ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

# ΒΑΤΟΜΕΤΡΟ RF ΑΠΟ 1KHz ΈΩΣ 500MHz


## Πτυχιακή Εργασία


## Φωτεινός Κώστας - Μαρτινούτσι Δημήτρης

**Υπεύθυνος Καθηγητής**
**Λαζαρίδης Παύλος**


2008

Περιεχόμενα

# Εισαγωγή

Τα βατόμετρα και γενικά η ισχύς και η μέτρηση της αποτελούν πολύ μεγάλο κεφαλαίο των ηλεκτρονικών. Η εργασία αυτή έχει σαν αντικείμενο την μέτρηση watt στο πεδίο των RF συχνοτήτων.

Αποτελείται λοιπόν από ένα σχετικά απλό στην κατασκευή και στη λειτουργία βαρόμετρο που μετρά την ισχύ σημάτων από 1KHz έως 500MHz. Επιπλέον τα κείμενα αναλύουν το πώς λειτουργεί ο μετρητής πως τον κατασκευάσαμε και τι δυσκολίες αντιμετωπίσαμε κατά την κατασκευή του.

Στη συνεχεία , επειδή ο μετρητής λειτουργεί με μικροελεγκτή παρουσιάζεται ο κώδικας που χρησιμοποιήθηκε για τον προγραμματισμό του συνοδευόμενος από λίγες και κατανοητές επεξηγήσεις.

Επειδή στα όργανα μέτρησης η ακρίβεια είναι το Α και το Ω έγιναν πολλές μετρήσεις με σκοπό να προσδιοριστεί ακριβώς αυτό. Ακόμα προσφέρεται η δυνατότητα παρουσίασης των αποτελεσμάτων σε όλα τα μεγέθη μέτρησης.(dB, dBm, watt, , RMS)

Τέλος για να είναι ποιο κατανοητά τα αποτελέσματα των μετρήσεων έχουν συμπυκνωθεί σε γραφήματα και πίνακες.

## Τεχνικά Χαρακτηρίστηκα

| | |
|---|---|
| Κάλυψη συχνοτήτων: | 1 KHz έως 500 MHz (βαθμονομημένη)<br>1 KHz έως 1000 MHz (μη βαθμονομημένη. μόνον για σχετικές μετρήσεις ισχύος) |
| Ονομαστική εμπέδηση εισόδου: | 50 Ω |
| Εύρος ισχύος εισόδου: | -60 dBm έως +30 dBm (1 nanowatt έως 1 watt) |
| Δυναμικό εύρος: | 90 dB με περίβλημα θωρακισμένο σε RF. |
| Ανάλυση: | 0,1 dBm (1 dBm σε ραβδόγραμμα) |
| Απώλειες επιστροφής εισόδου:<br><br><br><br>SWR εισόδου: | 300 KHz: -35 dB<br><br>100 MHz: -27 dB<br><br>500 MHz: -25 dB<br><br>300 KHz: 1,036<br><br>100 MHz: 1,094<br><br>500 MHz: 1,12 |
| Ακρίβεια πριν την βαθμονόμηση: | ± 1 dB από 1 MHZ έως 450 MHZ. |
| Μετά την βαθμονόμηση: | ±0,2 dB σε κάθε βαθμονομημένη συχνότητα. |
| Μέτρηση τάσης DC: | 0 έως 20 V. |
| Ανάλυση τάσης DC: | 20 mV |
| Ακρίβεια τάσης DC μετά την βαθμονόμηση: | ±20 mV |
| Τροφοδοσία: | 9 έως 20 VDC |
| Κατανάλωση ρεύματος: | χωρίς φωτισμό του LCD: 30 mA, με φωτισμό του LCD: 120mA. |

Ο συγκεκριμένος μετρητής ισχύος ('βατόμετρο'), χρησιμοποιεί για την μέτρηση της ισχύος ένα AD8307. Η βαθμίδα εισόδου του συγκεκριμένου ολοκληρωμένου διαθέτει αντιστάθμιση συχνότητας και είναι βελτιστοποιημένη όσον αφορά τα ανακλώμενα, ώστε να παρουσιάζει πολύ καλές τιμές SWR σε ένα μεγάλο εύρος συχνοτήτων.

Ένας ήδη προγραμματισμένος μικροελεγκτής τύπου PIC16F876 με ενσωματωμένους αναλογικό-ψηφιακούς μετατροπείς 10 ψηφίων, αναλαμβάνει την μετατροπή των αναλογικών τάσεων στην έξοδο του AD8307 σε ψηφιακές τιμές. Στην συνέχεια ένα σύνολο πινάκων αναφοράς, αναλαμβάνει την μετατροπή των τιμών dBm σε τάση RF και ισχύ RF (watts). Η απεικόνιση όλων τιμών συμπεριλαμβανομένου και ενός ραβδογράμματος γίνεται πάνω σε μία μεγάλη, φωτιζόμενη οθόνη υγρών κρυστάλλων 2 γραμμών των 20 χαρακτήρων, ενώ υπάρχει και ένα βολτόμετρο DC με δυνατότητα αποθήκευσης μέγιστης και ελάχιστης τιμής καθώς και πολλά άλλα χαρακτηριστικά .

## Η μονάδα decibel milli watt (dBm)

Όταν μιλάμε για ραδιοσυχνότητες (RF), τα 0 dBm αντιστοιχούν σε 1 milliwatt στα 50 Ω, ενώ αντίστοιχα τα 0 dBW αντιστοιχούν σε 1 watt στην ίδια εμπέδηση.

Σύμφωνα λοιπόν με τα παραπάνω ισχύουν οι παρακάτω αντιστοιχίες: + 10 dBm = 10 mW, +20 dBm = 100 mW, +30 dBm = 0 dBm =1 W, K.O.K.

Στο ραδιοφωνικό χώρο, ο όρος dBm χρησιμοποιείται στις διαδικασίες ανάπτυξης, ετιισκευής, υποστήριξης, καθώς και από τους ραδιοερασιτέχνες για την περιγραφή της (σχετικής) στάθμης ισχύος RF. Πολλοί μηχανικοί είναι εξοικειωμένοι με την μονάδα dBm, άλλοι προτιμούν τα 'watt', ενώ κάποιοι άλλοι προτιμούν να αναφέρονται σε 'τάση RMS'. Η συγκεκριμένη λοιπόν οθόνη απεικονίζει και τις τρεις μονάδες ταυτόχρονα .

## Λίγα λόγια για το AD8307

Το AD8307 είναι ένας μονολιθικός λογαριθμικός ενισχυτής της Analog Devices.
Παραθέτουμε στο Σχήμα 1 το σχηματικό διάγραμμα αυτού του πραγματικά πολύ επιτυχημένου ολοκληρωμένου. Το AD8307 είναι ένα σχετικά φτηνό εξάρτημα.
Η έκδοση Dil του AD8307 είναι πιο εύκολη στο κόλλημα και η χρήση του είναι πιο εύκολη σε σχέση με το SMD, το μεγάλο μήκος των ακροδεκτών το καθιστούν ουσιαστικά άχρηστο σε συχνότητες μεγαλύτερες από περίπου 100 MHz. Η έκδοση SMD μπορεί να χρησιμοποιηθεί μέχρι περίπου τα 500 MHz.

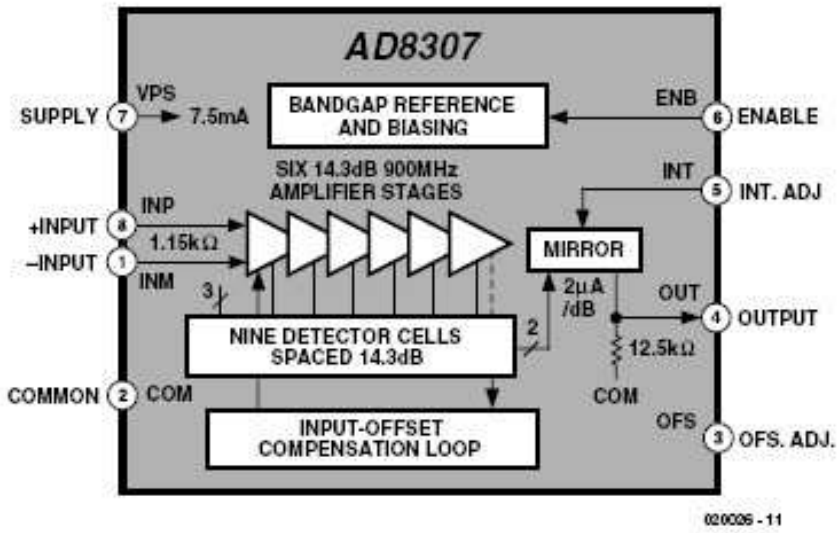
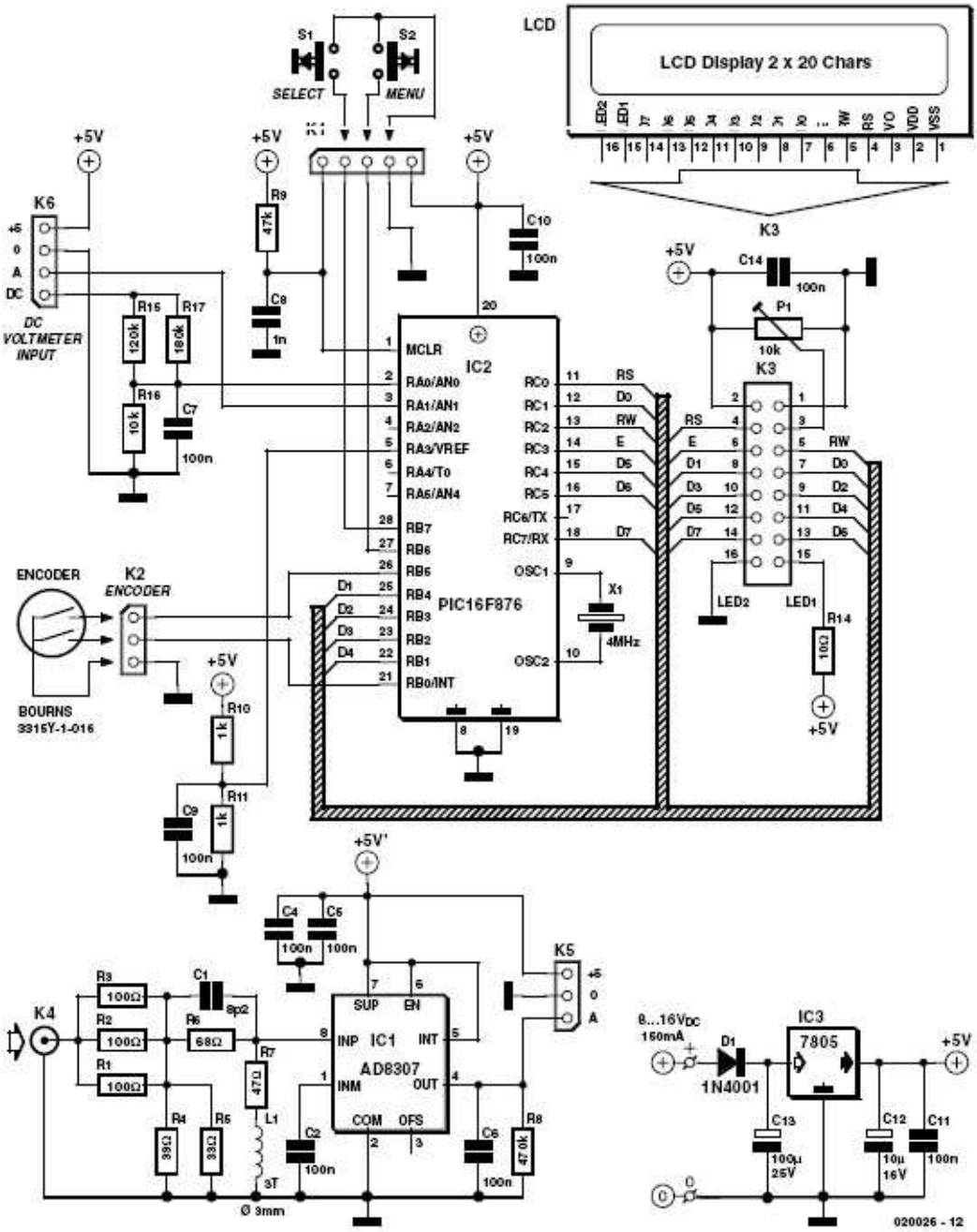



Figure 1. Block diagram of the AD8307 (courtesy Analog Devices).

Για να υπάρχει ακρίβεια, σε συχνότητες μεγαλύτερες από 300 MHZ η ισχύς του σήματος εισόδου δεν πρέπει να υπερβαίνει τα +20 dBm (100 mW). Η συγκεκριμένη μάλιστα αδυναμία του AD8307 αναφέρεται και στην βιβλιογραφία του. Το πρόβλημα βέβαια -εφ' όσον το γνωρίζει κανείς- δεν είναι ιδιαίτερα σοβαρό. Το μόνο που χρειάζεται είναι να παρεμβάλλεται ο κατάλληλος εξασθενητής, ώστε η πραγματική ισχύς RF που εφαρμόζεται στο ολοκληρωμένο να βρίσκεται σε μία στάθμη κάτω από τα 20 dBm, οπότε πλέον η ακρίβεια των ενδείξεων είναι εξασφαλισμένη.
Ο AD8307 έχει μια δυναμική περιοχή από τα 92dB με +- 3 dB μέχρι τα 100 MHz




Σχήμα 2. Κυκλωματικό διάγραμμα του μετρητή ισχύος RF. Τα δύο κύρια εξαρτήματα είναι το AD8307 και το PIC16F876 στην βαθμίδα ελέγχου.

## Περιγραφή του κυκλώματος

Στο Σχήμα 2 περιγράφεται το κυκλωματικό διάγραμμα του μετρητή ισχύος RF. Αποτελείται από τέσσερις βαθμίδες,

Ο μετατροπέας τάσης RF σχεδιάστηκε σαν ξεχωριστή μονάδα γύρω από το AD8307. Το ισοδύναμο της τάσης που εκπροσωπεί την εφαρμοζόμενη στον σύνδεσμο K5 ισχύ RF, εμφανίζεται στον K5 με την μορφή βηματικής στάθμης στην περιοχή από 0 έως

2.5 V. Το δικτύωμα των αντιστάσεων της εισόδου έχει σχεδιαστεί για εμπέδηση 50 Ω, η οποία αποτελεί αναμφισβήτητο πρότυπο στον χώρο των RF και. είναι σε θέση να εξυπηρετήσει στάθμες ισχύος μέχρι και 1 watt. Ο C 1 και το L 1 αντισταθμίζουν παρασιτικές χωρητικότητες ή αυτεπαγωγές, οπότε συμβάλλουν και στην βελτιστοποίηση του SWR εισόδου, στις υψηλότερες συχνότητες.

Η δεύτερη βαθμίδα είναι ο ψηφιακός ελεγκτής γύρω από το IC2 Το συγκεκριμένο 'μαύρο κουτί', εκτελεί ένα λογισμικό γραμμένο από τον κατασκευαστή του βατομέτρου το οποίο διαχειρίζεται τις παρακάτω λειτουργίες:

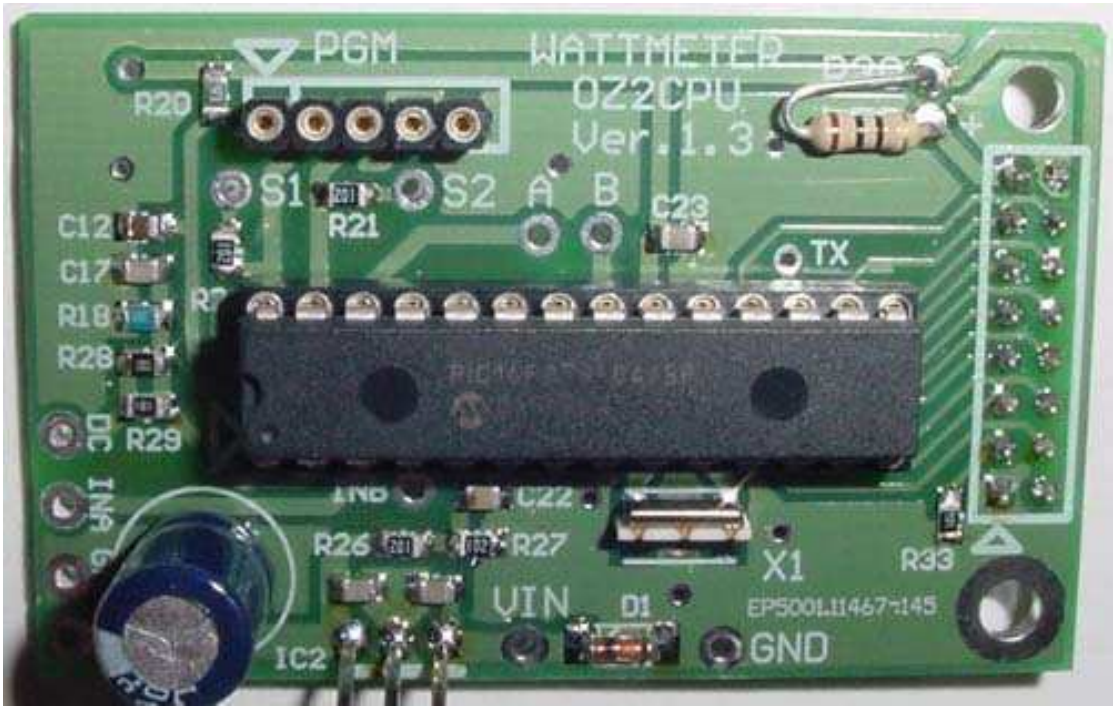
Α)επεξεργασία της εξόδου του AD8307 σε μία μορφή κατανοητή για τους περισσότερους χρήστες.

Β)ανάγνωση των πλήκτρων ελέγχου του χρήστη (πιεστικοί διακόπτες S1, S2 και ο περιστροφικός κωδικοποιητής στον K2),

Γ)οδήγηση της οθόνης LCD, παρέχοντας την δυνατότητα απεικόνισης μενού επιλογών, τιμών, κ.λ.π.

Με την έναρξη της τροφοδοσίας, το PIC επανατοποθετείται μέσω των R9-C8, ενώ ο χρονισμός του γίνεται μέσω ενός κεραμικού ταλαντωτή στα 4 MHz.
Η Τρίτη μονάδα είναι το LCD, το οποίο διαθέτει δύο γραμμές των 20 χαρακτήρων η κάθε μία ενώ η αντίθεση απεικόνισης ρυθμίζεται από το τρίμμερ προ ρύθμισης P1.

Η τέταρτη μονάδα αναφέρεται στο κύκλωμα τροφοδοσίας, το οποίο αναπτύσσεται γύρω από το IC3. Η σχεδίαση του είναι τελείως συμβατική και δεν νομίζουμε ότι χρήζει περαιτέρω μελέτης. Η ισχύς εισόδου είναι δυνατόν να ληφθεί από οποιοδήποτε μικρό τροφοδοτικό δικτύου που είναι σε θέση να παράσχει ρεύμα περίπου 150 mA, στα 8 έως 16 VDC.



## Λίγα λόγια για το PIC 16F876

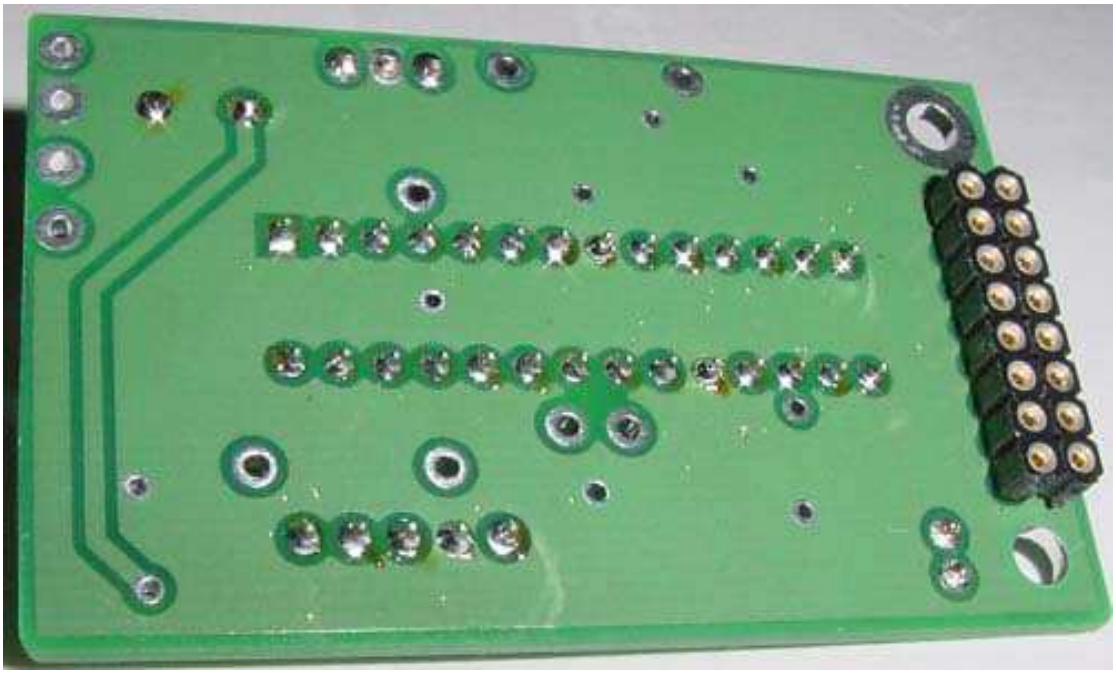
Οι απαιτήσεις όσον αφορά την ψηφιακή βαθμίδα της κατασκευής, περιελάμβαναν ένα φτηνό μικροελεγκτή με αναλογικό-ψηφιακό μετατροπέα (ADC) 1 0 ψηφίων, φτηνή και απλή διασύνδεση για τον προγραμματισμό καθώς και μνήμη για το λογισμικό τύπου Flash, ώστε να είναι εύκολη η ανάπτυξη και εκσφαλμάτωση του προγράμματος. Πέραν αυτών, θέλαμε και 4 ψηφιακές εισόδους, 2 για τους πιεστικούς διακόπτες και άλλες 2 για τον κωδικοποιητή, 7 εξόδους για την οθόνη υγρών κρυστάλλων σε κατάσταση 4 ψηφίων, ή 11 εξόδους για κατάσταση 8 ψηφίων, Οι μικροελεγκτές PIC 16F873 και 16F876 της Microchip, με μνήμη προγράμματος Flash 4 και 8 kwords αντίστοιχα, αποδείχθηκαν ιδανική επιλογή. Δεδομένου ότι η τιμή τους είναι πρακτικά η ίδια, προτιμήσαμε την έκδοση με τα 8 k μνήμης.

Το 16F876 διαθέτει 5 αναλογικές εισόδους με ανάλυση 10 ψηφίων, η οποία αντιστοιχεί σε ένα εύρος διακριτών τιμών από 0 έως 1023, όταν η τάση εισόδου μεταβάλλεται από 0 έως 5 V. Σε όλο το εύρος λειτουργίας, το συνεχές σήμα από το AD8307 κυμαίνεται στην περιοχή από 0 έως 2.5 V. Για πλήρη ψηφιακή ανάλυση, ο μετατροπέας ADC που περιλαμβάνεται στο PIC θα μπορούσε να χρησιμοποιήσει ως τάση πλήρους κλίμακας μία εξωτερική θετική τάση αναφοράς. Οι R 10 και R 11 λοιπόν, αναλαμβάνουν να δημιουργήσουν από την τάση τροφοδοσίας των 5 V μία τάση αναφοράς 2.5 V. Η συγκεκριμένη τάση δεν είναι κρίσιμη και στην περίπτωση που παρουσιάζεται κάποια απόκλιση, θα διορθωθεί στην συνέχεια από το λογισμικό μέσω του σημείου μετατόπισης των 0 dBm.

Ο Johann Aichinger σχεδίασε για το συγκεκριμένο PIC ένα πολύ καλό και απλό προγραμματιστή, τον οποίο καλεί PROPIC.

Υπάρχουν πολλοί άλλοι προγραμματιστές PIC που υποστηρίζουν το PIC 16F876, συμπεριλαμβανομένου του 'IC-PROG' του Bonny Gijzen. Ο συγκεκριμένος προγραμματιστής είναι και αυτός καλός και μάλιστα υποστηρίζει σχεδόν οποιοδήποτε IC μπορεί να προγραμματιστεί. Βρίσκεται στην διεύθυνση
www.ic-prog.com

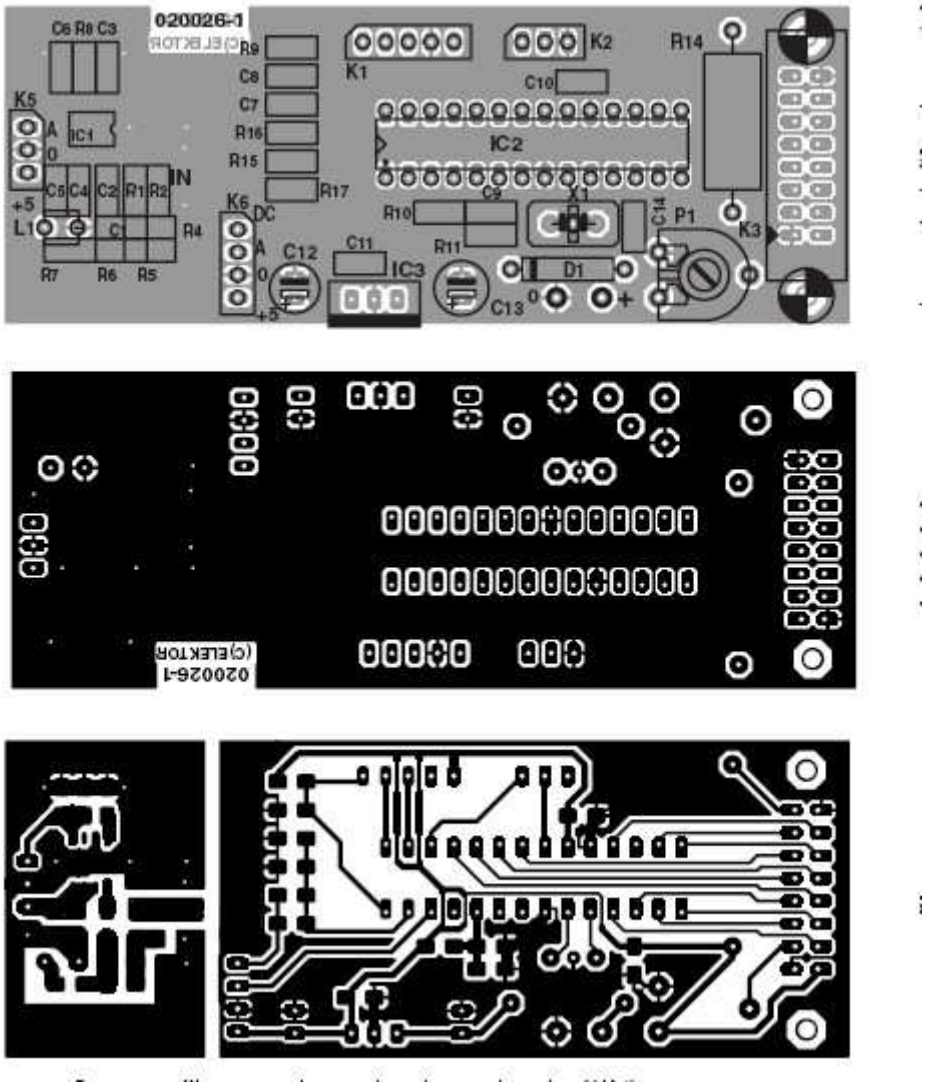
Το λογισμικό για την κατασκευή αναπτύχθηκε από τον Thomas Scherrer με την βοήθεια της εφαρμογής προγραμματισμού PIC 'MPLAB' της Microchip, σε συνδυασμό με ένα εξωτερικό μεταφραστή (compiler) C της Hi-Tech, καλούμενο PICC. Η διεύθυνση της Hi-Tech στο διαδίκτυο είναι www.htsoft.com



και από εκεί μπορεί κανείς να κατεβάσει δωρεάν την 'demo' έκδοση. Στην περίπτωση που κάποιος θελήσει να μεταφράσει 'εκ νέου το λογισμικό της κατασκευής, μπορεί εάν το επιθυμεί να τροποποιήσει και την εισαγωγική οθόνη έτσι ώστε να εμφανίζεται το όνομα του ή το ψευδώνυμο του. Τα αρχεία με τον πηγαίο κώδικα του λογισμικού για το βατόμετρο, διατίθενται δωρεάν από τον δικτυακό χώρο του περιοδικού Ελεκτορ.

Ένα όργανο σαν τον μετρητή ισχύος RF θα πρέπει να κατασκευαστεί με την δέουσα προσοχή όσον αφορά την μηχανική του θωράκιση, δεδομένου ότι ποτέ δεν πρόκειται να λειτουργήσει σωστά, εάν δεν δοθεί η απαραίτητη σημασία στην θωράκιση.

Για να αρχίσουμε λοιπόν, στο Σχήμα 3 έχουμε την επιμεταλλωμένη διπλής όψεως πλακέτα, η οποία κόβεται στα δύο για να χωρίσει η βαθμίδα ελέγχου από την πλακέτα εισόδου.




**Η πλακέτα εισόδου**

Η συγκεκριμένα πλακέτα (Σχήμα 4) περιλαμβάνει υλικά επιφανειακής στήριξης (SMD = Surface Mount Devices), δεδομένου ότι τα μικρά εξαρτήματα παρουσιάζουν εν γένει καλύτερη συμπεριφορά στις ψηλές συχνότητες.

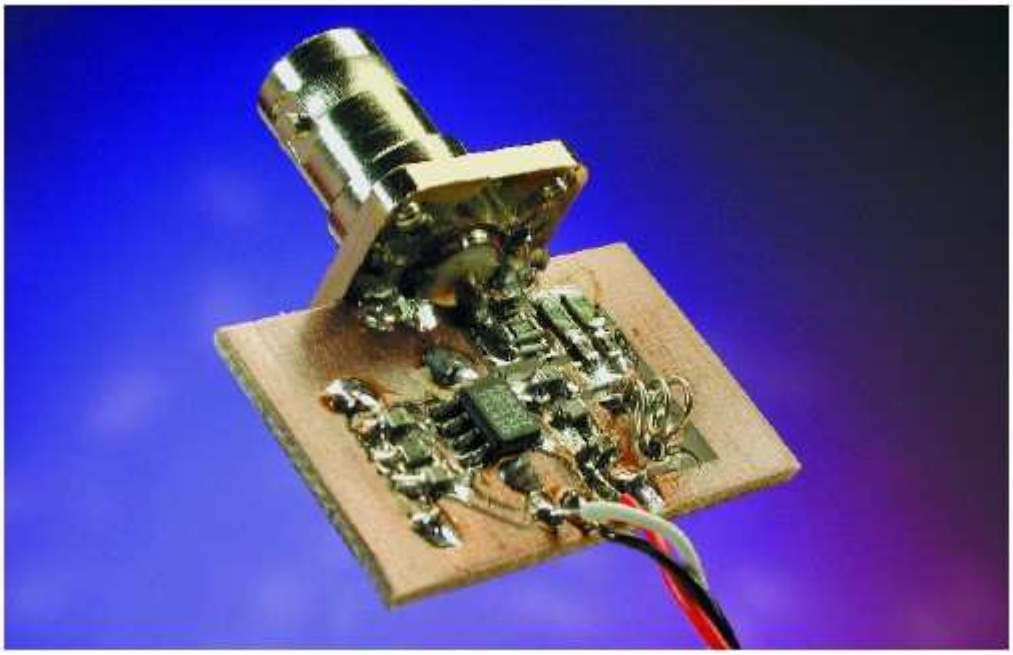


Figure 4. Input board with BNC flange socket attached by soldering along the edge.

Η πλακέτα εισόδου διαθέτει ειδικά σχεδιασμένα σχήματα, με άνετα ίχνη κόλλησης τα οποία επιτρέπουν την τοποθέτηση εξαρτημάτων SMD τόσο τύπου 1206 όσο και 0805.
Το πηνίο L 1 αποτελείται από 3 τυλίγματα επιβερνικωμένου χάλκινου σύρματος διαμέτρου 0,5 mm. Η εσωτερική διάμετρος του πηνίου είναι 3 mm και τα τυλίγματα έχουν μεταξύ τους απόσταση 0.5 mm.
Η πλακέτα εισόδου κολλιέται απ' ευθείας στην στεφάνη μίας υποδοχής BNC, με τέσσερις οπές στήριξης M2.5.

**Η πλακέτα του ελεγκτή**

Στο Σχήμα 5 απεικονίζεται ένα από τα πρώτα πρωτότυπα της εν λόγω πλακέτας.
Το ολοκληρωμένο είναι δυνατόν να ενημερωθεί με τυχόν νέο λογισμικό ακόμη και όταν είναι επάνω στην πλακέτα.

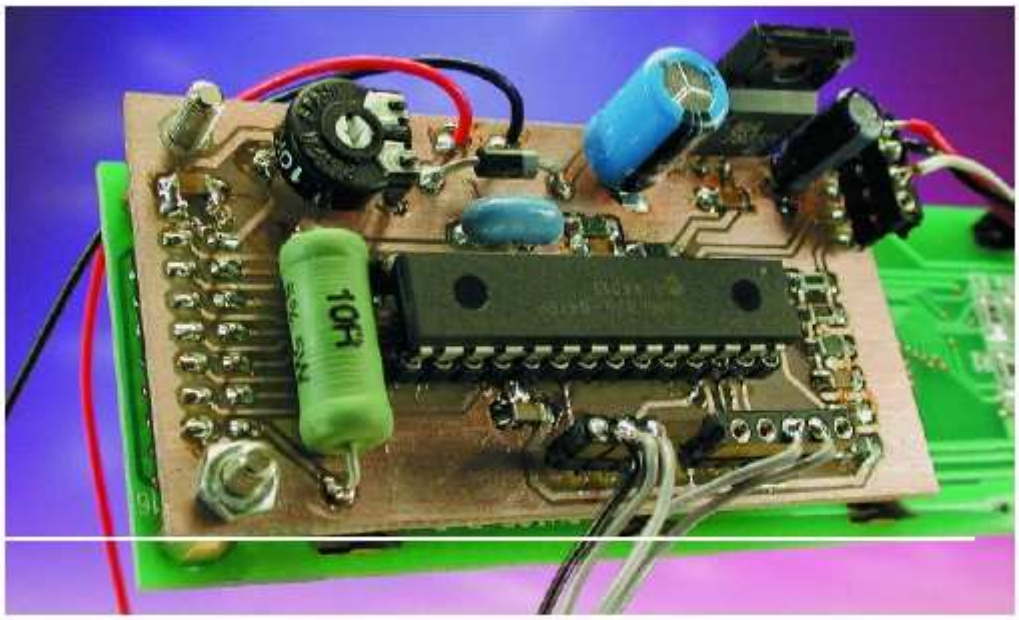



Figure 5. Controller board attached to back side of LCD module.

Η συχνότητα του κεραμικού ταλαντωτή είναι 4 MHZ, αλλά δεν είναι κρίσιμη. Εάν δεν θα να χρησιμοποιήσουμε την προαιρετική σειριακή έξοδο, (που δεν θα χρησιμοποιήσουμε) μπορούμε να χρησιμοποιήσουμε ένα ταλαντωτή τύπου 3 ακροδεκτών με ενσωματωμένους πυκνωτές. Αρκετά καλά πάντως θα κάνει την δουλειά και η τυπική έκδοση με 2 ακροδέκτες.

Η πλακέτα του ελεγκτή στερεώνεται στην πίσω πλευρά της μονάδας LCD, με την βοήθεια βίδας και παξιμαδιού μήκους 10 περίπου mm ή με βίδες στήριξης πλακετών.

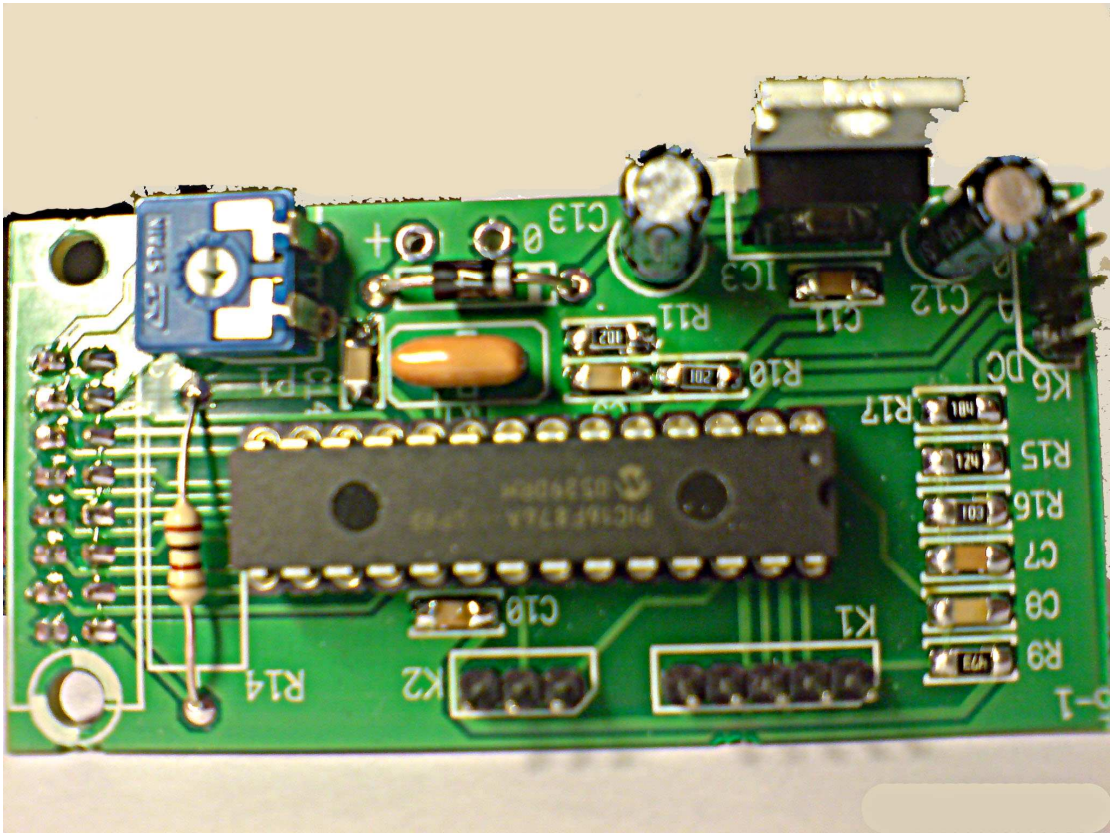
Η αντίσταση R 14 είναι στο 1 Watt και ελέγχει τον φωτισμό της οθόνης LCD. Με μία τιμή γύρω στα 10 Ω επιτυγχάνεται κανονικός φωτισμός της οθόνης, αλλά εάν προτιθέμεθα να τροφοδοτήσουμε την μονάδα με μπαταρίες καλό είναι η τιμή της R14 να ανέβει στα 20 Ω περίπου. Διαφορετικά, μπορούμε να τοποθετήσουμε ένα διακόπτη με την βοήθεια του οποίου θα έχουμε την δυνατότητα να διακόπτουμε τον φωτισμό της οθόνης όταν η μονάδα λειτουργεί με μπαταρίες.

Στην περίπτωση που η τάση εισόδου υπερβαίνει τα 10 V, ο σταθεροποιητής τάσης 7805 μάλλον θα θερμαίνεται κατά την λειτουργία. Μπορούμε είτε να τον τσακίσουμε λίγο για να στερεωθεί με μία βίδα M3 επάνω στο κουτί, είτε να τον εξοπλίσουμε με μια ψύκτρα. Μια μικρή ψύκτρα από αυτές που 'κουμπώνουν' είναι αρκετή.

**Στρεφόμενος κωδικοποιητής και διακόπτες**

Ο στρεφόμενος κωδικοποιητής είναι ένα σχετικά φτηνό εξάρτημα τύπου διπλών επαφών ο οποίος παράγει κώδικα Grey και χρησιμοποιείται σαν επιλογέας μεταξύ των διαφόρων δυνατοτήτων του μενού καθώς και για την αλλαγή διαφόρων ρυθμίσεων .

Οι δύο διακόπτες S1 και S2 πρέπει να είναι πιεστικού τύπου. Ο S1 εξυπηρετεί στην αποδοχή κάποιων επιλογών από τα μενού και ο S2 αποτελεί το πλήκτρο πρόσβασης στα μενού.




**Αρχική δοκιμή (σε επίπεδο υλικού)**

Στο σημείο αυτό και οι δύο πλακέτες είναι συναρμολογημένες και ετοιμες. Ο PIC, είναι κολλημένος επάνω στην πλακέτα χρειάζεται ένα τροφοδοτικό συνεχούς με μεταβαλλόμενη τάση για να υλοποιηθεί η δοκιμή που ακολουθεί.

Ξεκινάμε με την πλακέτα ελέγχου, η οποία περιλαμβάνει το 7805 και ενδεχομένως και το PIC. Η οθόνη και η πλακέτα εισόδου, παραμένουν για την ώρα εκτός.

Πριν προχωρήσουμε, θα χρειαστεί να βεβαιωθούμε ότι ο σταθεροποιητής 7805 λειτουργεί σωστά: τροφοδοτουμε την πλακέτα με 9 V και ελέγχουμε εάν υπάρχουν τα +5 V στο σημείο σύνδεσης της πλακέτας εισόδου κοντά στον C12, όπως επίσης θα πρέπει να μετρήσουμε +2.5 V στον ακροδέκτη 5 του IC1 και +5 V στον ακροδέκτη 1 του ίδιου ολοκληρωμένου.

Διακόπτουμε την τροφοδοσία, συνδέουμε την οθόνη και το PIC και τροφοδοτούμε εκ νέου. Στρίβουμε το P1 αντίθετα από την φορά κίνησης των δεικτών του ρολογιού, ώστε η οθόνη να έχει 0 V στον ακροδέκτη ρύθμισης 3. Με τον τρόπο αυτό παρουσιάζεται μέγιστη αντίθεση, οπότε μπορουμε να αντιληφθουμε εάν η οθόνη 'παίζει'. Στην συνέχεια ρυθμίζουμε το P1 ώστε σε σχέση με την γωνία θέασης να έχουμε επαρκή αντίθεση στην οθόνη, οπότε και εμφανίζεται το μήνυμα υποδοχής του μετρητή ισχύος RF . Αν ακουμπήσουμε το δάκτυλο στον ακροδέκτη εισόδου A θα δούμε την οθόνη να ανταποκρίνεται. ανεβοκατεβαίνει η ένδειξη του μετρητή.

Συνδέουμε στην συνέχεια την πλακέτα εισόδου με την κύρια πλακέτα, χρησιμοποιώντας καλώδιο όπως αυτό που χρησιμοποιείται για την μεταφορά σημάτων στα ηχοσυστήματα, ή στον υπολογιστή.

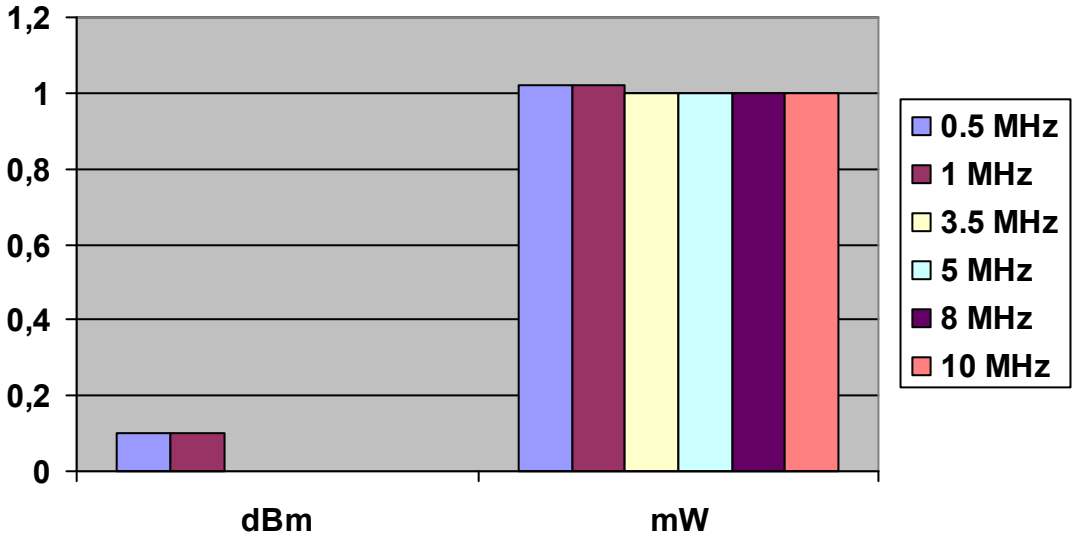
Τροφοδοτούμε ξανά το κύκλωμα και βλεπουμε ότι η κατανάλωση ρεύματος είναι μικρότερη από 150 mA.

Επειδή δεν διαθέτουμε γεννήτρια RF για να δοκιμάσουμε και να ρυθμίσουμε την κατασκευή, πήγαμε στην σχολή Ηλεκτρονικής στο εργαστήριο των κεραιών όπου και έγινε η ρύθμιση και η δοκιμή.

Αφού λοιπόν ολοκληρώσαμε με επιτυχία την αρχική αυτή δοκιμή στην συνέχεια πήραμε μετρήσεις για να προσδιορίσουμε την ακρίβεια του οργάνου και την περιοχή συχνοτήτων και ισχύων στην οποία αυτή είναι ικανοποιητική. Καλιμπράρουμε το όργανο, θέτοντας την στάθμη 0 dBm, αρχικά για τις συχνότητες: 3.5, 14, 145, 430, 440 MHz.

Ενδιάμεσες μετρήσεις από 0.5 MHz – 10 MHz

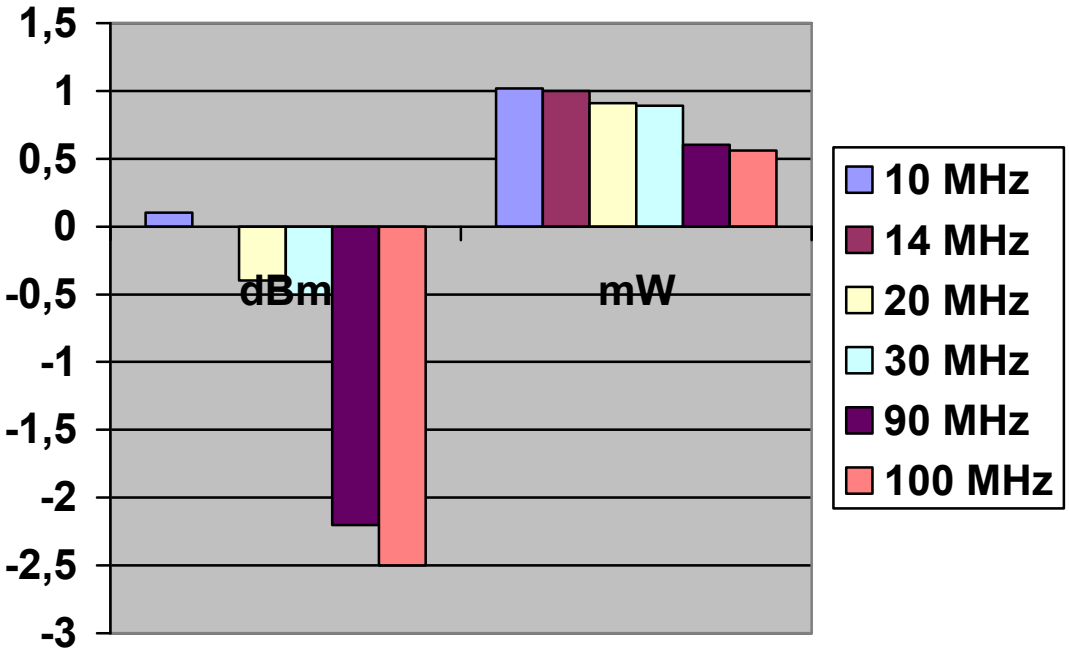
| LF | dBm | mW |
|---|---|---|
| 0.5 MHz | 0.1 | 1.02 |
| 1 MHz | 0.1 | 1.02 |
| 3.5 MHz | 0 | 1 |
| 5 MHz | 0 | 1 |
| 8 MHz | 0 | 1 |
| 10 MHz | 0 | 1 |




Ενδιάμεσες μετρήσεις από 10 MHz – 100 MHz

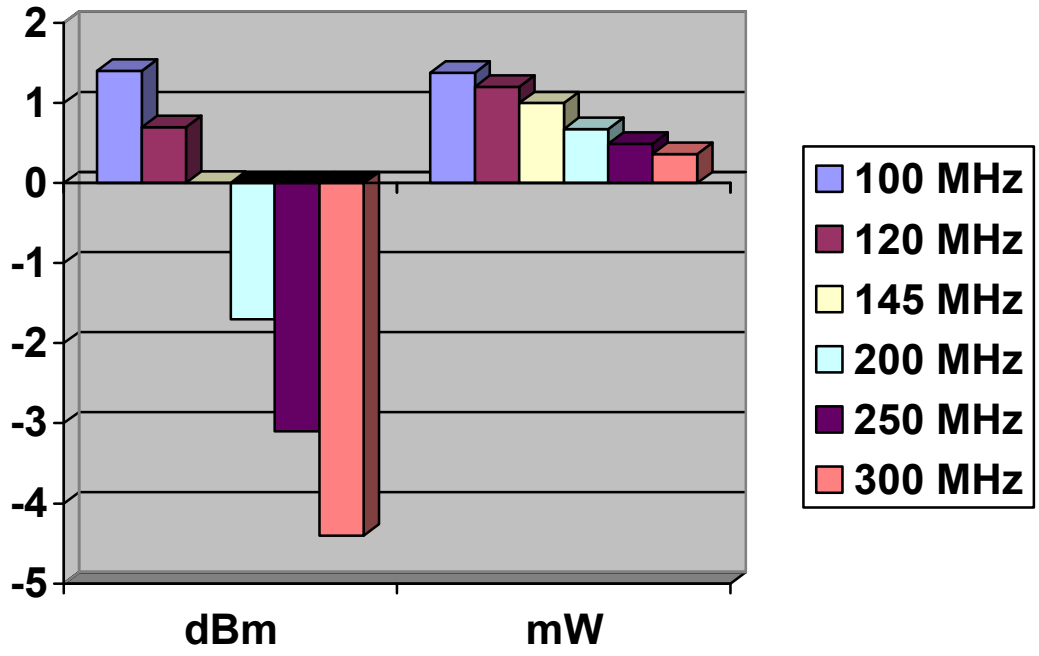
| HF | dBm | mW |
|---|---|---|
| 10 MHz | 0,1 | 1,02 |
| 14 MHz | 0 | 1 |
| 20 MHz | -0,4 | 0,912 |
| 30 MHz | -0,5 | 0,891 |

| | | |
|---|---|---|
| 90 MHz | -2,2 | 0,603 |
| 100 MHz | -2,2 | 0,562 |



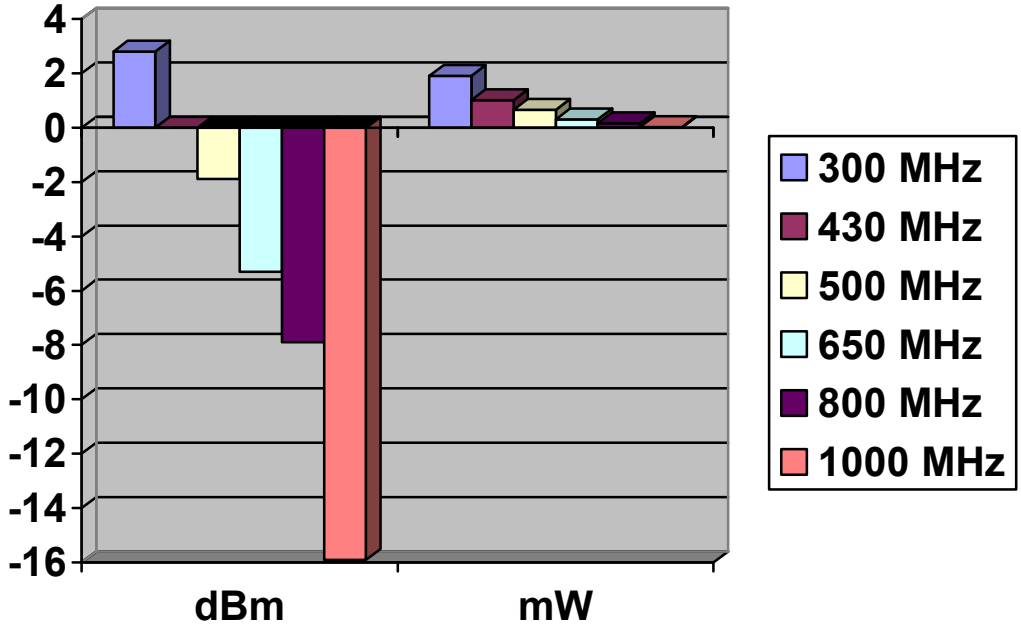

Ενδιάμεσες μετρήσεις από 100 MHz – 300 MHz

| VHF | dBm | mW |
|---|---|---|
| 100MHz | 1,4 | 1,38 |
| 120 MHz | 0,7 | 1,20 |
| 145 MHz | 0 | 1 |
| 200 MHz | -1,7 | 0,676 |
| 250 MHz | -3,1 | 0,490 |
| 300 MHz | -4,4 | 0,363 |

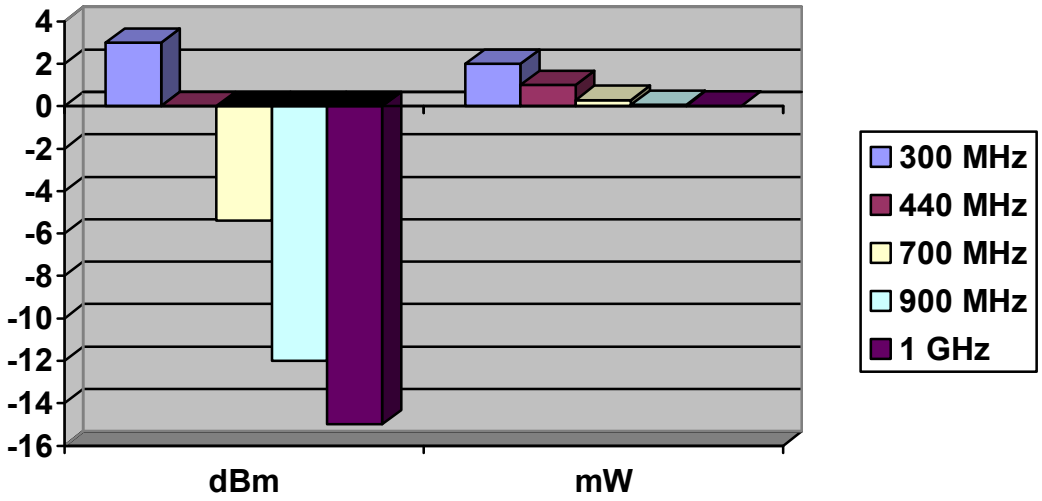



Ενδιάμεσες μετρήσεις από 300 MHz – 1000 MHz

| UHF | dBm | mW |
|---|---|---|
| 300 MHz | 2,8 | 1,91 |
| 430 MHz | 0 | 1 |
| 500 MHz | -1,9 | 0,646 |
| 650 MHz | -5,3 | 0,295 |
| 800 MHz | -7,9 | 0,192 |
| 1000 MHz | -15,9 | 0,025 |



| SHF | dBm | mW |
|---|---|---|
| 300 MHz | 3 | 2 |
| 440 MHz | 0 | 1 |
| 700 MHz | -5,4 | 0,288 |
| 900 MHz | -12 | 0,063 |
| 1 GHz | -15 | 0,026 |




**Συμπεράσματα**

Η κατασκευή μας δεν έχει πολύ μεγάλες αποκλίσεις λόγο ανοχών της κατασκευής ,οι μετρήσεις έγιναν σε συχνότητες που μπορούμε να μετρήσουμε για να έχουμε ένα ενδεικτικό της διακριτικής ικανότητας του Βατομέτρου.
Παρακάτω παραθέτουμε τις μετρήσεις που έχει κάνει ο εμπνευστής της κατασκευής

## Ακρίβεια στο σήμα εισόδου

| Input | 10Mhz | 100Mhz | 200Mhz | 300Mhz | 400Mhz | 500Mhz |
|---|---|---|---|---|---|---|
| +10 | +9.8 | +9.8 | +10.2 | +10.6 | +10.3 | +9.5 |
| 0 dBm | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| -10 | -10.1 | -10.0 | -10.0 | -9.9 | -10.2 | -10.4 |
| -20 | -20.4 | -20.1 | -19.8 | -19.7 | -20.3 | -20.9 |
| -30 | -30.4 | -29.9 | -29.8 | -29.9 | -30.5 | -31.5 |
| -40 | -40.4 | -40.0 | -40.4 | -40.3 | -40.6 | -42.2 |
| -50 | -50.3 | -50.1 | -51.4 | -50.5 | -50.1 | -52.2 |
| -60 | -58.8 | -58.3 | -58.2 | -58.7 | -58.3 | -60.3 |

Ημερομηνία . 21 Νοεμβριου 2001 OZ2CPU Thomas
Γεννητρια . Boonton 102D
Επιπλέον Πληροφοριες . με καλά ρυθμισμένο SWR το βατομετρο πάει μέχρι τα +30dBm(1W)αλλά η γεννήτρια του δεν έφτανε μέχρι εκεί οπότε το τεστ αυτό δεν είναι ακριβείς.

| Input | 10Mhz | 100Mhz | 200Mhz | 300Mhz | 400Mhz | 500Mhz |
|---|---|---|---|---|---|---|
| +9 | +8.8 | +8.8 | +9.2 | +9.6 | +9.2 | +8.5 |
| +8 | +7.8 | +7.7 | +8.1 | +8.5 | +8.4 | +7.7 |
| +7 | +6.8 | +6.7 | +7.0 | +7.4 | +7.4 | +6.8 |
| +6 | +5.8 | +5.7 | +5.9 | +6.3 | +6.3 | +5.9 |
| +5 | +4.8 | +4.7 | +4.9 | +5.2 | +5.3 | +5.1 |
| +4 | +3.9 | +3.8 | +3.8 | +4.1 | +4.2 | +4.2 |
| +3 | +3.0 | +2.9 | +2.9 | +3.1 | +3.2 | +3.2 |
| +2 | +2.0 | +1.9 | +1.9 | +2.0 | +2.1 | +2.1 |
| +1 | +1.0 | +1.0 | +0.9 | +1.0 | +1.1 | +1.1 |
| 0 dBm | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| -1 | -1.0 | -1.0 | -1.0 | -1.0 | -1.0 | -1.1 |
| -2 | -2.0 | -2.0 | -2.0 | -2.0 | -2.0 | -2.2 |
| -3 | -3.1 | -3.1 | -2.9 | -3.0 | -3.1 | -3.2 |
| -4 | -4.2 | -4.1 | -4.0 | -4.1 | -4.1 | -4.3 |
| -5 | -5.2 | -5.2 | -5.0 | -5.0 | -5.1 | -5.3 |
| -6 | -6.2 | -6.2 | -6.0 | -5.9 | -6.0 | -6.3 |
| -7 | -7.2 | -7.2 | -7.0 | -6.9 | -7.1 | -7.3 |
| -8 | -8.2 | -8.2 | -8.0 | -8.0 | -8.1 | -8.4 |
| -9 | -9.1 | -9.1 | -9.0 | -9.0 | -9.1 | -9.4 |

Το ίδιο τεστ με 1dB ανά βήμα χρησιμοποιώντας 1dB εξωτερικό εξασθένητη

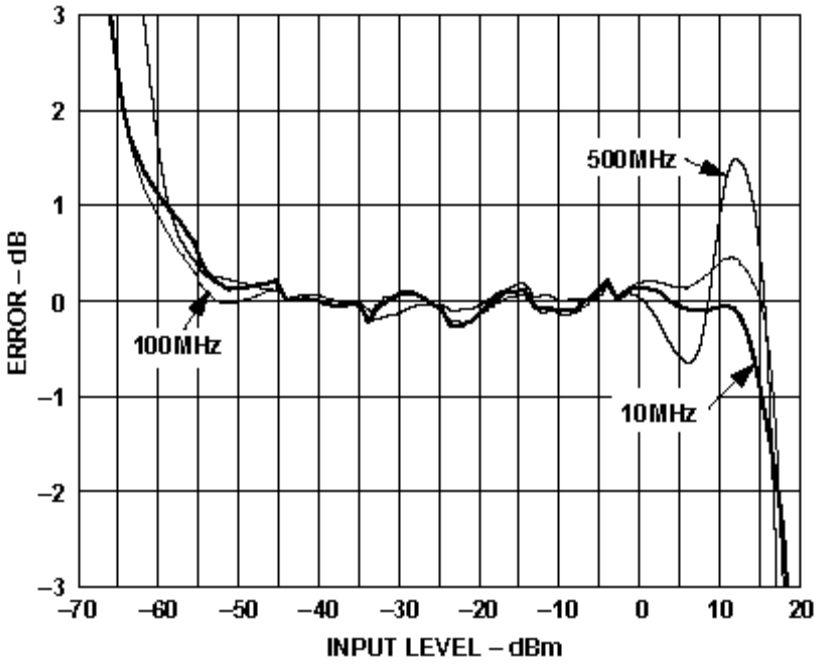



*Figure 12. Log Conformance vs. Input Level at 3 V Supply Using AD820 as Buffer, Gain = +2*

Το σήμα εισόδου χωρίς εξασθένιση. από την γραφική φαίνεται το σφάλμα dB στην έξοδο (V) του AD8307.Το σφάλμα του +1dB μπορεί να εμφανιστεί εάν το σήμα εισόδου είναι πάνω από +15dBm στο UHF

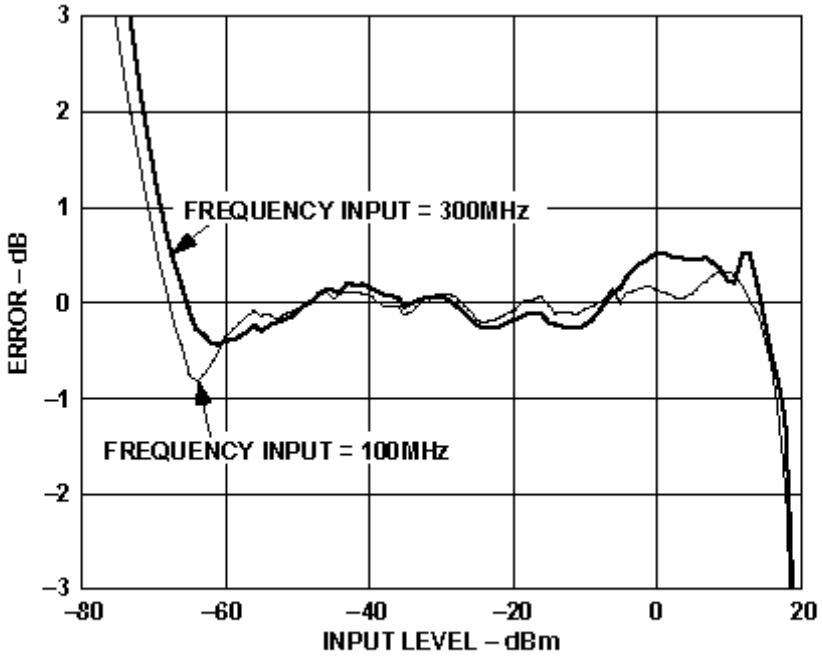



*Figure 3. Log Conformance vs. Input Level (dBm) @ 100 MHz, 300 MHz*

Το σήμα εισόδου χωρίς εξασθένιση. από την γραφική φαίνεται το σφάλμα dB στην έξοδο (V) του AD8307

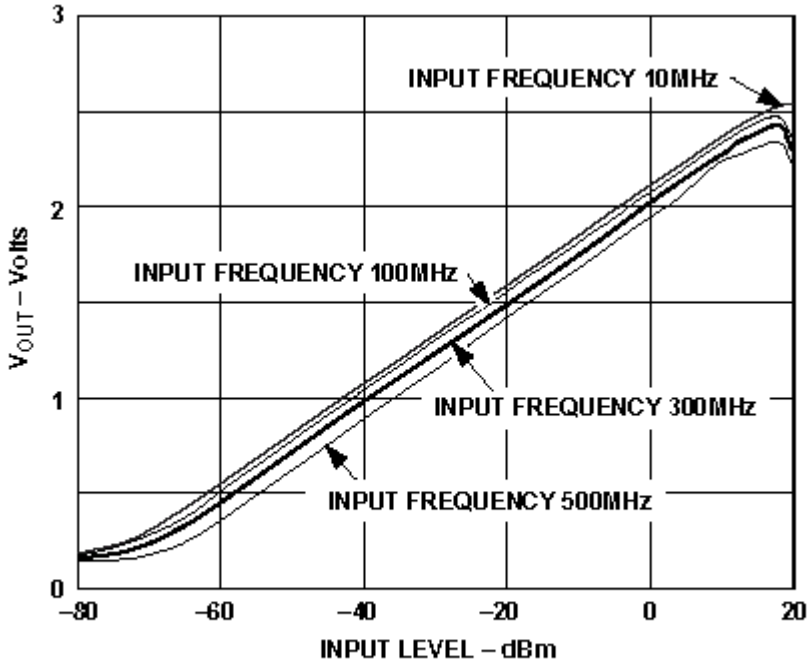


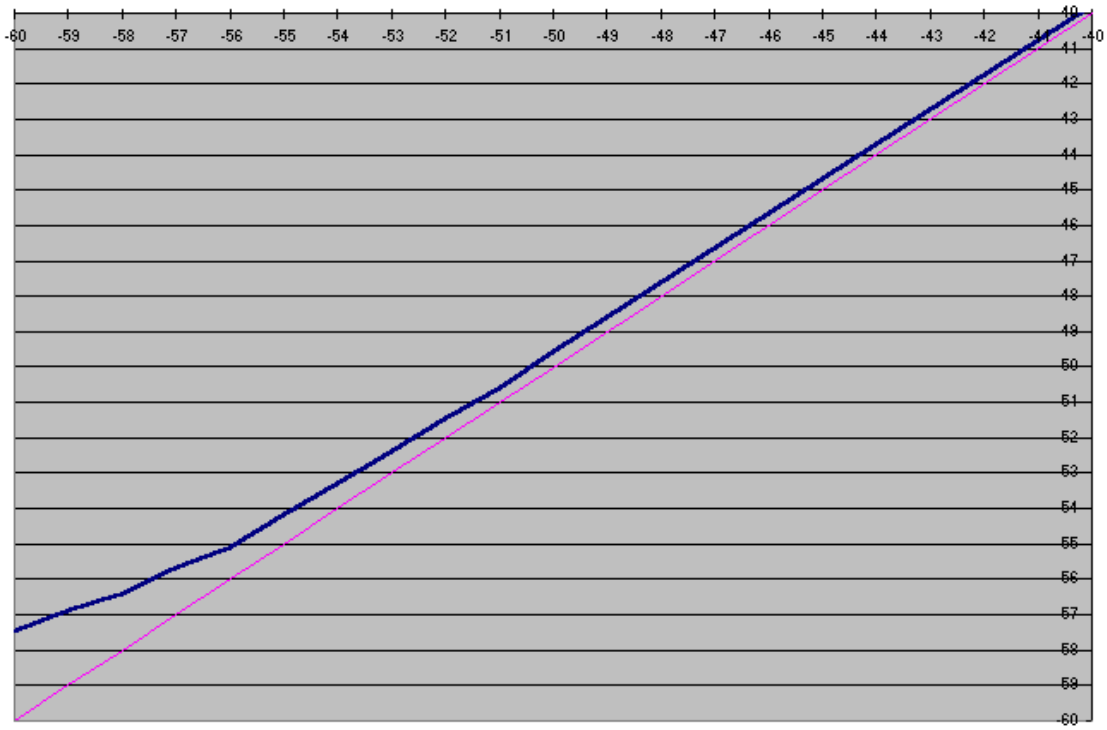

*Figure 5. $V_{OUT}$ vs. Input Level (dBm) at Various Frequencies*

Η αντιστάθμιση στης συχνότητες δεν είναι πολύ καλή

Η γραμμικοτητα της εισοδου -40 εως -60 dBm

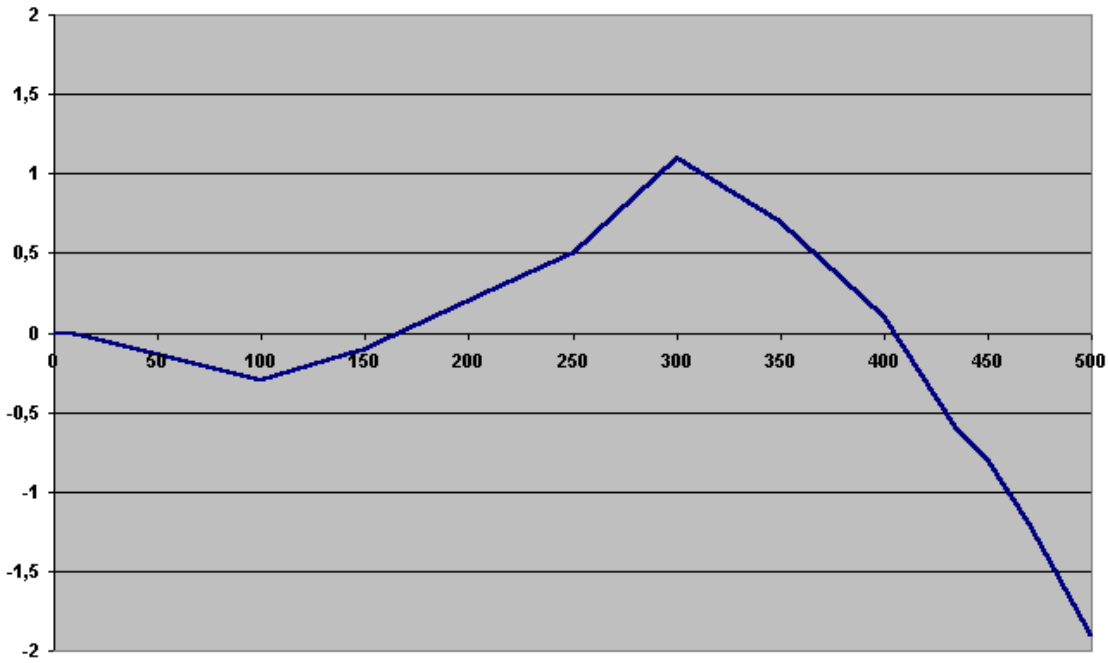
| | |
|---|---|
| -40 | -39,8 |
| -45 | -44,7 |
| -50 | -49,6 |
| -51 | -50,6 |
| -52 | -51,5 |
| -53 | -52,4 |
| -54 | -53,3 |
| -55 | -54,2 |
| -56 | -55,1 |
| -57 | -55,7 |
| -58 | -56,4 |
| -59 | -56,9 |
| -60 | -57,5 |
| | |
| 10 mhz 9mo | |

Μετρήθηκε από τον OZ9MO στα 10MHz

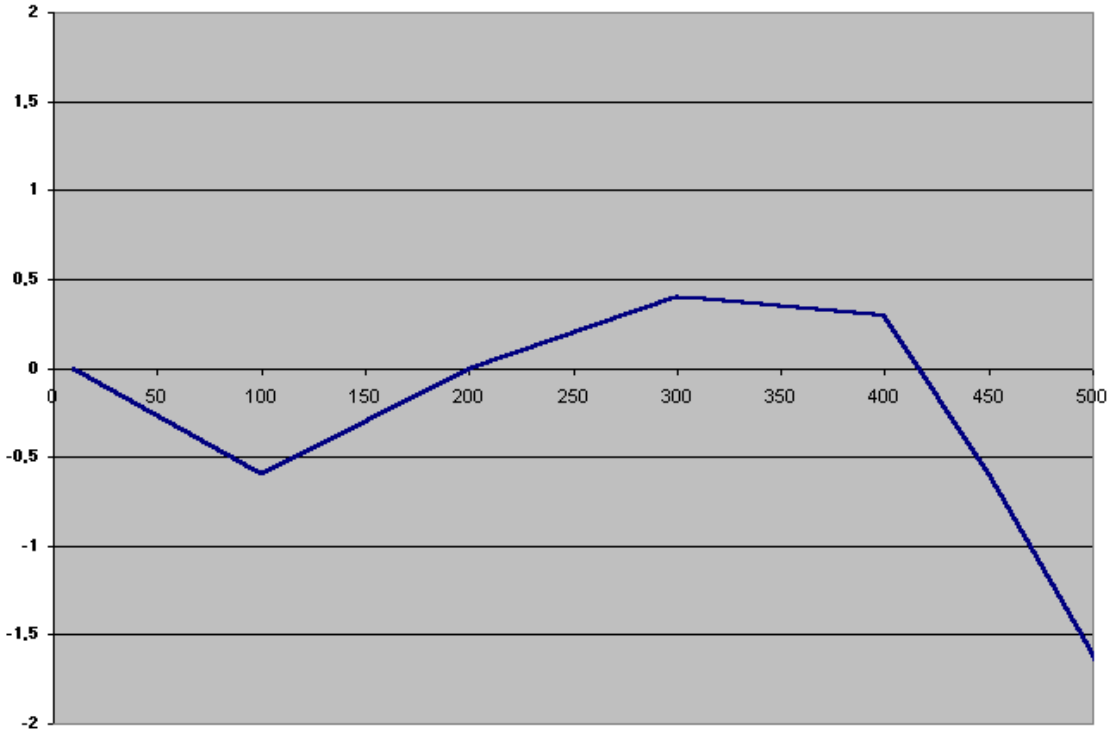



Η ίδια μέτρηση όπως φαίνεται σαν κυματομορφή. Η λεπτή γραμμή είναι η σωστή. Η μπλε γραμμή είναι αυτή που μετρήθηκε και όπως φαίνεται το μέγιστο λάθος είναι 2.5dB στα -60dBm.

Κυματομορφή συχνοτήτων




Το επίπεδο του σήματος εισόδου είναι 0dBm , η συχνότητα έχει αλλάξει και η μέτρηση των dBm έχει καταγραφεί
Το βατόμετρο έχει βαθμονομηθεί στα 0 για 10MHz
Η Γεννήτρια που χρησιμοποιήθηκε ήταν η Boonton 102D

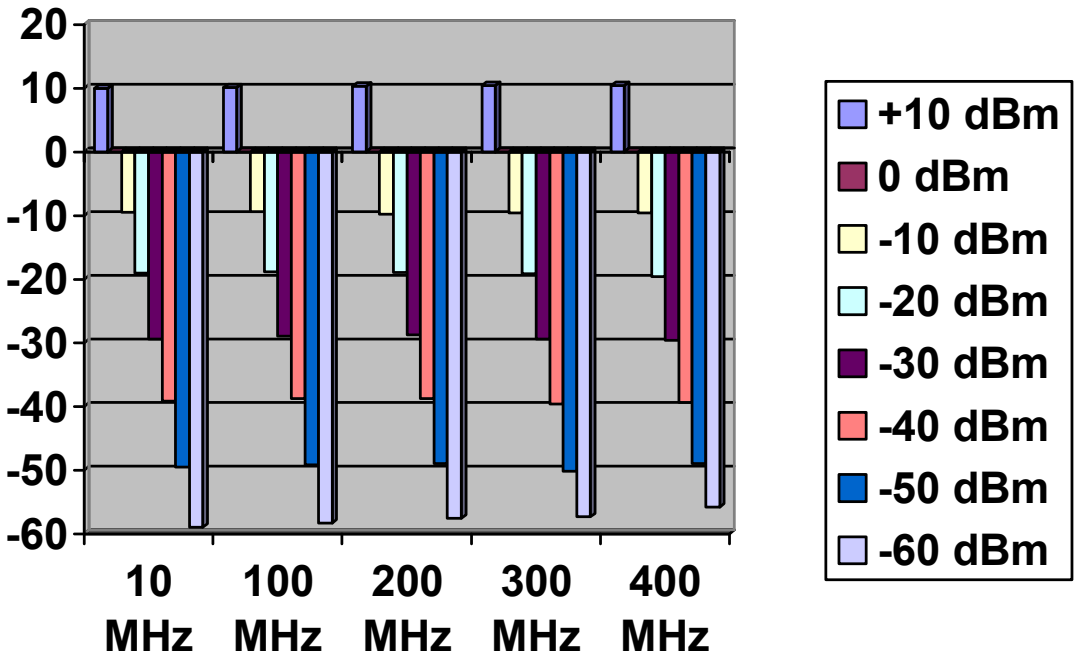



Η ίδια μέτρηση χρησιμοποιώντας την γεννήτρια marconi

Κάναμε τις ίδιες μετρήσεις και στο δικό μας βατόμετρο ,με την βοήθεια της γεννήτρια Agilent 8648C (9KHz – 3200MHz) με τάση τροφοδοσίας 9V
Για να γίνει αυτό χρειάστηκε να βαθμονομήσουμε το όργανο σε διαφορετικές συχνότητες.

| | |
|---|---|
| LF | 10 MHz |
| HF | 100 MHz |
| VHF | 200 MHz |
| UHF | 300 MHz |
| SHF | 400 MHz |

| | **LF** | **HF** | **VHF** | **UHF** | **SHF** |
|---|---|---|---|---|---|
| **Input** | **10 MHz** | **100 MHz** | **200 MHz** | **300 MHz** | **400 MHz** |
| **+10 dBm** | 9.9 | 10.1 | 10.3 | 10.4 | 10.4 |
| **0 dBm** | 0 | 0 | 0 | 0 | 0 |
| **-10 dBm** | -9.5 | -9.4 | -9.8 | -9.6 | -9.6 |
| **-20 dBm** | -19.1 | -18.9 | -19 | -19.2 | -19.6 |
| **-30 dBm** | -29.4 | -29 | -28.8 | -29.4 | -29.6 |
| **-40 dBm** | -39.2 | -38.8 | -38.7 | -39.6 | -39.4 |
| **-50 dBm** | -49.5 | -49.2 | -49 | -50.2 | -49 |
| **-60 dBm** | -59 | -58.3 | -57.6 | -57.3 | -55.8 |




| | LF | HF | VHF | UHF | SHF |
|---|---|---|---|---|---|
| **Input** | **10 MHz** | **100 MHz** | **200 MHz** | **300 MHz** | **400 MHz** |
| **+9** | 8.9 | 9.1 | 9.3 | 9.3 | 9.3 |
| **+8** | 8 | 8.2 | 8.2 | 8.3 | 8.2 |
| **+7** | 7.1 | 7.2 | 7.2 | 7.2 | 7.2 |
| **+6** | 6.1 | 6.2 | 6.1 | 6.1 | 6.1 |
| **+5** | 5.1 | 5.2 | 5.1 | 5 | 5 |
| **+4** | 4.1 | 4.1 | 4 | 4 | 4 |
| **+3** | 3.1 | 3.1 | 2.9 | 2.9 | 2.9 |
| **+2** | 2 | 2 | 1.9 | 1.9 | 1.9 |
| **+1** | 1 | 1 | 0.9 | 1 | 0.9 |
| **0 dBm** | 0 | 0 | 0 | 0 | 0 |
| **-1** | -1 | -0.9 | -0.9 | -0.9 | -1 |
| **-2** | -2 | -1.9 | -1.9 | -1.9 | -1.9 |
| **-3** | -3 | -2.9 | -2.8 | -2.8 | -2.9 |
| **-4** | -3.9 | -3.7 | -3.9 | -3.9 | -3.9 |
| **-5** | -4.8 | -4.7 | -4.7 | -4.8 | -4.9 |
| **-6** | -5.7 | -5.6 | -5.7 | -5.8 | -5.9 |
| **-7** | -6.4 | -6.5 | -6.4 | -6.4 | -6.7 |
| **-8** | -7.4 | -7.4 | -7.4 | -7.5 | -7.7 |
| **-9** | -8.4 | -8.5 | -8.4 | -8.5 | -8.7 |

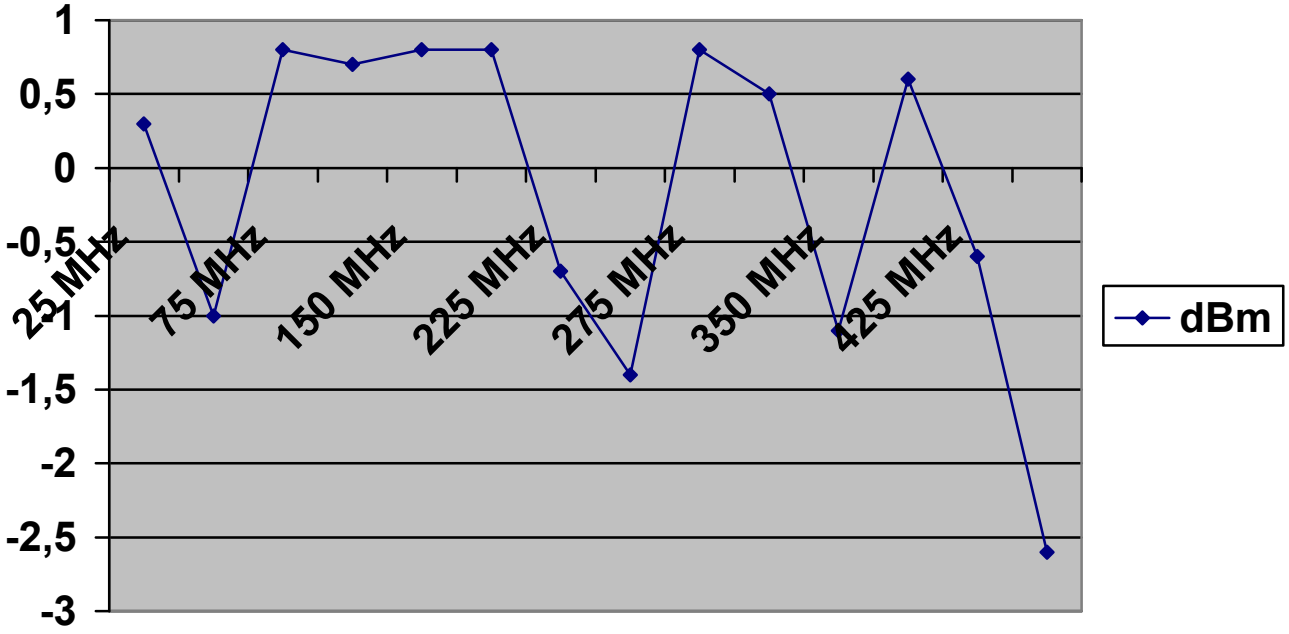
Παρακάτω παρουσιάζονται κάποιες ενδιάμεσες συχνότητες.

| | **LF** | | **HF** | | |
|---|---|---|---|---|---|
| **Input** | **25 MHz** | **50 MHz** | **75 MHz** | **125 MHz** | **150 MHz** |
| +10 | 9.6 | 8.9 | 10.8 | 9.5 | 11 |
| 0 | 0.3 | -1 | 0.8 | 0.7 | 0.8 |
| -10 | -9.7 | -10.3 | -8.5 | -10.1 | -8.6 |
| -30 | -29.7 | -30.3 | -28.3 | -29.8 | -27.9 |
| -60 | -59.2 | -59.6 | -57.5 | -58.7 | -56.5 |

| **VHF** | | | **UHF** | | |
|---|---|---|---|---|---|
| **Input** | **175 MHz** | **225 MHz** | **250 MHz** | **275 MHz** | **325 MHz** |
| +10 | 11 | 9.7 | 9 | 11.2 | 9.9 |
| 0 | 0.8 | -0.7 | -1.4 | 0.8 | 0.5 |
| -10 | -8.6 | -10.1 | -10.9 | -8.8 | -10 |
| -30 | -27.9 | -29.7 | -30.6 | -28.6 | -29.9 |
| -60 | -56.5 | -58.6 | -59.3 | -56.9 | -57.7 |

| | **UHF** | **SHF** | | |
|---|---|---|---|---|
| **Input** | **350 MHz** | **375 MHz** | **425 MHz** | **500 MHz** |
| +10 | 9.4 | 11 | 9.7 | 7.5 |
| 0 | -1.1 | 0.6 | -0.6 | -2.6 |
| -10 | -10.7 | -9 | -10.3 | -12.5 |
| -30 | -30.5 | -31.1 | -30.3 | -32.8 |
| -60 | -57.9 | -57.9 | -55.8 | -56.6 |

Μέτρηση κάποιων συχνοτήτων στις κοντινές κλίμακες (0dBm)




| 25 MHz | | 50 MHz | | 75 MHz | |
|---|---|---|---|---|---|
| LF | | LF | HF | LF | HF |
| 0.3 | | -1 | 1.6 | -1.8 | 0.8 |

| 125 MHz | | 150 MHz | | 175 MHz | |
|---|---|---|---|---|---|
| HF | | HF | VHF | HF | VHF |
| 0.7 | | -1.5 | 1.5 | -2.2 | 0.8 |

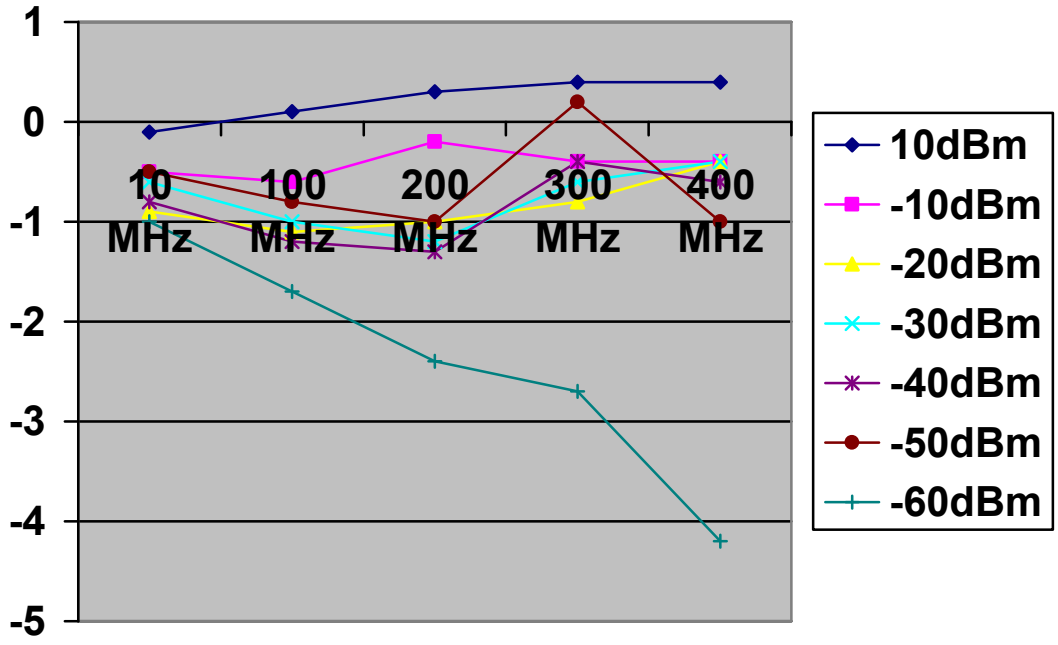
| 225 MHz | | 250 MHz | | 275 MHz | |
|---|---|---|---|---|---|
| VHF | | VHF | UHF | VHF | UHF |
| -0.7 | | -1.4 | 1.4 | -2 | 0.8 |

| 325 MHz | | 350 MHz | | 375 MHz | |
|---|---|---|---|---|---|
| UHF | | UHF | SHF | UHF | SHF |
| 0.5 | | -1.1 | 1.1 | -1.6 | 0.6 |

| 425 MHz | | 450 MHz | | 500 MHz | |
|---|---|---|---|---|---|
| SHF | | SHF | | SHF | |
| -0.6 | | -1.1 | | -2.6 | |

Πίνακας. Σφάλμα σε dB που παρουσιάζει η κατασκευή από 10MHz – 400MHz και ισχύ μέτρησης από -60 έως +10 dBm.

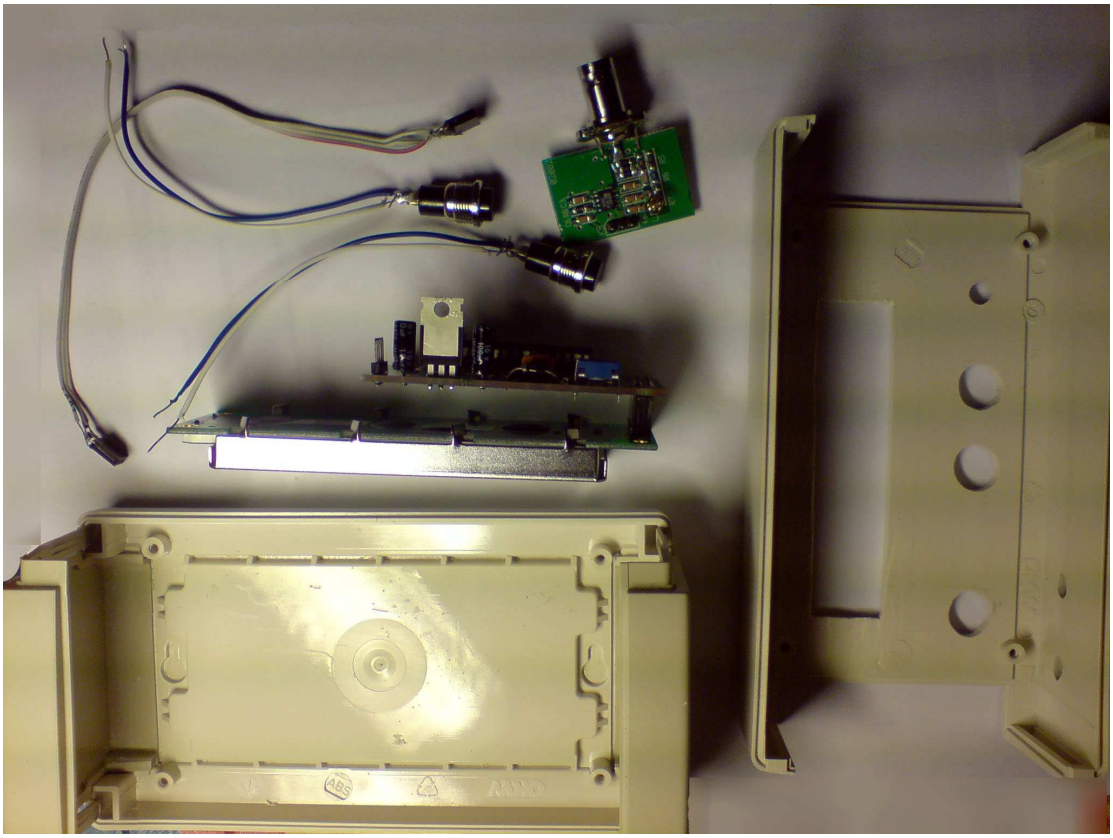
| | **10 MHz** | **100 MHz** | **200 MHz** | **300 MHz** | **400 MHz** |
|---|---|---|---|---|---|
| **10dBm** | -0,1 | 0,1 | 0,3 | 0,4 | 0,4 |
| **-10dBm** | -0,5 | -0,6 | -0,2 | -0,4 | -0,4 |
| **-20dBm** | -0,9 | -1,1 | -1 | -0,8 | -0,4 |
| **-30dBm** | -0,6 | -1 | -1,2 | -0,6 | -0,4 |
| **-40dBm** | -0,8 | -1,2 | -1,3 | -0,4 | -0,6 |
| **-50dBm** | -0,5 | -0,8 | -1 | 0,2 | -1 |
| **-60dBm** | -1 | -1,7 | -2,4 | -2,7 | -4,2 |




Έχουμε στα θετικά σφάλμα μέχρι +0,4dB ενώ στα αρνητικά έχουμε μέχρι -1,3dB στα -50 dBm. Στα -60 dBm φτάνει μέχρι τα -4,2 dB. **Επομένως, το μέγιστο σφάλμα μέτρησης του βαττομέτρου της κατασκευής είναι (-1.3 dB …. +0.4 dB) για ισχύ εισόδου από -50 μέχρι +10 dBm και περιοχή συχνοτήτων 10 – 400 MHz. Αυτή είναι η χρήσιμη περιοχή λειτουργίας του οργάνου.**
Αυτό το σφάλμα είναι ικανοποιητικό για μια φτηνή κατασκευη. Για να έχουμε μεγαλύτερη ακρίβεια θα πρέπει να δαπανήσουμε αρκετά χιλιάδες ευρώ και την αγορά ακριβού βατομέτρου.

**Το κουτί**

Υπάρχουν πάρα πολλοί τρόποι κατασκευής του κουτιού. θα μπορούσαμε να το κατασκευάσουμε από το υλικό κατασκευής πλακετών που υπάρχει άφθονο σε καταστήματα ηλεκτρονικών ειδών αλλά εμείς επιλέξαμε σκληρό πλαστικό κουτί συγκεκριμένων διαστάσεων κυρίως για λόγους εμφανισιακους καθώς και για λόγους χρηστικότητας.
Ο σκοπός μας είναι να κατασκευάσουμε έναν βατόμετρο πολύ μικρών διαστάσεων έτσι ώστε να μπορεί να είναι φορητό και εύχρηστο γι αυτό το λόγο άλλωστε δε βάλαμε και μπαταρία για να μην προσθέσουμε βάρος στη κατασκευή.



Χρειάστηκε ένα ειδικό πριονάκι το οποίο μας βοήθησε στο να κόψουμε το πλαστικό καθώς και ένα τρυπάνι για να κάνουμε τις τρύπες που θα μπουν οι διακόπτες καθώς και πολύ υπομονή!

**Ρύθμιση της εισόδου RF και της εισόδου DC**

Ο μετρητής RF ρυθμίζεται μέσω του λογισμικού, εφαρμόζοντας στον ακροδέκτη εισόδου 0 dBm. Επιλέγουμε πρώτα μέσω του κωδικοποιητή την μνήμη της αντίστοιχης περιοχής συχνοτήτων, στην συνέχεια καλούμε το μενού, διαλέγουμε 'Calíbrate 0 dBm' (ρύθμιση 0 dBm) και πατάμε το πλήκτρο επιλογής.

Η ένταση σήματος 0 dBm αποθηκεύεται στην εσωτερική μνήμη του μετρητή και από τούδε, ο μετρητής θα 'παίζει' ±2 dB στην συγκεκριμένη περιοχή συχνοτήτων. Η διαδικασία αυτή επαναλαμβάνεται για όλες τις περιοχές συχνοτήτων. Υπάρχουν συνολικά πέντε μνήμες περιοχής συχνοτήτων: LF, HF, VHF, UHF και SHF.

Ο μετρητής τάσης DC ρυθμίζεται μέσω του υλικού. Ρυθμίζουμε προσεκτικά την πάνω αντίσταση R 15 του διαιρέτη τάσης που βρίσκεται παράλληλα με την R 17 και την κάτω αντίσταση R16.
θέτουμε την οθόνη του μετρητή σε κατάσταση μετρητή τάσης DC, μέσω της διαδικασίας Menu-> Dial-> Select (Μενου-> Πίνακας επιλογής-> Επιλογή).

Εφαρμόζουμε 20.00 V στην είσοδο DC και κοιτάζουμε την ένδειξη που αναγράφεται στο LCD. Εάν η αναγραφόμενη τιμή είναι μικρότερη από 20.00 V, τοποθετούμε

παράλληλα με την επάνω αντίσταση R 17 μία αντίσταση 10 ΜΩ. Εάν η ένδειξη είναι μικρότερη από 20.00V, τοποθετούμε την αντίσταση στην κάτω αντίσταση R16.

**Προφυλάξεις**

Το AD8307 δεν πρόκειται να αντέξει σε εκ παραδρομής εφαρμογή τάσης +5 V στον ακροδέκτη εξόδου, γι' αυτό δώσαμε ιδιαίτερη προσοχή στις καλωδιώσεις που έρχονται από την κύρια πλακέτα.

Η πλακέτα εισόδου είναι σχεδιασμένη για τάσεις μέχρι 1 watt. Εάν εφαρμοστεί μεγαλύτερη ισχύ η ένδειξη στην οθόνη εξαφανίζεται! Εάν εφαρμοστεί περισσότερα από 1 Watt στο όργανο, ενδέχεται να καεί και το AD8307.

**Menu**

Η χρήση των μενού (έκδοση λογισμικού 1.03)

Όλη η νοημοσύνη που διαθέτει το όργανο βρίσκεται στο λογισμικό που ανάπτυξε ο Thomas Scherrer και αποθήκευσε στον μικροελεγκτή PIC.

Η οθόνη υποδοχής του οργάνου δείχνει κάπως έτσι:


Η κύρια οθόνη εκκίνησης δείχνει:

dBm, κατάσταση, τάση AI:: ραβδόγραμμα, ισχύ RF σε watt

Εάν δεν χρησιμοποιείται εξασθένηση, η ένδειξη dBm βρίσκεται μεταξύ -63 dBm (κατώφλι θορύβου) έως +30 dBm (1 Watt).

Η ένδειξη κατάστασης απεικονίζει την επιλεγείσα περιοχή συχνοτήτων, και την κατάσταση εξασθένησης. Χρησιμοποιήστε τον επιλογέα ζώνης συχνοτήτων για να διαλέξετε μεταξύ μνημών βαθμονόμησης LF, HF, VHF, UHF και SHF. Η βαθμονόμηση του οργάνου στα Ο dBm προτείνεται να γίνεται στα: LF = 3.5 MHZ, HF = 14 MHZ, VHF =145 MHZ, UHF = 430 MHZ, SHF = 440 MHZ. Εννοείται βέβαια ότι για καλύτερες επιδόσεις μπορούμε να βαθμονομήσουμε τις δικές μας συχνότητες .

Στο μενού του μετρητή ισχύος RF, πατήστε το πλήκτρο SELECT (επιλογή) για να μπείτε σε κατάσταση RELATIVE (σχετική). Όταν εισέρχεστε στην κατάσταση αυτή η ένδειξη dB μηδενίζεται και απεικονίζονται dBm και ραβδόγραμμα,


**Το μενού**

Για να μπείτε στο μενού / ρυθμίσεις χρησιμοποιούμε το πλήκτρο MENU. Όντας μέσα στο μενού, χρησιμοποιούμε τον στρεφόμενο κωδικοποιητή για να εμφανίσουμε τις επιθυμητές ρυθμίσεις. Όταν φτάσουμε σε αυτές, πατάμε SELECT για να τις ενεργοποιήσουμε, πληροφορία που απεικονίζεται και στην οθόνη.

Διαθέσιμες επιλογές στο μενού:

0: 0 dB, δεν υπάρχει εξασθενητής, μέγιστο 1 Watt

1: -10 dB, υπάρχει εξασθενητής, μέγιστο 10 Watt

2: -20 dB, υπάρχει εξασθενητής, μέγιστο 100 Watt

3: -30 dB, υπάρχει εξασθενητής, μέγιστο 1 KWatt

4: -40 dB, υπάρχει εξασθενητής, μέγιστο 10 KWatt

5: -50 dB, υπάρχει εξασθενητής, μέγιστο 100 KWatt

6: Ένδειξη βολτομέτρου DC, τρέχουσα τιμή, ελάχιστη και μέγιστη

7: Μετρητής ισχύος RF, η προεπιλεγμένη οθόνη κατά την εκκίνηση

8: SSB PEP (peak envetope power = μέγιστη περιβάλλουσα ισχύς) μέτρηση ισχύος, με συγκράτηση μεγίστου και μεταβαλλόμενη εξασθένηση.

## Ο κώδικας

# ΚΩΔΙΚΑΣ ΚΑΙ ΤΡΟΠΟΣ ΛΕΙΤΟΥΡΓΙΑΣ ΤΟΥ PIC

Γράφτηκε από τον Thomas Scherrer στο MPLAB
Υπάρχει στην σελίδα του www.elektor.com

```
// Εδώ καθορίζονται όλες οι μεταβλητές τα bit και το hardware
// Digital Wattmeter PCB version 1.3 OZ2CPU

 Unsigned char   LCDPORT @ 0x75;
 Unsigned char   sera;
 unsigned char   serb;
 unsigned char   user1;
 unsigned char   user2;
 unsigned char   user3;
 unsigned char   user4;
 unsigned char   user5;
 unsigned char   wpref;
 unsigned char   wpointer;
 unsigned char   w2;
 unsigned char   w1;
 unsigned char   wpoint;
 unsigned char   vpref;     // καθορισμος προθεματος
 unsigned char   vpointer;
 unsigned char   v2;       // καθορισμος δεκαδικου μερους
 unsigned char   v1;       // καθορισμος bit δυναμικοτητας
 unsigned char   vpoint;  // θεση υποδιαστολής
 unsigned char   del0;
 unsigned char   del1;
 unsigned char   del2;
 unsigned char   DELAYTIME;
 unsigned char   x;
 unsigned int    iuser1;
 unsigned int    peak;
 unsigned int    rfraw;
 unsigned int    rfrawB;
 unsigned int    rfraw2;  // αντιστάθμιση της αξίας βαθμολόγησης
 unsigned int    rfRawAtt; // μετατροπή τιμών
 unsigned int    rfdbm;
 unsigned char   dbswr;  // τιμή σε db για τον υπολογισμό του loss SWR
 unsigned int    voltraw;
 unsigned int    voltmax;
 unsigned int    voltmin;
 unsigned int    konv;
 unsigned int    LFcal;
 unsigned int    HFcal;
 unsigned int    VHFcal;
 unsigned int    UHFcal;
```

```
unsigned int    SHFcal;
unsigned int    ActualCal; // επιλογη μονάδας υπολογισμού
unsigned int    RelativeCal;
unsigned char   message1;
unsigned char   message2;
unsigned char   message3;
unsigned char   messagenumber;
unsigned char   messagetimer;
unsigned char   messagetype;
unsigned char   bandmode;  // καθορισμός μπάντας συχνοτήτων
unsigned char   attmode;
unsigned char   menutype;
unsigned char   displaydelay;  // θέτει την καθυστέρηση της οθόνης σε ms
unsigned char   peakholddelay; // για την απεικόνιση SSB
unsigned char   metertype;     // RF, SSB, DC, RTN, ABOUT osv...


////////////////////////// bitfield :Flags  ////////////////////////////////

 bit       ENA    @ (unsigned)&PORTC*8+3 ;  // LCD controll BIT in PORTC
 bit       RW     @ (unsigned)&PORTC*8+2 ;
 bit       RS     @ (unsigned)&PORTC*8+0 ;

 bit       LC0      @ (unsigned)&LCDPORT*8+0 ;
 bit       LC1      @ (unsigned)&LCDPORT*8+1 ;
 bit       LC2      @ (unsigned)&LCDPORT*8+2 ;
 bit       LC3      @ (unsigned)&LCDPORT*8+3 ;
 bit       LC4      @ (unsigned)&LCDPORT*8+4 ;
 bit       LC5      @ (unsigned)&LCDPORT*8+5 ;
 bit       LC6      @ (unsigned)&LCDPORT*8+6 ;
 bit       LC7      @ (unsigned)&LCDPORT*8+7 ;

 bit       LCD0     @ (unsigned)&PORTC*8+1 ;
 bit       LCD1     @ (unsigned)&PORTB*8+4 ;
 bit       LCD2     @ (unsigned)&PORTB*8+3 ;
 bit       LCD3     @ (unsigned)&PORTB*8+2 ;
 bit       LCD4     @ (unsigned)&PORTB*8+1 ;
 bit       LCD5     @ (unsigned)&PORTC*8+4 ;
 bit       LCD6     @ (unsigned)&PORTC*8+5 ;
 bit       LCD7     @ (unsigned)&PORTC*8+7 ;

 bit       BSEL     @ (unsigned)&PORTB*8+7 ;
 bit       BMENU    @ (unsigned)&PORTB*8+6 ;
 bit       BA       @ (unsigned)&PORTB*8+5 ; // encoder A
 bit       BB       @ (unsigned)&PORTB*8+0 ; // encoder B

 bit       SLUT;    // to end a sub
 bit   MENU;
 bit   BARGLO;  // εάν το bargraph ειναι hi ή lo side
 bit   DISPSPD;
 bit   oldpos;    // encoder A ανιχνευτης ακρων
 bit   oldmenu;   // menu button, ανιχνευτης ακρων
 bit   oldselect; // select button, ανιχνευτης ακρων
 bit   RELATIVE;  // αν είναι on τα rf volt και η rf ισχύς δεν θα παρουσιάζονται
 bit   SWRERROR;

////////////////////////// GLOBAL CONSTANTS//////////////////////////////////

 //             db   0   1   2   3   4   5   6   7
```

```
const char swrconv[] =
{255,255,255,255,255,255,200,161,132,110,92,78,67,57,49,43,37,32,28,25,22,19,17,15,13,11,10,9,8,7,
6,6,5,4,4,3,3,3,2,2,2,2,1,1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

const char wten[] =
{1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3,3,3,3
,3,3,3,3,3,3,4,4,4,4,4,4,4,4,4,5,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,7,7,7,7,7,7,8,8,8,8,8,9,9,9,9,10};
const char wdec[] =
{0,2,5,7,10,12,15,17,20,23,26,29,32,35,38,41,45,48,51,55,58,62,66,70,74,78,82,86,91,95,0,4,9,14,19,2
4,29,34,40,45,51,57,63,69,75,82,88,95,2,9,16,24,31,39,47,55,63,72,80,89,98,7,17,27,37,47,57,68,79,90
,1,13,25,37,50,62,75,89,3,17,31,46,61,76,92,8,24,41,59,76,94,13,32,51,71,91,12,33,55,77,0};

const char vten[] =
{22,22,22,23,23,23,23,24,24,24,25,25,25,25,26,26,26,27,27,27,28,28,28,29,29,29,30,30,30,
 31,31,31,32,32,33,33,33,34,34,35,35,35,36,36,37,37,37,38,38,39,39,40,40,41,41,42,42,43,43,44,44,45,
 45,46,46,47,47,48,48,49,50,50,51,51,52,53,53,54,54,55,56,56,57,58,58,59,60,60,61,62,63,63,64,65,65,
 66,67,68,69,69,70,71,72,73,74,74,75,76,77,78,79,80,81,82,83,84,85,85,86,88,89,90,91,92,93,94,95,96,

97,98,99,101,102,103,104,105,107,108,109,110,112,113,114,116,117,118,120,121,122,124,125,127,12
8,130,

131,133,134,136,137,139,141,142,144,146,147,149,151,152,154,156,158,160,161,163,165,167,169,17
1,173,
 175,177,179,181,183,185,188,190,192,194,197,199,201,203,206,208,211,213,216,218,221,223};

const char vdec[] =
{36,62,88,15,41,69,96,24,52,80,9,38,67,97,27,58,88,19,51,83,15,48,81,14,48,82,16,

51,87,22,59,95,32,70,7,46,84,24,63,3,44,85,26,68,11,54,97,41,86,31,76,22,69,16,64,12,61,10,60,10,62,
 13,65,18,72,26,81,36,92,49,6,64,23,82,42,3,64,26,89,52,17,82,48,14,81,50,18,88,59,30,2,75,49,24,99,

76,53,31,10,90,71,53,36,20,4,90,77,65,53,43,34,26,19,13,8,4,1,100,99,0,2,5,9,15,21,29,39,49,61,74,88,

4,21,39,59,80,2,26,52,79,7,37,68,1,35,71,8,47,88,30,74,20,67,16,67,19,74,30,87,47,9,72,37,4,74,45,18,

93,70,49,30,13,99,86,76,68,62,59,57,58,62,67,75,86,99,14,32,52,75,1,29,60,93,29,68,10,54,2,52,5,61};


#define FALSE       0
#define TRUE        1
#define OFF         0
#define ON          1
#define DISABLE             0
#define ENABLE      1
#define SET         1

// ρυθμίσεις μεταβλητών για τον τύπο μέτρησης
#define ABOUT       0
#define DC          2
#define SSB         3
#define RF          4
#define RTN         5

// ρυθμίσεις μεταβλητών για LCD goto
#define LINE1       0x80
#define LINE2       0xA9
#define CLEAR       0x01
#define GFXSET      0x40
```

```
////////////////// Defining MACROES

#define NOP() asm("nop");
```

**Μετατροπέας ακέραιων σε ASCII**

```
void KONVERTER()
{
 user1 = '0';
 user2 = '0';
 user3 = '0';
 user4 = '0';
 while (konv>=1000)
 {
  konv=konv-1000;
  user1++;
 }
 while (konv>=100)
 {
  konv=konv-100;
  user2++;
 }
 while (konv>=10)
 {
  konv=konv-10;
  user3++;
 }
 user4=konv+0x30;
}
```

**Ρουτίνες σειριακής μεταφοράς**

```
// φτιάχνονται στον assembler για τον σωστό συγχρονισμό
// 4.00 MHz PIC clock gives 38400 Baud
// σειριακός μεταφορέας software λόγο των hardware programming facilities
// ο PIC χρησιμοποιεί 2 pins (RX and TX) αν το hardware serial module είναι
// επιλεγμένο, το PCB είναι συνδεδεμένο στην LCD, και μόνο το TX pin ήταν //κρατημένο
// καιρό πριν φτιαχτεί η σειριακή έξοδος
// By OZ2CPU 2001, 2002 and beyond.

void Ser1(user1)
{
 LCDPORT = user1;
 NOP();
 #asm
 bcf  3,5        ;
 bcf  3,6
 goto Serstart
 SerSpeed  // αυτό παράγει το 26uS bit speed για 38400 Baud
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
```

```
nop
nop
nop
nop
nop
nop
return
Serstart
bcf  7,6       ; RC6=0;  // startbit
nop
nop
call SerSpeed
btfss  117,0    ; check LC0     1   1
goto  ser00     ; if zero jump  2   1
nop             ;                   1
bsf  7,6        ; RC6 = 1           1*
goto  ser01     ;                   2
ser00:
bcf  7,6        ; RC6 = 0;      1*
nop             ;               1
nop             ;               1
ser01:
call SerSpeed
btfss  117,1    ; check LC1
goto  ser10     ; if zero jump
nop
bsf  7,6        ; RC6 = 1
goto  ser11
ser10:
bcf  7,6        ; RC6 = 0;
nop
nop
ser11:
call SerSpeed
btfss  117,2    ; check LC2
goto  ser20     ; if zero jump
nop
bsf  7,6        ; RC6 = 1
goto  ser21
ser20:
bcf  7,6        ; RC6 = 0;
nop
nop
ser21:
call SerSpeed
btfss  117,3    ; check LC3
goto  ser30     ; if zero jump
nop
bsf  7,6        ; RC6 = 1
goto  ser31
ser30:
bcf  7,6        ; RC6 = 0;
nop
nop
ser31:
call SerSpeed
btfss  117,4    ; check LC4
goto  ser40     ; if zero jump
nop
bsf  7,6        ; RC6 = 1
```

```
 goto  ser41
 ser40:
 bcf  7,6       ; RC6 = 0;
 nop
 nop
 ser41:
 call SerSpeed
 btfss  117,5    ; check LC5
 goto  ser50     ; if zero jump
 nop
 bsf  7,6       ; RC6 = 1
 goto  ser51
 ser50:
 bcf  7,6       ; RC6 = 0;
 nop
 nop
 ser51:
 call SerSpeed
 btfss  117,6    ; check LC6
 goto  ser60     ; if zero jump
 nop
 bsf  7,6       ; RC6 = 1
 goto  ser61
 ser60:
 bcf  7,6       ; RC6 = 0;
 nop
 nop
 ser61:
 call SerSpeed
 btfss  117,7    ; check LC7
 goto  ser70     ; if zero jump
 nop
 bsf  7,6       ; RC6 = 1
 goto  ser71
 ser70:
 bcf  7,6       ; RC6 = 0;
 nop
 nop
 ser71:
 call SerSpeed
 nop
 nop
 nop
 bsf  7,6       ; RC6 = 1  stopbit
 call SerSpeed
 #endasm
}
```

**Ρουτίνες καθυστέρησης**

```
// στα 4 MHz clock PIC clock freq.
// OZ2CPU Wattmeter

void Del(del0)    // input in 10uS
{
 while (del0>=1)
 {
   CLRWDT();
```

```
   del0--;
  }
}


void DelaymS(del1)                    //  input in mS
{
 while (del1>=1)
 {
  Del(36);
  del1--;
 }
}



void Delay(del1)                      //  input in mS
{
 while (del1>=1)
 {
  Del(100);
  del1--;
 }
}



void Delay100(del2)                   //  input in mS
{
            while (del2>=1)
            {
            Delay(100);
            del2--;
            }
}
```

**Ρουτίνα για τον έλεγχο της LCD**

```
void LCDswap()
{
 LCD0 = LC0;
 LCD1 = LC1;
 LCD2 = LC2;
 LCD3 = LC3;
 LCD4 = LC4;
 LCD5 = LC5;
 LCD6 = LC6;
 LCD7 = LC7;
}


void LCDskriv1(user1)  // γραφει ενα 1 γραμμα στην LCD display
{
  LCDPORT = user1;
  LCDswap();
  ENA = 1;
  ENA = 0;
  Del(1);  // adjusted to max accepted writing speed
}
```

```
void LCDgoto(user1)  // μετακινεί τον αόρατο κέρσορα
{
  ENA = 0;
  RS  = 0;
  RW  = 0;
  LCDPORT = user1;
  LCDswap();
  ENA = 1;
  ENA = 0;
  Delay(2);
  RS  = 1;
}



void LCDsetgfx()  // χαρακτήρες που καθορίζονται από τον χρήστη για το bar-graf
{
 LCDgoto(GFXSET);

 LCDskriv1(0x00);  // 0 caracter code
 LCDskriv1(0x00);
 LCDskriv1(0x10);
 LCDskriv1(0x10);
 LCDskriv1(0x10);
 LCDskriv1(0x10);
 LCDskriv1(0x10);
 LCDskriv1(0x10);

 LCDskriv1(0x00);  // 1
 LCDskriv1(0x00);
 LCDskriv1(0x18);
 LCDskriv1(0x18);
 LCDskriv1(0x18);
 LCDskriv1(0x18);
 LCDskriv1(0x18);
 LCDskriv1(0x18);

 LCDskriv1(0x00);  // 2
 LCDskriv1(0x00);
 LCDskriv1(0x1c);
 LCDskriv1(0x1c);
 LCDskriv1(0x1c);
 LCDskriv1(0x1c);
 LCDskriv1(0x1c);
 LCDskriv1(0x1c);

 LCDskriv1(0x00);  // 3
 LCDskriv1(0x00);
 LCDskriv1(0x1e);
 LCDskriv1(0x1e);
 LCDskriv1(0x1e);
 LCDskriv1(0x1e);
 LCDskriv1(0x1e);
 LCDskriv1(0x1e);

 LCDskriv1(0x00);  // 4
 LCDskriv1(0x00);
```

```
  LCDskriv1(0x1F);
  LCDskriv1(0x1F);
  LCDskriv1(0x1F);
  LCDskriv1(0x1F);
  LCDskriv1(0x1F);
  LCDskriv1(0x1F);

  LCDskriv1(0x01);  // 5
  LCDskriv1(0x00);
  LCDskriv1(0x1F);
  LCDskriv1(0x1F);
  LCDskriv1(0x1F);
  LCDskriv1(0x1F);
  LCDskriv1(0x1F);
  LCDskriv1(0x1F);

   ENA = 0;
   RS  = 1;
   RW  = 0;
   LCDPORT = 0x80;  // 1.digit
   LCDswap();
   ENA = 1;
   ENA = 0;
   Delay(3);
   RS  = 0;
}


void LCDinit()  // init of LCD display
{              // απαραιτητο για κάθε τυπο LCD, στο power up
   ENA = 0;  // LCD control bits is καθοριζονται στο def.h
   RS  = 0;
   RW  = 0;

   LCDPORT = 0x38;   // RESET, LCDPORT see def.h
   LCDswap();
   Delay(3);
   ENA = 1; CLRWDT();
   ENA = 0;
   Delay(20);

   ENA = 1; CLRWDT();
   ENA = 0;
   Delay(3);

   ENA = 1;CLRWDT();
   ENA = 0;
   Delay(3);

            // Display reset
   ENA = 1;CLRWDT();
   ENA = 0;
   Delay(3);


     // Display entry mode set
   LCDPORT = 0x06;  // 4 or 8 bits mode, 8 επιλεγεται
   LCDswap();
   Delay(3);
   ENA = 1;CLRWDT();
```

```
  ENA = 0;

     // Display on
  LCDPORT = 0x0C;  // 0C = no cursor, no blink.
  LCDswap();
  Delay(3);
  ENA = 1;CLRWDT();

  LCDgoto(LINE1);
}



void LCDskriver(user2) // γεμίζει την 20 x 2 οθόνη
{
 user1=0;  // pointer for text
 while (user1<20)
  {
   if (user2==6) LCDPORT = TEXT6[user1];
   else
   if (user2==7) LCDPORT = TEXT7[user1];
   else
   if (user2==8) LCDPORT = TEXT8[user1];
   else
   if (user2==9) LCDPORT = TEXT9[user1];
   else
   if (user2==15) LCDPORT = TEXT15[user1];
   else
   if (user2==16) LCDPORT = TEXT16[user1];
   else
   if (user2==17) LCDPORT = TEXT17[user1];
   else
   if (user2==18) LCDPORT = TEXT18[user1];
   else
   if (user2==19) LCDPORT = TEXT19[user1];
   LCDswap();
   ENA = 1;
   ENA = 0;
   Del(1);
   user1++;
  }
}



void Welcome()
{
  LCDgoto(LINE1);
  LCDskriver(6); // ο χρηστης καθοριζει τις γραμμες
  LCDgoto(LINE2);
  LCDskriver(8);
  user1=30;
  while (user1>=1)
  {
   Delay(51);
   user1--;
  }
 LCDgoto(CLEAR);
}
```

```
void KonvSkriv()
{
 LCDskriv1(user2);
 LCDskriv1(user3);
 LCDskriv1(user4);
}
```

**Σχετικα με τον A/D converter**

```
// By oz2cpu
// all ADC

void GetADC()
{
   ADGO = 1;       // ξεκινα την μετατροπη RF A επιπεδου
   while(ADGO) CLRWDT();  // αναμονη για AD μετατροπη να γινει
   rfraw = (ADRESL+(ADRESH*256));   // παρε αποτελεσματα απο L=8bit H=2bit
   CHS2 = 0; CHS1 = 1;
   CHS0 = 0;  // Select AN2 input RF B
   if (metertype==SSB)
   {
    if (rfraw > peak)  // αν η μετρουμενη τιμη είναι μεγαλυτερη από την
                                //αναγραφομενη, αντεγραψε
    {
     peak = rfraw;
     peakholddelay = 50;
    }
    rfraw = peak;  // αποθηκευση του module για χρηση.
   }
   konv = rfraw;  // RF input A
   KONVERTER();
   Ser1(user1); Ser1(user2); Ser1(user3); Ser1(user4);
   Ser1(' ');

   ADGO = 1;       // start RF B level converting
   while(ADGO) CLRWDT();  // αναμονη για AD converter
   rfrawB = (ADRESL+(ADRESH*256));   // get result from adres L=8bit H=2bit
   CHS2 = 0; CHS1 = 0;
   CHS0 = 0;  // Select AN0 input Volt
   konv = rfrawB;  // RF input B
   KONVERTER();
   Ser1(user1); Ser1(user2); Ser1(user3); Ser1(user4);
   Ser1(' ');

   ADGO = 1;       // start VOLT converting
   while(ADGO) CLRWDT();  // wait for AD converter to be done
   voltraw = (ADRESL+(ADRESH*256));   // get result from adres L=8bit H=2bit
   CHS2 = 0; CHS1 = 0;
   CHS0 = 1;  // Select AN1 input RF Level
   konv = voltraw;
   KONVERTER();
   Ser1(user1); Ser1(user2); Ser1(user3); Ser1(user4);
   Ser1(10);  Ser1(13);

}
```

**Ρουτινες**

```
void WriteBand()
{
   if (bandmode==0)
   {
    LCDskriv1(' ');
    LCDskriv1('L');
   }
   else
   if (bandmode==1)
   {
    LCDskriv1(' ');
    LCDskriv1('H');
   }
   else
   if (bandmode==2)
   {
    LCDskriv1('V');
    LCDskriv1('H');
   }
   else
   if (bandmode==3)
   {
    LCDskriv1('U');
    LCDskriv1('H');
   }
   else
   if (bandmode==4)
   {
    LCDskriv1('S');
    LCDskriv1('H');
   }
   LCDskriv1('F');
}

void MessageNumber()
{
   if (messagenumber==0)
   {
    message1 = ' ';
    message2 = 'L';
    message3 = 'F';
    ActualCal = LFcal;
   }
   else
   if (messagenumber==1)
   {
    message1 = ' ';
    message2 = 'H';
    message3 = 'F';
    ActualCal = HFcal;
   }
   else
   if (messagenumber==2)
   {
    message1 = 'V';
    message2 = 'H';
    message3 = 'F';
    ActualCal = VHFcal;
   }
```

```
else
if (messagenumber==3)
{
  message1 = 'U';
  message2 = 'H';
  message3 = 'F';
  ActualCal = UHFcal;
}
else
if (messagenumber==4)
{
  message1 = 'S';
  message2 = 'H';
  message3 = 'F';
  ActualCal = SHFcal;
}
else
if (messagenumber==5)
{
  message1 = 'I';
  message2 = 'N';
  message3 = 'A';
}
else
if (messagenumber==6)
{
  message1 = 'I';
  message2 = 'N';
  message3 = 'B';
}
else
if (messagenumber==7)
{
  message1 = 'A';
  message2 = 'T';
  message3 = 'T';
}
else
if (messagenumber==8)
{
  message1 = '0';
  message2 = 'd';
  message3 = 'B';
}
else
if (messagenumber==9)
{
  message1 = '-';
  message2 = '1';
  message3 = '0';
}
else
if (messagenumber==10)
{
  message1 = '-';
  message2 = '2';
  message3 = '0';
}
else
if (messagenumber==11)
```

```
    {
      message1 = '-';
      message2 = '3';
      message3 = '0';
    }
    else
    if (messagenumber==12)
    {
      message1 = '-';
      message2 = '4';
      message3 = '0';
    }
    else
    if (messagenumber==13)
    {
      message1 = '-';
      message2 = '5';
      message3 = '0';
    }
}


void MessageUpdate()
{
 messagetimer++;
 if (displaydelay<30) user1 = 50;
 else
 user1 = 20;
 if (messagetimer>=user1)  // ταχύτητα πληροφοριών οθόνης
 {
  messagetype++;
  messagetimer = 0;
  if (messagetype>=3) messagetype = 0;
 }
 if (messagetype == 0)
 {
  if (bandmode==0) messagenumber = 0;  // LF
  if (bandmode==1) messagenumber = 1;  // HF
  if (bandmode==2) messagenumber = 2;  // VHF
  if (bandmode==3) messagenumber = 3;  // UHF
  if (bandmode==4) messagenumber = 4;  // SHF
 }
 if (messagetype == 1) messagenumber = 7;  // ATT
 if (messagetype == 2)
 {
  if (attmode==0) messagenumber = 8;  // 0dB
  if (attmode==1) messagenumber = 9;  // -10dB
  if (attmode==2) messagenumber = 10;  // -20dB
  if (attmode==3) messagenumber = 11;  // -30dB
  if (attmode==4) messagenumber = 12;  // -40dB
  if (attmode==5) messagenumber = 13;  // -50dB
 }
 MessageNumber();
}


void ZeroCal()
{
 LCDgoto(CLEAR);
 LCDgoto(LINE1);
```

```
  LCDskriver(17);  // γραψε τις σβησμενες πληροφοριες
  eeprom_write(0,0);  // σβησε το EEPROM location
  for (;;) NOP();   // wait for the watchdog to reset
}

void ReadCal()
{
  LCDgoto(CLEAR);
  LCDgoto(LINE1);
  LCDskriv1('C');  LCDskriv1('a');  LCDskriv1('l');  LCDskriv1('i');
  LCDskriv1('b');  LCDskriv1('.');  LCDskriv1('=');  LCDskriv1(' ');
  LCDskriv1('L');
  LCDskriv1('F');
  konv = LFcal;
  KONVERTER();
  KonvSkriv();
  LCDskriv1(' ');
  LCDskriv1(' ');

  LCDskriv1('H');
  LCDskriv1('F');
  konv = HFcal;
  KONVERTER();
  KonvSkriv();

  LCDgoto(LINE2);
  LCDskriv1('V');
  LCDskriv1('H');
  LCDskriv1('F');
  konv = VHFcal;
  KONVERTER();
  KonvSkriv();
  LCDskriv1(' ');

  LCDskriv1('U');
  LCDskriv1('H');
  LCDskriv1('F');
  konv = UHFcal;
  KONVERTER();
  KonvSkriv();
  LCDskriv1(' ');

  LCDskriv1('S');
  LCDskriv1('H');
  LCDskriv1('F');
  konv = SHFcal;
  KONVERTER();
  KonvSkriv();
  Delay(20);
  while (!BSEL) CLRWDT(); // αναμονή για κλείσιμο διακόπτη
  Delay(20);
  while (BSEL) CLRWDT();  // αναμονή για πάτημα διακόπτη
}


void Calibrate()
{
  LCDgoto(CLEAR);
  LCDgoto(LINE1);
  if ((rfraw>605)&&(rfraw<850))  // έλεγχος ότι το εισερχόμενο σήμα είναι εντος
```

```
                                                    //εμβέλειας.
 {
  LCDskriv1('C'); LCDskriv1('A'); LCDskriv1('L');
  LCDskriv1(' '); LCDskriv1('0'); LCDskriv1('d');
  LCDskriv1('B'); LCDskriv1('m'); LCDskriv1('=');
  konv = rfraw;
  KONVERTER();
  KonvSkriv();
  LCDgoto(LINE2);
  LCDskriv1('S'); LCDskriv1('t'); LCDskriv1('o'); LCDskriv1('r');
  LCDskriv1('e'); LCDskriv1('d'); LCDskriv1(' '); LCDskriv1('i');
  LCDskriv1('n'); LCDskriv1(' ');
  if (bandmode==0)
  {
   LFcal = rfraw;  // LF
   user1 = rfraw - 600;  // 600 is offset for EEPROM
   eeprom_write(0,user1);  // 600 προστίθεται ξανά όταν διαβάζεται το byte cal data
   LCDskriv1('L');
  }
  else
  if (bandmode==1)
  {
   HFcal = rfraw;  // HF
   user1 = rfraw - 600;  // 600 is offset for EEPROM
   eeprom_write(1,user1);  // 600 προστίθεται ξανά όταν διαβάζεται το byte cal data
   LCDskriv1('H');
  }
  else
  if (bandmode==2)
  {
   VHFcal = rfraw;  // VHF
   user1 = rfraw - 600;  // 600 is offset for EEPROM
   eeprom_write(2,user1);  // 600 προστίθεται ξανά όταν διαβάζεται το byte cal data
   LCDskriv1('V'); LCDskriv1('H');
  }
  else
  if (bandmode==3)
  {
   UHFcal = rfraw;  // UHF
   user1 = rfraw - 600;  // 600 is offset for EEPROM
   eeprom_write(3,user1);  // 700 προστίθεται ξανά όταν διαβάζεται το byte cal data
   LCDskriv1('U'); LCDskriv1('H');
  }
  else
  if (bandmode==4)
  {
   SHFcal = rfraw;  // SHF
   user1 = rfraw - 600;  // 600 is offset for EEPROM
   eeprom_write(4,user1);  // 700 προστίθεται ξανά όταν διαβάζεται το byte cal data
   LCDskriv1('S'); LCDskriv1('H');
  }
  LCDskriv1('F'); LCDskriv1(' ');
  LCDskriv1('B'); LCDskriv1('a'); LCDskriv1('n'); LCDskriv1('d');
 }
 else LCDskriver(18); // write Error message
 Delay100(20);
 messagetimer = 0;
 messagetype = 0;
}
```

```
void go_up()
{
 if (MENU) // in menu
 {
  menutype++;
  if (menutype>14) menutype = 14;
 }
 else  // not in menu
 {
  bandmode++;
  if (bandmode>=5) bandmode = 4;
  messagetimer = 0;
  messagetype = 0;
 }
 if (DISPSPD) displaydelay++;
}


void go_down()
{
 if (MENU) // in menu
 {
  menutype--;
  if (menutype>=128) menutype = 0;
 }
 else
 {
  bandmode--;
  if (bandmode>=128) bandmode = 0;
  messagetimer = 0;
  messagetype = 0;
 }
 if (DISPSPD) displaydelay--;
}

void EncoderActions()  // and buttons
{
   if (!BB) oldpos=0;  // if low

   if ((BB) && (oldpos==0)) // if hi and low before, positive edge
   {
    if (BA) go_up();
    else go_down();
    if (displaydelay>=80) displaydelay = 80;
    if (displaydelay<=2) displaydelay = 2;
    Delay(5);    // debounce delay
    oldpos = 1;  // mark that it's hi
   }
   if (!BMENU) oldmenu = 0;
   if ((BMENU) && (oldmenu==0))
   {
    oldmenu = 1;
    MENU = !MENU;
   }
}


void DisplaySpeed()
{
```

```
 SLUT = FALSE;
 DISPSPD = TRUE;
 LCDgoto(CLEAR);
 LCDskriver(9);
 while(!SLUT)
 {
  if (!BSEL) SLUT = TRUE;
  EncoderActions();
  LCDgoto(LINE2);
  konv = displaydelay;
  KONVERTER();
  LCDskriv1(user3);
  LCDskriv1(user4);
  LCDskriv1('m');
  LCDskriv1('S');  LCDskriv1(' ');  LCDskriv1(' ');  LCDskriv1(' ');
  LCDskriv1(' ');  LCDskriv1(' ');  LCDskriv1(' ');
  LCDskriv1(' ');  LCDskriv1('S');  LCDskriv1('e');  LCDskriv1('l');
  LCDskriv1('e');  LCDskriv1('c');  LCDskriv1('t');
  LCDskriv1('=');  LCDskriv1('O');  LCDskriv1('K');
 }
 DISPSPD = FALSE;
 eeprom_write(10,displaydelay);  // save value in EEPROM for all eternaty
}



void MenuEnd()
{
  MENU = 0;  // exit menu
  Delay(20);
  while (!BSEL) CLRWDT(); // wait intil select button released
  Delay(20);
}

void MenuActions()
{
 LCDgoto(CLEAR);
 LCDgoto(LINE1);
 LCDskriv1('M'); LCDskriv1('e'); LCDskriv1('n'); LCDskriv1('u');
 LCDskriv1(':'); LCDskriv1(' ');
 LCDgoto(LINE2);
 LCDskriver(19);  // TEXT : Select = Activate
 while (MENU)
 {
  CLRWDT();
  GetADC();
  EncoderActions();
  LCDgoto(0x85);
  if (menutype==0)
  {
   LCDskriv1(' '); LCDskriv1('0'); LCDskriv1(' ');
   LCDskriv1('A'); LCDskriv1('t'); LCDskriv1('t');
   LCDskriv1('='); LCDskriv1('0'); LCDskriv1('d');
   LCDskriv1('B'); LCDskriv1(' '); LCDskriv1(' ');
   LCDskriv1(' '); LCDskriv1(' '); LCDskriv1(' ');
   if (BSEL==0)  // the select button
   {
     metertype = RF;
     attmode = 0;
     eeprom_write(11,0);  // αποθήκευση setting εξασθενιτη
```

```
    MenuEnd();
  }
}
else
if (menutype==1)
{
  LCDskriv1(' '); LCDskriv1('1'); LCDskriv1(' ');
  LCDskriv1('A'); LCDskriv1('t'); LCDskriv1('t');
  LCDskriv1('='); LCDskriv1('-'); LCDskriv1('1');
  LCDskriv1('0'); LCDskriv1('d'); LCDskriv1('B');
  LCDskriv1(' '); LCDskriv1(' '); LCDskriv1(' ');
  if (BSEL==0)  // the select button
  {
    metertype = RF;
    attmode = 1;
    eeprom_write(11,1);  // αποθήκευση setting εξασθενιτη
    MenuEnd();
  }
}
else
if (menutype==2)
{
  LCDskriv1(' '); LCDskriv1('2'); LCDskriv1(' ');
  LCDskriv1('A'); LCDskriv1('t'); LCDskriv1('t');
  LCDskriv1('='); LCDskriv1('-'); LCDskriv1('2');
  LCDskriv1('0'); LCDskriv1('d'); LCDskriv1('B');
  LCDskriv1(' '); LCDskriv1(' '); LCDskriv1(' ');
  if (BSEL==0)  // the select button
  {
    metertype = RF;
    attmode = 2;
    eeprom_write(11,2);  // αποθήκευση setting εξασθενιτη
    MenuEnd();
  }
}
else
if (menutype==3)
{
  LCDskriv1(' '); LCDskriv1('3'); LCDskriv1(' ');
  LCDskriv1('A'); LCDskriv1('t'); LCDskriv1('t');
  LCDskriv1('='); LCDskriv1('-'); LCDskriv1('3');
  LCDskriv1('0'); LCDskriv1('d'); LCDskriv1('B');
  LCDskriv1(' '); LCDskriv1(' '); LCDskriv1(' ');
  if (BSEL==0)  // the select button
  {
    metertype = RF;
    attmode = 3;
    eeprom_write(11,3);  // αποθήκευση setting εξασθενιτη
    MenuEnd();
  }
}
else
if (menutype==4)
{
  LCDskriv1(' '); LCDskriv1('4'); LCDskriv1(' ');
  LCDskriv1('A'); LCDskriv1('t'); LCDskriv1('t');
  LCDskriv1('='); LCDskriv1('-'); LCDskriv1('4');
  LCDskriv1('0'); LCDskriv1('d'); LCDskriv1('B');
  LCDskriv1(' '); LCDskriv1(' '); LCDskriv1(' ');
  if (BSEL==0)  // the select button
```

```
  {
    metertype = RF;
    attmode = 4;
    eeprom_write(11,4);  // αποθήκευση setting εξασθενιτη
    MenuEnd();
  }
}
else
if (menutype==5)
{
  LCDskriv1(' '); LCDskriv1('5'); LCDskriv1(' ');
  LCDskriv1('A'); LCDskriv1('t'); LCDskriv1('t');
  LCDskriv1('='); LCDskriv1('-'); LCDskriv1('5');
  LCDskriv1('0'); LCDskriv1('d'); LCDskriv1('B');
  LCDskriv1(' '); LCDskriv1(' '); LCDskriv1(' ');
  if (BSEL==0)  // the select button
  {
    metertype = RF;
    attmode = 5;
    eeprom_write(11,5);  // αποθήκευση setting εξασθενιτη
    MenuEnd();
  }
}
else
if (menutype==6)
{
  LCDskriv1(' '); LCDskriv1('6'); LCDskriv1(' ');
  LCDskriv1('D'); LCDskriv1('C'); LCDskriv1(' ');
  LCDskriv1('V'); LCDskriv1('o'); LCDskriv1('l');
  LCDskriv1('t'); LCDskriv1('M'); LCDskriv1('e');
  LCDskriv1('t'); LCDskriv1('e'); LCDskriv1('r');

  if (BSEL==0)  // το select button
  {
    metertype = DC;
    MenuEnd();
  }
}
else
if (menutype==7)
{
  LCDskriv1(' '); LCDskriv1('7'); LCDskriv1(' ');
  LCDskriv1('R'); LCDskriv1('F'); LCDskriv1(' ');
  LCDskriv1('P'); LCDskriv1('o'); LCDskriv1('w');
  LCDskriv1('.'); LCDskriv1('M'); LCDskriv1('e');
  LCDskriv1('t'); LCDskriv1('e'); LCDskriv1('r');

  if (BSEL==0)  // το select button
  {
    metertype = RF;
    MenuEnd();
  }
}
else
if (menutype==8)
{
  LCDskriv1(' '); LCDskriv1('8'); LCDskriv1(' ');
  LCDskriv1('S'); LCDskriv1('S'); LCDskriv1('B');
  LCDskriv1(' '); LCDskriv1('P'); LCDskriv1('E');
  LCDskriv1('P'); LCDskriv1(' '); LCDskriv1('W');
```

```
  LCDskriv1('a'); LCDskriv1('t'); LCDskriv1('t');

  if (BSEL==0)  // το select button
  {
    metertype = SSB;
    MenuEnd();
  }
}

else
if (menutype==9)
{
  LCDskriv1(' '); LCDskriv1('9'); LCDskriv1(' ');
  LCDskriv1('R'); LCDskriv1('T'); LCDskriv1('N');
  LCDskriv1('.'); LCDskriv1('L'); LCDskriv1('o');
  LCDskriv1('s'); LCDskriv1('s'); LCDskriv1(' ');
  LCDskriv1('S'); LCDskriv1('W'); LCDskriv1('R');

  if (BSEL==0)  // το select button
  {
    metertype = RTN;
    RelativeCal = rfraw;  // αποθηκευση τιμης relative zero
    MenuEnd();
  }
}
else
if (menutype==10)
{
  LCDskriv1('1'); LCDskriv1('0'); LCDskriv1(' ');
  LCDskriv1('C'); LCDskriv1('a'); LCDskriv1('l');
  LCDskriv1(' '); LCDskriv1('0'); LCDskriv1('d');
  LCDskriv1('B'); LCDskriv1('m'); LCDskriv1(' ');
  WriteBand();
  if (BSEL==0)  // το select button
  {
   Calibrate();
   metertype = RF;
   MENU = 0;
  }
}
else
if (menutype==11)
{
  LCDskriv1('1'); LCDskriv1('1'); LCDskriv1(' ');
  LCDskriv1('R'); LCDskriv1('e'); LCDskriv1('a');
  LCDskriv1('d'); LCDskriv1(' '); LCDskriv1('C');
  LCDskriv1('a'); LCDskriv1('l'); LCDskriv1('D');
  LCDskriv1('a'); LCDskriv1('t'); LCDskriv1('a');
  if (BSEL==0)  // το select button
  {
   ReadCal();
   metertype = RF;
   MENU = 0;
  }
}
else
if (menutype==12)
{
  LCDskriv1('1'); LCDskriv1('2'); LCDskriv1(' ');
  LCDskriv1('Z'); LCDskriv1('e'); LCDskriv1('r');
```

```
    LCDskriv1('o'); LCDskriv1(' '); LCDskriv1('A');
    LCDskriv1('l'); LCDskriv1('l'); LCDskriv1(' ');
    LCDskriv1('C'); LCDskriv1('a'); LCDskriv1('l');
    if (BSEL==0)  // the select button
    {
     ZeroCal();  // ends with a watchdog reset !
    }
   }
   else
   if (menutype==13)
   {
    LCDskriv1('1'); LCDskriv1('3'); LCDskriv1(' ');
    LCDskriv1('D'); LCDskriv1('i'); LCDskriv1('s');
    LCDskriv1('p'); LCDskriv1('l'); LCDskriv1('a');
    LCDskriv1('y'); LCDskriv1('S'); LCDskriv1('p');
    LCDskriv1('e'); LCDskriv1('e'); LCDskriv1('d');
    if (BSEL==0)  // the select button
    {
     MenuEnd();  // βεβαιωση ότι το select button είναι ελευθερο
     DisplaySpeed();
     metertype = RF;
     MenuEnd();
    }
   }
   else
   if (menutype==14)
   {
    LCDskriv1('1'); LCDskriv1('4'); LCDskriv1(' ');
    LCDskriv1('A'); LCDskriv1('b'); LCDskriv1('o');
    LCDskriv1('u'); LCDskriv1('t'); LCDskriv1(' ');
    LCDskriv1('i'); LCDskriv1('n'); LCDskriv1('f');
    LCDskriv1('o'); LCDskriv1(' '); LCDskriv1(' ');
    if (BSEL==0)  // the select button
    {
      metertype = ABOUT;
      MenuEnd();
    }
   }
  }  // end while menu
  LCDgoto(CLEAR);  // at exit menu
}
```

**Σχετικα με τις ενδειξεις help**

```
void InfoWait()
{
 user3 = 20; // screen wait time
 while ((user3>=1)&&(SLUT))
 {
  if ((!BSEL)||(!BMENU))  // the select button
  {
   SLUT = 0;
  }
  Delay100(1);
  user3--;
 }
 while ((!BSEL)||(!BMENU)) CLRWDT();  // αναμονή για ελευθέρωση διακόπτη
}
```

```
void WriteAbout()
{
 SLUT = 1;
 while (SLUT)
 {
  LCDgoto(LINE1);
  LCDskriver(7);
  LCDgoto(LINE2);
  LCDskriver(8);
  InfoWait();

  LCDgoto(LINE1);
  LCDskriver(15);
  LCDgoto(LINE2);
  LCDskriver(16);
  InfoWait();
 }  // end while about
 metertype = RF;  // επιλογη RF powermeter mode μετα την about screen
}
```

**Ρουτίνες για μετρήσεις**

```
void WriteWattmeter()
{
   if (rfraw2<=99)
   {
    wpointer=rfraw2;  // pointer to Watt table 0-100 positions
    wpref='p';       // pico Watt
    wpoint = 3;      // where to place the point .
   }
   else
   if (rfraw2<=199)
   {
    wpointer=rfraw2-100;  // pointer to Watt table 0-100 positions
    wpref='p';       // pico Watt
    wpoint = 4;      // where to place the point .
   }
   else
   if (rfraw2<=299)
   {
    wpointer=rfraw2-200;  // pointer to Watt table 0-100 positions
    wpref='n';       // pico Watt
    wpoint = 2;      // where to place the point .
   }
   else
   if (rfraw2<=399)
   {
    wpointer=rfraw2-300;  // pointer to Watt table 0-100 positions
    wpref='n';       // nano Watt
    wpoint = 3;      // where to place the point .
   }
   else
   if (rfraw2<=499)
   {
```

```
  wpointer=rfraw2-400;  // pointer to Watt table 0-100 positions
  wpref='n';       // nano Watt
  wpoint = 4;      // where to place the point .
}
else
if (rfraw2<=599)
{
  wpointer=rfraw2-500;  // pointer to Watt table 0-100 positions
  wpref='u';       // micro Watt
  wpoint = 2;      // where to place the point .
}
else
if (rfraw2<=699)
{
  wpointer=rfraw2-600;  // pointer to Watt table 0-100 positions
  wpref='u';       // micro Watt
  wpoint = 3;      // where to place the point .
}
else
if (rfraw2<=799)
{
  wpointer=rfraw2-700;  // pointer to Watt table 0-100 positions
  wpref='u';       // micro Watt
  wpoint = 4;      // where to place the point .
}
else
if (rfraw2<=899)
{
  wpointer=rfraw2-800;  // pointer to Watt table 0-100 positions
  wpref='m';       // milli Watt
  wpoint = 2;      // where to place the point .
}
else
if (rfraw2<=999)
{
  wpointer=rfraw2-900;  // pointer to Watt table 0-100 positions
  wpref='m';       // milli Watt
  wpoint = 3;      // where to place the point .
}
else
if (rfraw2<=1099)
{
  wpointer=rfraw2-1000;  // pointer to Watt table 0-100 positions
  wpref='m';       // milli Watt
  wpoint = 4;      // where to place the point .
}
else
if (rfraw2<=1199)
{
  wpointer=rfraw2-1100;  // pointer to Watt table 0-100 positions
  wpref=' ';       // Watt
  wpoint = 2;      // where to place the point .
}
else
if (rfraw2<=1299)
{
  wpointer=rfraw2-1200;  // pointer to Watt table 0-100 positions
  wpref=' ';       //  Watt
  wpoint = 3;      // where to place the point .
}
```

```
   else
   if (rfraw2<=1399)
   {
    wpointer=rfraw2-1300;  // pointer to Watt table 0-100 positions
    wpref=' ';       // Watt
    wpoint = 4;      // where to place the point .
   }
   else
   if (rfraw2<=1499)
   {
    wpointer=rfraw2-1400;  // pointer to Watt table 0-100 positions
    wpref='k';       // Watt
    wpoint = 2;      // where to place the point .
   }
   else
   if (rfraw2<=1599)
   {
    wpointer=rfraw2-1500;  // pointer to Watt table 0-100 positions
    wpref='k';       // Watt
    wpoint = 3;      // where to place the point .
   }
   else
   if (rfraw2<=1699)
   {
    wpointer=rfraw2-1600;  // pointer to Watt table 0-100 positions
    wpref='k';       // Watt
    wpoint = 4;      // where to place the point .
   }
    w1=wten[wpointer];  // get absolutte part from table
    w2=wdec[wpointer];  // get decimal part from table
    konv = 0;
    konv = w1;
    KONVERTER();
    if (wpoint==4) LCDskriv1(' ');
    LCDskriv1(user4);
    if (wpoint==2) LCDskriv1('.');  // if this is the right place, write the point
    konv = w2;
    KONVERTER();
    LCDskriv1(user3);
    if (wpoint==3) LCDskriv1('.');  // if this is the right place, write the point
    LCDskriv1(user4);
    LCDskriv1(wpref);
    LCDskriv1('W');
}


void BarGraph()
{
   user2 = 1;  // carracter pointer
   user1 = (rfraw/10);  // now max rf level is 100
   while (user1>=5)  // how meny full black carracters to draw
   {
    LCDskriv1(4);
    user1=user1-5;
    user2++;
   }
   if (user1==5) LCDskriv1(4);  // Draw the last carracter
   else
   if (user1==4) LCDskriv1(3);
   else
```

```
   if (user1==3) LCDskriv1(2);
   else
   if (user1==2) LCDskriv1(1);
   else
   if (user1==1) LCDskriv1(0);
}


void BarEraseRest()
{
   while (user2<=20)  // erase the rest
   {
    LCDskriv1(' ');
    user2++;
   }
}

void BarGraphHI()  // write the hight part
{
   WriteWattmeter();
   LCDskriv1(' ');

   LCDgoto(0xc7);

   user2 = 7;  // carracter pointer
   user1 = (rfraw/10);   // now max rf level is 100
   user1 = user1-user2*5;  // find real starting point
   while (user1>=5)  // how meny full black carracters to draw
   {
    LCDskriv1(4);
    user1=user1-5;
    user2++;
   }
   if (user1==5) LCDskriv1(4);  // Draw the last carracter
   else
   if (user1==4) LCDskriv1(3);
   else
   if (user1==3) LCDskriv1(2);
   else
   if (user1==2) LCDskriv1(1);
   else
   if (user1==1) LCDskriv1(0);
}


void WriteRFVolt()
{
   if (rfraw2<=199)
   {
    vpointer=rfraw2;  // pointer to Watt table 0-100 positions
    vpref='u';       // Mirco Volt
    vpoint = 3;      // where to place the point .
   }
   else
   if (rfraw2<=399)
   {
    vpointer=rfraw2-200;  // pointer to Watt table 0-100 positions
    vpref='m';       // milli Volt
    vpoint = 1;      // where to place the point .
   }
```

```
else
if (rfraw2<=599)
{
  vpointer=rfraw2-400;  // pointer to Watt table 0-100 positions
  vpref='m';       // milli Volt
  vpoint = 2;      // where to place the point .
}
else
if (rfraw2<=799)
{
  vpointer=rfraw2-600;  // pointer to Watt table 0-100 positions
  vpref='m';       // milli Volt
  vpoint = 3;      // where to place the point .
}
else
if (rfraw2<=999)
{
  vpointer=rfraw2-800;  // pointer to Watt table 0-100 positions
  vpref=' ';       // Volt
  vpoint = 1;      // where to place the point .
}
else
if (rfraw2<=1199)
{
  vpointer=rfraw2-1000;  // pointer to Watt table 0-100 positions
  vpref=' ';       // Volt
  vpoint = 2;      // where to place the point .
}
else
if (rfraw2<=1399)
{
  vpointer=rfraw2-1200;  // pointer to Watt table 0-100 positions
  vpref=' ';       // Volt
  vpoint = 3;      // where to place the point .
}
else
if (rfraw2<=1599)
{
  vpointer=rfraw2-1400;  // pointer to Watt table 0-100 positions
  vpref='k';       // Volt
  vpoint = 1;      // where to place the point .
}
else
if (rfraw2<=1799)
{
  vpointer=rfraw2-1600;  // pointer to Watt table 0-100 positions
  vpref='k';       // Volt
  vpoint = 2;      // where to place the point .
}


  v1=vten[vpointer];  // get absolutte part from table
  v2=vdec[vpointer];  // get decimal part from table
  konv = v1;
  KONVERTER();
  if (vpoint==0)
  {
    LCDskriv1('0');
    LCDskriv1('.');  // if this is the right place, write the point
  }
```

```
    LCDskriv1(user2);
    if (vpoint==1) LCDskriv1('.');  // if this is the right place, write the point
    LCDskriv1(user3);
    if (vpoint==2) LCDskriv1('.');  // if this is the right place, write the point
    LCDskriv1(user4);
    if (vpoint==3) LCDskriv1('.');  // if this is the right place, write the point

    konv = v2;
    KONVERTER();
    LCDskriv1(user3);
    LCDskriv1(vpref);
    LCDskriv1('V');
}


void WriteDBM()
{
 if (RELATIVE)
 {
  if (rfraw>=RelativeCal)
  {
   rfdbm = rfraw-RelativeCal;
   dbswr = 0;  // the return loss and swr value does not have a meaning at +dB
   if (rfdbm>6) SWRERROR = TRUE;
   else SWRERROR = FALSE;
   LCDskriv1('+');  // this can also be changed to a space
  }
  else
  {
   rfdbm = RelativeCal-rfraw;
   dbswr = rfdbm/10;
   SWRERROR = FALSE;
   LCDskriv1('-');
  }
 }
 else  // absolute dBm read out
 {
  if (attmode==0) rfRawAtt=rfraw;  // add nothing if no att. is mounted
  if (attmode==1) rfRawAtt=rfraw+100;  // add 10dB if 10dB att. is mounted
  if (attmode==2) rfRawAtt=rfraw+200;  // add 20dB if 20dB att. is mounted
  if (attmode==3) rfRawAtt=rfraw+300;  // add 30dB if 30dB att. is mounted
  if (attmode==4) rfRawAtt=rfraw+400;  // add 40dB if 40dB att. is mounted
  if (attmode==5) rfRawAtt=rfraw+500;  // add 50dB if 50dB att. is mounted

  if (rfRawAtt>=ActualCal)
  {
   rfdbm = rfRawAtt-ActualCal;
   rfraw2 = 800 + rfdbm;  // calibrated raw to volt and power
   LCDskriv1('+');  // this can also be changed to a space
  }
  else
  {
   rfdbm = ActualCal-rfRawAtt;
   rfraw2 = 800 - rfdbm;  // used for the look-up tabels
   LCDskriv1('-');
  }
 }
 konv = rfdbm;
 KONVERTER();
 LCDskriv1(user2); LCDskriv1(user3);
```

```
  LCDskriv1('.');  LCDskriv1(user4);  LCDskriv1('d');  LCDskriv1('B');

  if (RELATIVE) LCDskriv1(' ');
  else LCDskriv1('m');
}


void WriteDCmeter()
{
   konv = voltraw*2;
   KONVERTER();
   if (user1=='0')  LCDskriv1(' ');
   else
   LCDskriv1(user1);
   LCDskriv1(user2);  LCDskriv1('.');
   LCDskriv1(user3); LCDskriv1(user4);
   LCDskriv1('V');  LCDskriv1(' ');
   LCDskriv1(' ');  LCDskriv1('S'); LCDskriv1('e'); LCDskriv1('l');
   LCDskriv1('e');  LCDskriv1('c'); LCDskriv1('t');
   LCDskriv1('=');  LCDskriv1('R'); LCDskriv1('e'); LCDskriv1('s');
   LCDskriv1('e');  LCDskriv1('t');

   if (BSEL==0)  // the select button
   {
     voltmin=voltraw;
     voltmax=voltraw;
   }

   if (voltmin>voltraw) voltmin=voltraw;
   if (voltmax<voltraw) voltmax=voltraw;
   LCDgoto(LINE2);
   LCDskriv1('M');    LCDskriv1('i');    LCDskriv1('n'); LCDskriv1('=');
   konv = voltmin*2;
   KONVERTER();
   if (user1=='0')  LCDskriv1(' ');
   else
   LCDskriv1(user1);
   LCDskriv1(user2);  LCDskriv1('.');
   LCDskriv1(user3); LCDskriv1(user4);

   LCDskriv1(' '); LCDskriv1(' ');

   LCDskriv1('M');    LCDskriv1('a');    LCDskriv1('x'); LCDskriv1('=');
   konv = voltmax*2;
   KONVERTER();
   if (user1=='0')  LCDskriv1(' ');
   else
   LCDskriv1(user1);
   LCDskriv1(user2);  LCDskriv1('.');
   LCDskriv1(user3); LCDskriv1(user4);

}


void WriteRFmeter()
{
   WriteDBM();
   if (!BSEL)
   {
     while (!BSEL) CLRWDT();  // wait for select to be released
```

```
    RELATIVE=!RELATIVE;     // togle relative bit
    if (RELATIVE) RelativeCal=rfraw;  // store relative zero value
    Delay(20);
   }
   EncoderActions();
   LCDskriv1(' ');
   if (RELATIVE)
   {
    LCDskriv1(' ');
    LCDskriv1('R');
    LCDskriv1('E');
    LCDskriv1('L');
    LCDskriv1('A');
    LCDskriv1('T');
    LCDskriv1('I');
    LCDskriv1('V');
    LCDskriv1('E');
    LCDskriv1(' ');
    LCDskriv1(' ');
    LCDgoto(LINE2);
    BarGraph(); // and no Wattmeter
    BarEraseRest();
   }
   else
   {
    LCDskriv1(message1);
    LCDskriv1(message2);
    LCDskriv1(message3);
    LCDskriv1(' ');
    EncoderActions();
    WriteRFVolt();
    EncoderActions();
    LCDskriv1(' ');
    LCDgoto(LINE2);
    if (rfraw>=575) BARGLO = TRUE;
    if (rfraw<=545) BARGLO = FALSE;
    EncoderActions();
    if (BARGLO)
    {
     BarGraphHI();  // and Wattmeter
     BarEraseRest();
    }
    else
    {
     BarGraph();
     while (user2<=14)  // erase the rest
     {
      LCDskriv1(' ');
      user2++;
     }
     LCDgoto(0xce);
     WriteWattmeter();
     LCDskriv1(' ');
    }
   }
}



void WriteSSBmeter()
```

```
{
 WriteDBM();
 EncoderActions();
 LCDskriv1(' ');
 LCDskriv1(message1);
 LCDskriv1(message2);
 LCDskriv1(message3);
 LCDskriv1(' ');
 WriteWattmeter();

 LCDgoto(LINE2);
 BarGraph(); // and no Wattmeter
 BarEraseRest();

 if (peakholddelay >= 10) peakholddelay--;
 else
 {
  if (peak >=10) peak--;  // decay, make sure the peak value can not go under 10
 }
}



void WriteRTNmeter()
{
  RELATIVE = 1;
  WriteDBM();
  if (!BSEL)
  {
    while (!BSEL) CLRWDT();  // wait for select to be released
    RelativeCal = rfraw;  // store relative zero value
    Delay(20);
  }

  LCDskriv1(' ');
  LCDskriv1('R');
  LCDskriv1('E');
  LCDskriv1('T');
  LCDskriv1('U');
  LCDskriv1('R');
  LCDskriv1('N');
  LCDskriv1(' ');
  LCDskriv1('L');
  LCDskriv1('O');
  LCDskriv1('S');
  LCDskriv1('S');

  LCDgoto(LINE2);
  if (dbswr<=60);  // check value is within table size, else crash !!
  {
    user1 = swrconv[dbswr];  // get swr value from table
    konv = user1;
    KONVERTER();
    user5 = '.';
    user2++;  // add 1 to the hundred
    user1=' ';
  }
  if (dbswr>=61)
  {
```

```
    user1='-'; user2='-';
    user5='.';
    user3='-'; user4='-';
   }
   if (dbswr==0) { user1='C'; user2='A'; user5='L'; user3='E'; user4='D'; }
   else
   if (dbswr==1) { user1='1'; user2='7'; user5='.'; user3='3'; user4='9'; }
   else
   if (dbswr==2) { user1=' '; user2='8'; user5='.'; user3='7'; user4='2'; }
   else
   if (dbswr==3) { user1=' '; user2='5'; user5='.'; user3='8'; user4='4'; }
   else
   if (dbswr==4) { user1=' '; user2='4'; user5='.'; user3='4'; user4='2'; }
   else
   if (dbswr==5) { user1=' '; user2='3'; user5='.'; user3='5'; user4='7'; }
   if (SWRERROR) { user1='E'; user2='R'; user5='R'; user3='O'; user4='R'; }

   LCDskriv1(user1); LCDskriv1(user2);
   LCDskriv1(user5);
   LCDskriv1(user3); LCDskriv1(user4);
   LCDskriv1(' ');
   LCDskriv1('S');
   LCDskriv1('W');
   LCDskriv1('R');
   LCDskriv1(' ');
   LCDskriv1(' ');
   LCDskriv1(' ');
   LCDskriv1('S');
   LCDskriv1('E');
   LCDskriv1('L');
   LCDskriv1('=');
   LCDskriv1('Z');
   LCDskriv1('E');
   LCDskriv1('R');
   LCDskriv1('O');
}
```

**Hardware και μεταβλητές**

```
void INIThardware()
{
  TRISB = 0b11100001;  // DISP DATA OUTPUT
  TRISC = 0b00000000;
  PORTC = 0b01000000;  // TX = 1
  RBPU = 0;  // port B pull up
  // INIT ADC
  ADCON1 = 0b11100010;       // 10 bit convert ADRESH,ADRESL, PORTA=Analog VCC=REF
(PORTE=Digital)
  ADCON0 = 0b01000001;    // 8T clock, Analog input RA0, adon
  PCFG3 = 0;
  PCFG2 = 1;
  PCFG1 = 0;
  PCFG0 = 1;  // AN3 is Vref AN0 and AN1 analog rest digital
  CHS2 = 0; CHS1 = 0;
  CHS0 = 1;  // Select AN1 input RF Level
  CLRWDT();

 // get cal settings from EEPROM !!!
  user1 = eeprom_read(0);  // get first cal
```

```
 if ((user1==0)||(user1==255))  // check if data is bad
 {
  eeprom_write(0,66);  // erase to normal harware cal 666
  Delay(20);
  eeprom_write(1,66);  // erase to normal harware cal 666
  Delay(20);
  eeprom_write(2,68);  // erase to normal harware cal 668
  Delay(20);
  eeprom_write(3,64);  // erase to normal harware cal 664
  Delay(20);
  eeprom_write(4,62);  // erase to normal harware cal 662
  Delay(20);
  eeprom_write(10,50);  // erase to normal harware cal 50mS  displaydelay
  Delay(20);
  eeprom_write(11,0);  // erase to 0 dB attenuator
  Delay(20);
 }
 user1 = eeprom_read(0);  // get value from EEPROM byte size
 LFcal = 600 + user1;     // Add 600 to the byte from EEPROM
 user1 = eeprom_read(1);  // 600 was subtracted, before saving value in EEPROM
 HFcal = 600 + user1;
 user1 = eeprom_read(2);
 VHFcal = 600 + user1;
 user1 = eeprom_read(3);
 UHFcal = 600 + user1;
 user1 = eeprom_read(4);
 SHFcal = 600 + user1;
 displaydelay = eeprom_read(10);  // get value from EEPROM
 attmode = eeprom_read(11);
 Ser1(' '); Ser1(' '); Ser1(10); Ser1(13);
 Ser1('O'); Ser1('Z'); Ser1('2'); Ser1('C'); Ser1('P'); Ser1('U');
 Ser1(' '); Ser1('d'); Ser1('w'); Ser1('m'); Ser1('1'); Ser1('0'); Ser1('3');
 Ser1(10); Ser1(13);
 konv = LFcal;
 KONVERTER();
 Ser1(user2); Ser1(user3); Ser1(user4); Ser1('-');
 konv = HFcal;
 KONVERTER();
 Ser1(user2); Ser1(user3); Ser1(user4); Ser1('-');
 konv = VHFcal;
 KONVERTER();
 Ser1(user2); Ser1(user3); Ser1(user4); Ser1('-');
 konv = UHFcal;
 KONVERTER();
 Ser1(user2); Ser1(user3); Ser1(user4); Ser1('-');
 konv = SHFcal;
 KONVERTER();
 Ser1(user2); Ser1(user3); Ser1(user4);
 Ser1(10); Ser1(13);
}

void InitVars()
{
 message1 = ' ';
 message2 = ' ';
 message3 = ' ';
 ActualCal = 666;
 bandmode = 0;   // LF
 messagenumber = 0;
 messagetimer = 0;
```

```
  messagetype = 0;
  oldmenu = 1;  // at powerup no menu
  metertype = RF;
  RELATIVE = 0;
  MENU = 0;
}
```

### `Βολτόμετρο DC

Στην περίπτωση ανάστροφης πόλωσης, τίποτε δεν καίγεται. Στην οθόνη του βολτομέτρου απεικονίζονται η τρέχουσα τάση, όπως επίσης η μέγιστη και η ελάχιστη τιμή, ενώ με το πάτημα του πλήκτρου SELECT, οι ενδείξεις μεγίστου και ελαχίστου μηδενίζονται. Το βολτόμετρο μπορεί να χρησιμοποιηθεί για να παρακολουθείται η τάση της μπαταρίας (στην περίπτωση που χρησιμοποιείται μπαταρία για την τροφοδοσία). Λόγω έλλειψης χώρου και μπαταρίας δεν βγάλαμε εξόδους για το βολτόμετρο.

### Επί πλέον χαρακτηριστικά

Η ύπαρξη σειριακής εξόδου, Το PIC οδηγεί στον ακροδέκτη 17 μία σειριακή ακολουθία δεδομένων η οποία μπορεί να μετατραπεί σε στάθμες RS232 με την βοήθεια του ολοκληρωμένου MAX232 σε συνήθη συνδεσμολογία. Τα δεδομένα αυτά είναι δυνατόν στην συνέχεια να οδηγηθούν σε κάποια ελεύθερη σειριακή θύρα του υπολογιστή. Οποιοδήποτε πρόγραμμα επικοινωνίας ή εξομοίωσης τερματικού (όπως είναι το Hyper TerminaI των Windows) μπορεί να διαβάσει την ακολουθία των δεδομένων. Οι παράμετροι επικοινωνίας είναι: 38400 baud, 8 bit ASCII, όχι ισοτιμία, 1 ψηφίο παύσης. Εν συντομία: 38K4 8 N 1. Λόγω έλλειψης χώρου δεν βγάλαμε έξοδο σειριακή.

### Συμπέρασμα

Το μέγιστο σφάλμα μέτρησης του βαττομέτρου της κατασκευής είναι (-1.3 dB …. +0.4 dB) για ισχύ εισόδου από -50 μέχρι +10 dBm και περιοχή συχνοτήτων 10 – 400 MHz. Αυτή είναι η χρήσιμη περιοχή λειτουργίας του οργάνου.

Η οριακά μεγαλύτερη ακρίβεια, εύρος συχνοτήτων και δυναμικό εύρος των επαγγελματικών μετρητών ισχύος RF, απαιτούν ένα κόστος 20-40 φορές μεγαλύτερο από αυτό του οργάνου που έχουμε κατασκευάσει.

Σαν μελλοντική επέκταση θεωρούμε ότι θα μπορούσαν να συνδεθούν δύο πλακέτες εισόδου, όπου το λογισμικό θα αφαιρεί την είσοδο Β από την είσοδο Α για να απεικονίζει μαζί με την υπολογιζόμενη τιμή SWR την εξερχόμενη και την ανακλώμενη ισχύ.

Επίσης θα μπορούσε να μονωθεί καλύτερα ή το κουτί να ήταν μεταλλικό έτσι ώστε να προστατευόταν περισσότερο από εξωτερικές παρεμβολές.

Τέλος θα μπορούσαμε να αλλάξουμε την βαθμονόμηση έτσι ώστε να έχει μεγαλύτερη ακρίβεια σε συχνότητες μεγαλύτερες από τα 400MHz

**Προβλήματα που αντιμετωπίσαμε**

Το μεγαλύτερο πρόβλημα ήταν η παλαιότητα της κατασκευής. Τα περισσότερο σημαντικά υλικά δεν έβγαιναν πλέον στην παραγωγή η είχαν αντικατασταθεί από νεότερες εκδόσεις και κάποια απλά δεν τα έφερναν στην ελληνική αγορά.
Το Ελέκτορ που φιλοξένησε την κατασκευή αυτή δεν είχε πλέον τα υλικά οπότε η αγορά έγινε από το εξωτερικό με πολλά προβλήματα.
Πιο συγκεκριμένα ο PIC 16F876 σταμάτησε να παράγεται και αντικατασταθηκε με τον 16F876A ο οποίος έχει κάποιες τροποποιήσεις οι οποίες όμως δεν επηρεάζουν την κατασκευή μας
Ο περιστρεφόμενος επιλογεας της Bourns είχε σταματήσει να βγαίνει με αποτέλεσμα να βάλουμε έναν άλλο της ιδίας εταιρίας με τις εξής διαφορές. Είναι περισσοτέρων θέσεων και οι χρόνοι μετάβασης είναι γρηγορότεροι από ότι ο παλιός. Δηλαδή χρειάζεται να το γυρίζουμε πιο αργά για να <<πιάσει>> και να αλλάξει η ένδειξη
Τέλος αλλάξαμε την οθόνη από πράσινη σε μπλε με αποτέλεσμα να αλλάξουμε την αντίσταση της με μεγαλύτερη για να έχουμε το ίδιο αποτέλεσμα.

## Τα υλικά

**Αντιστάσεις**

SMD case 1206 or 0805:
R1,R2,R3 = 100Ω (R3 on top of R1/R2)
R4 = 39Ω
R5 = 33Ω
R6 = 68Ω
R7 = 47Ω
R8 = 470kΩ
R9 = 47kΩ
R10,R11 = 1kΩ
R15 = 120kΩ
R16 = 10kΩ
R17 = 180kΩ
R14 = 10Ω 1W
P1 = 10kΩ preset

**Πυκνωτές**

SMD case 1206 or 0805:
C1 = 8pF2
C2-C7,C9,C10,C11,C14 = 100nF
C8 = 1nF
C12 = 10µF 16V radial
C13 = 100µF 25V radial

**Πυνίο**

L1 = 3 turns, 0.5mm dia. ECW (SWG #30),
turns spaced at 0.5mm, internal dia. 3mm.

**Ημιαγωγοί**

D1 = 1N4001
IC1 = AD8307AR (SMD)
IC2 = PIC16F876-04/SP, programmed,
order code 020026-41, see Readers
Services page
IC3 = 7805

**Διάφορα**
K1 = 5-way SIL pinheader
K2,K5 = 3-way SIL pinheader
K3 = 16-way SIL pinheader
K4 = BNC socket with flange
K6 = 4-way SIL pinheader
S1,S2 = pushbutton, 1 make contact, chassis
mount
PC1,PC3,PC8,PC10,PC12 = solder pin
X1 = 4MHz ceramic resonator (3 pins)

LCD module with 2 lines of 20 characters,
e.g., LM032L (PC2002LRS-BEA-C)
Rotary encoder type 3315Y-1-016 (Bourns)
Mains adapter socket, chassis mount
IC socket, 28 pins, narrow

PCB, order code 020026-1 (see Readers
Services page).
Disk, source code files, order code 020026-
11 (files also available from Free Downloads)

## Περίληψη

Η εργασία αυτή αφορά στην κατασκευή ενός βατομέτρου RF για συχνότητες από 1KHz έως 500MHz .Πρόκειται για μια κατασκευή σχετικά απλή σε λειτουργία και χειρισμό που δίνει την δυνατότητα στον χρήστη να μετρήσει ισχύς σημάτων

Επειδή στην μέτρηση ισχύος επικρατούν πολλές μονάδες μέτρησης, ο μετρητής αυτός παρουσιάζει τα αποτελέσματα του σε όλες τις μονάδες αυτές (dBm Watt Watt rms).Ακόμα η παραμετροποίηση του βατόμετρο για κάθε μέτρηση, μηδενίζοντας τον κάθε φορά στην προς μέτρηση συχνότητα, εξασφαλίζει ακρίβεια στα αποτελέσματα του.

Επιπλέον περιγράφονται οι επιμέρους μονάδες του κυκλώματος όπως ο μετατροπέας RF,ο μικροελεγκτής, ο Analog to Digital μετατροπέας και η LCD οθόνη.

Επιπλέον την εργασία συμπληρώνουν αποτελέσματα εργαστηριακών μετρήσεων που παρουσιάζονται σε πίνακες και δίνουν μια εικόνα της ακρίβειας μέτρησης του βατομέτρου.

Τέλος επειδή όπως αναφέρθηκε προηγουμένως το όργανο αυτό λειτουργεί με μικροελεγκτής έχουμε συμπεριλάβει τον κώδικα με τον οποίο είναι προγραμματισμένος ο PIC δίνοντας μερικές εξηγήσεις σχετικές με τον τρόπο λειτουργίας του.

# FEW WORDS ABOUT THE THESIS

This thesis regards to the construction of a RF wattmeter for frequencies varying from 1 KHz to 500MHz. The specific wattmeter is actually a device quite simple in operation and handling which gives the operator the ability to measure signal powers

Because in power measuring there are many measures such as Watts. dBm, etc. this instrument produces its results to all sorts of values. In addition the fact that the wattmeter can be calibrated before each measurent and be parameterized for the specific frequency that it is about to test reassures us that its results will be as accurate as possible.

The different sub-units of the meter (PIC, analog to digital converter, LCD screen, RF converter) are also presented and explained in the text.

Furthermore the thesis is completed with results from laboratory measurements that are presented in graphs and tables so that the accuracy of the metering capability of the instrument can be evaluated.

Finally because the wattmeter uses a PIC to calculate and present results we have included the code with which the PIC is programmed supplemented with some basic explanations.

## Βιβλιογραφία

Ελέκτορ Νοέμβριος 2002 (αριθ. τεύχους 243)

www.webx.dk

http://jaichi.virtualave.net/

www.ic-prog.com

www.htsoft.com

Datasheets AD8307, PIC16F876