold
ReleaseButton:
```

```
                sbrs    MyStat,EnMeasure
                rjmp    StopMeasurements

                ;sbr    MyStat,1<<NewMeasure
                sbrc    MyStat,ContMeasure
                sbr     MyStat,1<<NewMeasure

                ;nop
                ;ldi    data,'W'
                ;rcall  putchar

                sbrc    MyStat,NewMeasure       ;
                rjmp    StartMeasurement        ;If flag for continuous A/D
                                                ; measurement
                                                ; is set, do it.
StopMeasurements:

                sei

        ;       tst     LEDcounter          ;Turn On LED for a while to show
        ;       breq    OffLED              ; activity in UART.
        ;       sbi     ddrd,LEDBUT
        ;???????????????????????????????????
        ;       dec     LEDcounter
        ;       brne    OffLED
        ;       ;cbr    MyStat,LEDOn
        ;       cbi     ddrd,LEDBUT
        ;???????????????????????????????????
OffLED:


                rjmp    MainLoop    ;Loop forever to MainLoop.
                rjmp    MainLoop
                rjmp    MainLoop


;********************* Interrupt Service routines. ************************

;---------------------- Rx Interrupt service routine. ----------------------
UARTRxC:
                ;sbi    portb,DB7                   ;Test command.
                push    r1                          ;Preserve registers.
                in      r1,SREG                     ;Preserve main OS status reg.
                push    data
                push    zl
                push    zh
```

```
            push    temp

            clr     data
            ldi     zl,low(RxBufAddr)   ;Pointer z to our RxBuffer's tail.
            ldi     zh,high(RxBufAddr)
            add     zl,RxPointer
            adc     zh,data

            in      data,UDR            ;Get the incomming char.

            ;rcall  putchar             ;Echoes back everything received.

            sbic    USR,FE              ;If Framming error occurs,
            rjmp    UARTRxC_FE          ;
            sbic    USR,OR              ; or Overrun error,
            rjmp    UARTRxC_OR          ; do Rx-error routine.
            sbrc    UARTStat,RxBufFull      ;If RxBuffer is Full,
            rjmp    UARTRxC_BF              ;
            sbrc    UARTStat,RxBufOv        ; or RxBuffer overflowed,
            rjmp    UARTRxC_BO              ; do Rx-error routine.


            st      z,data                  ;RxBuffer[RxTail] = UDR
            cbr     UARTStat,1<<RxBufEmp    ;

            cpi     data,$0D                ;<CR> (<-Terminal send this).
            brne    UARTRxC_1
            sbr     UARTStat,1<<RxJobPend
UARTRxC_1:
            cpi     data,'#'                ;# (<-Terminal send this).
            brne    UARTRxC_2
            ldi     zl,low(RxBufAddr)       ;Pointer z to our RxBuffer's
                                            ; start.
            ldi     zh,high(RxBufAddr)
            st      z,data                  ;Store char '#' there.
            clr     RxPointer               ;RxPointer points at RXBuffer
                                            ; start.
UARTRxC_2:
            inc     RxPointer               ;Increase Rx tail pointer.
            cpi     RxPointer,RxBufSize     ;If RxBuffer is full, set flag.
            brne    UARTRxC_3
            sbr     UARTStat,1<<RxBufFull
UARTRxC_3:

;***** Xoff calcs go here *****

            ;rjmp   UARTRxC_end
UARTRxC_Er:
UARTRxC_FE:
            ;rjmp   UARTRxC_end
```

```
UARTRxC_BO:
UARTRxC_OR:
                rjmp    UARTRxC_end

UARTRxC_BF:
                sbr     UARTStat,1<<RxBufOv     ;Set Buffer Overflow flag.
                rjmp    UARTRxC_end

;***** Decoding commands could be here if necessary but now they are in a
;                                                                routine. *****

UARTRxC_end:
                sbrc    UARTStat,RxJobPend      ;If anyone of these Flags is
                                                ; set,
                rcall   UARTRxDec               ; go to decode the command
                                                ; in RX buffer.
                sbrc    UARTStat,RxBufFul       ;
                rcall   UARTRxDec               ;
                sbrc    UARTStat,RxBufOv        ;
                rcall   UARTRxDec               ;

                ldi     LEDcounter,LEDTTL
                pop     temp
                pop     zh
                pop     zl
                pop     data
                out     SREG,r1             ;Restore previous status reg.
                pop     r1
                ;cbi    portb,DB5           ;Test command.
                reti                        ;Return to normal program status.

;---------------------- Tx Interrupt service routine. ----------------------


;---------------------- Interrupt service routines end. ----------------------

.include        "ADwithMAX197b.asm"                 ;Subroutines file.
```

Β) Υπορουτίνες μικροελεγκτή σε assembly.

```
;***************************************************************************
;*                      Subroutines start here.
;***************************************************************************
;* Version:                 1.3e
;* Last updated:    2/3/2004
;***************************************************************************
;*Subroutine:       ADCWr8bit
;*Description:This subroutine puts the 8 bits from "data" in ADC.
;*Registers:  data - As input data.Unchanged.
;*              temp,cntr8 - As temporary file.Unchanged.
;***************************************************************************
ADCWr8bit:
                push  data              ;
                push  temp              ;

                sbi   portd,WR
                sbi   portd,RD
                ldi   temp,$FF
                out   ddrb,temp
                out   portb,data
                rcall Delay10us
                cbi   portd,WR
                rcall Delay10us
                sbi   portd,WR

                pop   temp      ;
                pop   data      ;
                ret             ;Return.
;_____________________________________________________________________
_____________

;***************************************************************************
;*Subroutine:       ADCRd16bit
;*Description:This subroutine gets 16 bits from ADC and puts them
;*              in "dataH:data".
;*Registers:  dataH,data - As output data.
;*              temp - As temporary file.Unchanged.
;***************************************************************************
ADCRd16bit:
                push  temp              ;

                sbi   portd,WR
                sbi   portd,RD
                cbi   portd,HBEN

                ldi   temp,$00
                out   ddrb,temp
```

```
                cbi     portd,RD

                rcall   Delay10us

                in      data,pinb

                sbi     portd,HBEN

                rcall   Delay10us

                in      dataH,pinb

                sbi     portd,RD

                pop     temp                ;
                ret                         ;Return.
;_____________________________________________________________________________

;*****************************************************************************
;*Subroutine: ReadADC
;*Description:This subroutine makes an ADC measure.
;*Registers:  data - As input ADC's channel low 3-bit data sign.
;*           dataH,data - As output data.
;*****************************************************************************
ReadADC:
                ;push  temp
                andi    data,0b00000111     ;Mask the 3 LS bits.

                ;   ++------- 01          |Normal operation/Internal Mode
                ;   ||+------ 0   |Internally controlled acquisition
                ;   |||++---- 00 |Input range: 0 to 5V
                ;   |||||+++- xxx|Channel
                ori     data,0b01000000         ;
                sbrc    Scale,RNG
                ori     data,0b00010000         ;
                sbrc    Scale,BIP
                ori     data,0b00001000         ;
                rcall   ADCWr8bit           ;Write to control register.

                ser     temp
ReadADC_INT:
                dec     temp
                breq    ReadADC_Err

                rcall   Delay10us
                sbic    pind,ADCINT         ;
                rjmp    ReadADC_INT         ;

                rcall   ADCRd16bit          ;Read Volt Data.MSByte
```

```
ReadADC_Err:
                ret                     ;4 Return.
;_____________________________________________________________________

;***************************************************************************
;*Subroutine: Delay10us
;*Description:This subroutine makes a delay for ~10us.
;*Registers:   temp - As temporary file.Unchanged.
;***************************************************************************
Delay10us:
                push    temp            ;3+2
                ldi     temp,1          ;1
                rcall   Delay           ;-> 28
                pop     temp            ;2
                ret                     ;4 Return.
;_____________________________________________________________________

;***************************************************************************
;*Subroutine: Delay1ms
;*Description:This subroutine makes a delay for 1.002332899ms.
;*Registers:   temp - As temporary file.Unchanged.
;***************************************************************************
Delay1ms:
                push    temp        ;
                ldi     temp,10     ;
                rcall   Delay       ;Delay for 908,5us
                ldi     temp,4      ;
                rcall   Delay       ;Delay for 73.5us = 982us
                ldi     temp,2      ;
                rcall   Delay       ;Delay for 16.5us = 998us
                pop     temp        ;
                ret                 ;adding the ldis and push/pops =1ms
;_____________________________________________________________________

;***************************************************************************
;*Subroutine: Delay
;*Description:This subroutine makes a delay for:
;*              t(sec)=[6*2+4+3+3*(x^3+x^2+x)]/3,6864MHz.
;*Finding "x" formula:
;*                          x=????????????.
;*Registers:   temp - As input x data.Unchanged.
;*              DelayL,DelayM,DelayH - As temporary delay
counters.Unchanged.
;***************************************************************************
Delay:                                  ;3 for rcall        |"temp" ~> time
            push    DelayL              ;Store"DelayL";+2   | 6       215us
            push    DelayM              ;Store"DelayM";+2   | 17      4.25ms
            push    DelayH              ;Store"DelayH";+2   | 138     2.15sec
            mov     DelayL,temp         ;1---\              | 172     4.16sec
DelayLoopL: mov     DelayM,temp         ;1    \             | 200     6.54sec
```

```
DelayLoopM: mov   DelayH,temp      ;1    \                | 255   13.55sec
DelayLoopH:                        ;     \
            dec   DelayH           ;1     \ Delay (cycles):
            brne  DelayLoopH       ;1/2    /
            dec   DelayM           ;1     /  3*temp+
            brne  DelayLoopM       ;1/2   /  +3*temp^2+
            dec   DelayL           ;1    /   +3*temp^3+
            brne  DelayLoopL       ;1/2-/    +temp^3(with nop)
            pop   DelayH           ;+2 Restore "DelayH".
            pop   DelayM           ;+2 Restore "DelayM".
            pop   DelayL           ;+2 Restore "DelayL".
            ret                    ;+4 Return.
;_____________________________________________________________________________

;*****************************************************************************
;
;*Subroutine: ASCIIHextoBIN
;*Description: It converts an ASCII-hex number to 4-bit binary in "data".
;*Registers:  data   -      As input/output file.
;*****************************************************************************
;
ASCIIHextoBIN:
                    push  temp
                    mov   temp,data
                    andi  temp,$F0
                    cpi   temp,$30
                    brne  ASCIIHextoBIN1
                    mov   temp,data
                    andi  temp,$0F
                    cpi   temp,10
                    brge  ASCIIHextoBINer
                    mov   data,temp
                    rjmp  ASCIIHextoBINend
ASCIIHextoBIN1:
                    cpi   temp,$40
                    brne  ASCIIHextoBINer
                    mov   temp,data
                    andi  temp,$0F
                    cpi   temp,0
                    breq  ASCIIHextoBINer
                    cpi   temp,7
                    brge  ASCIIHextoBINer
                    mov   data,temp
                    subi  data,-9
ASCIIHextoBINend:
                    andi  data,$0F
                    pop   temp
                    ret                       ;Return.
ASCIIHextoBINer:
                    ldi   data,$80            ;Error code goes here.
                    pop   temp
                    ret                       ;Return.
```

```
;_____________________________________________________________________

;*********************************************************************
;
;*Subroutine: ASCIIDectoBIN
;*Description: It converts an ASCII-dec number to 4-bit binary in "data".
;*Registers:   data   -      As input/output file.
;*********************************************************************
;
ASCIIDectoBIN:
                    push   temp
                    mov    temp,data
                    andi   temp,$F0
                    cpi    temp,$30
                    brne   ASCIIDectoBINer
                    mov    temp,data
                    andi   temp,$0F
                    cpi    temp,10
                    brge   ASCIIDectoBINer
                    mov    data,temp
                    rjmp   ASCIIDectoBINend
ASCIIDectoBINend:
                    andi   data,$0F
                    pop    temp
                    ret                         ;Return.
ASCIIDectoBINer:
                    ldi    data,$80             ;Error code goes here.
                    pop    temp
                    ret                         ;Return.
;_____________________________________________________________________

;*********************************************************************
;
;*Subroutine: BINtoASCIIHex
;*Description: It converts a 4-bit binary number to ASCII-HEX in "data".
;*Registers:   data   -      As temporary file.Unchange.
;*********************************************************************
;
BINtoASCIIHex:
                    ;push  data                 ;Store "data".
                    andi   data,$0F             ;
                    cpi    data,10              ;
                    brlo   BINtoASCIIHex_1      ;
                    subi   data,9               ;
                    ori    data,$40             ;
                    rjmp   BINtoASCIIHex_2      ;
BINtoASCIIHex_1:
                    ori    data,$30             ;
BINtoASCIIHex_2:
                    ;pop   data                 ;Restore "data".
                    ret                         ;Return.
;_____________________________________________________________________

;*********************************************************************
;
```

```
;*Subroutine: BINtoASCIIDec
;*Description: It converts a 4-bit binary number to ASCII-dec in "data".
;*Registers:   data   -      As temporary file.Unchange.
;*************************************************************************
;
;BINtoASCIIDec:
;                    ;push  data                ;Store "data".
;                    andi   data,$0F            ;
;                    cpi    data,10             ;
;                    brlo   BINtoASCIIDec_1     ;
;                    subi   data,9              ;
;                    ori    data,$40            ;
;                    rjmp   BINtoASCIIDec_2     ;
;BINtoASCIIDec_1:
;                    ori    data,$30            ;
;BINtoASCIIDec_2:
;                    ;pop   data                ;Restore "data".
;                    ret                        ;Return.
;_________________________________________________________________________

;*************************************************************************
;
;*Subroutine: putchar
;*Description: It sends out a character from "data" through UART.
;*Registers:   data   -      As input data.Unchange.
;*************************************************************************
;
putchar:
                    ldi          LEDcounter,LEDTTL

                    sbis   USR,UDRE            ;Wait until Tx register
                    rjmp   putchar             ; is ready and then send
                    out    UDR,data            ; data to it.

                    ret                        ;Return.
;_________________________________________________________________________

;*************************************************************************
;
;*Subroutine: UARTRxDec
;*Description: It decodes the Serial Command which just arrived.
;*Registers:   MyStat       -      As output data.
;*************************************************************************
;
UARTRxDec:
              push   zl
              push   zh
              push   yl
              push   yh
              push   data
              push   temp
              push   DelayL

              ;cli
```

```
;;              ldi     zl,low(RxBufAddr)       ;Pointer z to our RxBuffer's
                                                ; tail.
;;              ldi     zh,high(RxBufAddr)
;;UARTRxDec_Report:
;;              ld      data,z+         ;Echoes back everything between
;;              rcall   putchar         ; received 'A' and <CR>, including them.
;;              cpi     data,$0D
;;              brne    UARTRxDec_Report



                ldi     zl,low(RxBufAddr)   ;Pointer z to our RxBuffer's tail.
                ldi     zh,high(RxBufAddr)

                ld      data,z+             ;Every incoming command must start
                cpi     data,'#'            ; with an '#'.
                brne    UARTRxDec_Err_M

                ld      data,z+

;---------------------------------------;Check if it is Command No0
                cpi     data,'C'            ; [Continuous measurements set]
                brne    UARTRxDec_1

                ld      data,z+
                cpi     data,$0D
                brne    UARTRxDec_Err_M
                cbr     MyStat,1<<EnMeasure     ;Disable measurements.
                sbr     MyStat,1<<ContMeasure   ;Set Continuously
                                                ; Measurements Mode.
                cbr     MyStat,1<<NewMeasure    ;Do NOT start new
                                                ; measurement.
;               ldi     DebCntr,$39

                rjmp    UARTRxDec_CR;End

;---------------------------------------;Check if it is Command No1
UARTRxDec_1:
                cpi     data,'S'        ; [Single measurements set]
                brne    UARTRxDec_2

                ld      data,z+
                cpi     data,$0D
                brne    UARTRxDec_Err_M

                cbr     MyStat,1<<EnMeasure     ;Disable measurements.
                cbr     MyStat,1<<ContMeasure   ;Set Single Measurements
                                                ; Mode.
                cbr     MyStat,1<<NewMeasure    ;Do NOT start new
                                                ; measurement.
```

```
;               ldi     DebCntr,$39

                rjmp    UARTRxDec_CR;End

;---------------------------------------;Check if it is Command No2
UARTRxDec_2:
                cpi     data,'H'                ; [cHannel set]
                brne    UARTRxDec_3

                ld      data,z+
                rcall   ASCIIDectoBIN           ;Convert from ASCII-Decimal
                                                ; to binary.
                sbrc    data,7                  ;
UARTRxDec_Err_M:
                rjmp    UARTRxDec_Err           ;

                cpi     data,9;8
                brsh    UARTRxDec_Err_M         ;
                mov     temp,data

                ld      data,z+
                cpi     data,$0D
                brne    UARTRxDec_Err_M

                mov     Channel,temp            ;Set new default Channel.

                rjmp    UARTRxDec_CR;End

;---------------------------------------;Check if it is Command No3
UARTRxDec_3:
                Cpi     data,'T'            ; [wait Time set]
                brne    UARTRxDec_4

                ld      data,z+             ;Decode the hundrents.
                rcall   ASCIIDectoBIN
                sbrc    data,7              ;
                rjmp    UARTRxDec_Err       ;

                clr     yl
                clr     yh;temp

                inc     data                ;
UARTRxDec_3_1b:
                dec     data
                breq    UARTRxDec_3_1a
                adiw    yl,50
                adiw    yl,50;subi      temp,-100
                rjmp    UARTRxDec_3_1b
UARTRxDec_3_1a:
```

```
                ld      data,z+             ;Decode the decades.
                rcall   ASCIIDectoBIN
                sbrc    data,7              ;
                rjmp    UARTRxDec_Err       ;

                inc     data
UARTRxDec_3_2b:
                dec     data
                breq    UARTRxDec_3_2a
                adiw    yl,10;subi      temp,-10
                rjmp    UARTRxDec_3_2b
UARTRxDec_3_2a:

                ld      data,z+             ;Decode the units.
                rcall   ASCIIDectoBIN
                sbrc    data,7              ;
                rjmp    UARTRxDec_Err       ;

                inc     data
UARTRxDec_3_3b:
                dec     data
                breq    UARTRxDec_3_3a
                adiw    yl,1;subi       temp,-1
                rjmp    UARTRxDec_3_3b
UARTRxDec_3_3a:
                ld      data,z+
                cpi     data,$0D
                brne    UARTRxDec_Err

                mov     temp,yl             ;Is received time = 0?
                or      temp,yh;cpi   temp,0
                breq    UARTRxDec_Err       ;
                tst     yh                  ;Is received time > 255?
                brne    UARTRxDec_Err       ;

                mov     Time,yl         ;Finally load the new Time value.

        ;       mov     data,Time
        ;       swap    data                ;Debugging
        ;       rcall   BINtoASCIIHex
        ;       rcall   putchar             ;

        ;       mov     data,Time
        ;       rcall   BINtoASCIIHex
        ;       rcall   putchar             ;
        ;       ldi     data,$0D
        ;       rcall   putchar             ;

                rjmp    UARTRxDec_CR;End
```

```
;---------------------------------------;Check if it is Command No4
UARTRxDec_4:
                cpi     data,'G'        ; [Go to start a new measurement cycle]
                brne    UARTRxDec_5

                ld      data,z+
                cpi     data,$0D
                brne    UARTRxDec_Err

                ;sbr    MyStat,1<<NewMeasure    ;Set "NewMeasure" flag.

                ;ldi    temp,1<<EnMeasure       ;Toggle "EnMeasure" flag.
                ;eor    MyStat,temp
                ;sbrs   MyStat,EnMeasure        ;If Measurements are
                                                ; disabled,
                ;cbr    MyStat,1<<NewMeasure    ; clear "NewMeasure" flag.

                sbr     MyStat,1<<EnMeasure     ;Enable measurements.
                sbr     MyStat,1<<NewMeasure    ;Start new measurements.

                rjmp    UARTRxDec_CR;End

;---------------------------------------;Check if it is Command No5
UARTRxDec_5:
                cpi     data,'P'        ; [stoP measurements]
                brne    UARTRxDec_6

                ld      data,z+
                cpi     data,$0D
                brne    UARTRxDec_Err

                cbr     MyStat,1<<EnMeasure     ;Disable measurements.
                cbr     MyStat,1<<NewMeasure    ;Do NOT start new
measurement.

                rjmp    UARTRxDec_CR;End

;---------------------------------------;Check if it is Command No6
UARTRxDec_6:
                cpi     data,'L'                ; [scaLe set]
                brne    UARTRxDec_Err

                ld      data,z+
                rcall   ASCIIDectoBIN           Convert from ASCII-Decimal
to binary.
                sbrc    data,7                  ;
                rjmp    UARTRxDec_Err           ;

                mov     temp,data               ;Bit0 = RNG
```

```
                                                ;Bit1 = BIP

                ld      data,z+
                cpi     data,$0D
                brne    UARTRxDec_Err

                mov     Scale,temp              ;Set new Scale.

                rjmp    UARTRxDec_CR;End

;---------------------------------------;Command Error
UARTRxDec_Err:

                ldi     zl,low(UARTErrorMes*2)
                ldi     zh,high(UARTErrorMes*2)
                rcall   PutMes          ;Report general UART command error.

        ;Place here code for detailed error reports.
                rjmp    UARTRxDec_End

UARTRxDec_CR:
                ldi     data,$0D        ;Send <CR> if command received ok.
                rcall   putchar

UARTRxDec_End:
                cbr     UARTStat,(1<<RxBufFull)|(1<<RxBufOv)|
                                                        (1<<RxJobPend)
                sbr     UARTStat,(1<<RxBufEmp)
                clr     RxPointer
                ldi     LEDcounter,LEDTTL
                pop     DelayL
                pop     temp
                pop     data
                pop     yh
                pop     yl
                pop     zh
                pop     zl
                ret
;_____________________________________________________________________________

;*****************************************************************************
;*Subroutine: EEWrite
;*Description:This subroutine waits until EEPROM is ready to be accessed
;*            and writes a byte from "data" to EEPROM address "EEa".
;*Registers:  data    -       As input data.
;*            EEa     -       As input EEPROM address byte.Unchanged.
;*****************************************************************************
EEWrite:
                push    DelayL                  ;
                mov     DelayL,data             ;
```

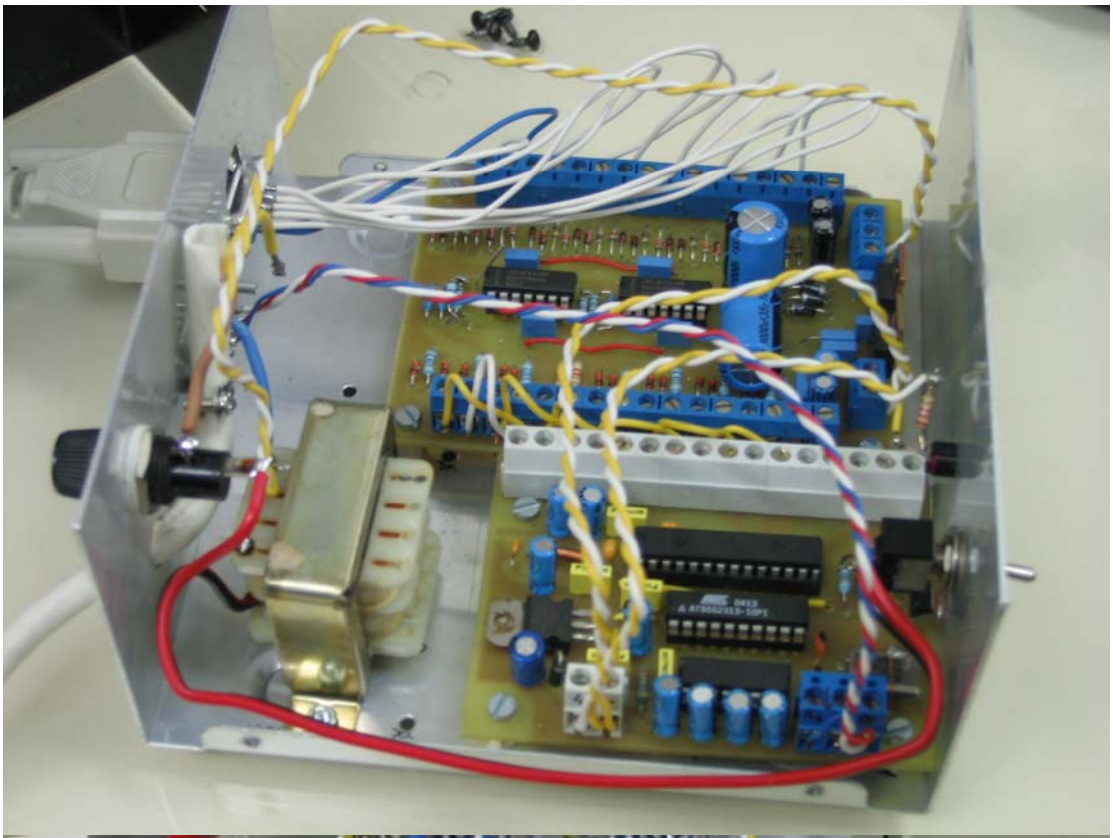
```
                rcall   EERead          ;
                cp      data,DelayL     ;
                breq    EEWriteNoWrite  ;
                mov     data,DelayL     ;
EEWriteWait:
                ;sbic   EECR,EEWE       ;If EEWE not clear
                ;rjmp   EEWriteWait     ;    wait more.
                out     EEAR,EEa        ;Output address.
                out     EEDR,data       ;Output data.
                sbi     EECR,EEMWE      ;Set master write enable.
                sbi     EECR,EEWE       ;Set EEPROM Write strobe.
EEWriteNoWrite:
                pop     DelayL          ;
                ret                     ;Return.
;_____________________________________________________________________

;***************************************************************************
;*Subroutine: EERead
;*Description:This subroutine waits until EEPROM is ready to be accessed
;*           and reads a byte from EEPROM address "EEa" to "data".
;*Registers:  data   -       As output data.
;*           EEa    -       As input EEPROM Low address byte.Unchanged.
;***************************************************************************
EERead:
                sbic    EECR,EEWE       ;If EEWE not clear
                rjmp    EERead          ;    wait more.
                out     EEAR,EEa        ;Output address.
                sbi     EECR,EERE       ;Set EEPROM Read strobe.
                in              data,EEDR ;Get data.
                ret                     ;Return.
;_____________________________________________________________________

;***************************************************************************
;*Subroutine: PutMes
;*Description:This subroutine reads characters from FLASH ROM (start
;*           address "zH:zL") and sends them to UART. The end of
;*           message is defined by the byte $FF.
;*Registers:  data,DelayL - As temporary file.Unchanged.
;*           zH - As input start FLASH Hi address byte.Unchanged.
;*           zL - As input start FLASH Low address byte.Unchanged.
;*           temp - As input start DDRAM address.Unchanged.
;***************************************************************************
PutMes:
                push    temp            ;Store "temp".
                push    zH              ;Store "zH".
                push    zL              ;Store "zL".
                push    data            ;Store "data".
                push    DelayL          ;Store "DelayL".
PutMes_1:
                lpm                             ;
```

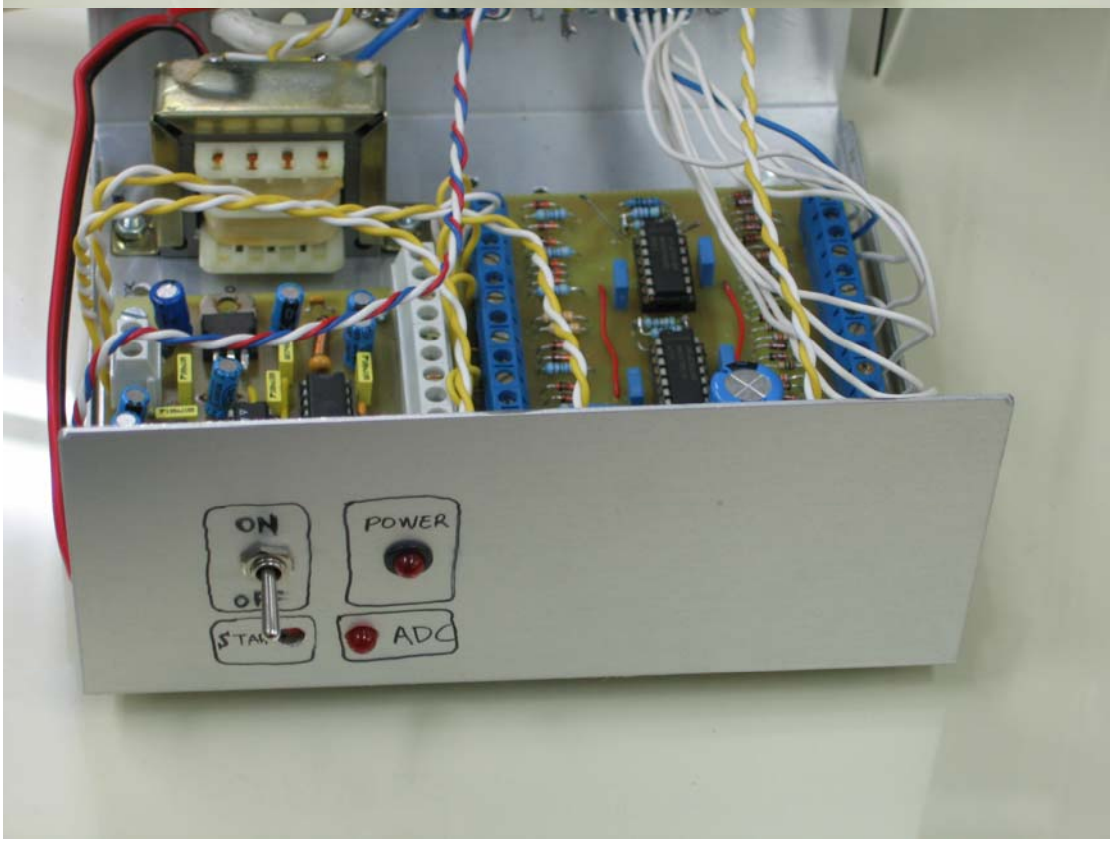
```
                mov     data,DelayL             ;
                cpi     data,$FF                ;
                breq    PutMes_End
                rcall   PutChar                 ;
                adiw    zL,1                    ;
                rjmp    PutMes_1                ;
PutMes_End:
                pop     DelayL                  ;Restore "DelayL".
                pop     data                    ;Restore "data".
                pop     zL                      ;Restore "zL".
                pop     zH                      ;Restore "zH".
                pop     temp                    ;Restore "temp".
                ret                             ;Return.
;_____________________________________________________________________

WelcomeMes:
.db     $0D,$0A,"Serial A/D with MAX197 - Ver1.3e - 2/3/2004 - Kyriakos
Kontakos",$0D,$0A,$0D,$0A,$FF

ADCErrorMes:
.db     $0D,$0A,"ADC FATAL ERROR -> NOT RESPONDING!
",$0D,$0A,$FF

UARTErrorMes:
.db     $0D,$0A,"UART COMMAND ERROR! ",$0D,$0A,$0D,$0A    ;20

CommandList:
.db     "Command List: ",$0D,$0A
.db     " #Hx<CR>   : sets ADC's cHannel. <x> can be from 0 to 8.
8=ALL",$0D,$0A
.db     " #Lx<CR>   : sets ADC's input scaLe. <x> can be from 0 to
3.",$0D,$0A
.db     " #Txxx<CR> : sets sampling Time period in ms. <xxx> can be from
001 to 255. ",$0D,$0A
.db     " #C<CR>    : sets Continuously measure mode.",$0D,$0A
.db     " #S<CR>    : sets Single shot measure mode. ",$0D,$0A
.db     " #G<CR>    : It means Go... It starts a new measurement.",$0D,$0A
.db     " #P<CR>    : It means stoP... It stops measurements.",$0D,$0A
.db     " Caution! <CR> = 0D hex",$0D,$0A,$FF

DefaultSetMes:
.db     "Default Settings are: ",$0D,$0A
.db     " A/D Channel = 0",$0D,$0A
.db     " A/D Input Scale = 0 (0..5V)",$0D,$0A
.db     " Sample Period Time = 10ms",$0D,$0A
.db     " Single Shot Measure Mode",$0D,$0A,$0D,$0A,$FF

.exit
;*****************************************************************************
;*                      End of code.
```

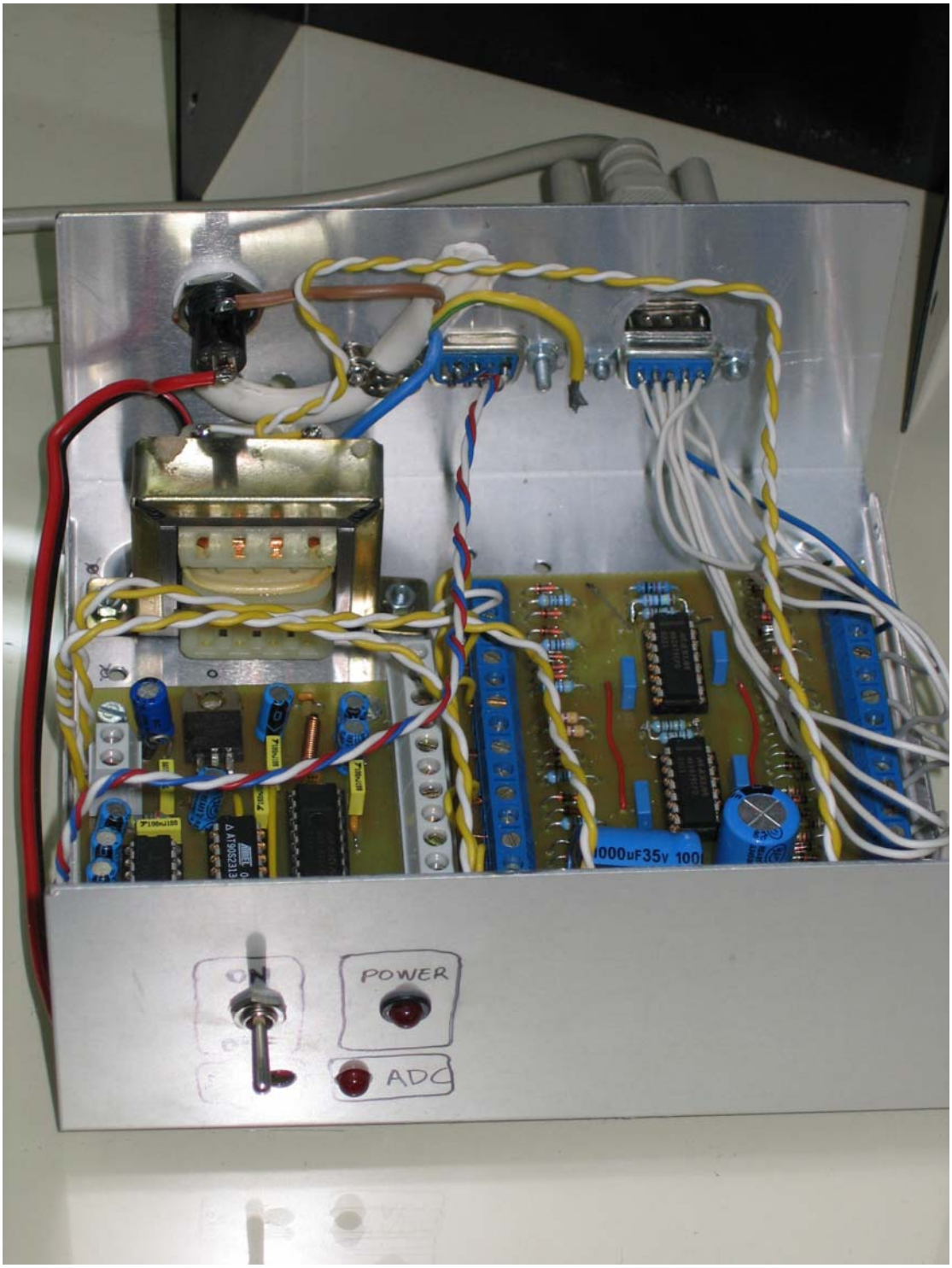
```
;*************************************************************************
```

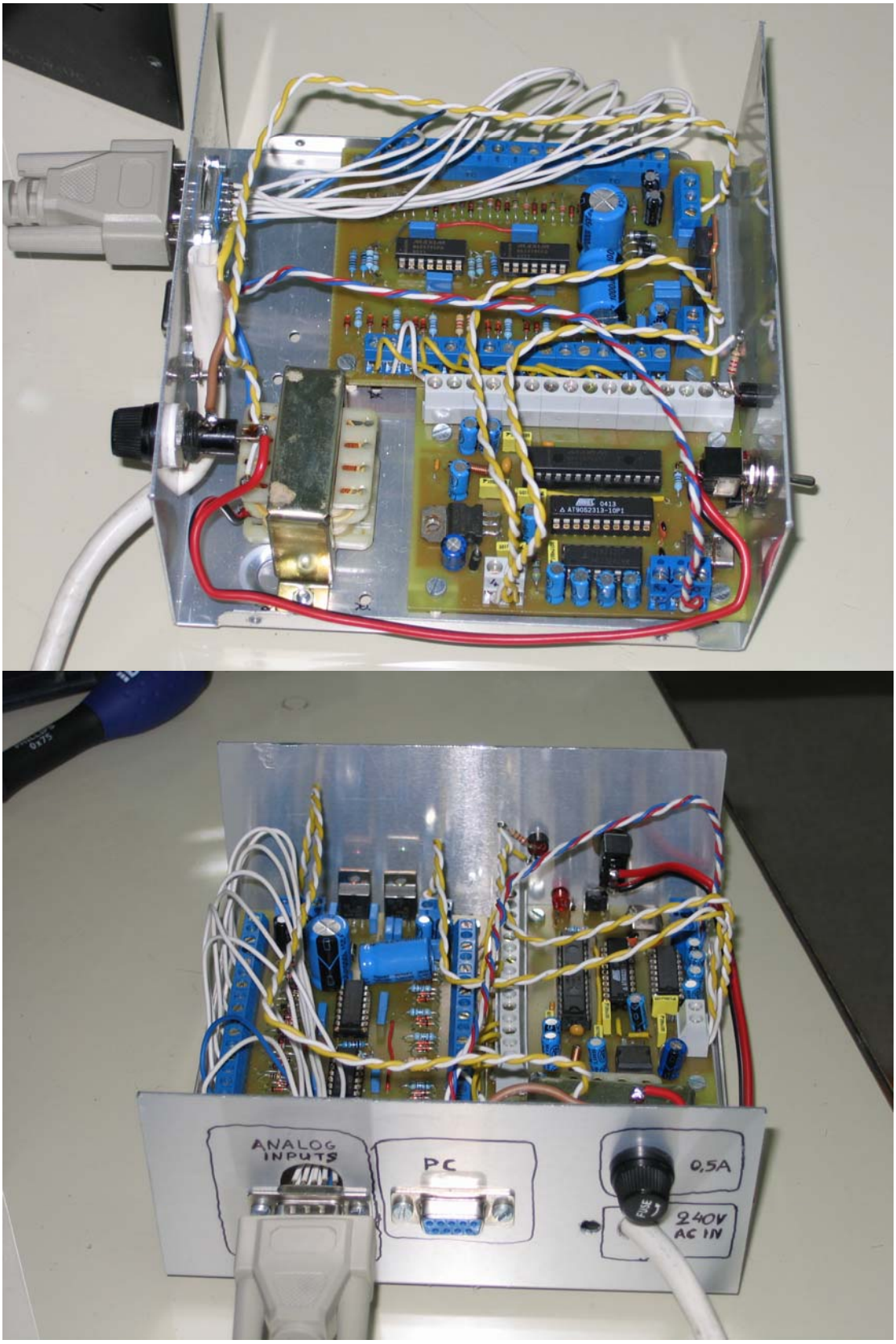
ΠΑΡΑΡΤΗΜΑ Γ

Screen Shots της Συσκευής.






ON
POWER
ADC


ANALOG INPUTS
PC
0,5A
240V AC IN

ΠΑΡΑΡΤΗΜΑ Δ

Βιβλιογραφία.

1) “Mastering Delphi 7” – Marco Cantu, Cybex 2003, ISBN: 0-7821-4201-X.
2) Datasheet AT90S2313, Atmel Corporation, www.atmel.com.
3) Datasheet MAX197, Maxim Intergrated Products, www.maxim-ic.com.
4) Datasheet MAX232, Maxim Intergrated Products, www.maxim-ic.com.
5) Datasheet MAX479, Maxim Intergrated Products, www.maxim-ic.com.