Τμήμα Ηλεκτρονικής

Α.Τ.Ε.Ι. Αθήνας

Ανάπτυξη Μετρητικής Διάταξης της Χλωροφύλλης σε Υγρά με Φασματοφωτομετρική Μέθοδο

Πτυχιακή εργασία

Δημήτριος Σ. Αγιακάτσικας

2008, ΑΘΗΝΑ

Περιεχόμενα

Λίστα Εικόνων

Λίστα Πινάκων

Ακρωνύμια

Ακρωνύμιο	Περιγραφή
ADC	Analogue-to-Digital Converter
ASCII	American Standard Code for Information Interchange
ASIC	Application Specific Integrated Circuit
CMOS	Complementary metal oxide semiconductor
CPU	Central Processing Unit
CSV	Comma Separated Values
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
EEPROM	Electrical Erasable Programmable Read Only Memory
FFT	Fast Fourier Transform
RF	Radio frequency
μC	Microcontroller
ISP	In-System Programming
MIPS	Million Instructions Per Second
PC	Personal Computer
PCB	Printed Circuit Board
RAM	Random Access Memory
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver-Transmitter

Ευχαριστίες

Θέλω να ευχαριστήσω ορισμένους ανθρώπους που έπαιξαν ρόλο για την ύλη και την προετοιμασία αυτή της πτυχιακής εργασίας. Πρώτα θέλω να ευχαριστήσω θερμά τους επιβλέποντες καθηγητές μου, κ Κωνσταντίνο Νομικό και τον κ Γρηγόρη Κουλούρα για το ιδιαίτερο ενδιαφέρον που επέδειξαν καθ'όλη τη πορεία των εργασιών, καθώς και την ουσιαστική βοήθεια που μου παρείχαν μέσω του Ιδρύματος. Ευχαριστίες ανήκουν στην οικογένεια μου, Σαράντο, Πιπίνα, Μυρσίνη και στα υπόλοιπα μέλη της, για την υπομονή, την υποστήριξη και την αγάπη που επέδειξαν κατά τα φοιτητικά μου χρόνια. Τέλος θέλω να ευχαριστήσω τη φίλη μου Λύσια για τη συνεχή ενθάρρυνση στην ολοκλήρωση του έργου μου.

Περίληψη

Αυτή η εργασία έχει σαν σκοπό τον σχεδιασμό και κατασκευή ενός πρότυπου ψηφιακού οργάνου μέτρησης *χλωροφύλλης* στο θαλάσσιο νερό. Οι εγγενείς οπτικές ιδιότητες του θαλάσσιου νερού, όπως είναι η εξασθένηση, η σκέδαση και ο φθορισμός, εξαρτώνται άμεσα από την ποιότητα του. Αυτή η συσκευή θα επιτρέπει στον χρήστη να μετράει την χλωροφύλλη με απευθείας μέτρηση της συνολικής ποσότητας της φθορίζουσας εκπομπής από το δείγμα νερού.

Η συνεχής καταγραφή τους σε περιοχές με πιθανή επιβάρυνση (ιχθυοκαλλιέργειες, λιμάνια, κοίτες ποταμών) είναι απαραίτητη για τον έλεγχο του μεγέθους ρύπανσης και διασφάλισης της καθαρότητάς του. Στην αγορά αν και υπάρχουν συστήματα που μετρούν αυτές τις ιδιότητες, αποτελούν συνήθως συνδυασμό περισσοτέρων οργάνων με απαγορευτικές τιμές για ευρεία χρήση. Στα πλαίσια της πτυχιακής εργασίας θα αναπτυχθεί και θα βαθμονομηθεί ερευνητική διάταξη που θα μετράει τις συγκεκριμένες παραμέτρους με γνώμονα το χαμηλό κόστος κατασκευής. Η πτυχιακή αυτή εργασία έγινε στα πλαίσια της υποστήριξης ερευνητικού προγράμματος του καθηγητή του τμήματος Οπτικής Δρ. Παναγιώτη Δρακόπουλο

1 Εισαγωγή

1.1 Ρύπανση του Νερού

Το νερό είναι η βάση της ζωής στον πλανήτη. Περίπου το 70% της επιφάνειας της Γης σκεπάζεται με νερό. Οι ωκεανοί ρυθμίζουν το κλίμα και είναι ο βιότοπος πολλών ζωντανών οργανισμών, οι οποίοι αποτελούνται σε μεγάλο βαθμό από νερό. Το νερό συνιστά το 60% του συνολικού βάρους ενός δένδρου και το 50-65% του βάρους των ζωικών οργανισμών, συμπεριλαμβανομένου και του ανθρώπου.

Είναι ζωτικός πόρος για τη γεωργία, τη βιοτεχνία, τις μεταφορές και άλλες αμέτρητες ανθρώπινες δραστηριότητες. Από το συνολικό διαθέσιμο νερό στη Γη, το 97% βρίσκεται στους ωκεανούς, ενώ το υπόλοιπο 3% είναι παγιδευμένο σε παγετώνες και παγόβουνα ή βρίσκεται βαθιά στη Γη, ώστε είναι πρακτικά απροσπέλαστο.

Ρύπανση υδάτων ονομάζεται οποιαδήποτε μεταβολή των φυσικών, χημικών και βιολογικών παραμέτρων του νερού (θαλασσών, ποταμών, λιμνών), λόγω της παρουσίας σε αυτό ουσιών σε ποσότητα που υπερβαίνει τα φυσιολογικά όρια. Η μεταβολή αυτή μπορεί να έχει αρνητικές επιπτώσεις στον άνθρωπο, σε άλλους ζωικούς ή φυτικούς οργανισμούς και γενικότερα να διαταράξει την ισορροπία των οικοσυστημάτων σε μικρή ή μεγάλη γεωγραφική κλίμακα. Οι ουσίες αυτές διαλύονται στο νερό, επιπλέουν ή κατακάθονται στον πυθμένα και προέρχονται κυρίως από ανθρωπογενείς δραστηριότητες, όπως το πετρέλαιο και τα λιπάσματα. Επίσης, είναι πιθανή η απελευθέρωση ενέργειας υπό τη μορφή θερμότητας ή ραδιενέργειας, η οποία προκαλεί αύξηση της θερμοκρασίας του νερού, οπότε έχουμε τη «θερμική ρύπανση των υδάτων».

1.2 Μέτρηση Οπτικών Παραμέτρων Νερού

Η ποιότητα του νερού μπορεί να μετρηθεί, από τις οπτικές ιδιότητές του, όπως είναι η εξασθένηση, η σκέδαση και ο φθορισμός του. Η συνεχής καταγραφή των παραμέτρων αυτών σε περιοχές με πιθανή επιβάρυνση (ιχθυοκαλλιέργειες, λιμάνια, κοίτες ποταμών) είναι απαραίτητη για τον έλεγχο του μεγέθους ρύπανσης και διασφάλισης της καθαρότητάς του.

Αυτό μπορεί να επιτευχθεί με χρήση συσκευών μέτρησης βιολογικών, χημικών και γεωλογικών παραμέτρων στη θάλασσα, στις λίμνες και στα ποτάμια. Αυτές οι συσκευές είναι υποβρύχιοι αισθητήρες που μετρούν τις οπτικές ιδιότητες του θαλάσσιου περιβάλλοντος όπως της μέτρησης βίο-φωσφορισμού (fluorometers), οπτικής διάδοσης (transmissometers), οπτικής σκέδασης (scattering) και ανάλυσης οπτικού φάσματος (spectrophotometers).

Αυτή η συσκευή θα επιτρέπει στον χρήστη να μετράει την χλωροφύλλη με απευθείας μέτρηση της συνολικής ποσότητας της φθορίζουσας εκπομπής από το δείγμα νερού.

1.3 Προτεινόμενες σχεδιαστικές λύσεις

Μετά από έρευνα πάνω στην σχεδίαση συστημάτων μέτρησης των οπτικών παραμέτρων του νερού και κυρίως την μέτρηση του φθορισμού της χλωροφύλλης, οι τρόποι υλοποίησης ενός τέτοιου συστήματος είναι οι παρακάτω: (α) Σχεδίαση Αναλογικού Μετρητή και (β) Σχεδίαση Ψηφιακού Μετρητή.

Στην εκδοχή του αναλογικού μετρητή η έξοδος του συστήματος είναι μια αναλογική DC τάση που είναι ανάλογη της τιμής του φθορισμού της χλωροφύλλης στο νερό.

Στην εκδοχή του ψηφιακού μετρητή η έξοδος του συστήματος καταγράφεται στην εσωτερική μνήμη του μικροελεκτή και στο τέλος των μετρήσεων αποστέλλονται οι μετρήσεις στον Η/Υ για περαιτέρω ανάλυση. Μετά από τεχνοοικονομική μελέτη που ακολούθησε, υπερίσχυσε η εκδοχή του ψηφιακού μετρητή.

1.4 Περίγραμμα πτυχιακής εργασίας

Η πτυχιακή εργασία χωρίζεται σε 4 κεφάλαια τα οποία είναι τα παρακάτω. Το πρώτο κεφάλαιο είναι η εισαγωγή της πτυχιακής εργασίας. Το Κεφάλαιο 2 περιγράφει την θεωρία μέτρησης των οπτικών παραμέτρων του νερού και πόσο σημαντικές είναι για την προστασία του θαλάσσιου βιότοπου.

Το κεφάλαιο 3 παρουσιάζει τον σχεδιασμό των λογισμικών υποστήριξης του κυκλώματος καθώς και τα μαθηματικά εργαλεία για την επίτευξη της ορθής ανάπτυξης και λειτουργίας του πηγαίου κώδικα.

Τέλος το κεφάλαιο 4 παρουσιάζει τα συμπεράσματα της εργασίας καθώς επίσης και μελλοντικές βελτιώσεις του ηλεκτρονικού συστήματος μέτρησης που κατασκευάστηκε.

2 Σχεδιασμός μετρητή χλωροφύλλης υγρών

2.1 Εισαγωγή

Οι οπτικές ιδιότητες του θαλάσσιου νερού, όπως είναι η εξασθένηση, η σκέδαση και ο φθορισμός, εξαρτώνται άμεσα από την ποιότητά του. Αυτή η εργασία έχει σαν σκοπό τον σχεδιασμό και κατασκευή ενός πρότυπου ψηφιακού οργάνου μέτρησης χλωροφύλλης στο θαλάσσιο νερό. Αυτή η διάταξη θα επιτρέπει στον χρήστη να μετράει την χλωροφύλλη με απευθείας μέτρηση της συνολικής ποσότητας της φθορίζουσας εκπομπής από το δείγμα νερού.

2.2 Φωτοσυνθετικές χρωστικές

Οι χρωστικές είναι ουσίες που απορροφούν το ορατό φως. Η χλωροφύλλη, η πιο σημαντική χρωστική ουσία που παίρνει μέρος στην φωτοσύνθεση απορροφά φως κυρίως από την κυανή, ιώδη και κόκκινη περιοχή του ορατού φάσματος. Έτσι εξηγείται γιατί δεν απορροφάται καθόλου πράσινο φως, με αποτέλεσμα αυτό να ανακλάται. Τα περισσότερα φυτά είναι πράσινα, επειδή τα φύλλα τους ανακλούν το μεγαλύτερο τμήμα του πράσινου φωτός που δέχονται.

Διαφορετικοί τύποι χρωστικών απορροφούν φως διαφορετικών μηκών κύματος. Σήμερα γνωρίζουμε ότι υπάρχουν κίτρινες, πορτοκαλιές και καφέ φωτοσυνθετικές χρωστικές που είναι γνωστές ως καροτενοειδή και ξανθοφύλλες. Ορισμένα φυτά έχουν επιπλέον χρωστικές τις ανθόκυανίνες, οι οποίες αποθηκεύονται στα χυμοπότια των κυττάρων τους. Όταν φθάνει φθινόπωρο, στα φυτά που ρίχνουν τα φύλλα τους την περίοδο αυτή, παρατηρούμε μια μαζική μετατροπή σακχάρων σε ανθοκυανίνες, οι οποίες παράγουν κόκκινο ή μπλε χρώμα, κάτι που συμβάλει στη δημιουργία των φθινοπωρινών χρωμάτων της φύσης.

Το μόριο της χλωροφύλλης είναι μια οργανική ένωση με μαγνήσιο και έχει μια υδρόφοβη ουρά. Πολλά μόρια χλωροφύλλης στοιβάζονται το ένα δίπλα στο άλλο μέσα στα θυλακοειδή των χλωροπλαστών.

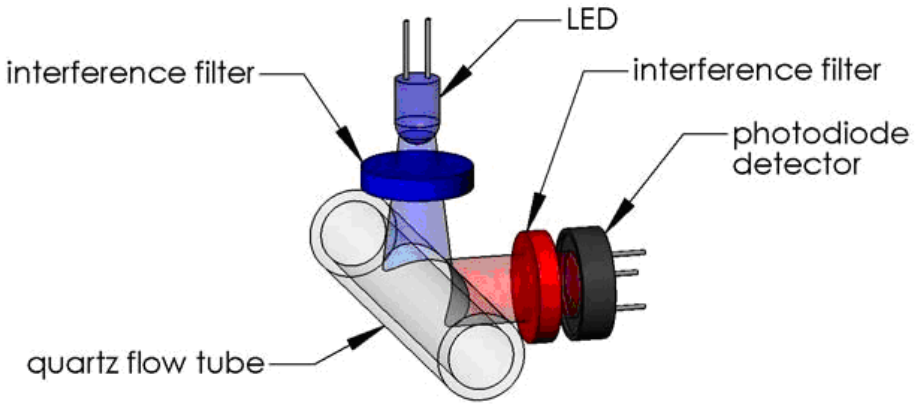
Υπάρχουν διάφοροι τύποι χλωροφυλλών οι οποίες εξελικτικά δείχνουν να συνδέονται στενά. Η πιο σημαντική είναι η χλωροφύλλη **«α»**, η χρωστική που ευθύνεται για τις πρώτες αντιδράσεις της φωτοσύνθεσης. Η χλωροφύλλη **«β»** που δρα ως βοηθητικό μόριο στη φωτοσύνθεση, διαφέρει ελάχιστα χημικά από τη χλωροφύλλη **«α»**. Η χλωροφύλλη **«β»** ευθύνεται για το κιτρινοπράσινο χρώμα, ενώ η χλωροφύλλη **«α»** για το έντονο πράσινο χρώμα. Το αποτέλεσμα είναι ότι κάθε φυτό εκμεταλλεύεται το μεγαλύτερο τμήμα του φάσματος της ορατής ηλιακής ακτινοβολίας.

Υπάρχουν δύο εξειδικευμένες μορφές χλωροφύλλης **«α»**, οι οποίες ονομάζονται P680 και P700. Το P είναι το αρχικό γράμμα της λέξης pigment που σημαίνει χρωστική. Η P680 εμφανίζει ένα μέγιστο απορρόφησης στο κόκκινο χρώμα που αντιστοιχεί σε μήκος κύματος 680nm. Η P700 εμφανίζει μέγιστο απορρόφησης ελαφρά μετατοπισμένο προς μεγαλύτερο μήκος κύματος που αντιστοιχεί σε 700nm

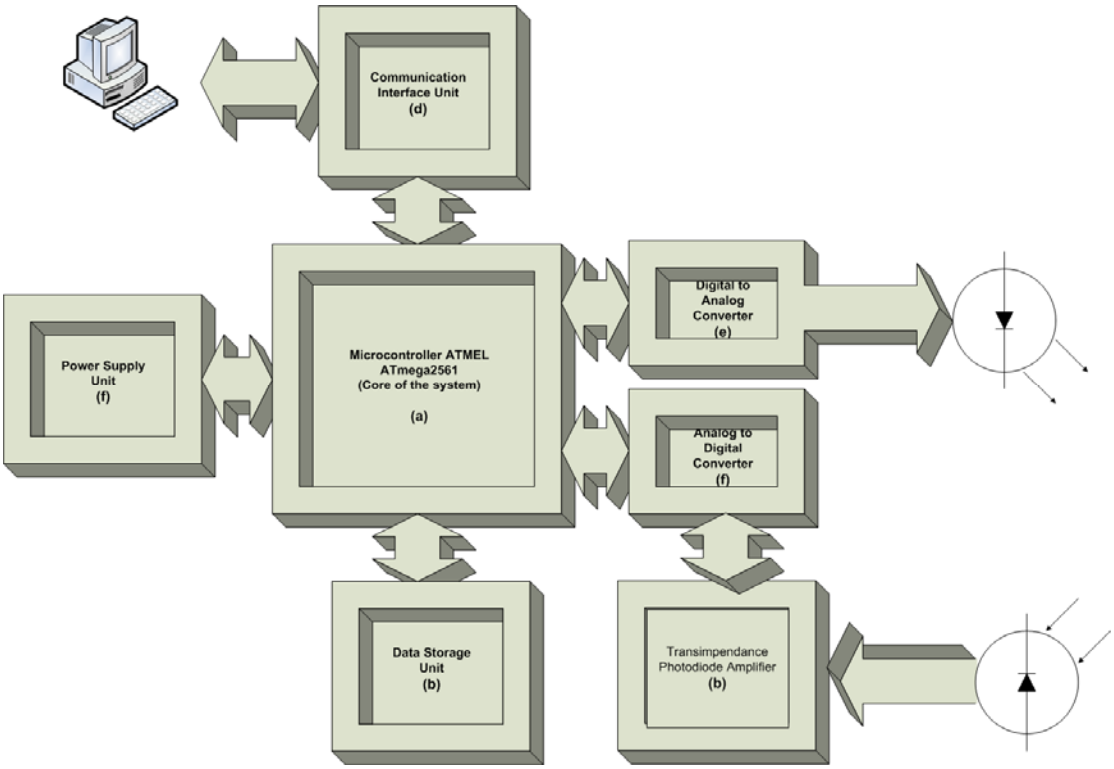
2.3 Θεωρία οργάνου μέτρησης χλωροφύλλης

Αυτό το όργανο σχεδιάστηκε για να μετράει τον αντανακλαστικό φθορισμό της χλωροφύλλης που περιέχει φωτοπλακτόν, το οποίο απορροφάει φως με μήκος κύματος μεταξύ 400 και 520nm και εκπέμπει φως μεταξύ 670 και 730nm. Το ψηφιακό όργανο μέτρησης χλωροφύλλης χρησιμοποιεί ένα υψηλής φωτεινότητας μπλε LED, το οποίο έχει κεντρικό μήκος κύματος 470nm και διαμορφωμένο με πηγή ρεύματος ημιτονικής μορφής 500Hz για να παρέχει διέγερση. Μπλε φίλτρα χρησιμοποιούνται για να εμποδίσουν την μικρή ποσότητα εκπομπής κόκκινου φωτός από το LED. Όπως φαίνεται στην Εικόνα 1 ένας ανιχνευτής (φωτοδίοδος) τοποθετημένος στις 90 μοίρες κεντραρισμένος στην δέσμη φωτός του LED μετράει το εκπεμπόμενο κόκκινο φως από το δείγμα. Ένα κόκκινο φίλτρο χρησιμοποιείται για να εμποδίσει την διασπαρμένη διέγερση του μπλε φωτός. Τα κόκκινα φθορίζων σωματίδια εκπεμπόμενα στις 90 μοίρες ανιχνεύονται συγχρονισμένα στα 500Hz με μία φωτοδίοδο πυριτίου. Η ενισχυμένη τάση εξόδου της φωτοδιόδου οδηγείται στον εσωτερικό 10-bit αναλογικό σε ψηφιακό μετατροπέα του μικροελεκτή για περαιτέρω

ψηφιακή επεξεργασία του μετρούμενου σήματος. Στο πλέον ψηφιακό σήμα γίνεται μετασχηματισμός FFT (Fast Fourier Transform) ώστε να μεταφέρουμε το σήμα από το πεδίο του χρόνου στο πεδίο των συχνοτήτων. Υλοποιώντας ένα ψηφιακό φίλτρο στενής ζώνης διέλευσης αναδεικνύουμε τη συχνότητα των 500Hz αποκλείοντας έτσι το θόρυβο.



Εικόνα 1 - Σχεδιάγραμμα οπτικού μέρους συσκευής

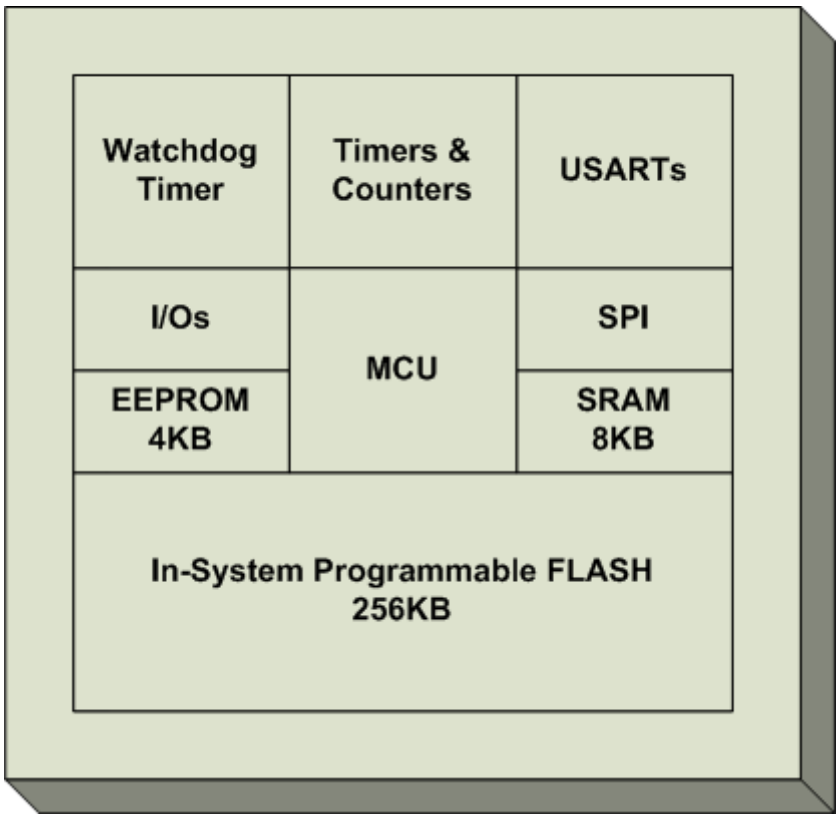


Εικόνα 2 - Μπλοκ διάγραμμα ηλεκτρονικού μέρους συσκευής

Στην Εικόνα 2 βλέπουμε το μπλοκ διάγραμμα του ηλεκτρονικού μέρους της συσκευής

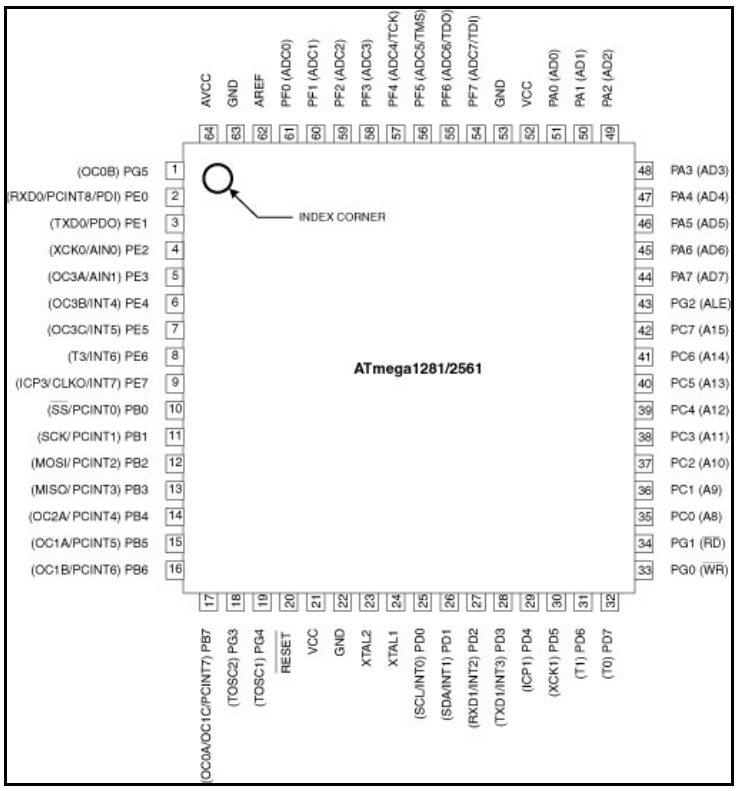
2.4 Ο μικροελεκτής του συστήματος

Σε αυτό το κεφάλαιο περιγράφεται η αρχιτεκτονική του μικροελεκτή του συστήματος. Ο μικροελεκτή του συστήματος είναι ο ATmega2561, της σειράς AVR, της εταιρίας ATMEL. Ο "ATmega2561" είναι ένας χαμηλής κατανάλωσης CMOS 8-bit μικροελεκτής βασισμένος στην αρχιτεκτονική AVR enhanced RISC. Εκτελεί τις περισσότερες εντολές σε ένα μόνο κύκλο ρολογιού. Έτσι καθώς φτάνει σε ταχύτητες 1MIPS/MHz έχει την μέγιστη απόδοση κατανάλωσης και επεξεργαστικής ταχύτητας [ATMEL Corporation, ATMega2561 (2007)]. Το σύστημα λειτουργεί στην "UART friendly" συχνότητα των 14.7456MHz, επιτρέποντας του να έχει εξαιρετική επίδοση.



Εικόνα 3 - Μπλοκ διάγραμμα του μικροελεκτή του συστήματος

Ο ATmega2561 έχει τις ακόλουθες μνήμες: 256KB Flash για τον κώδικα, 4KB EEPROM και 8KB SRAM για τις προσωρινές μεταβλητές.



Εικόνα 4 - Μικροελεκτής ATmega2561

2.5 Αποθήκευση δεδομένων

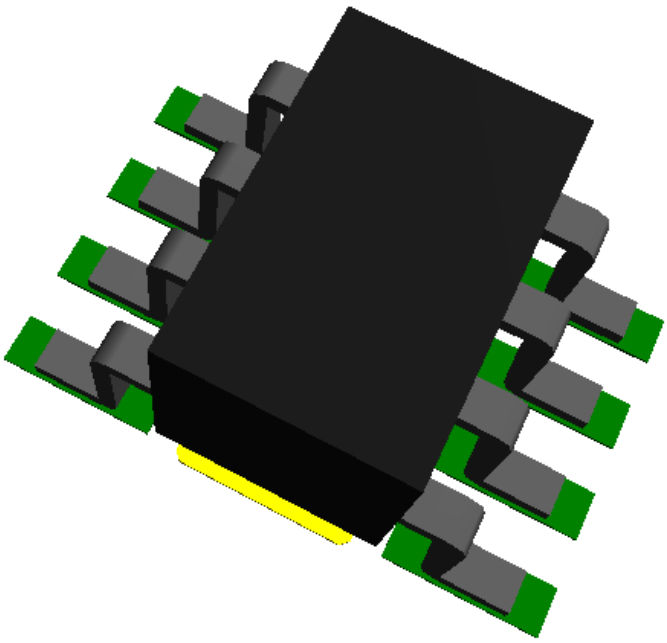
Τα δεδομένα αποθηκεύονται σε μορφή δυαδική μέσα στην μνήμη Flash του μικροελεκτή ATmega2561. Ο μικροελεκτής αυτός έχει 256KB προγραμματιζόμενης μνήμης συστήματος Flash με αντοχή 10.000 κύκλους εγγραφής/διαγραφής. Τα 56KB είναι δεσμευμένα για τον πηγαίο κώδικα του συστήματος. Τα υπόλοιπα 200KB είναι για την αποθήκευση των ψηφιακών μετρήσεων. Σε μελλοντική έκδοση της συσκευής τα ψηφιακά δεδομένα θα αποθηκεύονται σε εξωτερική μνήμη τύπου Flash μεγαλύτερης χωρητικότητας.

Εικόνα 5 - Μνήμη Compact Flash™

Οι Compact Flash™ κάρτες είναι μέσα στις καλύτερες επιλογές για αποθήκευση δεδομένων σε μετρητικά συστήματα υψηλών προδιαγραφών. Οι χωρητικότητες των καρτών αυτών σήμερα (4/2008) είναι της τάξης των GB. Επιπλέον αυτές οι μνήμες δεν χάνουν τα δεδομένα τους κατά την αφαίρεση της τροφοδοσίας τους καθώς είναι τύπου non-volatile.

2.6 Μετατροπέας Αναλογικού σήματος σε Ψηφιακό

Για να μετατραπεί η χρήσιμη πληροφορία του αναλογικού σήματος που ενισχύεται από τον τελεστικό ενισχυτή LMP7721 (Εικόνα 6) σε ψηφιακή, οδηγείται στον εσωτερικό 10-bit μετατροπέα αναλογικού σήματος σε ψηφιακό του ATmega2651. Η τάση αναφοράς που χρειάζεται ο A/D παράγεται από το ίδιο το ολοκληρωμένο ATmega2561 και έχει τιμή 2,56V. Για το φιλτράρισμα και την σταθεροποίηση της τάσης αυτής που εμφανίζεται στον ακροδέκτη 62, υπάρχουν δύο πυκνωτές αποσύζευξης C11(4,7μF/16V) και C10(100nF)



Εικόνα 6 - Τελεστικός Ενισχυτής LMP7721

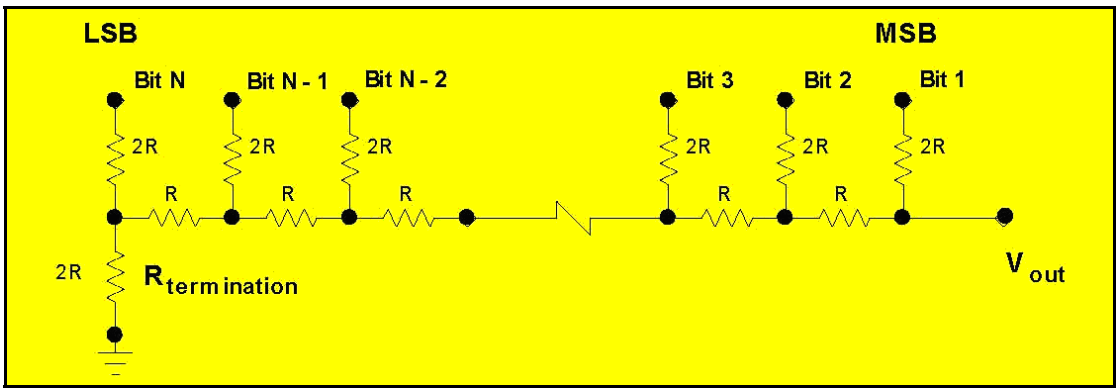
Στους ακροδέκτες του μικροελεκτή (54 ως 61) βρίσκονται τα οκτώ αναλογικά κανάλια εισόδου του A/D εκ τον οποίων χρησιμοποιείται μόνο το πρώτο (ADC0) , ενώ τα υπόλοιπα γειώνονται για μείωση του θορύβου .

Ένα σημαντικό σημείο του κυκλώματος είναι το σημείο όπου συνδέονται 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.

2.7 Μετατροπέας Ψηφιακού σήματος σε Αναλογικό

Στο πραγματικό κόσμο όλα τα σήματα είναι αναλογικά. Τα ψηφιακά συστήματα προκειμένου να διασυνδεθούν με τον πραγματικό κόσμο θα πρέπει να είναι εξοπλισμένα με μεταλλάκτες που πραγματοποιούν μετατροπές των αναλογικών σημάτων σε ψηφιακά (ADC) καθώς επίσης και μεταλλάκτες που κάνουν την αντίστροφη διαδικασία, δηλαδή την μετατροπή των ψηφιακών σημάτων σε αναλογικά (DAC). Το δικτύωμα R2R χρησιμοποιείται συνήθως για μετατροπή ψηφιακού σήματος σε αναλογικό (DAC). Πολλές φορές το συναντάμε στην εσωτερική δομή ολοκληρωμένων ADC διαδοχικών προσεγγίσεων.

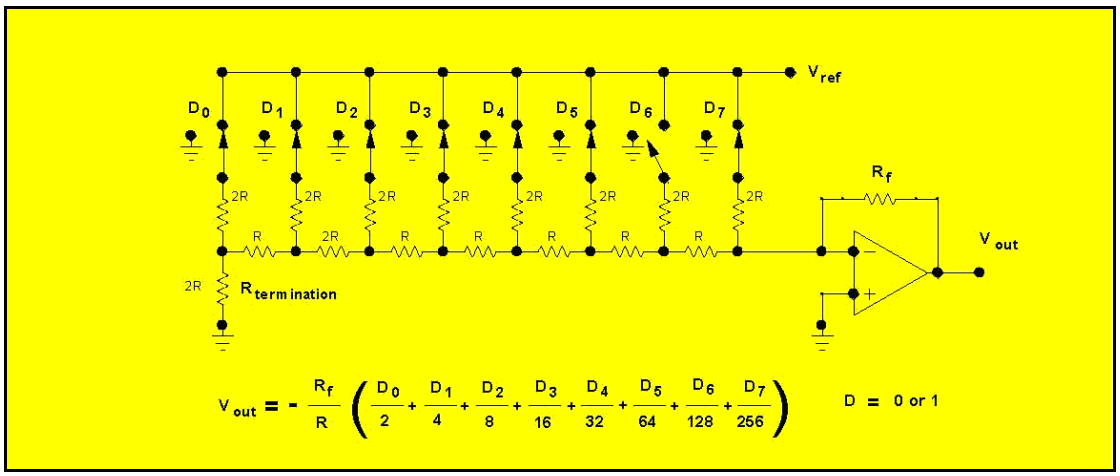
Στην εργασία αυτή θα επισημάνουμε τις σημαντικές παραμέτρους που πρέπει να λάβουμε υπόψη μας όταν σχεδιάζουμε συστήματα DAC με χρήση δικτυώματος αντιστάσεων ladder τύπου R2R.



Εικόνα 7 - Σχηματικό διάγραμμα DAC R2R

Ένα βασικό N bit R2R δικτύωμα αντιστάσεων ladder φαίνεται στην Εικόνα 7. Το εύρος των ψηφιακών εισόδων (bits) ξεκινάνε από το πιο σημαντικό ψηφίο (MSB) στο λιγότερο σημαντικό ψηφίο (LSB). Τα bits εναλλάσσονται μεταξύ 0V και $V_{αναφ}$

(τάση αναφοράς) και ανάλογα με την λογική κατάσταση των ψηφίων (bit) η τάση εξόδου (Vout) θα αλλάζει μεταξύ 0V και $V_{αναφ}$. Το πιο σημαντικό bit (MSB) προκαλεί την μεγαλύτερη αλλαγή στην τάση εξόδου ενώ το λιγότερο σημαντικό bit (LSB) προκαλεί την μικρότερη αλλαγή. Το R2R δικτύωμα αντιστάσεων ladder είναι πολύ φτηνό και κατασκευάζεται σχετικά εύκολα, αφού χρειάζονται μόνο δύο τιμές αντιστάσεων για την υλοποίηση του. Είναι γρήγορο και έχει σταθερή εμπέδηση αντίστασης εξόδου. Το R2R ladder χρησιμοποιείται ως μια διάταξη από διαιρέτες τάσεως των οποίων η ακρίβεια εξαρτάται αποκλειστικά στο πόσο καλά κάθε αντίσταση είναι ταιριασμένη στις άλλες. Υπάρχουν R2R δικτυώματα σε ολοκληρωμένα κυκλώματα των οποίων οι αντιστάσεις είναι τυπωμένες κατευθείαν σε ένα μονό υπόστρωμα, χρησιμοποιώντας ένα μονό φιλμ ώστε να μοιράζονται ίδια ηλεκτρικά χαρακτηριστικά. Επίσης τριμάρονται με laser ώστε να παρέχουν την μέγιστη δυνατή ακρίβεια. Αυτά τα μονολιθικά δικτυώματα έχουν έμφυτη ακρίβεια σε σχέση με τα R2R δικτυώματα κατασκευασμένα από διακριτές αντιστάσεις, λόγω της χαμηλής ανοχής και του χαμηλού συντελεστή μεταβολής της θερμοκρασίας λειτουργίας.



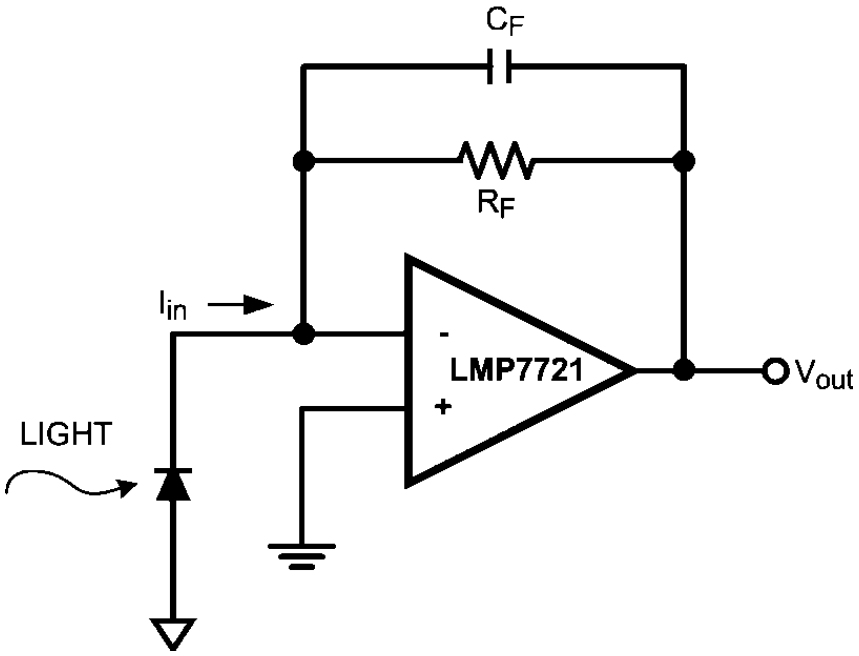
Εικόνα 8 - R2R με χρήση τελεστικού ενισχυτή

Η τερματική αντίσταση είναι πάντα ενωμένη στη γείωση και εξασφαλίζει ότι η thevenin αντίσταση του δικτυώματος (μετρώντας ως προς τη γείωση), με κατεύθυνση το LSB (με όλα τα bits γειωμένα), είναι R. Η Thevenin αντίσταση ενός δικτυώματος R2R ladder είναι R, ασχέτως από τον αριθμό των bits του ladder. Η Εικόνα 8 δείχνει μια συνηθισμένη εφαρμογή ενός δικτυώματος R2R ladder που χρησιμοποιείται ως μετατροπέας ψηφιακού σήματος σε αναλογικό χρησιμοποιώντας την έξοδο ενός

τελεστικού ενισχυτή. Αυτό το είδος του DAC είναι γνωστό ως μετατροπέας πολλαπλασιασμού. Η τάση εξόδου είναι ανάλογη της ψηφιακής εισόδου και το εύρος μπορεί να ρυθμιστεί αλλάζοντας την τάση αναφοράς $V_{αναφ.}$ Ανεξάρτητα από την ψηφιακή είσοδο, η εμπέδηση της πηγής που βλέπει ο τελεστικός ενισχυτής είναι πάντα σταθερή και ίση με R. Η τιμή της R μπορεί να γίνει μικρή, ώστε να φορτίζει γρήγορα η χωρητικότητα εισόδου του τελεστικού ενισχυτή και να επιτευχθεί απαιτούμενο εύρος ζώνης. Βασικός κανόνας είναι ότι πρέπει το εύρος ζώνης του τελεστικού ενισχυτή να είναι αρκετά πιο μικρό από την συχνότητα με την οποία αλλάζουν τα ψηφιακά δεδομένα (π.χ. χρησιμοποιώντας ένα φίλτρο χαμηλών συχνοτήτων) ώστε να περνάνε μόνο οι αναμενόμενες αλλαγές της ψηφιακής εισόδου. Οποιοδήποτε περιττό εύρος ζώνης προσθέτει θόρυβο στην τάση εξόδου. Επίσης ο τελεστικός ενισχυτής πρέπει να έχει πολύ καλή θερμική και ηλεκτρική σταθεροποίηση.

2.8 Υψηλής εμπέδησης ενισχυτής φωτοδιόδου

Ένας ενισχυτής φωτοδιόδου μετατρέπει μία μικρή ποσότητα ρεύματος σε τάση. Η συνάρτηση μεταφοράς του ενισχυτή είναι $V_{out} = -I_{in} \cdot R_F$. Η Εικόνα 9 δείχνει ένα τυπικό ενισχυτή φωτοδιόδου.



Εικόνα 9 - Ενισχυτής Υψηλής Εμπέδισης Φωτοδιόδου

Το ρεύμα δημιουργείται από την φωτοδίοδο πυριτίου. Η συνολική ποσότητα του ρεύματος είναι πολύ μικρή οπότε απαιτείται μεγάλη ενίσχυση από τον ενισχυτή για να μετατρέψει το μικρό ρεύμα σε εύκολα αναγνωρίσιμες τάσεις.

Όσο μεγαλύτερη είναι η ενίσχυση, τόσο μεγαλύτερη τιμή της R_F χρειάζεται. Όταν η R_F είναι μεγάλη, το σφάλμα που προκαλείται από το $I_{bias} \cdot R_F$ αυξάνεται. Για παράδειγμα, εάν η R_F είναι 1000MΩ, και χρησιμοποιείται ένας τελεστικός ενισχυτής με $I_{bias} = \cdot 3nA$ σφάλμα $I_{bias} \cdot R_F$ στην έξοδο θα είναι 3V. Αυτό το σφάλμα μπορεί να μειωθεί δραματικά στα 3μV χρησιμοποιώντας τον τελεστικό ενισχυτή LMP7721.

Οι φωτοδίοδοι είναι υψηλής εμπέδησης αισθητήρες οι οποίοι απαιτούν προσεχτική σχεδίαση του κυκλώματος σήματος για να ικανοποιήσουν τις απαιτήσεις του συστήματος. Συνήθως τελεστικοί ενισχυτές με CMOS εισόδους χρησιμοποιούνται σε εφαρμογές με φωτοδιόδους αφού έχουν υψηλή εμπέδηση εισόδου. Μία CMOS είσοδος με πάρα πολύ χαμηλό ρεύμα I_{bias}, σχεδόν μηδενίζει το ρεύμα θορύβου. Συνεπώς προκύπτει χαμηλή τάση θορύβου. Το LMP7721 είναι ιδανικό για αυτή την λειτουργία καθώς παρέχει υψηλής πιστότητας ενίσχυση καθώς έχει εύρους ζώνης 17MHz. Αυτές οι ιδιότητες κάνουν τον τελεστικό αυτό ιδανικό για ενισχυτή φωτοδιόδου.

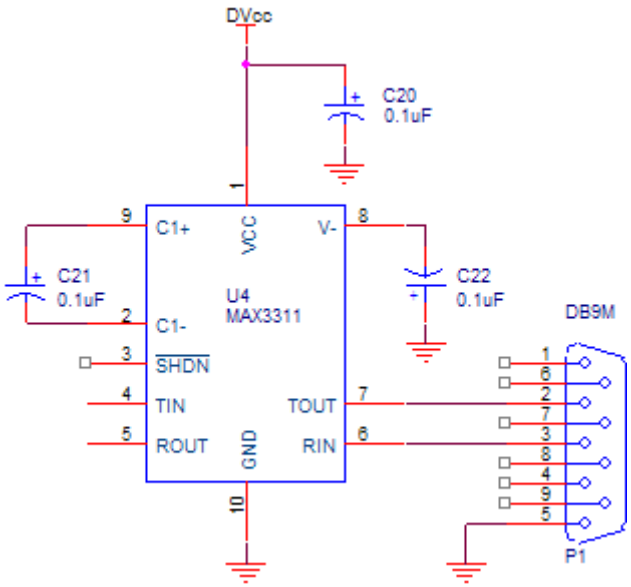
Το LMP7721 είναι ο χαμηλότερος εγγυημένος σε ρεύμα εισόδου I_{bias} ενισχυτής ακριβείας στην βιομηχανία. Το πολύ χαμηλό ρεύμα εισόδου I_{bias} είναι 3fA, με εγγυημένο όριο ±20fA στους 25°C και ±900fA στους 85°C. Αυτό επιτυγχάνεται με την τελευταίας τεχνολογίας πατέντα του κυκλώματος διόρθωσης – ελαχιστοποίησης του ρεύματος εισόδου I_{bias} του ενισχυτή.

Με άλλα λόγια, μια τόσο μικρή τάση θορύβου ($6.5nV/\sqrt{Hz}$), χαμηλή DC τάση ολίσθησης (±150μV μέγιστη στους 25°C) και χαμηλή ολίσθηση τάσης λόγω θερμοκρασίας (1.5μV/°C), αναβαθμίζει την ευαισθησία και την ακρίβεια του συστήματος. Με ένα εύρος τάσεων τροφοδοσίας 1.8V μέχρι 5.5V, το LMP7721 είναι η ιδανική επιλογή για φορητές διατάξεις με μπαταρία.

2.9 Επικοινωνία με τον Ηλεκτρονικό Υπολογιστή

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 μικροελεκτής παράγει και δέχεται σήματα των 0V και +5V οπότε χρειάζεται τον μετατροπέα που φαίνεται στην Εικόνα 10.



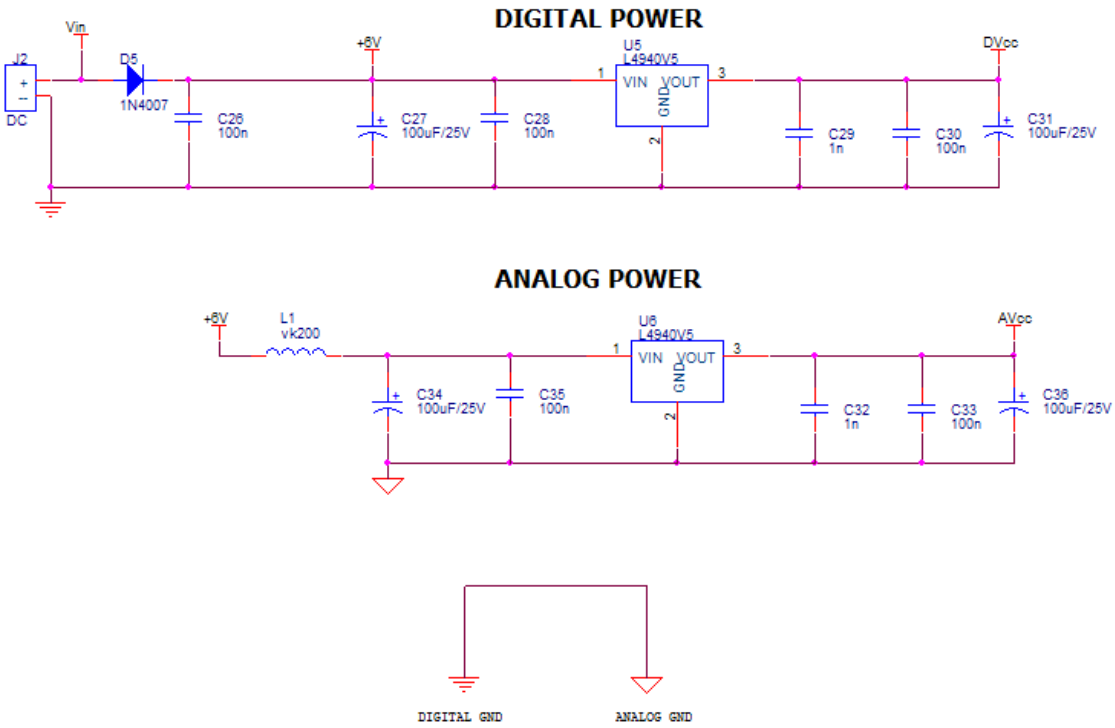
Εικόνα 10 - Σχηματικό διάγραμμα μονάδας επικοινωνίας

Ο μετατροπέας (MAX3311) δέχεται σήματα TTL/CMOS της τάξης των 5V στον ακροδέκτη 4 και τα μετατρέπει σε -12V στον ακροδέκτη 7 ενώ τα 0V τα μετατρέπει σε +12. Αντίστοιχα, δέχεται σήματα της τάξης των -12V στον ακροδέκτη 13 και τα μετατρέπει σε 0V στον ακροδέκτη 12 και τα +12V 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 C21 και C22 (100nF).

Όπως και στα υπόλοιπα ολοκληρωμένα του κυκλώματος, έτσι και εδώ υπάρχει πυκνωτής απόζευξης τροφοδοσίας, ο C20 (100nF) που είναι τοποθετημένος όσο γίνεται πιο κοντά στους ακροδέκτες τροφοδοσίας του MAX3311. Ο πυκνωτής αυτός έχει σκοπό την μείωση του θορύβου που παράγει το ίδιο το ολοκληρωμένο στην γραμμή των +5V λόγω των εσωτερικών κυκλωμάτων ανύψωσης τάσης μέσω διακοπτικών συστημάτων.

2.10 Τροφοδοτικό συσκευής

Τα κυκλώματα τροφοδοσίας έχουν σαν κύρια στοιχεία τους σταθεροποιητές τάσης LS4940V5. Η τροφοδοσία εισόδου του τροφοδοτικού είναι από (6V – 12V).



Εικόνα 11 - Σχηματικό τροφοδοτικού

Για προστασία από τυχόν εσφαλμένη ανάστροφη συνδεσμολογία, έχει προστεθεί η δίοδος D5 (1N4007) που είναι μια δίοδος γενικής χρήσης με ονομαστικό επιτρεπόμενο ρεύμα διέλευσης 1A, υπεραρκετό για την ορθή λειτουργία του κυκλώματος. Ο πυκνωτής C27 (100μF/25V) φροντίζει ώστε να υπάρχει πάντα διαθέσιμο φορτίο, ώστε να καλύπτει τις ανάγκες ρεύματος του κυκλώματος κατά τις απότομες αλλαγές κατάστασης, δεδομένου ότι τροφοδοτεί ψηφιακό κύκλωμα. Οι πυκνωτές C28 (100nF), C29 (1nF) και C30 (100nF) βρίσκονται κοντά στους ακροδέκτες του ολοκληρωμένου σταθεροποίησης U1(LS4940V5) 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 C31 (100μF/50V) υπάρχει στο κύκλωμα τροφοδοσίας ώστε να ελαχιστοποιούνται οι βυθίσεις της τάσης. Η τροφοδοσία του A/D παρέχεται από το δεύτερο κύκλωμα τροφοδοσίας AVcc. Το U6 (LS4940V5) τροφοδοτείται με τάση εισόδου από 6V έως 12V μέσω του πηνίου L1 (10μH) το οποίο μαζί με τους πυκνωτές C35 (100nF) και C34 (100μF/25V), αποτελούν ένα φίλτρο χαμηλών συχνοτήτων. Το φίλτρο αυτό έχει σκοπό την απομάκρυνση τυχόν παρασιτικών συχνοτήτων και άλλων θορύβων που έχουν επικαθίσει πάνω στην γραμμή τροφοδοσίας των +5V.

2.11 Συμπεράσματα

Στην αγορά αν και υπάρχουν συστήματα που μετρούν τις ιδιότητες του θαλάσσιου νερού, αποτελούν συνήθως συνδυασμό περισσοτέρων οργάνων με απαγορευτικές τιμές για ευρεία χρήση. Στα πλαίσια της πτυχιακής εργασίας αναπτύχθηκε τεχνοοικονομική μελέτη ώστε να σχεδιαστεί η ερευνητική διάταξη που θα μετράει τις συγκεκριμένες παραμέτρους με γνώμονα το χαμηλό κόστος κατασκευής χωρίς να μειωθεί η υψηλή ποιότητα και πιστότητα των μετρήσεων του οργάνου.

3 Σχεδιασμός λογισμικού

3.1 Εισαγωγή

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. Ο κώδικας που αφορά τον μικροελεκτή έχει γραφτεί στο Codevision σε γλώσσα προγραμματισμού C, ενώ ο κώδικας για τον Η/Υ έχει γραφτεί σε Delphi για Windows XP.

3.2 Μεθοδολογία

Για την υλοποίηση του λογισμικού, χρησιμοποιήθηκε η τεχνολογία ψηφιακής επεξεργασίας σήματος (DSP – Digital Signal Processing). Η ψηφιακή επεξεργασία σήματος είναι η μαθηματική διαδικασία, οι αλγόριθμοι και οι "τεχνικές" που χρησιμοποιούνται για την επεξεργασία των σημάτων, με σκοπό την βελτίωση και την "αριστοποίησή" τους. Βέβαια, ο όρος "βελτίωση" είναι αρκετά ασαφής και αποκτά διαφορετικό νόημα ανάλογα με την περίπτωση. Συνήθως όμως, αφορά την ανάδειξη του σήματος μέσα από το θόρυβο, την απόρριψη της πλεονάζουσας πληροφορίας και τη μορφοποίηση του, ώστε να αποκτήσει περαιτέρω "δυνατότητες". Το σήμα εξόδου ενός μετατροπέα ή αισθητήρα "μολύνεται" συνήθως από ηλεκτρικό θόρυβο. Η επεξεργασία με ένα φίλτρο μπορεί να τον αφαιρέσει ή τουλάχιστο να τον ελαττώσει. Παλαιότερα, η διαδικασία αυτή αποτελούσε πεδίο εφαρμογής αναλογικών φίλτρων, τα οποία είχαν υψηλό κόστος και χαμηλή απόδοση. Σήμερα είναι αποκλειστικό προνόμιο της ψηφιακής επεξεργασίας, η οποία στο συγκεκριμένο τομέα επιτελεί θαύματα. Η Ψηφιακή Επεξεργασία Σήματος γίνεται από μικροεπεξεργαστές, που η αρχιτεκτονική τους, το ρεπερτόριο των εντολών τους και το λογισμικό που τους ελέγχει, αποσκοπούν στην ταχύτατη διεξαγωγή ορισμένων μαθηματικών πράξεων. Ουσιαστικά, το DSP είναι ταχύτατη πολλαπλασιασμοί, αθροίσεις και μετακινήσεις μεγάλων αριθμών. Αν και η μαθηματική θεωρία που βρίσκεται πίσω από τις εφαρμογές DSP, όπως ο γρήγορος μετασχηματισμός Fourier (FFT), ο

μετασχηματισμός Hilbert, η θεωρία των φίλτρων και οι αλγόριθμοι συμπίεσης, είναι αρκετά πολύπλοκοι, οι αριθμητικές πράξης που απαιτούνται για να υλοποιηθούν είναι απλές και μπορούν να γίνουν και από μία αριθμομηχανή τεσσάρων πράξεων. Ο Μικροϋπολογιστής έχει την ικανότητα να επεξεργάζεται ένα σήμα "ζωντανά", αμέσως μετά τη δειγματοληψία και να διαμορφώνει το αποτέλεσμα πριν έρθει το νέο δείγμα. Η επεξεργασία κάθε δείγματος πρέπει να τελειώνει πριν εμφανιστεί το επόμενο. Όλες οι εφαρμογές DSP, από τους ελεγκτές των σκληρών δίσκων μέχρι τα κινητά τηλέφωνα, απαιτούν λειτουργία πραγματικού χρόνου.

3.2.1 Μαθηματικά Εργαλεία

Με την βοήθεια της τεχνολογίας DSP μπορεί να υλοποιηθεί ο μετασχηματισμός του φάσματος των σημάτων, ούτως ώστε να βελτιωθούν ή να αποκτήσουν διαφορετικές ιδιότητες. Για να γίνει όμως αυτό πρέπει να χρησιμοποιηθεί το κατάλληλο "μαθηματικό εργαλείο". Για τα περιοδικά σήματα, το εργαλείο αυτό είναι η Ανάλυση κατά Φουριέ (Fourier) ενώ για τα μη περιοδικά (ασυνεχή) ο Μετασχηματισμός Φουριέ. Είναι γνωστή σε θεωρητικό επίπεδο από το 1822, αλλά εφαρμόστηκαν στη θεωρία φασμάτων τη δεκαετία του 1950. Είχαν όμως μεγάλες υπολογιστικές απαιτήσεις. Το 1965 δημοσιεύτηκε ένας νέος μαθηματικός αλγόριθμος, που επιτάχυνε θεαματικά την εξέλιξη του DSP. Έγινε γνωστός σαν Γρήγορος Μετασχηματισμός Φουριέ (Fast Fourier Transform, FFT) και μείωσε κατά 100 φορές τον αριθμό των πολλαπλασιασμών που χρειαζόταν για να υπολογιστεί το φάσμα ενός ασυνεχούς σήματος. Η ανακάλυψη αυτού του μαθηματικού εργαλείου απετέλεσε ορόσημο στην εξελικτική πορεία του DSP και παραμένει ακόμη και σήμερα ακρογωνιαίος λίθος στο λαμπρό οικοδόμημα την επεξεργασίας των ψηφιακών σημάτων.

3.2.2 Ψηφιακά Φίλτρα

Η Τα ψηφιακά φίλτρα είναι υπολογιστικές μονάδες DSP, που εκτελούν αριθμητικές πράξεις σε ψηφιακά δείγματα του αναλογικού σήματος. Έχουν το ίδιο πεδίο εφαρμογών με τα αναλογικά. Το μόνο που αλλάζει είναι ο τρόπος με τον οποίο

επιτυγχάνονται οι επιδόσεις και η υψηλότερη ποιότητα στην απόδοση τους. Θεμελιώδες πλεονέκτημα της ψηφιακής τεχνολογίας είναι η ομοιομορφία στην απόδοση. Κάθε αντίγραφο της ηλεκτρονικής συσκευής έχει ακριβώς τα ίδια χαρακτηριστικά με το πρωτότυπο, κάτι που δεν είναι ούτε προφανές, ούτε εύκολο κατασκευαστικά στις αναλογικές σχεδιάσεις. Πέρα όμως από το εγγενές χαρακτηριστικό των ψηφιακών φίλτρων, υπάρχουν και αρκετά άλλα σημαντικά πλεονεκτήματα:

- Ένα ψηφιακό φίλτρο είναι σε θέση να προγραμματιστεί. Αυτό σημαίνει ότι το φίλτρο μπορεί να προσαρμοστεί εύκολα στις νέες παραμέτρους ή ανάγκες της εφαρμογής, χωρίς κόστος. Αντίθετα, ένα αναλογικό φίλτρο μπορεί να αλλάξει συμπεριφορά μόνο με πλήρη επανασχεδίαση και αντικατάσταση.
- Τα Ψηφιακά φίλτρα είναι εύκολα στη σχεδίαση, στη δοκιμή και την υλοποίηση.
- Τα χαρακτηριστικά ενός αναλογικού φίλτρου εξαρτώνται από τη την τάση λειτουργίας, τη θερμοκρασία και την ηλικία των ηλεκτρονικών εξαρτημάτων (τιμές πυκνωτών – αντιστάσεων – πηνίων). Τα ψηφιακά φίλτρα έχουν υψηλή θερμική και λειτουργική σταθερότητα.
- Αντίθετα από τα αναλογικά, τα ψηφιακά φίλτρα μπορούν να χειριστούν χαμηλής συχνότητας σήματα με μεγάλη ακρίβεια. Καθώς η ταχύτητα της τεχνολογίας DSP συνεχίζει να αυξάνεται, χρησιμοποιούνται ήδη και σε εφαρμογές ραδιοσυχνοτήτων (RF), περιοχή που ήταν παλαιότερα αποκλειστικότητα των αναλογικών φίλτρων.
- Τα ψηφιακά φίλτρα είναι προσαρμοσμένα (adaptive) στις ανάγκες της εφαρμογής, ακόμη και κατά τη διάρκεια λειτουργίας. Αυτή η συνεχής αλλαγή και προσαρμογή της συμπεριφοράς τους «εν θερμώ», τα έχει καταστήσει απαραίτητα στις σύγχρονες εφαρμογές κωδικοποίησης.

Σημαντικό μειονέκτημα των ψηφιακών φίλτρων είναι η χαμηλή ταχύτητα λειτουργίας, η οποία, αν και συνεχώς βελτιώνεται, παρόλα αυτά δεν μπορεί να συγκριθεί με την άμεση απόκριση των αναλογικών φίλτρων.

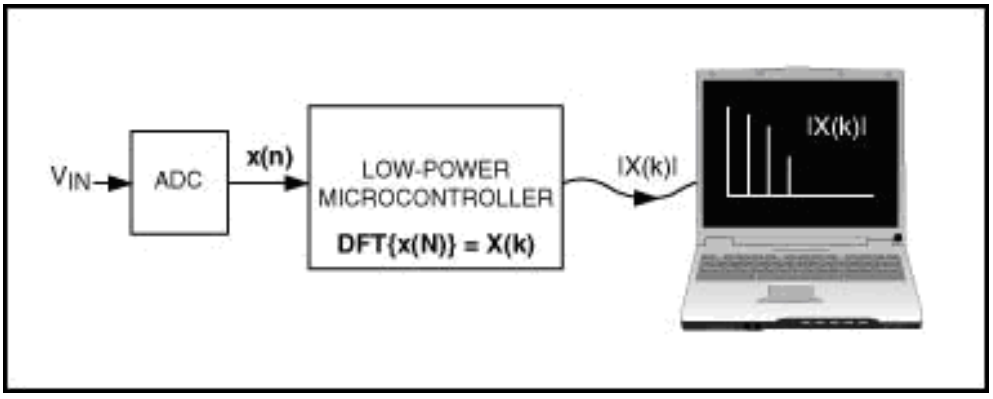
3.3 Λογισμικό για τον μικροελεκτή

Μετά την κατανόηση των κυκλωμάτων του οργάνου υλοποιήθηκαν λογικά διαγράμματα για να αναπτυχθεί το λογισμικό για τον έλεγχο της διάταξης του μC.

3.3.1 Ανάπτυξη FFT εφαρμογής με μικροελεκτή

Καθώς οι χαμηλής κατανάλωσης μικροελεκτές (μCs) ξεκίνησαν να περιλαμβάνουν περιφερειακά που ήταν παλιότερα προνόμιο μεγαλύτερων επεξεργαστών ASICs και DSPs, καινούριες ευκαιρίες για επεξεργασία σύνθετων αλγορίθμων σε μικρότερης τάξης μικροελεκτές γίνονται εφικτές. Το παρακάτω κείμενο περιγράφει την εφαρμογή γρήγορης μετατροπής Fourier (FFT) που έχει υλοποιηθεί στην διάταξη μέτρησης χλωροφύλλης.

Αυτή η FFT εφαρμογή επεξεργάζεται, σε πραγματικό χρόνο, το φάσμα μιας τάσης εισόδου (V_{IN}) στην Εικόνα 12. Για να επιτευχθεί αυτό, ο αναλογικός σε ψηφιακός μετατροπέας (ADC) του μC δειγματοληπτεί την τάση εισόδου V_{IN} και εφαρμόζει ένα 256 σημείων FFT στα δείγματα, ώστε να αποκτήσει το φάσμα της τάσης εισόδου. Για εργαστηριακούς λόγους, ο μC υπολογίζει το φάσμα και μεταφέρει τα αποτελέσματα στο Η/Υ όπου εμφανίζονται σε πραγματικό χρόνο.



Εικόνα 12 - FFT με μC

- **ΥΠΟΒΑΘΡΟ**

Για να προσδιορίσουμε το φάσμα του δειγματοληπτημένου σήματος εισόδου, πρέπει να επεξεργαστούμε τη διακριτή μετατροπή Fourier - Discrete Fourier Transform (DFT) των δειγμάτων εισόδου. Το DTF ορίζεται ως :

$$X(k) = \sum_{N=0}^{N-1} X(n) \cdot e^{\frac{-j \cdot 2 \cdot \pi \cdot k \cdot n}{N}} \quad for \quad 0 \le k \le N-1$$

όπου N είναι ο αριθμός των δειγμάτων, X(k) είναι το φάσμα, και x(n) είναι το σύνολο των δειγμάτων εισόδου. Αναλύοντας με τη ταυτότητα του Euler, και χωρίζοντας τα δείγματα εισόδου και το φάσμα στο πραγματικό και φανταστικό μέρος, ακολουθούν οι παρακάτω εξισώσεις:

Το δεύτερο μέρος των εξισώσεων 3.1 και 3.2 εξαφανίζονται γιατί τα δείγματα εισόδου αποτελούνται από πραγματικούς αριθμούς. Ανακεφαλαιώνοντας έχουμε N δείγματα, η επεξεργασία των εξισώσεων 3.1 και 3.2 απαιτεί $2 \cdot N^2$ πολλαπλασιασμούς και $2 \cdot N \cdot (N-1)$ προσθέσεις. Οπότε, το DTF με 256 δείγματα εισόδου θα απαιτούσε 131.072 πολλαπλασιασμούς και 130.560 προσθέσεις. Για αυτό το λόγω επιλέγουμε το FFT.

$$X_{\mathrm{Re}}(k) = \sum_{N=0}^{N-1}\left[X_{\mathrm{Re}}(n) \cdot \cos\left(\frac{2 \cdot \pi \cdot \kappa \cdot n}{N}\right) + X_{\mathrm{Im}}(n) \sin\left(\frac{2 \cdot \pi \cdot \kappa \cdot n}{N}\right)\right] = \sum_{N=0}^{N-1}\left[X_{\mathrm{Re}}(n) \cdot \cos\left(\frac{2 \cdot \pi \cdot \kappa \cdot n}{N}\right)\right]$$

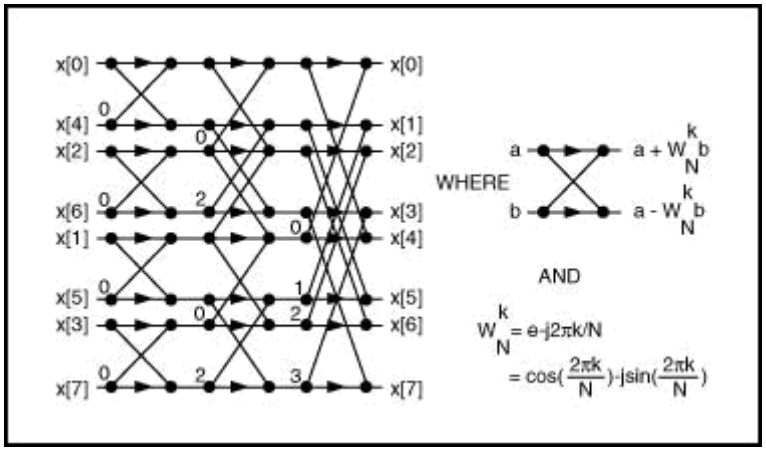
Εξίσωση 3.1

$$X_{\mathrm{Im}e}(k) = \sum_{N=0}^{N-1}\left[X_{\mathrm{Re}}(n) \cdot \sin\left(\frac{2 \cdot \pi \cdot \kappa \cdot n}{N}\right) - X_{\mathrm{Im}}(n) \cos\left(\frac{2 \cdot \pi \cdot \kappa \cdot n}{N}\right)\right] = \sum_{N=0}^{N-1}\left[X_{\mathrm{Re}}(n) \cdot \sin\left(\frac{2 \cdot \pi \cdot \kappa \cdot n}{N}\right)\right]$$

Εξίσωση 3.2

Υπάρχουν πολλοί αλγόριθμοι FFT. Σε αυτή την εφαρμογή χρησιμοποιείται ο συνηθισμένος radix-2 αλγόριθμος που συνεχώς χωρίζει το DFT σε μικρότερα DFTs. Για τα είναι εφικτό αυτό, τα N πρέπει να είναι δύναμη του 2. Τα βήματα του radix-2 FFT αλγόριθμου φαίνονται εικονογραφημένα με τους υπολογισμούς πεταλούδα στην Εικόνα 13.

Παρατηρώντας αυτές τις επεξεργασίες, βλέπουμε ότι ο radix-2 αλγόριθμος απαιτεί μόνο $\frac{N}{2}\log_2(N)$ πολλαπλασιασμούς και $N\log_2(N)$ προσθέσεις. Οι τιμές του W_N που χρησιμοποιούνται στην Εικόνα 13 είναι γνωστές ως "στρέψη παράγοντες" και μπορούν να υπολογιστούν προτού εφαρμοστεί ο αλγόριθμος.



Εικόνα 13 - Radix-2 FFT

Στην Εικόνα 13, φαίνεται η διαδικασία της ανασύνταξης της θέσης των δειγμάτων εισόδου με τεχνική αντιστροφής των bits του δείκτη της θέσης των δειγμάτων. Συνεπώς, υπολογίζοντας τον Radix-2 FFT αλγόριθμο με N = 8 απαιτεί την αλλαγή σειράς των δεδομένων εισόδου από:

0 (000b), 1 (001b), 2 (010b), 3 (011b), 4 (100b), 5 (101b), 6 (110b), 7 (111b)

σε:

0 (000b), 4 (100b), 2 (010b), 6 (110b), 1 (001b), 5 (101b), 3 (011b), 7 (111b)

Η FFT έξοδος εμφανίζεται στη σωστή σειρά. Στην Εικόνα 13 παρατηρούμε ότι τα αποτελέσματα των υπολογισμών πεταλούδα είναι τα μόνα δεδομένα που απαιτούνται για το επόμενο στάδιο του FFT. Λόγω ότι οι υπολογισμοί γίνονται «στον ίδιο πίνακα», οι καινούριες τιμές αντικαθιστούν τις παλιές, οπότε μόνο 2N μεταβλητές απαιτούνται για τον υπολογισμό ενός FFT με N δείγματα (2N μεταβλητές απαιτούνται διότι κάθε τιμή έχει πραγματικό και φανταστικό μέρος).

Όταν το FFT ολοκληρωθεί, τα αποτελέσματα είναι σε μιγαδική μορφή. Οι εξισώσεις 4 και 5 μετατρέπουν τα αποτελέσματα σε πολική μορφή:

Η DSP βιβλιογραφία περιέχει πολλές βελτιώσεις για τον αλγόριθμο DFT/FFT που περιγραφικέ παραπάνω ώστε να τον κάνει γρηγορότερο και μικρότερο. Μια από τις πιο σημαντικές βελτιώσεις (και ίσως ο πιο εύκολος τρόπος να εφαρμοστεί)

πηγάζει από τον ρεαλισμό ότι η ισχύς του DFT ενός πραγματικού σήματος είναι συμμετρικό γύρω από το X(N/2), συνεπώς :

$$X_{MAGN}(k) = \sqrt{X^2{}_{RE}(k) + X^2{}_{IM}(k)}$$

Εξίσωση 3.3

$$X_{PHASE}(k) = \arctan\left(\frac{X_{\mathrm{Im}}(k)}{X_{\mathrm{Re}}(k)}\right)$$

Εξίσωση 3.4

Να γράψεις ένα FFT δεν είναι τόσο απλό και μερικοί περιορισμοί της μικρής υπολογιστικής ισχύος μπορεί να κάνουν πιο πολύπλοκο το γράψιμο του FFT αλγόριθμου για αυτές τις διατάξεις:

Μνήμη: Ο επιλεγμένος μC έχει 256kB RAM. Ξέροντας ότι ο αλγόριθμος απαιτεί 2N 16-bit μεταβλητές για FFT δεδομένα ο μικροελεκτής μας μπορεί να εφαρμόσει FFT με N μέχρι 2048. Ωστόσο, άλλα μέρη του firmware απαιτούν μερικά Bytes της SRAM. Για την εφαρμογή μας βάζουμε όριο στο N το 256. Χρησιμοποιώντας 16-bits μεταβλητές στο να αναπαραστήσεις το πραγματικό και φανταστικό μέρος της κάθε τιμής, 1024Bytes της RAM απαιτούνται για FFT δεδομένα.

Πρόγραμμα μικροελεκτή: Η παρακάτω παράγραφος περιγράφει το firmware το οποίο υπολογίζει ένα radix-2 FFT σε ένα χαμηλής ισχύος μC. Τα δείγματα που διαβάζει ο ADC αποθηκεύονται στο x_n_re_array (πίνακα). Αυτός ο πίνακας απεικονίζει τις πραγματικές τιμές του x(n). Οι φανταστικές τιμές που αρχικοποιούνται με μηδέν πριν αρχίσει το FFT, αποθηκεύονται στο x_n_im_array (πίνακα). Όταν το FFT ολοκληρωθεί, τα αποτελέσματα του φάσματος θα έχουν αντικαθιστίσει τις αρχικές τιμές (πεδίο του χρόνου) και θα αποθηκευτούν στους πίνακες x_n_re (πραγματικό μέρος) και x_n_im (φανταστικό μέρος).

Δειγματοληψία: Ο FFT αλγόριθμος υποθέτει ότι τα δείγματα παίρνονται σε σταθερή συχνότητα δειγματοληψίας. Ένα FFT δεν μπορεί να υλοποιηθεί σωστά αν δεν γίνει σωστή δειγματοληψία. Για παράδειγμα, αν δεν είναι σταθερή η δειγματοληψία (ύπαρξη jitter) τότε θα προκληθούν σφάλματα στα FFT αποτελέσματα.

Τριγωνομετρική Πίνακες αναφοράς: Ο FFT αλγόριθμος χρησιμοποιεί πίνακες αναφοράς (lookup tables LUTs) αντί να υπολογίζει τη τιμή του συνημίτονου ή του

ημίτονου. Οι δηλώσεις των πινάκων αναφοράς του ημίτονου και συνημίτονου δίνονται στον Πίνακας 3.1. Και οι δύο πίνακες αναφοράς έχουν N / 2 στοιχεία διότι είναι συμμετρικοί.

LUT'S

```
const float cosLUT[N_DIV_2] =
{
1.0000,0.9997,0.9988,0.9973,0.9952,0.9925,0.9892,0.9853,0.9808,0.9757,0.9700,0.9638,0.9569,0.9495,
0.9415,0.9330,0.9239,0.9142,0.9040,0.8932,0.8819,0.8701,0.8577,0.8449,0.8315,0.8176,0.8032,0.7883,
0.7730,0.7572,0.7410,0.7242,0.7071,0.6895,0.6716,0.6532,0.6344,0.6152,0.5957,0.5758,0.5556,0.5350,
0.5141,0.4929,0.4714,0.4496,0.4276,0.4052,0.3827,0.3599,0.3369,0.3137,0.2903,0.2667,0.2430,0.2191,
0.1951,0.1710,0.1467,0.1224,0.0980,0.0736,0.0491,0.0245,0.0000,-0.0245,-0.0491,-0.0736,-0.0980,
-0.1224,-0.1467,-0.1710,-0.1951,-0.2191,-0.2430,-0.2667,-0.2903,-0.3137,-0.3369,-0.3599,-0.3827,
-0.4052,-0.4276,-0.4496,-0.4714,-0.4929,-0.5141,-0.5350,-0.5556,-0.5758,-0.5957,-0.6152,-0.6344,
-0.6532,-0.6716,-0.6895,-0.7071,-0.7242,-0.7410,-0.7572,-0.7730,-0.7883,-0.8032,-0.8176,-0.8315,
-0.8449,-0.8577,-0.8701,-0.8819,-0.8932,-0.9040,-0.9142,-0.9239,-0.9330,-0.9415,-0.9495,-0.9569,
-0.9638,-0.9700,-0.9757,-0.9808,-0.9853,-0.9892,-0.9925,-0.9952,-0.9973,-0.9988,-0.9997
};

const float sinLUT[N_DIV_2] =
{
0.0000,0.0245,0.0491,0.0736,0.0980,0.1224,0.1467,0.1710,0.1951,0.2191,0.2430,0.2667,0.2903,0.3137,
0.3369,0.3599,0.3827,0.4052,0.4276,0.4496,0.4714,0.4929,0.5141,0.5350,0.5556,0.5758,0.5957,0.6152,
0.6344,0.6532,0.6716,0.6895,0.7071,0.7242,0.7410,0.7572,0.7730,0.7883,0.8032,0.8176,0.8315,0.8449,
0.8577,0.8701,0.8819,0.8932,0.9040,0.9142,0.9239,0.9330,0.9415,0.9495,0.9569,0.9638,0.9700,0.9757,
0.9808,0.9853,0.9892,0.9925,0.9952,0.9973,0.9988,0.9997,1.0000,0.9997,0.9988,0.9973,0.9952,0.9925,
0.9892,0.9853,0.9808,0.9757,0.9700,0.9638,0.9569,0.9495,0.9415,0.9330,0.9239,0.9142,0.9040,0.8932,
0.8819,0.8701,0.8577,0.8449,0.8315,0.8176,0.8032,0.7883,0.7730,0.7572,0.7410,0.7242,0.7071,0.6895,
0.6716,0.6532,0.6344,0.6152,0.5957,0.5758,0.5556,0.5350,0.5141,0.4929,0.4714,0.4496,0.4276,0.4052,
0.3827,0.3599,0.3369,0.3137,0.2903,0.2667,0.2430,0.2191,0.1951,0.1710,0.1467,0.1224,0.0980,0.0736,
0.0491,0.0245
};
```

Πίνακας 3.1 - Τριγωνομετρικοί Πίνακες Αναφοράς

Οι πίνακες που περιέχουν αυτές τις αναφορές δηλώνονται ως **const** (σταθερές), αναγκάζοντας τον compiler να τις αποθήκευση στην FLASH αντί της RAM.

Ανασύνταξη θέσης των δειγμάτων: Η διαδικασία της ανασύνταξης της θέσης των δειγμάτων εισόδου με τεχνική αντιστροφής των bits του δείκτη της θέσης των δειγμάτων, μπορεί να υπολογιστεί κατά το χρόνο εκτέλεσης, με αναφορά τους LUTs, ή γράφοντας κατευθείαν ένα «unrolled loop». Κάθε μία από αυτές τις μεθόδους έχει μειονεκτήματα και τα πλεονεκτήματα (trade-offs) του μεγέθους του κώδικα και της ταχύτητας εκτέλεσης. Αυτή η FFT εφαρμογή εκτελεί αντιστροφή ψηφίων χρησιμοποιώντας «unrolled loop», που απαιτεί περισσότερο κώδικα, αλλά έχει γρηγορότερη εκτέλεση. Ο Πίνακας 3.2 δείχνει την υλοποίηση αυτής της «unrolled loop».

Ανασύνταξης της θέσης των δειγμάτων

```
i=x_n_re[ 1]; x_n_re[ 1]=x_n_re[128]; x_n_re[128]=i;
i=x_n_re[ 2]; x_n_re[ 2]=x_n_re[ 64]; x_n_re[ 64]=i;
i=x_n_re[ 3]; x_n_re[ 3]=x_n_re[192]; x_n_re[192]=i;
i=x_n_re[ 4]; x_n_re[ 4]=x_n_re[ 32]; x_n_re[ 32]=i; ...
i=x_n_re[207]; x_n_re[207]=x_n_re[243]; x_n_re[243]=i;
i=x_n_re[215]; x_n_re[215]=x_n_re[235]; x_n_re[235]=i;
i=x_n_re[223]; x_n_re[223]=x_n_re[251]; x_n_re[251]=i;
i=x_n_re[239]; x_n_re[239]=x_n_re[247]; x_n_re[247]=i;
```

Πίνακας 3.2 - Εντολές για αντιστροφή ψηφίων για N=256

Ο Radix-2 FFT Αλγόριθμος: Μετά την ανασύνταξη των δειγμάτων χρησιμοποιώντας την αντιστροφή ψηφίων, μπορεί να υπολογιστεί ο FFT. Το πρόγραμμα για αυτή την εφαρμογή του radix-2 FFT εκτελεί τους υπολογισμούς πεταλούδα που φαίνονται στην Εικόνα 13 με τρεις κύριους βρόγχους. Ο εξωτερικός βρόγχος μετράει (αυξάνεται) διαμέσου των $\log_2(N)$ σταδίων του FFT υπολογισμού. Ο εσωτερικός βρόγχος εκτελεί τους εσωτερικούς υπολογισμούς πεταλούδας του κάθε σταδίου. Η καρδιά του FFT αλγόριθμου είναι το μικρό μπλοκ κώδικα που εκτελεί κάθε υπολογισμό πεταλούδας (Πίνακας 3.3).

Radix-2

```
MUL_1(cosLUT[tf],x_n_re[b],resultMulReCos);
MUL_2(sinLUT[tf],resultMulReSin);
MUL_1(cosLUT[tf],x_n_im[b],resultMulImCos);
MUL_2(sinLUT[tf],resultMulImSin);
x_n_re[b] = x_n_re[a]-resultMulReCos+resultMulImSin;
x_n_im[b] = x_n_im[a]-resultMulReSin-resultMulImCos;
x_n_re[a] = x_n_re[a]+resultMulReCos-resultMulImSin;
x_n_im[a] = x_n_im[a]+resultMulReSin+resultMulImCos;
```

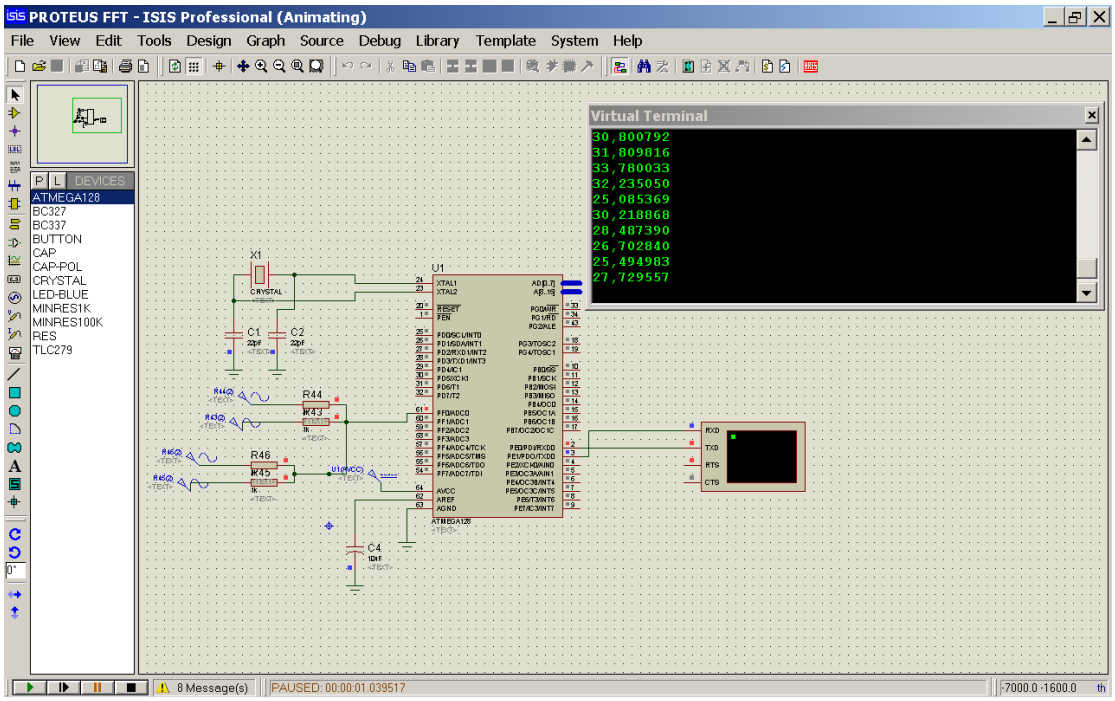
Πίνακας 3.3 - Ο υπολογισμός πεταλούδας σε C

3.4 Λογισμικό για τον Ηλεκτρονικό Υπολογιστή

Μετά την υλοποίηση των λογικών διαγραμμάτων, αναπτύχθηκε λογισμικό σε περιβάλλον Windows XP, σε γλώσσα προγραμματισμού Delphi 7.0 από τον Δρ. Κουλούρα προκειμένου να είναι δυνατή η επικοινωνία της συσκευής με τον Η/Υ.

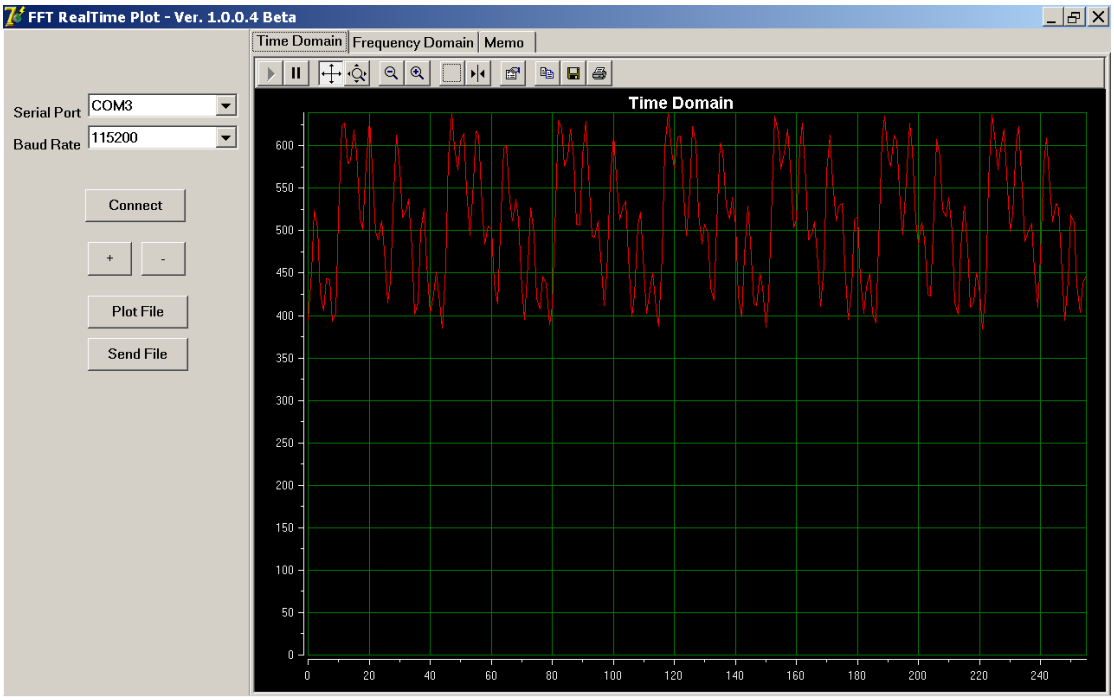
Κατά το πειραματικό στάδιο ανάπτυξης των λογισμικών του μικροελεκτή και του Η/Υ χρησιμοποιήθηκε το λογισμικό ISIS PROTEUS 7.1 SP4 για την ηλεκτρονική εξομοίωση συνθηκών λειτουργίας του κυκλώματος.

Στα παρακάτω σχήματα φαίνεται ο Γρήγορος Μετασχηματισμός Φουριέ ενός σήματος με τέσσερις διαφορετικές αρμονικές διαφορετικού πλάτους. Όπως φαίνεται στο παρακάτω παράδειγμα αθροίζονται τρία ημίτονα και οδηγούνται στο πρώτο κανάλι του ADC(0). Μετά την δειγματοληψία του σήματος ο μC υλοποιεί τον FFT αλγόριθμο και στέλνει τα αποτελέσματα στο Η/Υ μέσου της σειριακή πόρτας.

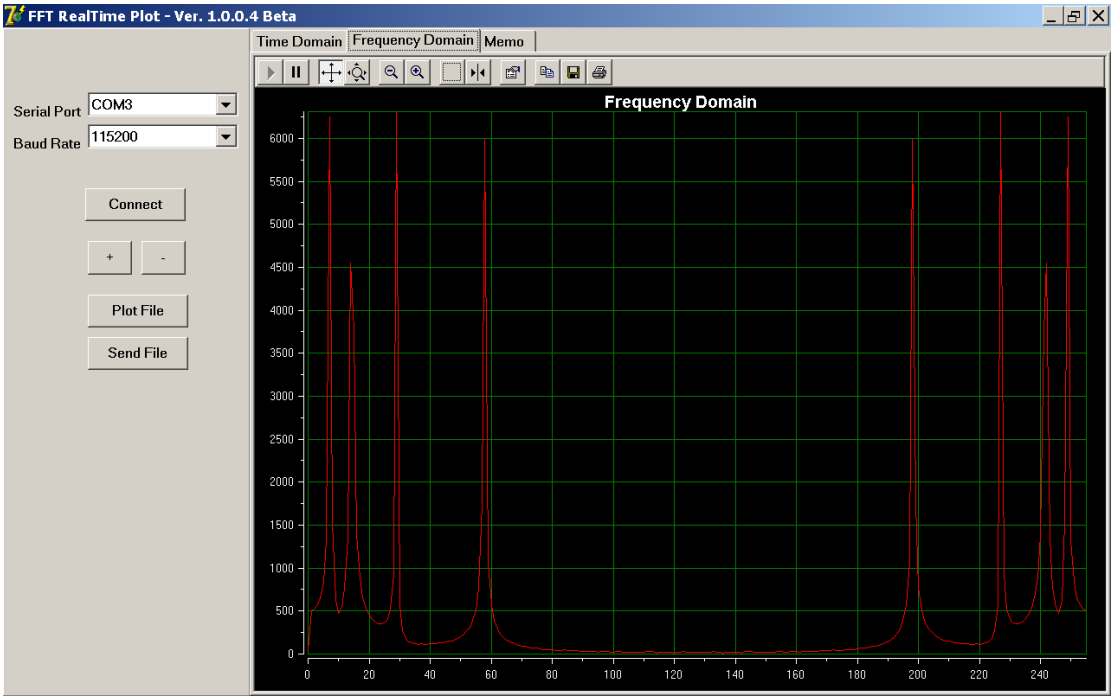


Εικόνα 14 - Εξομοίωση με λογισμικό PROTEUS 7.1 SP4

Τα Real Time αποτελέσματα απεικονίζονται στο πρόγραμμα FFT RealTime Plot Στην φαίνεται το σήμα στο πεδίο του χρόνου



Εικόνα 15 - Σήμα στο πεδίο του χρόνου



Εικόνα 16 - Σήμα στο πεδίο των συχνοτήτων

3.5 Συμπεράσματα

Στο κεφάλαιο αυτό αναλύθηκε το πρόγραμμα για τον έλεγχο της διάταξης, τόσο αυτό της ίδιας της διάταξης όσο και αυτό του υπολογιστή. Ο κώδικας που αφορά τον μικροελεκτή έχει γραφτεί στο Codevision σε γλώσσα προγραμματισμού C, ενώ ο κώδικας για τον Η/Υ έχει γραφτεί σε Delphi για Windows XP. Η Borland Delphi είναι μια πολύ καλά δομημένη γλώσσα προγραμματισμού για υψηλού επιπέδου προγραμματισμό. Το λογισμικό του μικροελεκτή AVR γράφτηκε σε γλώσσα προγραμματισμού C, για να είναι εύκολη και γρήγορη η ανάπτυξη και η μελλοντικές αναβαθμίσεις της εφαρμογής.

4 Συμπεράσματα – Μελλοντικές εργασίες

4.1 Συμπεράσματα

Θεμελιώδες πλεονέκτημα της ψηφιακής τεχνολογίας είναι η ομοιομορφία στην απόδοση. Κάθε αντίγραφο της ηλεκτρονικής συσκευής έχει ακριβώς τα ίδια χαρακτηριστικά με το πρωτότυπο, κάτι που δεν είναι ούτε προφανές, ούτε εύκολο κατασκευαστικά στις αναλογικές σχεδιάσεις. Είναι προφανές ότι σε γραμμή παραγωγής οργάνων ακριβείας πρέπει να κατασκευάζονται πανομοιότυπα «αντίτυπα» ώστε να οι ποιότητα και πιστότητα των συστημάτων να είναι ίδιες. Τα αναλογικά φίλτρα αποτελούνται από διακριτά στοιχεία R, L, C, που καθορίζουν τις ιδιότητές τους. Τα ηλεκτρικά χαρακτηριστικά των υλικών (κυρίως των πυκνωτών) αλλοιώνονται με το πέρασμα του χρόνου που έχει σαν συνέπεια να αλλάζουν τα χαρακτηριστικά του αναλογικού φίλτρου. Αντίθετα τα ψηφιακά φίλτρα δεν αλλάζουν τα χαρακτηριστικά τους με την πάροδο του χρόνου καθώς υλοποιούνται με λογισμικό (DSP). Για τους παραπάνω λόγους επιλέχθηκε η σχεδίαση του φίλτρου στενής ζώνης διέλευσης του οργάνου να είναι ψηφιακή.

4.2 Μελλοντικές εργασίες

Ενδέχεται μελλοντικά τα ψηφιακά δεδομένα να αποθηκεύονται σε εξωτερική μνήμη τύπου Flash μεγαλύτερης χωρητικότητας. Με αυτό το τρόπο θα υπάρχει δυνατότητα να αποθηκεύονται πολύ περισσότερες μετρήσεις αφού ο χώρος αποθήκευσης των μετρήσεων θα είναι της τάξης των GB. Επίσης θα χρησιμοποιηθεί Boot Loader στο μC ώστε να υπάρχει δυνατότητα αναβάθμισης του λογισμικού του μικροελεκτή. Προκειμένου να πετύχουμε καλύτερες επιδόσεις από πλευράς ταχύτητας επεξεργασίας, ο βασικός κορμός του προγράμματος θα γραφτεί σε γλώσσα μηχανής (Assembly). Τέλος το λογισμικό υποστήριξης στον Η/Υ θα βελτιωθεί, έτσι ώστε η διαχείριση του προγράμματος από τον χρήστη να είναι πιο εύκολη και φιλική (User Friendly) και να λειτουργεί σε περισσότερα λειτουργικά συστήματα όπως το Linux.

Αναφορές

ATMEL Corporation (2007) - “ATmega2561, 8-bit AVR Microcontroller with 2561K Bytes In-System Programmable Flash”, Rev. 2467H, AVR

Maxim Comporation (2001) – Max3311, 460kbps, 1μA Supply Current, 19-1901; Rev 0; 2/01

BI Technologies - R2R Resistor Ladder Networks

National Semiconductors(2008) – LMP7721, 3 Femtoampere Input Bias Current Precision Amplifier, March 14, 2008

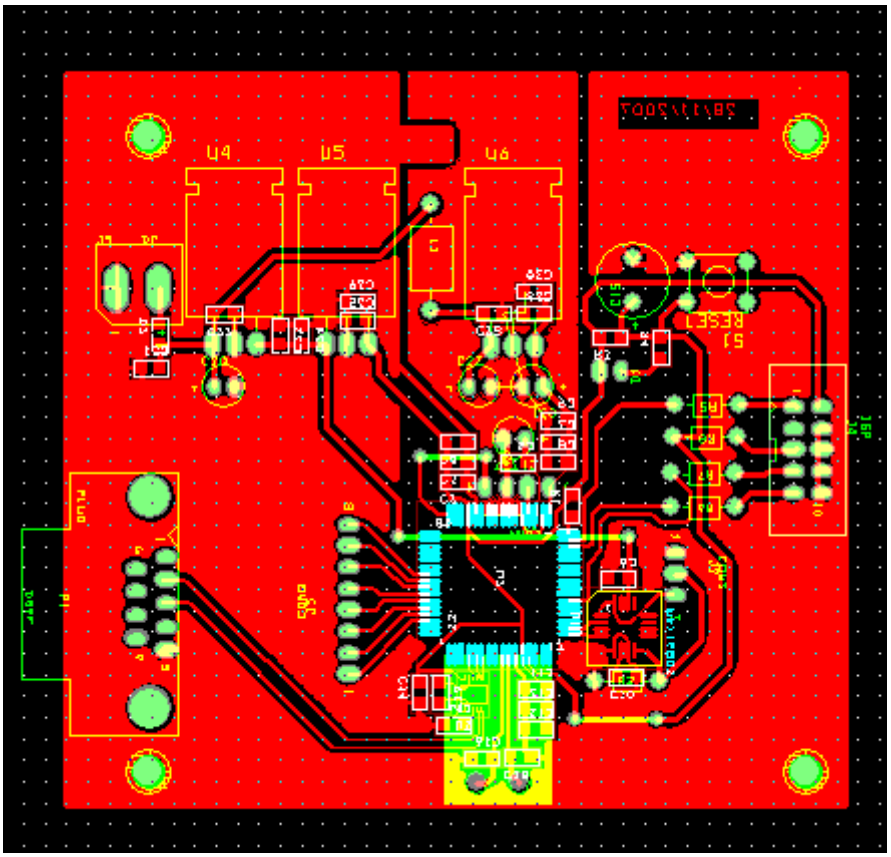
Maxim Comporation (2005) - APPLICATION NOTE 3722 Developing FFT Applications with Low-Power Microcontrollers, Dec 13, 2005

Szyperski C. (Editors), (2002) - “Component Software: Beyond Object-Oriented Programming”, Addison-Wesley Professional, 2nd Edition, pp. 624, ISBN: 978-0201745726

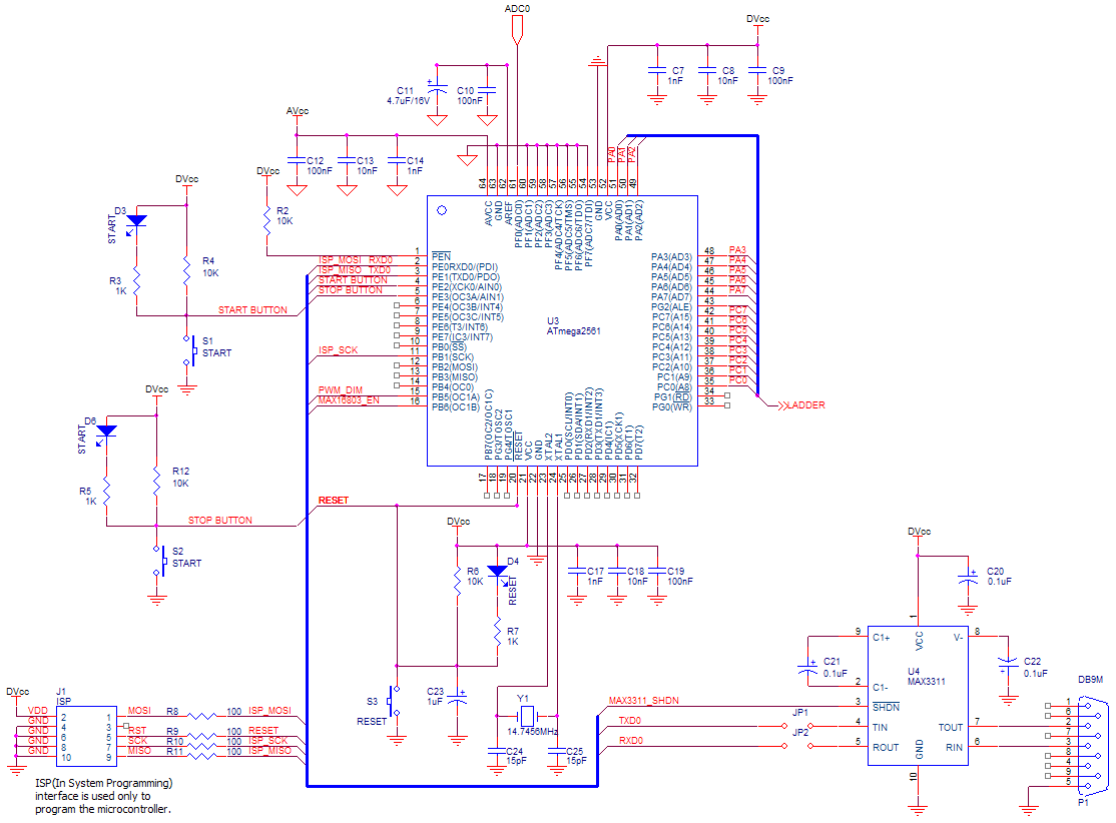
Texas Instruments Incorporated (2001) - “TLC279, LinCMOS™ Precision Quad Operation Amplifiers, SLOS092D”, Rev. 03/2001

Weber E., Ford N. and Weber C. (Editors), (1995) - “Developing With Delphi: Object-Oriented Techniques”, Prentice Hall, 1st Edition, pp. 418, ISBN: 978-0133781182

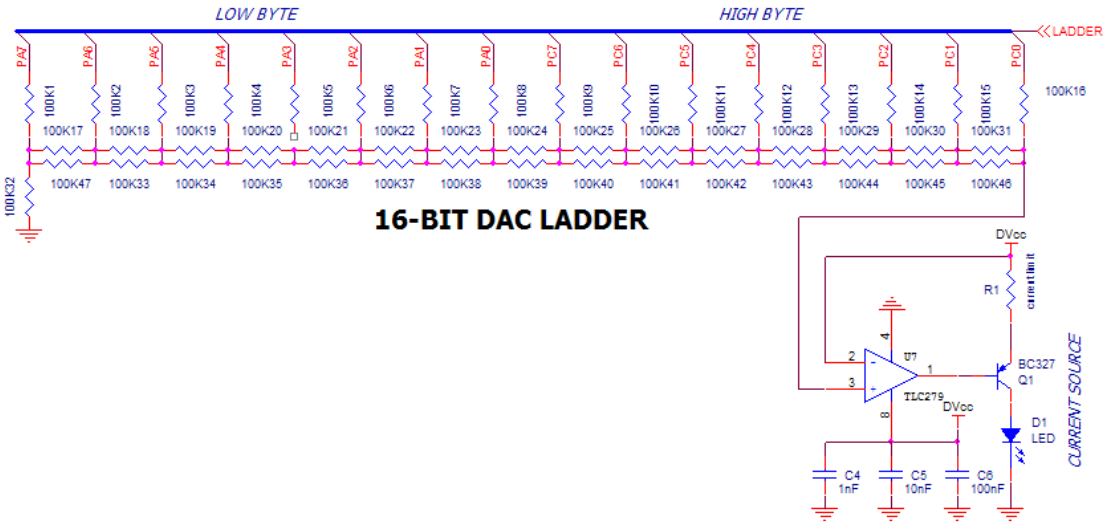
Παράρτημα Α’ (Πλακέτα)

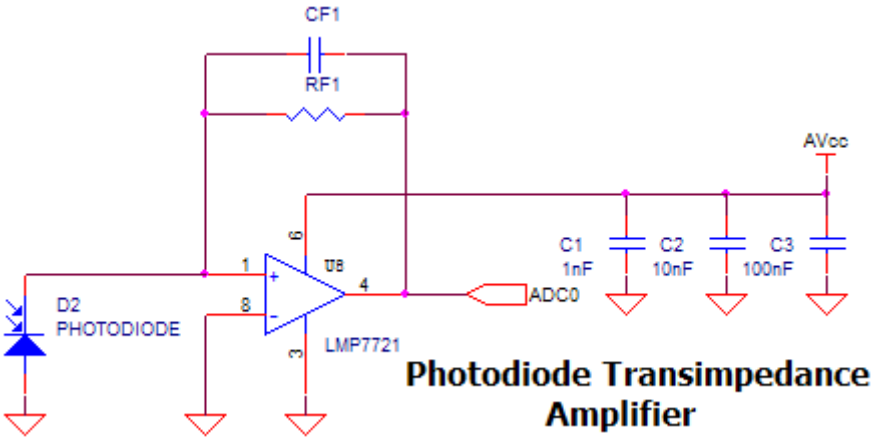


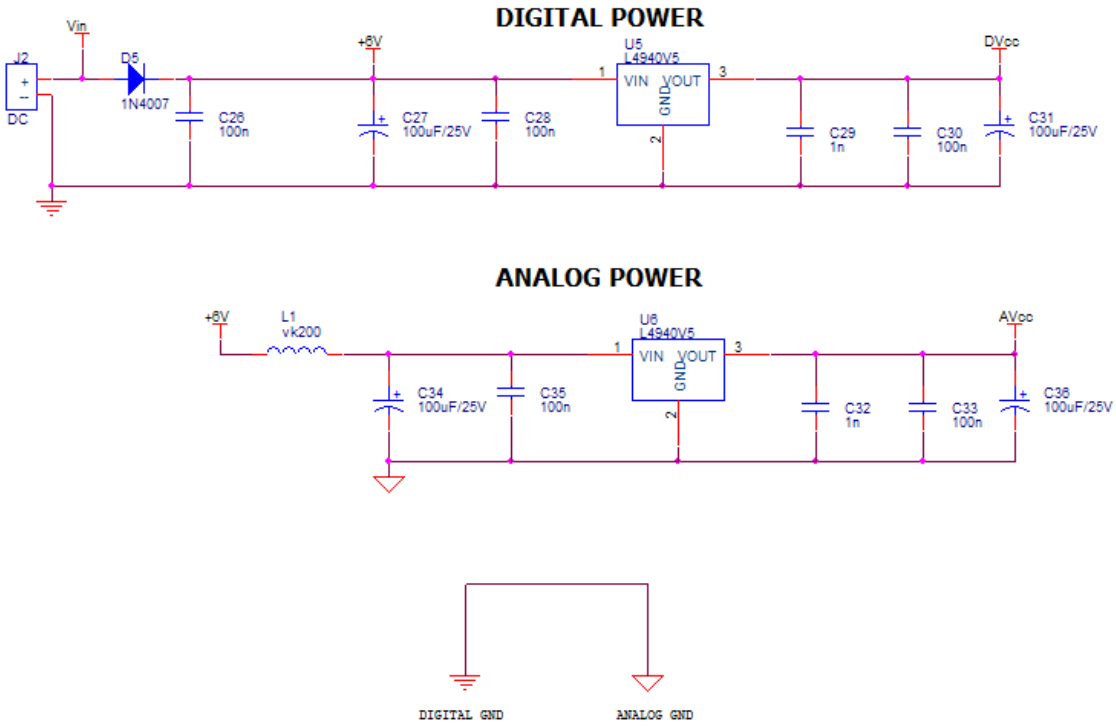
Παράρτημα Β’ (Σχηματικά Διαγράμματα)



MICROCONTROLLER


LOW BYTE
HIGH BYTE
LADDER
16-BIT DAC LADDER
DVcc
R1
BC327
Q1
U7
TLC279
D1
LED
CURRENT SOURCE
C4
1nF
C5
10nF
C6
100nF


CF1
RF1
AVcc
U8
LMP7721
D2
PHOTODIODE
ADC0
C1
1nF
C2
10nF
C3
100nF
Photodiode Transimpedance
Amplifier


DIGITAL POWER
Vin
J2
DC
D5
1N4007
C26
100n
+6V
C27
100uF/25V
C28
100n
U5
L4940V5
VIN
VOUT
GND
C29
1n
C30
100n
DVcc
C31
100uF/25V
ANALOG POWER
+6V
L1
vk200
C34
100uF/25V
C35
100n
U6
L4940V5
VIN
VOUT
GND
C32
1n
C33
100n
AVcc
C36
100uF/25V
DIGITAL GND
ANALOG GND

Παραρτήμα Γ’ (Κώδικας σε C για τον AVR)

Codevision

```
/*****************************************************
This program was produced by the
CodeWizardAVR V1.25.7 beta 5 Professional
Automatic Program Generator
© Copyright 1998-2007 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :Flurometer
Version :1.0
Date    : 21/3/2008
Author  : Dimitris Agiakatsikas
Company : PTEAM
Comments:


Chip type           : ATmega2561
Program type        : Application
Clock frequency     : 14,745600 MHz
Memory model        : Small
External SRAM size  : 0
Data Stack size     : 2048
*****************************************************/
/* DEFINE STATEMENTS */
#define N               256
#define N_DIV_2         128
#define N_DIV_2_PLUS_1    129
#define N_MINUS_1       255
#define LOG_2_N          8
#define amp             10
#define periods          120
#define offset          0

// include files
#include <mega2561.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <delay.h>

// Declare your constants here
const float cosLUT[N_DIV_2] =
{
1.0000,0.9997,0.9988,0.9973,0.9952,0.9925,0.9892,0.9853,0.9808,0.9757,0.9700,0.9638,0.9569,0.9495,0.9415,0.9330,
0.9239,0.9142,0.9040,0.8932,0.8819,0.8701,0.8577,0.8449,0.8315,0.8176,0.8032,0.7883,0.7730,0.7572,0.7410,0.7242,
0.7071,0.6895,0.6716,0.6532,0.6344,0.6152,0.5957,0.5758,0.5556,0.5350,0.5141,0.4929,0.4714,0.4496,0.4276,0.4052,
0.3827,0.3599,0.3369,0.3137,0.2903,0.2667,0.2430,0.2191,0.1951,0.1710,0.1467,0.1224,0.0980,0.0736,0.0491,0.0245,
0.0000,-0.0245,-0.0491,-0.0736,-0.0980,-0.1224,-0.1467,-0.1710,-0.1951,-0.2191,-0.2430,-0.2667,-0.2903,-0.3137,-0.3369,-0.3599,
-0.3827,-0.4052,-0.4276,-0.4496,-0.4714,-0.4929,-0.5141,-0.5350,-0.5556,-0.5758,-0.5957,-0.6152,-0.6344,-0.6532,-0.6716,-0.6895,
-0.7071,-0.7242,-0.7410,-0.7572,-0.7730,-0.7883,-0.8032,-0.8176,-0.8315,-0.8449,-0.8577,-0.8701,-0.8819,-0.8932,-0.9040,-0.9142,
-0.9239,-0.9330,-0.9415,-0.9495,-0.9569,-0.9638,-0.9700,-0.9757,-0.9808,-0.9853,-0.9892,-0.9925,-0.9952,-0.9973,-0.9988,-0.9997
};
```

```
const float sinLUT[N_DIV_2] =
{
0.0000,0.0245,0.0491,0.0736,0.0980,0.1224,0.1467,0.1710,0.1951,0.2191,0.2430,0.2667,0.2903,0.3137,0.3369,0.3599,
0.3827,0.4052,0.4276,0.4496,0.4714,0.4929,0.5141,0.5350,0.5556,0.5758,0.5957,0.6152,0.6344,0.6532,0.6716,0.6895,
0.7071,0.7242,0.7410,0.7572,0.7730,0.7883,0.8032,0.8176,0.8315,0.8449,0.8577,0.8701,0.8819,0.8932,0.9040,0.9142,
0.9239,0.9330,0.9415,0.9495,0.9569,0.9638,0.9700,0.9757,0.9808,0.9853,0.9892,0.9925,0.9952,0.9973,0.9988,0.9997,
1.0000,0.9997,0.9988,0.9973,0.9952,0.9925,0.9892,0.9853,0.9808,0.9757,0.9700,0.9638,0.9569,0.9495,0.9415,0.9330,
0.9239,0.9142,0.9040,0.8932,0.8819,0.8701,0.8577,0.8449,0.8315,0.8176,0.8032,0.7883,0.7730,0.7572,0.7410,0.7242,
0.7071,0.6895,0.6716,0.6532,0.6344,0.6152,0.5957,0.5758,0.5556,0.5350,0.5141,0.4929,0.4714,0.4496,0.4276,0.4052,
0.3827,0.3599,0.3369,0.3137,0.2903,0.2667,0.2430,0.2191,0.1951,0.1710,0.1467,0.1224,0.0980,0.0736,0.0491,0.0245
};

const unsigned int sin_dacA[256] =
{  //4,25Volt - 4,5Volt
57344,57384,57425,57465,57505,57545,57585,57625,57665,57704,57744,57783,57821,57860,57898,57936,57973,58010,58047,
58083,58119,58154,58189,58224,58257,58291,58323,58355,58387,58418,58448,58477,58506,58534,58562,58588,58614,58639,
58664,58687,58710,58732,58753,58773,58792,58811,58828,58845,58861,58875,58889,58902,58914,58925,58935,58944,58953,
58960,58966,58971,58975,58979,58981,58982,58982,58982,58980,58977,58973,58969,58963,58956,58949,58940,58930,58920,
58908,58896,58882,58868,58853,58837,58820,58802,58783,58763,58742,58721,58699,58675,58651,58627,58601,58575,58548,
58520,58492,58463,58433,58402,58371,58339,58307,58274,58241,58207,58172,58137,58101,58065,58029,57992,57955,57917,
57879,57841,57802,57763,57724,57685,57645,57605,57565,57525,57485,57445,57405,57364,57324,57283,57243,57203,57163,
57123,57083,57043,57003,56964,56925,56886,56847,56809,56771,56733,56696,56659,56623,56587,56551,56516,56481,56447,
56414,56381,56349,56317,56286,56255,56225,56196,56168,56140,56113,56087,56061,56037,56013,55989,55967,55946,55925,
55905,55886,55868,55851,55835,55820,55806,55792,55780,55768,55758,55748,55739,55732,55725,55719,55715,55711,55708,
55706,55706,55706,55707,55709,55713,55717,55722,55728,55735,55744,55753,55763,55774,55786,55799,55813,55827,55843,
55860,55877,55896,55915,55935,55956,55978,56001,56024,56049,56074,56100,56126,56154,56182,56211,56240,56270,56301,
56333,56365,56397,56431,56464,56499,56534,56569,56605,56641,56678,56715,56752,56790,56828,56867,56905,56944,56984,
57023,57063,57103,57143,57183,57223,57263,57304,57344
};
const unsigned int sin_dacB[256] =
{  //4Volt - 4,5Volt
55706,55786,55867,55948,56028,56108,56188,56268,56347,56426,56505,56583,56660,56737,56814,56889,56964,57038,57112,
57184,57256,57326,57396,57465,57532,57599,57664,57728,57791,57853,57913,57972,58030,58086,58141,58194,58246,58296,
58345,58392,58437,58481,58523,58563,58602,58639,58674,58707,58739,58768,58796,58822,58846,58868,58888,58907,58923,
58937,58950,58960,58968,58975,58979,58982,58982,58981,58977,58972,58964,58955,58944,58930,58915,58898,58878,58857,
58834,58809,58782,58754,58723,58691,58657,58621,58583,58543,58502,58459,58415,58368,58321,58271,58220,58168,58114,
58058,58001,57943,57883,57822,57760,57696,57632,57566,57499,57431,57361,57291,57220,57148,57075,57001,56927,56852,
56776,56699,56622,56544,56466,56387,56308,56228,56148,56068,55988,55907,55827,55746,55665,55585,55504,55423,55343,
55263,55183,55103,55024,54946,54867,54789,54712,54636,54560,54484,54410,54336,54263,54191,54120,54050,53981,53912,
53845,53780,53715,53651,53589,53528,53468,53410,53353,53298,53244,53191,53140,53091,53043,52997,52952,52909,52868,
52828,52791,52755,52720,52688,52657,52629,52602,52577,52554,52533,52514,52496,52481,52468,52456,52447,52439,52434,
52430,52429,52429,52432,52436,52443,52451,52462,52474,52488,52505,52523,52543,52565,52589,52615,52643,52672,52704,
52737,52772,52809,52848,52888,52930,52974,53019,53067,53115,53165,53217,53270,53325,53381,53439,53498,53558,53620,
53683,53747,53812,53879,53946,54015,54085,54155,54227,54299,54373,54447,54522,54598,54674,54751,54828,54906,54985,
55064,55143,55223,55303,55383,55464,55544,55625,55706
};
const unsigned int sin_dacC[256] =
{  //3,75Volt - 4,5Volt
54067,54188,54309,54430,54551,54671,54791,54911,55030,55148,55266,55383,55499,55615,55729,55843,55955,56066,56176,
56285,56393,56499,56603,56706,56807,56907,57005,57101,57196,57288,57379,57467,57553,57638,57720,57800,57877,57953,
58026,58096,58165,58230,58293,58354,58412,58467,58520,58570,58617,58661,58703,58742,58778,58811,58841,58869,58893,
58915,58933,58949,58961,58971,58978,58982,58982,58980,58975,58967,58955,58941,58924,58904,58881,58855,58826,58795,
58760,58723,58683,58639,58594,58545,58494,58440,58383,58324,58262,58198,58131,58061,57990,57915,57839,57760,57679,
57596,57511,57423,57334,57242,57149,57053,56956,56857,56757,56655,56551,56446,56339,56231,56122,56011,55899,55786,
55672,55557,55441,55325,55207,55089,54970,54851,54731,54611,54491,54370,54249,54128,54007,53886,53765,53644,53523,
53403,53283,53164,53045,52927,52810,52693,52577,52462,52348,52235,52123,52013,51903,51795,51689,51583,51480,51378,
51277,51178,51081,50986,50892,50801,50711,50624,50539,50455,50374,50295,50219,50145,50073,50004,49937,49872,49811,
49751,49695,49641,49589,49541,49495,49452,49412,49374,49340,49308,49279,49253,49230,49210,49193,49179,49168,49160,
49154,49152,49153,49157,49163,49173,49186,49201,49220,49241,49266,49293,49323,49357,49393,49431,49473,49518,49565,
49615,49667,49723,49781,49841,49904,49970,50038,50109,50182,50257,50335,50414,50497,50581,50667,50756,50846,50939,
51033,51129,51227,51327,51428,51531,51636,51742,51849,51958,52068,52179,52292,52405,52520,52635,52751,52868,52986,
53105,53224,53343,53463,53584,53704,53825,53946,54067
};
```

```
const unsigned int sin_dacD[256] =
{  //3,5Volt - 4,5Volt
52429,52590,52752,52913,53074,53234,53394,53554,53712,53870,54027,54183,54338,54492,54645,54796,54946,55094,55241,
55386,55529,55671,55810,55947,56082,56215,56346,56474,56600,56723,56844,56962,57077,57190,57299,57406,57509,57610,
57707,57801,57892,57979,58064,58144,58222,58295,58366,58432,58495,58554,58610,58662,58710,58754,58794,58831,58863,
58892,58917,58938,58954,58967,58976,58981,58982,58979,58972,58961,58946,58928,58905,58878,58847,58813,58774,58732,
58686,58636,58583,58525,58464,58399,58331,58259,58183,58104,58022,57936,57847,57754,57659,57560,57458,57353,57245,
57134,57020,56903,56784,56662,56537,56410,56281,56149,56015,55879,55740,55600,55458,55314,55168,55020,54871,54721,
54569,54416,54261,54105,53949,53791,53633,53474,53314,53154,52993,52832,52671,52510,52348,52187,52025,51864,51704,
51543,51384,51225,51066,50909,50752,50597,50442,50289,50137,49986,49837,49690,49544,49400,49257,49117,48979,48843,
48708,48577,48447,48320,48196,48074,47954,47838,47724,47613,47505,47400,47298,47199,47103,47011,46922,46836,46753,
46674,46599,46527,46458,46394,46332,46275,46221,46171,46125,46083,46045,46010,45979,45953,45930,45911,45896,45885,
45878,45875,45876,45881,45890,45903,45920,45941,45966,45994,46027,46063,46104,46148,46196,46248,46303,46363,46425,
46492,46562,46636,46713,46794,46878,46966,47057,47151,47248,47348,47452,47559,47668,47780,47896,48014,48134,48258,
48383,48512,48642,48775,48910,49048,49187,49328,49472,49617,49763,49912,50061,50213,50365,50519,50674,50830,50987,
51145,51304,51463,51623,51784,51945,52106,52267,52429
};

// Declare your global variables here
unsigned char dac_ptr=0;
unsigned int cap_ptr=0;
unsigned char sin_dacSRAM_MSB[256];
unsigned char sin_dacSRAM_LSB[256];
char cmd;

//for future use
float x_n_re[N];
float x_n_im[N];
unsigned int y=0;
char str[15];
int zkl;
//temporary variables
int i ;
char strr[10];
//functions
float magn_F(float re,float im)
{
float res=sqrt((re*re)+(im*im));
return res;
}

int forward_FFT_F(float re[N],float im[N])
{
float i;
int  n_of_b     = N_DIV_2; // Number of butterflies
int  s_of_b     = 1;      // Size   of butterflies
int  a_index    = 0;      // fft data index
int  a_index_ref = 0;      // fft data index reference
unsigned char stage      = 0;      // Stage of the fft, 0 to (Log2(N)-1)
int  nb_index;   // Number of butterflies index
int  sb_index;   // Size   of butterflies index
float resultMulReCos;
float resultMulImCos;
float resultMulReSin;
float resultMulImSin;
int b_index;
int tf_index;
int y;
for(y=0;y<=N-1;y++)
      {
      im[y]=0;
      }
i=re[  1]; re[  1]=re[128]; re[128]=i;
i=re[  2]; re[  2]=re[ 64]; re[ 64]=i;
i=re[  3]; re[  3]=re[192]; re[192]=i;
i=re[  4]; re[  4]=re[ 32]; re[ 32]=i;
```

```
i=re[  5]; re[  5]=re[160]; re[160]=i;
i=re[  6]; re[  6]=re[ 96]; re[ 96]=i;
i=re[  7]; re[  7]=re[224]; re[224]=i;
i=re[  8]; re[  8]=re[ 16]; re[ 16]=i;
i=re[  9]; re[  9]=re[144]; re[144]=i;
i=re[ 10]; re[ 10]=re[ 80]; re[ 80]=i;
i=re[ 11]; re[ 11]=re[208]; re[208]=i;
i=re[ 12]; re[ 12]=re[ 48]; re[ 48]=i;
i=re[ 13]; re[ 13]=re[176]; re[176]=i;
i=re[ 14]; re[ 14]=re[112]; re[112]=i;
i=re[ 15]; re[ 15]=re[240]; re[240]=i;
i=re[ 17]; re[ 17]=re[136]; re[136]=i;
i=re[ 18]; re[ 18]=re[ 72]; re[ 72]=i;
i=re[ 19]; re[ 19]=re[200]; re[200]=i;
i=re[ 20]; re[ 20]=re[ 40]; re[ 40]=i;
i=re[ 21]; re[ 21]=re[168]; re[168]=i;
i=re[ 22]; re[ 22]=re[104]; re[104]=i;
i=re[ 23]; re[ 23]=re[232]; re[232]=i;
i=re[ 25]; re[ 25]=re[152]; re[152]=i;
i=re[ 26]; re[ 26]=re[ 88]; re[ 88]=i;
i=re[ 27]; re[ 27]=re[216]; re[216]=i;
i=re[ 28]; re[ 28]=re[ 56]; re[ 56]=i;
i=re[ 29]; re[ 29]=re[184]; re[184]=i;
i=re[ 30]; re[ 30]=re[120]; re[120]=i;
i=re[ 31]; re[ 31]=re[248]; re[248]=i;
i=re[ 33]; re[ 33]=re[132]; re[132]=i;
i=re[ 34]; re[ 34]=re[ 68]; re[ 68]=i;
i=re[ 35]; re[ 35]=re[196]; re[196]=i;
i=re[ 37]; re[ 37]=re[164]; re[164]=i;
i=re[ 38]; re[ 38]=re[100]; re[100]=i;
i=re[ 39]; re[ 39]=re[228]; re[228]=i;
i=re[ 41]; re[ 41]=re[148]; re[148]=i;
i=re[ 42]; re[ 42]=re[ 84]; re[ 84]=i;
i=re[ 43]; re[ 43]=re[212]; re[212]=i;
i=re[ 44]; re[ 44]=re[ 52]; re[ 52]=i;
i=re[ 45]; re[ 45]=re[180]; re[180]=i;
i=re[ 46]; re[ 46]=re[116]; re[116]=i;
i=re[ 47]; re[ 47]=re[244]; re[244]=i;
i=re[ 49]; re[ 49]=re[140]; re[140]=i;
i=re[ 50]; re[ 50]=re[ 76]; re[ 76]=i;
i=re[ 51]; re[ 51]=re[204]; re[204]=i;
i=re[ 53]; re[ 53]=re[172]; re[172]=i;
i=re[ 54]; re[ 54]=re[108]; re[108]=i;
i=re[ 55]; re[ 55]=re[236]; re[236]=i;
i=re[ 57]; re[ 57]=re[156]; re[156]=i;
i=re[ 58]; re[ 58]=re[ 92]; re[ 92]=i;
i=re[ 59]; re[ 59]=re[220]; re[220]=i;
i=re[ 61]; re[ 61]=re[188]; re[188]=i;
i=re[ 62]; re[ 62]=re[124]; re[124]=i;
i=re[ 63]; re[ 63]=re[252]; re[252]=i;
i=re[ 65]; re[ 65]=re[130]; re[130]=i;
i=re[ 67]; re[ 67]=re[194]; re[194]=i;
i=re[ 69]; re[ 69]=re[162]; re[162]=i;
i=re[ 70]; re[ 70]=re[ 98]; re[ 98]=i;
i=re[ 71]; re[ 71]=re[226]; re[226]=i;
i=re[ 73]; re[ 73]=re[146]; re[146]=i;
i=re[ 74]; re[ 74]=re[ 82]; re[ 82]=i;
i=re[ 75]; re[ 75]=re[210]; re[210]=i;
i=re[ 77]; re[ 77]=re[178]; re[178]=i;
i=re[ 78]; re[ 78]=re[114]; re[114]=i;
i=re[ 79]; re[ 79]=re[242]; re[242]=i;
i=re[ 81]; re[ 81]=re[138]; re[138]=i;
i=re[ 83]; re[ 83]=re[202]; re[202]=i;
i=re[ 85]; re[ 85]=re[170]; re[170]=i;
i=re[ 86]; re[ 86]=re[106]; re[106]=i;
i=re[ 87]; re[ 87]=re[234]; re[234]=i;
```

```
i=re[ 89]; re[ 89]=re[154]; re[154]=i;
i=re[ 91]; re[ 91]=re[218]; re[218]=i;
i=re[ 93]; re[ 93]=re[186]; re[186]=i;
i=re[ 94]; re[ 94]=re[122]; re[122]=i;
i=re[ 95]; re[ 95]=re[250]; re[250]=i;
i=re[ 97]; re[ 97]=re[134]; re[134]=i;
i=re[ 99]; re[ 99]=re[198]; re[198]=i;
i=re[101]; re[101]=re[166]; re[166]=i;
i=re[103]; re[103]=re[230]; re[230]=i;
i=re[105]; re[105]=re[150]; re[150]=i;
i=re[107]; re[107]=re[214]; re[214]=i;
i=re[109]; re[109]=re[182]; re[182]=i;
i=re[110]; re[110]=re[118]; re[118]=i;
i=re[111]; re[111]=re[246]; re[246]=i;
i=re[113]; re[113]=re[142]; re[142]=i;
i=re[115]; re[115]=re[206]; re[206]=i;
i=re[117]; re[117]=re[174]; re[174]=i;
i=re[119]; re[119]=re[238]; re[238]=i;
i=re[121]; re[121]=re[158]; re[158]=i;
i=re[123]; re[123]=re[222]; re[222]=i;
i=re[125]; re[125]=re[190]; re[190]=i;
i=re[127]; re[127]=re[254]; re[254]=i;
i=re[131]; re[131]=re[193]; re[193]=i;
i=re[133]; re[133]=re[161]; re[161]=i;
i=re[135]; re[135]=re[225]; re[225]=i;
i=re[137]; re[137]=re[145]; re[145]=i;
i=re[139]; re[139]=re[209]; re[209]=i;
i=re[141]; re[141]=re[177]; re[177]=i;
i=re[143]; re[143]=re[241]; re[241]=i;
i=re[147]; re[147]=re[201]; re[201]=i;
i=re[149]; re[149]=re[169]; re[169]=i;
i=re[151]; re[151]=re[233]; re[233]=i;
i=re[155]; re[155]=re[217]; re[217]=i;
i=re[157]; re[157]=re[185]; re[185]=i;
i=re[159]; re[159]=re[249]; re[249]=i;
i=re[163]; re[163]=re[197]; re[197]=i;
i=re[167]; re[167]=re[229]; re[229]=i;
i=re[171]; re[171]=re[213]; re[213]=i;
i=re[173]; re[173]=re[181]; re[181]=i;
i=re[175]; re[175]=re[245]; re[245]=i;
i=re[179]; re[179]=re[205]; re[205]=i;
i=re[183]; re[183]=re[237]; re[237]=i;
i=re[187]; re[187]=re[221]; re[221]=i;
i=re[191]; re[191]=re[253]; re[253]=i;
i=re[199]; re[199]=re[227]; re[227]=i;
i=re[203]; re[203]=re[211]; re[211]=i;
i=re[207]; re[207]=re[243]; re[243]=i;
i=re[215]; re[215]=re[235]; re[235]=i;
i=re[223]; re[223]=re[251]; re[251]=i;
i=re[239]; re[239]=re[247]; re[247]=i;

for(stage=0; stage<LOG_2_N; stage++)
    {
      for(nb_index=0; nb_index<n_of_b; nb_index++)
      {
        tf_index = 0; // The twiddle factor index
        for(sb_index=0; sb_index<s_of_b; sb_index++)
        {
        b_index = a_index+s_of_b;
        resultMulReCos=cosLUT[tf_index]*re[b_index];
        resultMulReSin=sinLUT[tf_index]*re[b_index];
        resultMulImCos=cosLUT[tf_index]*im[b_index];
        resultMulImSin=sinLUT[tf_index]*im[b_index];
        re[b_index] = re[a_index]-resultMulReCos+resultMulImSin;
        im[b_index] = im[a_index]-resultMulReSin-resultMulImCos;
        re[a_index] = re[a_index]+resultMulReCos-resultMulImSin;
```

```
        im[a_index] = im[a_index]+resultMulReSin+resultMulImCos;

          if (((sb_index+1) & (s_of_b-1)) == 0)
            {
            a_index = a_index_ref;
            }
          else
          {
            a_index++;
          }
          tf_index += n_of_b;
        }
        a_index     = ((s_of_b<<1) + a_index) & N_MINUS_1;
        a_index_ref = a_index;
      }
      n_of_b >>= 1;
      s_of_b <<= 1;
    }

for (y=0;y<N;y++)
{
im[y]=magn_F(re[y],im[y]);
}
return y;
}


// Timer 1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{
// Place your code here
PORTC=sin_dacSRAM_MSB[dac_ptr];
PORTA=sin_dacSRAM_LSB[dac_ptr];
dac_ptr++;
//TIFR=TIFR | 0x10;
}

#define ADC_VREF_TYPE 0x00

// ADC interrupt service routine
interrupt [ADC_INT] void adc_isr(void)
{
unsigned int adc_data;
// Read the AD conversion result
adc_data=ADCW;
ADCSRA|=0x40; //start ad conversion
// Place your code here
if (cap_ptr<256)
{
x_n_re[cap_ptr]=adc_data;
cap_ptr++;
}

}


void main(void)
{
// Declare your local variables here

// Crystal Oscillator division factor: 1
#pragma optsize-
CLKPR=0x80;
CLKPR=0x00;
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
```

```
#endif

// Input/Output Ports initialization
// Port A initialization LSPORT
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
// Control Amplitude
PORTA=0x00;
DDRA=0xFF;

// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
DDRB=0x00;

// Port C initialization MSPORT
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
// Control Waveform
PORTC=0x00;
DDRC=0xFF;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Port E initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTE=0x00;
DDRE=0x00;

// Port F initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTF=0x00;
DDRF=0x00;

// Port G initialization
// Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State5=T State4=T State3=T State2=T State1=T State0=T
PORTG=0x00;
DDRG=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0A output: Disconnected
// OC0B output: Disconnected
TCCR0A=0x00;
TCCR0B=0x00;
TCNT0=0x00;
OCR0A=0x00;
OCR0B=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 14745,600 kHz
// Mode: CTC top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
// OC1C output: Discon.
```

```
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x09;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x73;
OCR1BH=0x00;
OCR1BL=0x00;
OCR1CH=0x00;
OCR1CL=0x00;



// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2A output: Disconnected
// OC2B output: Disconnected
ASSR=0x00;
TCCR2A=0x00;
TCCR2B=0x00;
TCNT2=0x00;
OCR2A=0x00;
OCR2B=0x00;

// Timer/Counter 3 initialization
// Clock source: System Clock
// Clock value: Timer 3 Stopped
// Mode: Normal top=FFFFh
// Noise Canceler: Off
// Input Capture on Falling Edge
// OC3A output: Discon.
// OC3B output: Discon.
// OC3C output: Discon.
// Timer 3 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR3A=0x00;
TCCR3B=0x00;
TCNT3H=0x00;
TCNT3L=0x00;
ICR3H=0x00;
ICR3L=0x00;
OCR3AH=0x00;
OCR3AL=0x00;
OCR3BH=0x00;
OCR3BL=0x00;
OCR3CH=0x00;
OCR3CL=0x00;

// Timer/Counter 4 initialization
// Clock source: System Clock
// Clock value: Timer 4 Stopped
// Mode: Normal top=FFFFh
```

```
// OC4A output: Discon.
// OC4B output: Discon.
// OC4C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 4 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR4A=0x00;
TCCR4B=0x00;
TCNT4H=0x00;
TCNT4L=0x00;
ICR4H=0x00;
ICR4L=0x00;
OCR4AH=0x00;
OCR4AL=0x00;
OCR4BH=0x00;
OCR4BL=0x00;
OCR4CH=0x00;
OCR4CL=0x00;

// Timer/Counter 5 initialization
// Clock source: System Clock
// Clock value: Timer 5 Stopped
// Mode: Normal top=FFFFh
// OC5A output: Discon.
// OC5B output: Discon.
// OC5C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 5 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR5A=0x00;
TCCR5B=0x00;
TCNT5H=0x00;
TCNT5L=0x00;
ICR5H=0x00;
ICR5L=0x00;
OCR5AH=0x00;
OCR5AL=0x00;
OCR5BH=0x00;
OCR5BL=0x00;
OCR5CH=0x00;
OCR5CL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
// INT3: Off
// INT4: Off
// INT5: Off
// INT6: Off
// INT7: Off
EICRA=0x00;
EICRB=0x00;
EIMSK=0x00;
// PCINT0 interrupt: Off
// PCINT1 interrupt: Off
// PCINT2 interrupt: Off
// PCINT3 interrupt: Off
```

```
// PCINT4 interrupt: Off
// PCINT5 interrupt: Off
// PCINT6 interrupt: Off
// PCINT7 interrupt: Off
// PCINT8 interrupt: Off
// PCINT9 interrupt: Off
// PCINT10 interrupt: Off
// PCINT11 interrupt: Off
// PCINT12 interrupt: Off
// PCINT13 interrupt: Off
// PCINT14 interrupt: Off
// PCINT15 interrupt: Off
// PCINT16 interrupt: Off
// PCINT17 interrupt: Off
// PCINT18 interrupt: Off
// PCINT19 interrupt: Off
// PCINT20 interrupt: Off
// PCINT21 interrupt: Off
// PCINT22 interrupt: Off
// PCINT23 interrupt: Off
PCMSK0=0x00;
PCMSK1=0x00;
PCMSK2=0x00;
PCICR=0x00;

// Timer/Counter 0 Interrupt(s) initialization
TIMSK0=0x00;
// Timer/Counter 1 Interrupt(s) initialization
TIMSK1=0x02;
// Timer/Counter 2 Interrupt(s) initialization
TIMSK2=0x00;
// Timer/Counter 3 Interrupt(s) initialization
TIMSK3=0x00;
// Timer/Counter 4 Interrupt(s) initialization
TIMSK4=0x00;
// Timer/Counter 5 Interrupt(s) initialization
TIMSK5=0x00;

// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 115200
UCSR0A=0x00;
UCSR0B=0x18;
UCSR0C=0x06;
UBRR0H=0x00;
UBRR0L=0x07;


// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
ADCSRB=0x00;

// ADC initialization
// ADC Clock frequency: 921,600 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: Free Running
// Digital input buffers on ADC0: On, ADC1: On, ADC2: On, ADC3: On
// ADC4: On, ADC5: On, ADC6: On, ADC7: On
DIDR0=0x00;
// Digital input buffers on ADC8: On, ADC9: On, ADC10: On, ADC11: On
// ADC12: On, ADC13: On, ADC14: On, ADC15: On
```

```
DIDR2=0x00;
/*
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0xAC;  //10101100
ADCSRB&=0xF8;   */

ADMUX=0x40;
ADCSRA=0x00;
ADCSRA|=0x20; //free running mode enable
ADCSRA|=0x80; //adc enable
ADCSRA|=0x08; //enable adc interrupt
ADCSRA|=0x05; //prescaler xtal/32
ADCSRB=0x00;


for (i=0;i<256;i++)
      {
    sin_dacSRAM_MSB[i]=sin_dacD[i]/0x100;
    sin_dacSRAM_LSB[i]=sin_dacD[i]&0xFF;
     }

// Global enable interrupts
#asm("sei")
ADCSRA|=0x40; //start ad conversion

while (1)
       {

       cmd=getchar();
       if ((cmd=='a')||(cmd=='A'))
           {
           for(y=0;y<256;y++)
                 {
                 sin_dacSRAM_MSB[y]=sin_dacA[y]/0x100;
                 sin_dacSRAM_LSB[y]=sin_dacA[y]&0xFF;
                 }
           cap_ptr=256;
           while(dac_ptr>0);
           cap_ptr=0;
           while(cap_ptr<256);
           //sent to uart time domain data
           putchar('t');
           for(y=0; y<N;y++)
                 {
                 ftoa(x_n_re[y],4,str);
                 for(zkl=0;zkl<strlen(str)-1;zkl++)
                       {
                       if(str[zkl]=='.')
                             {
                             str[zkl]=',';
                             }
                       }
                       puts(str);
                       putchar(0x0d);
                 }
           putchar('$');
           }
       else if ((cmd=='b')||(cmd=='B'))
           {
           for(y=0;y<256;y++)
                 {
                 sin_dacSRAM_MSB[y]=sin_dacB[y]/0x100;
                 sin_dacSRAM_LSB[y]=sin_dacB[y]&0xFF;
                 }
           cap_ptr=256;
           while(dac_ptr>0);
```

```
        cap_ptr=0;
        while(cap_ptr<256);
        //sent to uart time domain data
        putchar('t');
        for(y=0; y<N;y++)
                {
                ftoa(x_n_re[y],4,str);
                for(zkl=0;zkl<strlen(str)-1;zkl++)
                        {
                        if(str[zkl]=='.')
                                {
                                str[zkl]=',';
                                }
                        }
                        puts(str);
                        putchar(0x0d);
                }
        putchar('$');
        }
    else if ((cmd=='c')||(cmd=='C'))
        {
        for(y=0;y<256;y++)
                {
                sin_dacSRAM_MSB[y]=sin_dacC[y]/0x100;
                sin_dacSRAM_LSB[y]=sin_dacC[y]&0xFF;
                }
        cap_ptr=256;
        while(dac_ptr>0);
        cap_ptr=0;
        while(cap_ptr<256);
        //sent to uart time domain data
        putchar('t');
        for(y=0; y<N;y++)
                {
                ftoa(x_n_re[y],4,str);
                for(zkl=0;zkl<strlen(str)-1;zkl++)
                        {
                        if(str[zkl]=='.')
                                {
                                str[zkl]=',';
                                }
                        }
                        puts(str);
                        putchar(0x0d);
                }
        putchar('$');
        }
    else if ((cmd=='d')||(cmd=='D'))
        {
        for(y=0;y<256;y++)
                {
                sin_dacSRAM_MSB[y]=sin_dacD[y]/0x100;
                sin_dacSRAM_LSB[y]=sin_dacD[y]&0xFF;
                }
        cap_ptr=256;
        while(dac_ptr>0);
        cap_ptr=0;
        while(cap_ptr<256);
        //sent to uart time domain data
        putchar('t');
        for(y=0; y<N;y++)
                {
                ftoa(x_n_re[y],4,str);
                for(zkl=0;zkl<strlen(str)-1;zkl++)
                        {
                        if(str[zkl]=='.')
```

```
                    {
                    str[zkl]=',';
                    }
                }
                puts(str);
                putchar(0x0d);
            }
        putchar('$');
        }
    else if ((cmd=='s')||(cmd=='S'))
        {
        for(i=0;i<4;i++)
            {
            if (i==0)
                {
                for(y=0;y<256;y++)
                    {
                    sin_dacSRAM_MSB[y]=sin_dacA[y]/0x100;
                    sin_dacSRAM_LSB[y]=sin_dacA[y]&0xFF;
                    }
                }
            else if (i==1)
                {
                for(y=0;y<256;y++)
                    {
                    sin_dacSRAM_MSB[y]=sin_dacB[y]/0x100;
                    sin_dacSRAM_LSB[y]=sin_dacB[y]&0xFF;
                    }
                }
            else if (i==2)
                {
                for(y=0;y<256;y++)
                    {
                    sin_dacSRAM_MSB[y]=sin_dacC[y]/0x100;
                    sin_dacSRAM_LSB[y]=sin_dacC[y]&0xFF;
                    }
                }
            else if (i==3)
                {
                for(y=0;y<256;y++)
                    {
                    sin_dacSRAM_MSB[y]=sin_dacD[y]/0x100;
                    sin_dacSRAM_LSB[y]=sin_dacD[y]&0xFF;
                    }
                }


            cap_ptr=256;
            while(dac_ptr>0);
            cap_ptr=0;
            while(cap_ptr<256);
            //sent to uart time domain data
            putchar('t');

            for(y=0; y<N;y++)
                {

                ftoa(x_n_re[y],4,str);
                for(zkl=0;zkl<strlen(str)-1;zkl++)
                    {
                    if(str[zkl]=='.')
                        {
                        str[zkl]=',';
                        }
                    }
                    puts(str);
```

```
                        putchar(0x0d);
                    }
                putchar('$');
                } //for 4 tables
            } //if cmd='s'
   for (i=0;i<255;i++)
     {
  //   i=0xFF;
     PORTA=i;
     itoa(i,strr);
     puts(strr);
     putchar(0x0D);
     delay_ms(500);

     }
    // Place your code here
    //putsf("*");
    if(cap_ptr>=256)
    {
  //  #asm("cli")
    //sent to uart time domain data
    putchar('t');

    for(y=0; y<N;y++)
     {

     ftoa(x_n_re[y],4,str);
     for(zkl=0;zkl<strlen(str)-1;zkl++)
       {
       if(str[zkl]=='.'){str[zkl]=',';};
       }
     puts(str);
     putchar(0x0d);
     }
      putchar('$');



   //forward_FFT_F(x_n_re, x_n_im);
    //sent to uart frequency domain data
    putchar('f');
    for(y=5; y<N;y++)
     {
     ftoa(x_n_im[y],4,str);
     for(zkl=0;zkl<strlen(str)-1;zkl++)
       {
       if(str[zkl]=='.'){str[zkl]=',';};
       }
     puts(str);
     putchar(0x0d);
     }
     putchar('$');
    //start new capture
   // #asm("sei")
    cap_ptr=0;

    }

    };
}
```