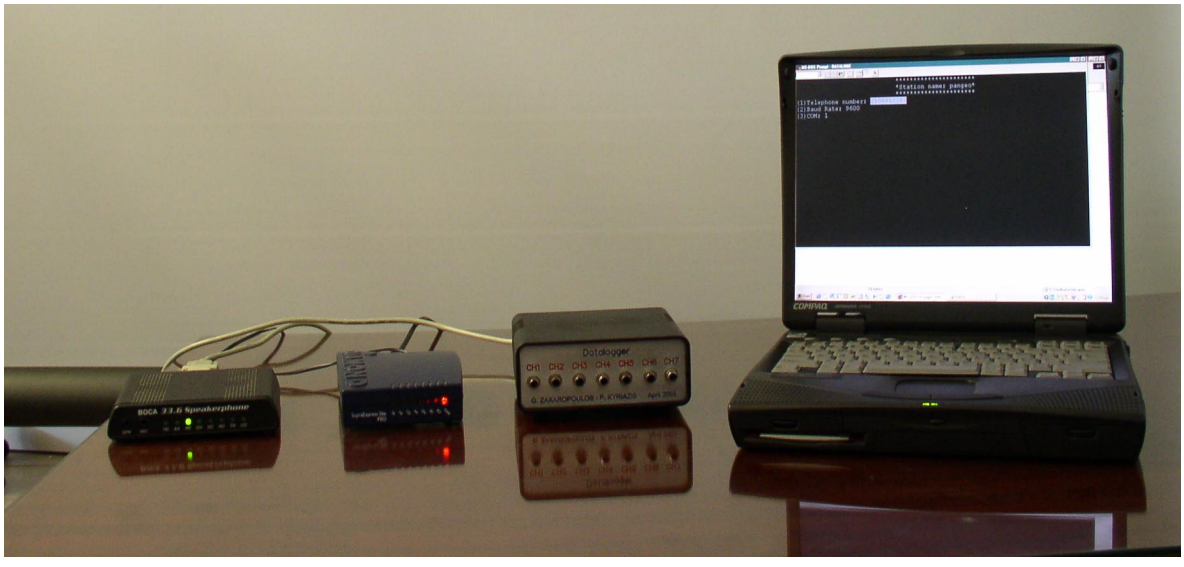
ΤΕΙ ΑΘΗΝΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

ΖΑΧΑΡΟΠΟΥΛΟΣ
ΓΕΩΡΓΙΟΣ
ΑΜ: 3362

ΚΥΡΙΑΖΗΣ
ΠΑΝΑΓΙΩΤΗΣ
ΑΜ: 3303

# ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

## ΜΕΛΕΤΗ, ΣΧΕΔΙΑΣΗ ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΟΛΟΚΛΗΡΩΜΕΝΟΥ ΣΥΣΤΗΜΑΤΟΣ ΤΗΛΕΜΕΤΡΙΑΣ ΜΕ ΤΗ ΜΕΘΟΔΟ DATALOGGING ΓΙΑ ΤΗ ΜΕΤΡΗΣΗ ΤΩΝ ΜΕΤΑΒΟΛΩΝ ΤΟΥ ΗΛΕΚΤΡΟΜΑΓΝΗΤΙΚΟΥ ΠΕΔΙΟΥ ΤΗΣ ΓΗΣ




**ΕΠΙΒΛΕΠΩΝ: Κ.ΝΟΜΙΚΟΣ**

# Περιεχόμενα

# Πρόλογος

Το έντυπο αυτό αποτελεί την πτυχιακή εργασία των συγγραφέων και έχει θέμα την περιγραφή του συστήματος τηλεμετρίας με τη μέθοδο datalogging, που αναπτύχθηκε για την μέτρηση των μεταβολών του ηλεκτρομαγνητικού πεδίου της Γης πριν τους σεισμούς.

Το κείμενο χωρίζεται σε οκτώ κεφάλαια, εκ των οποίων το πρώτο περιέχει εισαγωγικά ζητήματα σχετικά με την τηλεμετρία και το δεύτερο αξιόλογα συστήματα της αγοράς, ανάλογα του αναπτυχθέντος, ώστε να είναι εφικτή η σύγκριση από τον αναγνώστη. Στα υπόλοιπα έξι κεφάλαια περιγράφονται τα βασικότερα σημεία της σχεδίασης και υλοποίησης του συστήματος, καθώς και τα αποτελέσματα και συμπεράσματα που προέκυψαν από την εργασία. Ιδιαίτερη έμφαση έχει δοθεί στην πληρότητα και την ακρίβεια των σχετικών με τα χαρακτηριστικά και τη χρήση του συστήματος πληροφοριών, ώστε το παρόν να μπορεί να αποτελέσει και εγχειρίδιο χρήσης του συστήματος.

Υπάρχουν επίσης δύο παραρτήματα που περιέχουν τα χαρακτηριστικά του datalogger του συστήματος και τον πηγαίο κώδικα του λογισμικού των σταθμών βάσης και υπαίθρου.

Στο σημείο αυτό θέλουμε να ευχαριστήσουμε τον καθηγητή μας κ. Κ.Νομικό και τους Ηλεκτρονικούς της Υ.Π.Α για τις τεχνικές συμβουλές και την αρωγή τους στην ολοκλήρωση της εργασίας.

Μάιος 2003

Ζαχαρόπουλος Γεώργιος
Κυριαζής Παναγιώτης

# ΚΕΦΑΛΑΙΟ 1$^{Ο}$ : ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΤΗΛΕΜΕΤΡΙΑ

Τηλεμετρία ονομάζεται η επιστήμη που ασχολείται με τη μεταφορά μετρήσεων φυσικών μεγεθών από ένα σταθμό μέτρησης σε ένα κεντρικό σταθμό. Οι σταθμοί μέτρησης που βρίσκονται συνήθως στην ύπαιθρο μπορεί να είναι σταθμοί μετεωρολογικοί, σεισμολογικοί, θερμοκήπια ή γενικά κάθε σταθμός που συλλέγει αναλογικά δεδομένα με αισθητήρες και ανιχνευτές, τα οποία στη συνέχεια αποστέλλονται στον κεντρικό σταθμό για περαιτέρω επεξεργασία και μελέτη. Η τηλεμετρία είναι από τις πλέον αναπτυσσόμενες τεχνολογίες που έχει να επιδείξει η Ηλεκτρονική και η Πληροφορική.

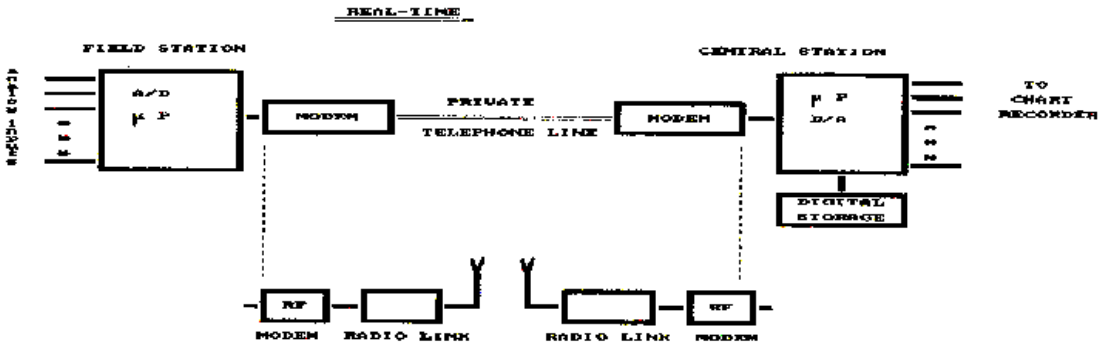
Η αναλογική τηλεμετρία που επικρατούσε πριν από μια εικοσιπενταετία είχε το πλεονέκτημα της απλότητας της σχεδίασης και του χαμηλού κόστους κατασκευής της, ωστόσο παρουσίαζε το μεγάλο μειονέκτημα της αδυναμίας εξάλειψης του θορύβου που παρεμβάλλεται στο μέσο μετάδοσης.

Με την ανάπτυξη της ψηφιακής τεχνολογίας παραγκωνίστηκε ο ρόλος των αναλογικών τηλεμετρικών συστημάτων και στη θέση τους αναπτύχθηκαν ψηφιακά τηλεμετρικά συστήματα, τα οποία χωρίζονται σε δυο μεγάλες κατηγορίες: α) Συστήματα πραγματικού χρόνου (Real time) και β) Συστήματα συλλογής δεδομένων στο σταθμό μέτρησης και εν συνεχεία αποστολής στον κεντρικό σταθμό (Datalogging).

## 1.1. Τηλεμετρία σε πραγματικό χρόνο.

Η τηλεμετρία σε πραγματικό χρόνο έχει το πλεονέκτημα της απ' ευθείας καταγραφής των δεδομένων του σταθμού μέτρησης στον κεντρικό σταθμό, ο οποίος παρακολουθεί σε πραγματικό χρόνο (real time) τις μεταβολές των καταγραφόμενων φυσικών μεγεθών. Η τεχνική αυτή, όμως, έχει το σοβαρό μειονέκτημα του υψηλού κόστους, καθώς για τη διασύνδεση του σταθμού μέτρησης με τον κεντρικό σταθμό απαιτείται η χρήση μισθωμένης γραμμής, της οποίας το κόστος είναι πολύ υψηλό.

Η δομή ενός συστήματος τηλεμετρίας σε πραγματικό χρόνο φαίνεται στο σχήμα 1.1.α που ακολουθεί και παρουσιάζει τη δομή τόσο του σταθμού μετρήσεων όσο και του κεντρικού σταθμού.

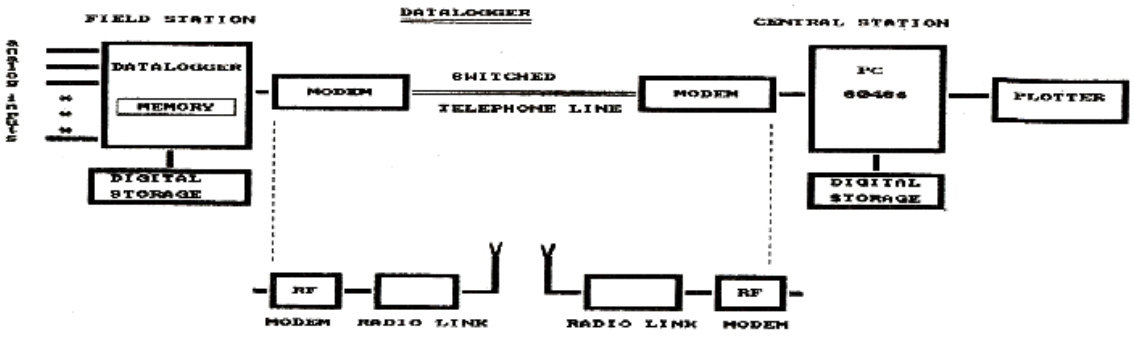



**Σχήμα 1.1.α** Τηλεμετρικό σύστημα πραγματικού χρόνου

## 1.2. Τηλεμετρία μέσω dataloggers.

Η τηλεμετρία με τη χρήση dataloggers υλοποιείται με τη λήψη δεδομένων στο σταθμό μέτρησης, τα οποία αποθηκεύονται στη μνήμη του ηλεκτρονικού υπολογιστή (datalogger) που βρίσκεται στο σταθμό αυτό. Η λήψη δεδομένων στον κεντρικό σταθμό από το σταθμό μέτρησης, γίνεται σε τακτά χρονικά διαστήματα με επικοινωνία μέσω κοινής τηλεφωνικής γραμμής. Το κυριότερο μειονέκτημα της τεχνικής αυτής είναι η αδυναμία αποθήκευσης μεγάλου όγκου πληροφορίας και εξ' αιτίας αυτού του χαρακτηριστικού της χρησιμοποιείται σε περιπτώσεις όπου ο όγκος πληροφοριών είναι μικρός. Στις περιπτώσεις αυτές είναι οικονομικά συμφερότερη σε σύγκριση με την τηλεμετρία πραγματικού χρόνου, καθώς το κόστος απασχόλησης μιας κοινής τηλεφωνικής γραμμής για μερικά λεπτά της ώρας ημερησίως, είναι σαφώς κατώτερο του κόστους μιας μισθωμένης γραμμής.

Στη τηλεμετρία με τη χρήση datalogger της οποίας η δομή φαίνεται στο σχήμα 1.2.α κρίσιμη είναι τόσο η δομή του σταθμού μέτρησης, όσο και η δομή του κεντρικού σταθμού. Η συνήθης δομή των δυο σταθμών αναλύεται παρακάτω.




**Σχήμα 1.2.α** Τηλεμετρικό δίκτυο μέσω dataloggers

### 1.2.1. Δομή του σταθμού μέτρησης

Η δομή του σταθμού μέτρησης είναι γενικότερα αυτή που σχηματικά αναπαριστάται στο σχήμα 1.2.α. Αναλυτικότερα διακρίνονται :

1) Οι αισθητήρες (sensors) που μετατρέπουν τις φυσικές παραμέτρους που μετρά ο σταθμός , σε αναλογικό ηλεκτρικό σήμα.

2) Σύστημα Η/Υ που στην είσοδό του λαμβάνει τα αναλογικά ηλεκτρικά σήματα από την έξοδο των αισθητήρων και τα μετατρέπει σε ψηφιακή πληροφορία.

Τα συστήματα αυτά ονομάζονται dataloggers και αποτελούνται από :

α) την κεντρική μονάδα μικροϋπολογιστή (CPU)

β) τη μνήμη RAM για την αποθήκευση προγραμμάτων και δεδομένων

γ) τη μνήμη ROM όπου αποθηκεύεται το λειτουργικό πρόγραμμα

δ) τη μονάδα αναλογικής πολυπλεξίας για τη μέτρηση περισσοτέρων του ενός αναλογικών καναλιών

ε) τη μονάδα μετατροπής του αναλογικού σήματος σε ψηφιακό (A/D converter) για να μπορεί η CPU να διαβάζει την ψηφιακή πληροφορία, να την επεξεργάζεται και να την καταχωρεί στη μνήμη του

στ) μια ή περισσότερες σειριακές πόρτες για επικοινωνία με modem, με μονάδες μαζικής αποθήκευσης και άλλα περιφερειακά

ζ) τις παράλληλες πόρτες εισόδου-εξόδου (I/O) για τον έλεγχο περιφερειακών.

Τα συστήματα datalogger, που χρησιμοποιούνται στους σταθμούς μέτρησης, πρέπει να λειτουργούν σε περιβάλλον multitasking, ώστε κατά τη διάρκεια της επικοινωνίας με τον κεντρικό σταθμό για την μεταφορά δεδομένων, να συνεχίζεται ταυτόχρονα η συλλογή, επεξεργασία και αποθήκευση νέων δεδομένων από τους αισθητήρες.

Την τελευταία εικοσαετία αρκετές ξένες εταιρίες κατασκευάζουν dataloggers για επιστημονικούς σκοπούς ή εφαρμογές της τηλεμετρίας, όπως η Campbell (USA), η Teledyne (USA), η Laplace Instruments Ltd (UK), κ.τ.λ. Κάθε κατασκευάστρια εταιρία, λόγω των επιμέρους ιδιαιτεροτήτων, εξειδικεύεται σε μια κυρίως επιστήμη. Για παράδειγμα τα dataloggers της Campbell προορίζονται κυρίως για τη μετεωρολογία, της Teledyne για τη σεισμολογία και της Laplace για εργαστηριακές μετρήσεις.

### 1.2.2. Δομή του Κεντρικού σταθμού

Ο κεντρικός σταθμός είναι ο αποδέκτης του συνόλου των πληροφοριών από τους σταθμούς μέτρησης. Στη δομή του κεντρικού σταθμού όπως σχηματικά απεικονίζεται στο σχήμα 1.2.α διακρίνονται :

α) ένα PC τρέχουσας τεχνολογίας, έτσι ώστε οι απεικονίσεις και οι πράξεις να γίνονται κατά το δυνατόν γρηγορότερα

β) ένα μέσο μόνιμης και μεγάλης αποθηκευτικής ικανότητας (zip disk ή hard disk)

γ) ένα modem και μια κοινή τηλεφωνική γραμμή για επικοινωνία με τους σταθμούς μέτρησης

δ) ένα χρονοδιακόπτη (timer) για να θέτει σε λειτουργία τον Η/Υ αν η επικοινωνία με τους σταθμούς μέτρησης χρειάζεται να γίνεται χωρίς παρακολούθηση το βράδυ, οπότε και οι γραμμές του σταθερού τηλεφωνικού δικτύου είναι φθηνότερες και λιγότερο κατειλημμένες

ε) έναν εκτυπωτή και έναν καταγραφέα (printer and plotter).

Είναι επομένως αντιληπτό ότι ο κεντρικός σταθμός αποτελείται από όργανα που υπάρχουν στην αγορά και μάλιστα σε σχετικά χαμηλή τιμή λόγω της ευρείας χρήσης τους.

Στον κεντρικό σταθμό το λογισμικό (software), που τρέχει ο Η/Υ, επικοινωνεί σε προκαθορισμένα χρονικά διαστήματα με τους σταθμούς μέτρησης και λαμβάνει τις μετρήσεις που έχουν αποθηκευθεί στα dataloggers. Το πρόγραμμα επικοινωνίας, πρέπει να διασφαλίζει με ειδικά πρωτόκολλα επικοινωνίας, ότι τα δεδομένα που λαμβάνει από το σταθμό υπαίθρου δεν εμφανίζουν σφάλματα, απαλείφοντας τις παρεμβολές που προέρχονται από το μέσο επικοινωνίας. Τα σύγχρονα modem έχουν αυτή τη δυνατότητα εντοπισμού και διόρθωσης λαθών (error correction). Η σημασία του προγράμματος του κεντρικού σταθμού είναι αντιληπτή με βάση τις παρακάτω απαιτήσεις που πρέπει να πληρούνται :

1) Διασφάλιση επικοινωνίας μεταξύ κεντρικού σταθμού και σταθμού μέτρησης σε κάθε περίπτωση, ανεξαρτήτως αν η σταθερή τηλεφωνική γραμμή είναι προσωρινά κατειλημμένη.

2) Διασφάλιση της ορθότητας των πληροφοριών που αποστέλλονται κατά την διάρκεια της επικοινωνίας σε ποσοστό 99.99%.

3) Διαδικασία συνένωσης των αρχείων των δεδομένων που έχουν αποθηκευθεί από την επικοινωνία με τους σταθμούς μέτρησης. Η διαδικασία είναι εξαιρετικά σημαντική, καθώς δημιουργεί ένα τελικό αρχείο διαχείρισης, το οποίο περιέχει όλες

τις μετρήσεις με την ημερομηνία και το χρόνο μέτρησης. Το πρόγραμμα λοιπόν πρέπει να είναι ικανό να ξεχωρίζει τις πληροφορίες της ίδιας χρονικής στιγμής που ελήφθησαν από τους σταθμούς μέτρησης, γιατί κάποιοι σταθμοί μέτρησης ήταν ενδεχομένως εκτός λειτουργίας για συγκεκριμένα χρονικά διαστήματα ή εξαιτίας λάθους κατά την επικοινωνία έχουν ξαναστείλει το ίδιο πακέτο δεδομένων.

4) Δυνατότητα γραφικής απεικόνισης στην οθόνη του Η/Υ των μετρήσεων συναρτήσει του χρόνου για τα ίδια αναλογικά κανάλια ή σε ομάδες καναλιών, από όλους τους σταθμούς μέτρησης, ώστε να είναι εφικτή η σύγκριση των ιδίων πειραματικών μετρήσεων από όλους τους σταθμούς. Επίσης απαραίτητη είναι η δυνατότητα εκτύπωσης των γραφικών παραστάσεων των δεδομένων στον εκτυπωτή ή τον καταγραφέα.

5) Δυνατότητα μαζικής αποθήκευσης των τελικών αρχείων δεδομένων σε περιφερειακή μονάδα (streamer ή οπτικού δίσκου), για περαιτέρω ανάλυση των δεδομένων.

6) Οι αναφερόμενες προδιαγραφές στα 1 και 2 απαιτείται να γίνονται απρόσκοπτα χωρίς παρακολούθηση του συστήματος κατά τις βραδινές ώρες.

### 1.3. Μέσο μεταβίβασης πληροφοριών

Η εγκατάσταση ενός τηλεμετρικού δικτύου με τη χρήση dataloggers, γίνεται αφού πρώτα αποφασιστεί το μέσο μεταβίβασης των δεδομένων, δηλαδή αν είναι προτιμότερο να χρησιμοποιηθεί ασύρματο δίκτυο ή δίκτυο σταθερής τηλεφωνίας.

Η απόφαση είναι δύσκολη και εξαρτάται από το πλήθος και τον όγκο της μεταδιδόμενης πληροφορίας. Αν ο όγκος δεδομένων είναι μικρός επιλέγεται η σταθερή τηλεφωνική γραμμή, ενώ εάν είναι μεγάλος, η λύση του ασυρμάτου δικτύου είναι πιο συμφέρουσα, όπως και η αλλαγή της τεχνικής μέτρησης στη μέθοδο real-time, με χρησιμοποίηση μισθωμένης γραμμής.

Σε αρκετές περιπτώσεις επιλέγεται ο συνδυασμός των δύο τεχνικών, έτσι ώστε να εξασφαλίζεται σε μεγαλύτερο βαθμό η απρόσκοπτη μετάδοση των δεδομένων. Το ασύρματο δίκτυο χρησιμοποιείται σαν εφεδρικό στην περίπτωση βλάβης ή φόρτου εργασίας των τηλεφωνικών γραμμών του σταθερού δικτύου.

# ΚΕΦΑΛΑΙΟ 2$^{Ο}$ : DATALOGGERS ΤΗΣ ΑΓΟΡΑΣ

Στο κεφάλαιο που ακολουθεί παρουσιάζονται κάποια αξιόλογα dataloggers της αγοράς, πριν ακόμα αναλυθούν οι επιμέρους λεπτομέρειες του αναπτυχθέντος συστήματος, προκειμένου να είναι εφικτή στη συνέχεια η σύγκριση.

Στην κατασκευή dataloggers ειδικεύονται πολλές εταιρείες ηλεκτρονικών, εκ των οποίων από τις πλέον αξιόλογες είναι η Campbell Scientific, της οποίας τα dataloggers χρησιμοποιούνται ευρέως σε εφαρμογές σχετικές με μετεωρολογία, γεωργικές καλλιέργειες και έρευνα γύρω από αυτές, γεωλογία και έρευνα γύρω από τις πηγές ύδατος, έρευνα στο διάστημα, με συλλογή δεδομένων για τη συντήρηση μνημείων, με έρευνα για την ποιότητα του αέρα και την υγρασία του εδάφους έως και με συλλογή δεδομένων σε βιομηχανίες αυτοκινήτων για τον ποιοτικό έλεγχο των οχημάτων.

Αρχικά θα δούμε τα βασικά χαρακτηριστικά των δύο δημοφιλών μοντέλων της εταιρείας που έχουν αποσυρθεί από την αγορά και έχουν αντικατασταθεί από νέα μοντέλα με βελτιωμένα χαρακτηριστικά, ωστόσο εξακολουθούν να χρησιμοποιούνται για αρκετές εφαρμογές, καθώς το κόστος αντικατάστασής τους είναι εξαιρετικά υψηλό.

**Datalogger CR10**

- Έχει αντικατασταθεί από το datalogger CR10X
- Κατασκευαζόταν στο χρονικό διάστημα 1987-1996
- Αποτελείται από μια βαθμίδα μετρήσεων και ελέγχου και το CR10WP, που είναι ένα σύστημα καλωδίων με αποσπώμενο πίνακα οργάνων ελέγχου
- Κατά τον κύκλο ζωής του προϊόντος χρησιμοποιήθηκαν τρεις διαφορετικές εκδοχές της μονάδας CR10WP
- Το αποσπώμενο πληκτρολόγιο και η οθόνη των ενδείξεων (CR10KD) μπορούν να μεταφέρονται σε οποιοδήποτε σταθμό (φορητότητα μονάδας)

•Δυνατότητες στον προγραμματισμό που εξαρτώνται από τα εσωτερικά αποτυπωμένα μικροπρογράμματα στη μνήμη ROM (firmware)

•Διαθέτει 29908 σημεία για δεδομένα στην μνήμη του προγράμματος (προσωρινής αποθήκευσης)

•Η διάταξη των δεδομένων είναι βασισμένη σε πίνακα

•Δεν γίνονται πλέον επισκευές

**21X και 21XL Microloggers®**


•Αντικαταστάθηκε από το CR23X Micrologger

•Κατασκευαζόταν από το 1984 έως το 1999

•Αποτελεί συμπαγές και αυτόνομο datalogger

•Διαθέτει ενσωματωμένο πληκτρολόγιο, οθόνη ενδείξεων και τροφοδοτικό

•Το 21X τροφοδοτείται από αλκαλικές D-κυψέλες

•Το 21XL τροφοδοτείται από σφραγισμένες επαναφορτιζόμενες μπαταρίες

•Δυνατότητες προγραμματισμού που εξαρτώνται από το εσωτερικό λογισμικό PROM

•Διαθέτει 19296 σημεία για αποθήκευση δεδομένων (μνήμη προσωρινής αποθήκευσης)

•Η διάταξη των δεδομένων είναι βασισμένη σε πίνακα

•Εξακολουθούν να γίνονται επισκευές

Στη συνέχεια παραθέτουμε τα νέα μοντέλα της εταιρείας που αντικατέστησαν τα προαναφερόμενα dataloggers. Τα κυριότερα χαρακτηριστικά των συσκευών που παρουσιάζονται στο διαδίκτυο από την εταιρία είναι τα ακόλουθα.

**Σύστημα μετρήσεων και ελέγχου CR10X**

- Το πιο δημοφιλές datalogger της εταιρίας
- Σχεδιασμένο για δικτυακές εφαρμογές χωρίς επίβλεψη (unattended applications)
- Αποτελείται από μία μονάδα μετρήσεων, μια μονάδα ελέγχου και από την αποσπώμενη μονάδα των οργάνων ελέγχου CR10XWP
- Διαθέτει 62000 σημεία για δεδομένα σε μνήμη μόνιμης αποθήκευσης (non-volatile)
- Υπάρχει η δυνατότητα αναβάθμισης της μνήμης (επάνω στην υπάρχουσα διάταξη) προκειμένου να είναι εφικτή η αποθήκευση πάνω από $10^6$ σημείων δεδομένων (μνήμη μόνιμης αποθήκευσης)

- Διαθέτει μνήμη 32K για ενεργά προγράμματα και προγράμματα αποθηκευμένα από το χρήστη (μνήμη μόνιμης αποθήκευσης)
- Η διάταξη των δεδομένων είναι βασισμένη σε πίνακες [array (standard) ή table]
- Διαθέσιμα λειτουργικά συστήματα : Standard, Table, Pakbus, Modbus και ALERT
- Λογισμική υποστήριξη προσφερόμενη από το LoggerNet ή από το PC208W (με πλήρη χαρακτηριστικά) ή με ShortCut προγραμματισμό
- Το αποσπώμενο πληκτρολόγιο και η οθόνη των ενδείξεων (CR10KD) μπορούν να μεταφέρονται σε οποιοδήποτε σταθμό (φορητότητα μονάδας)

**CR23X Micrologger**

- Είναι ιδανικό είτε για εφαρμογές εντός του εργαστηρίου ή για εφαρμογές όπου απαιτείται φορητότητα


- Συμπαγές και αυτόνομο datalogger
- Ενσωματωμένο αλφανουμερικό πληκτρολόγιο και οθόνη ενδείξεων
- Αποθηκεύει $10^6$ σημεία δεδομένων σε μνήμη μόνιμης αποθήκευσης
- Δυνατότητα αναβάθμισης της μνήμης (επάνω στην υπάρχουσα διάταξη) για την αποθήκευση πάνω από $2*10^6$ σημείων δεδομένων (μνήμη μόνιμης αποθήκευσης)
- Μνήμη μόνιμης αποθήκευσης 16K για προγράμματα
- Μνήμη μόνιμης αποθήκευσης 16K για αποθήκευση βοηθητικών προγραμμάτων
- Η διάταξη των δεδομένων είναι βασισμένη σε πίνακες [array (standard) ή table]
- Διαθέσιμα λειτουργικά συστήματα : Standard, Table, Pakbus
- Λογισμική υποστήριξη προσφερόμενη από το LoggerNet ή από το PC208W (με πλήρη χαρακτηριστικά) ή με ShortCut προγραμματισμό

Το σύνολο των dataloggers της εταιρίας Campbell Scientific διαθέτει τα ακόλουθα χαρακτηριστικά:

**Συμβατότητα αισθητήρων:**

Μπορούν να μετρηθούν οι ευρύτερα χρησιμοποιούμενοι αισθητήρες του εμπορίου συμπεριλαμβανομένων θερμοζευγών, thermistors, RTDs, μετατροπέων πιέσεως, μετατροπέων ροής, ποτενσιομέτρων, strain gates, ψηφιακών διακοπτών, μετρητών επιτάχυνσης, και LVDTs. Η εύχρηστη τερματική μονάδα επιτρέπει στο χρήστη να μετρήσει εύκολα και άνετα και άλλους τύπους αισθητήρων.

**Εσωτερική επεξεργασία:**

Το datalogger μπορεί να κάνει σε σημαντικό βαθμό συμπίεση δεδομένων για να εξοικονομεί χώρο αποθήκευσης, να μην αποθηκεύει πλεονάζουσα πληροφορία και

να κάνει μία προεπεξεργασία των δεδομένων. Για παράδειγμα αποθηκεύει τις μέγιστες και ελάχιστες τιμές, την ώρα των μετρήσεων, το σύνολο, το μέσο όρο και την τυπική απόκλιση για κάθε μέτρηση.

**Επεκτασιμότητα:**

Με τη χρήση των πολυπλεκτών της εταιρείας SDI-12 και τα περιφερειακά SDM (Σύγχρονες Συσκευές Μετρήσεων) ένα datalogger μπορεί να μετρήσει εκατοντάδες αισθητήρες.

**Πλήρως υποστηριζόμενες επικοινωνίες μεταξύ του datalogger και του PC:**

Δίνεται η δυνατότητα σύνδεσης με το datalogger απ' ευθείας, μέσω τηλεφωνικής γραμμής, κυψελωτού τηλεφώνου, δικτύου MD9, modem ή πρακτικά ενός συνδυασμού των παραπάνω για ανάκτηση δεδομένων, αποστολή και ανάκτηση προγραμμάτων και ρύθμιση του ρολογιού του datalogger. Οι μεταδόσεις μεταξύ του datalogger και του PC ελέγχονται αυτόματα για λάθη για να επιβεβαιωθεί η ακριβής μεταφορά των δεδομένων. Επίσης, υποστηρίζεται μεταφορά δεδομένων μέσω δορυφόρου για εφαρμογές σε πολύ απομακρυσμένα σημεία. Για τη χειροκίνητη μεταφορά δεδομένων από το σταθμό πεδίου στο γραφείο προσφέρονται κάρτες PC και μονάδες αποθήκευσης.

**Χαμηλή καταναλισκόμενη ισχύς σε κατάσταση ηρεμίας:**

Στα περισσότερα CSI dataloggers το ρεύμα είναι κατά μέσο όρο περίπου 1mA όταν αυτά δεν είναι προγραμματισμένα. Τα συστήματα της εταιρείας τροφοδοτούνται από D-cells ή επαναφορτιζόμενες μπαταρίες 12V DC, οι οποίες φορτίζονται από ηλιακά ή από 110-230V AC.

**Περιοχή θερμοκρασιών λειτουργίας:**

Οι τυπικές θερμοκρασίες λειτουργίας είναι -25°C έως +50°C, αλλά η λειτουργία είναι εφικτή και σε πιο εκτεταμένες περιοχές θερμοκρασίας.

# ΚΕΦΑΛΑΙΟ 3$^{Ο}$ : ΕΙΣΑΓΩΓΗ ΣΤΟ PROJECT

Στο τηλεμετρικό δίκτυο για τη μέτρηση των μεταβολών του ηλεκτρομαγνητικού πεδίου της γης, που παρατηρούνται πριν από τους σεισμούς, χρησιμοποιείται η τεχνική datalogger και κατά βάση τα μοντέλα 21X και CR10X της Campbell Scientific (USA).

Η παραπάνω τεχνική με τη χρήση των dataloggers της Campbell, όμως, παρουσιάζει τα ακόλουθα μειονεκτήματα:

1) Το κόστος αγοράς και συντήρησης των συγκεκριμένων συσκευών είναι εξαιρετικά υψηλό. Η συγκεκριμένη τεχνολογία χρησιμοποιείται σε ερευνητικούς σκοπούς, γεγονός που καθιστά το κόστος απαγορευτικό, ιδιαιτέρως όταν για ένα πλήρες τηλεμετρικό δίκτυο απαιτούνται δεκάδες τέτοιες συσκευές.
2) Η τεχνολογία κατασκευής των dataloggers της αγοράς συγκαταλέγεται στις πιο ανεπτυγμένες και για το λόγο αυτό είναι προστατευμένη από τις εταιρείες τόσο σε επίπεδο hardware όσο και σε επίπεδο software. Επομένως οι δυνατότητες επέμβασης στα συστήματα της αγοράς για προσαρμογή στις ανάγκες της εκάστοτε εφαρμογής αλλά και για επισκευές είναι από δύσκολες έως και αδύνατες.
3) Τα dataloggers της Campbell έχουν περιορισμένη μνήμη αποθήκευσης, στην καλύτερη των περιπτώσεων 64K. Η μειωμένη μνήμη έχει ως αποτέλεσμα την μικρή αυτονομία του datalogger αφού με τον περιορισμό του όγκου της αποθηκευόμενης πληροφορίας, περιορίζεται και ο χρόνος που μεσολαβεί μεταξύ των κλήσεων του σταθμού υπαίθρου στο σταθμό βάσης για αποστολή δεδομένων. Σε περιπτώσεις που παρατηρούνται προβλήματα στο τηλεφωνικό δίκτυο, η αδυναμία εγκαθίδρυσης επικοινωνίας μεταξύ κεντρικού και απομακρυσμένου σταθμού οδηγεί στην υπερκάλυψη της μνήμης και φυσικά στην απώλεια δεδομένων.
4) Η έλλειψη σημάτων handshaking για την εξασφάλιση του απαιτούμενου flow control αποτελεί σοβαρό μειονέκτημα των συγκεκριμένων dataloggers, αφού καθιστά την επικοινωνία σε υψηλότερες από 1200 bps αδύνατη. Επειδή δεν υπάρχουν σήματα handshaking, είναι δύσκολο να αυξηθεί ο ρυθμός μετάδοσης των δεδομένων, καθώς μια κακής ποιότητας γραμμή μπορεί να καθυστερεί τη μετάδοση και να υπερχειλίζει με δεδομένα το buffer του modem του σταθμού υπαίθρου, με αποτέλεσμα την απώλεια τμήματος αυτών.

5) Ο περιορισμός της επικοινωνίας σε συγκεκριμένα baud rate 1200, 9600 και 19200 bps. Πιο συγκεκριμένα, ο χρήστης είναι υποχρεωμένος να επιλέξει ένα από τα τρία baud rate περιορίζοντας έτσι τη δυνατότητα για μια ανώτερη ή ενδιάμεση ταχύτητα επικοινωνίας.
6) Η σειριακή θύρα του 21X είναι σε στάθμη TTL επειδή το συγκεκριμένο datalogger χρησιμοποιείται σε μετεωρολογικές κυρίως εφαρμογές, γεγονός που καθιστά τη χαμηλή κατανάλωση ισχύος εξαιρετικά σημαντική παράμετρο. Η συγκεκριμένη όμως στάθμη σήματος δεν είναι συμβατή με τα κοινά smart modem της αγοράς που απαιτούν στάθμη RS-232.

Λαμβάνοντας υπ΄ όψιν τα παραπάνω μειονεκτήματα οι συγγραφείς σχεδίασαν και κατασκεύασαν ολοκληρωμένο τηλεμετρικό σύστημα για μετρήσεις μεταβολών του ηλεκτρομαγνητικού πεδίου της γης, το οποίο περιέχει τα εξής :

α) Συσκευή Datalogger για τη συλλογή, επεξεργασία και αποθήκευση δεδομένων στο σταθμό υπαίθρου.

β) Λογισμικό απαραίτητο για την υλοποίηση των παραπάνω και την επικοινωνία του datalogger με τον κεντρικό σταθμό.

γ) Λογισμικό του κεντρικού σταθμού για την επιτήρηση του datalogger, την επικοινωνία με το σταθμό υπαίθρου για τη λήψη δεδομένων και τέλος για την επεξεργασία και μετατροπή τους σε αρχείο συγκεκριμένου format.

δ) Λογισμικό καταχώρησης παραμέτρων επικοινωνίας και ιδιαιτέρων πληροφοριών του εκάστοτε σταθμού βάσης που ανήκει στο τηλεμετρικό δίκτυο, για την πλήρη αυτοματοποίηση της λήψης δεδομένων.

Η σημασία της ανάπτυξης του παραπάνω συστήματος είναι εμφανής, δεδομένου ότι οι εταιρείες κατασκευής τέτοιων συσκευών δεν ανταποκρίνονται πάντα στις εξειδικευμένες απαιτήσεις του τηλεμετρικού δικτύου για την πρόβλεψη σεισμών.

### 3.1. Προδιαγραφές του συστήματος

Οι απαιτούμενες προδιαγραφές του συστήματος είναι οι ακόλουθες:

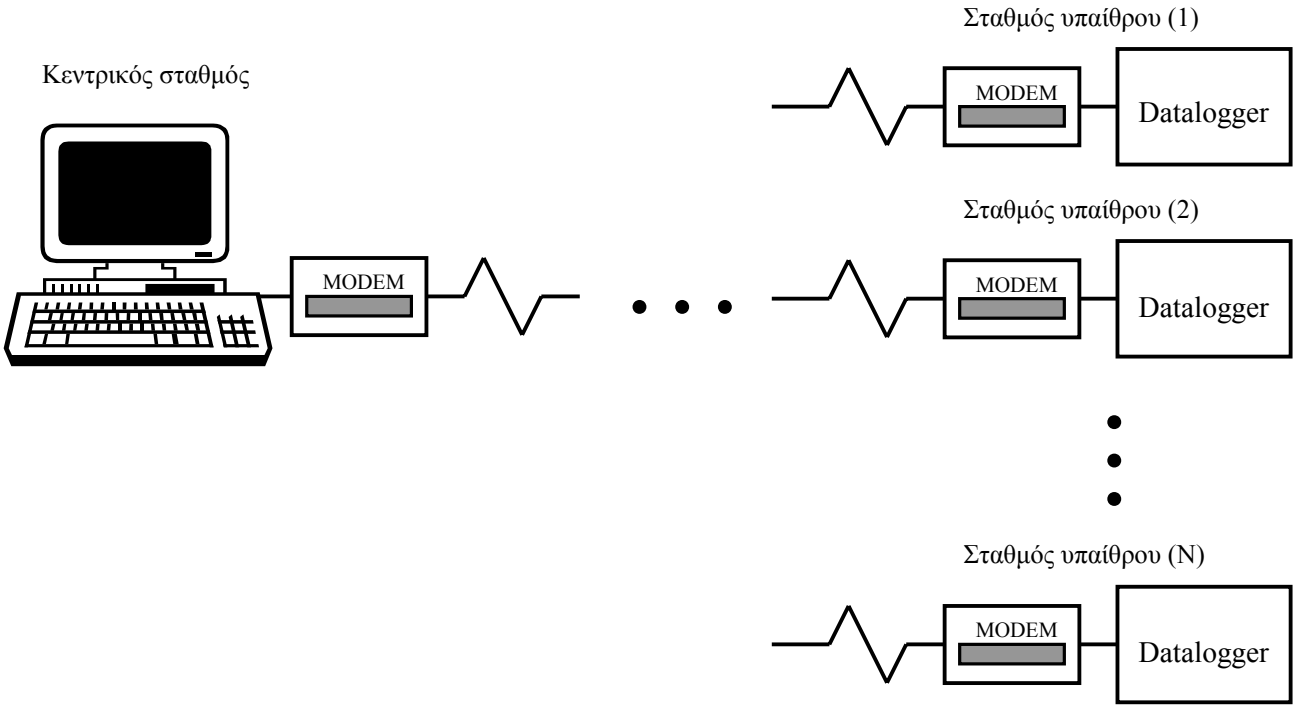
1) Επικοινωνία αμφίδρομη.
2) Δυνατότητα επικοινωνίας του σταθμού βάσης με πολλούς σταθμούς μετρήσεων.
3) Έλεγχος της λειτουργίας του σταθμού υπαίθρου και λήψη μετρήσεων από τον κεντρικό σταθμό ανά δώδεκα ώρες.

4) Ελάχιστη πιθανότητα απώλειας δεδομένων ή εσφαλμένης λήψης αυτών κατά τη διάρκεια της μεταφοράς.
5) Ελαχιστοποίηση των επισκέψεων τεχνικού στο σταθμό μέτρησης για συντήρηση του συστήματος.
6) Χαμηλή κατανάλωση που επιτρέπει την τροφοδότηση του συστήματος από μπαταρίες και φωτοβολταϊκά στοιχεία.
7) Διατήρηση των δεδομένων σε περίπτωση απώλειας της τροφοδοσίας.
8) Αυτόματη επανατοποθέτηση του συστήματος (reset) στην περίπτωση που το πρόγραμμα "κολλήσει" ή στην περίπτωση που η τάση τροφοδοσίας πέσει κάτω από ένα ορισμένο κατώφλι.
9) Ευέλικτα προγράμματα και εύκολη εκμάθηση χειρισμού.
10) Χαμηλό κόστος κατασκευής και συντήρησης.

## 3.2. Δομή του τηλεμετρικού συστήματος

Στο σχήμα 3.2.α παρουσιάζεται η δομή του τηλεμετρικού δικτύου για τη μέτρηση των μεταβολών του ηλεκτρομαγνητικού πεδίου της γης πριν την εκδήλωση σεισμών.

Το σύστημα αποτελείται από ένα κεντρικό σταθμό και από Ν το πλήθος σταθμούς υπαίθρου (δυνατότητα υποστήριξης απεριορίστου αριθμού σταθμών).




**Σχήμα 3.2.α** Δομή του τηλεμετρικού δικτύου

Ο κεντρικός σταθμός επικοινωνεί με καθένα από τους σταθμούς υπαίθρου ανά 12 ώρες, ελέγχει τη σωστή λειτουργία τους, λαμβάνει τα δεδομένα από αυτούς και τα αποθηκεύει για περαιτέρω επεξεργασία και ανάλυση. Η επικοινωνία γίνεται μέσω του σταθερού τηλεφωνικού δικτύου με τη χρήση κοινών smart modem της αγοράς.

Ο σταθμός υπαίθρου συλλέγει δεδομένα από 6 αναλογικά κανάλια με ρυθμό δειγματοληψίας 1sample/min. Επίσης, συλλέγει στο έβδομο αναλογικό κανάλι δεδομένα σχετικά με την τάση της μπαταρίας, η οποία τροφοδοτεί το σύστημα, στοιχείο απαραίτητο για την περαιτέρω ανάλυση των δεδομένων από τον κεντρικό σταθμό. Σημαντικό είναι ότι κατά τη διάρκεια της επικοινωνίας με τον κεντρικό σταθμό, το datalogger συνεχίζει να συλλέγει δεδομένα (περιβάλλον multitasking).

# ΚΕΦΑΛΑΙΟ 4$^{Ο}$ : HARDWARE ΤΟΥ DATALOGGER

Στο παρόν κεφάλαιο θα αναπτύξουμε τη λογική της σχεδίασης του datalogger, θα παρουσιάσουμε το δομικό του διάγραμμα, θα αναλύσουμε το κυκλωματικό του διάγραμμα και θα παραθέσουμε τα τυπωμένα κυκλώματα της κατασκευής.

## 4.1. Λογική της σχεδίασης

Κατά τη διάρκεια της μελέτης και σχεδίασης του datalogger εξετάστηκαν διάφορες τεχνικές σχεδίασης προκειμένου να βρεθεί η βέλτιστη λύση που θα ικανοποιούσε τις απαιτούμενες προδιαγραφές.

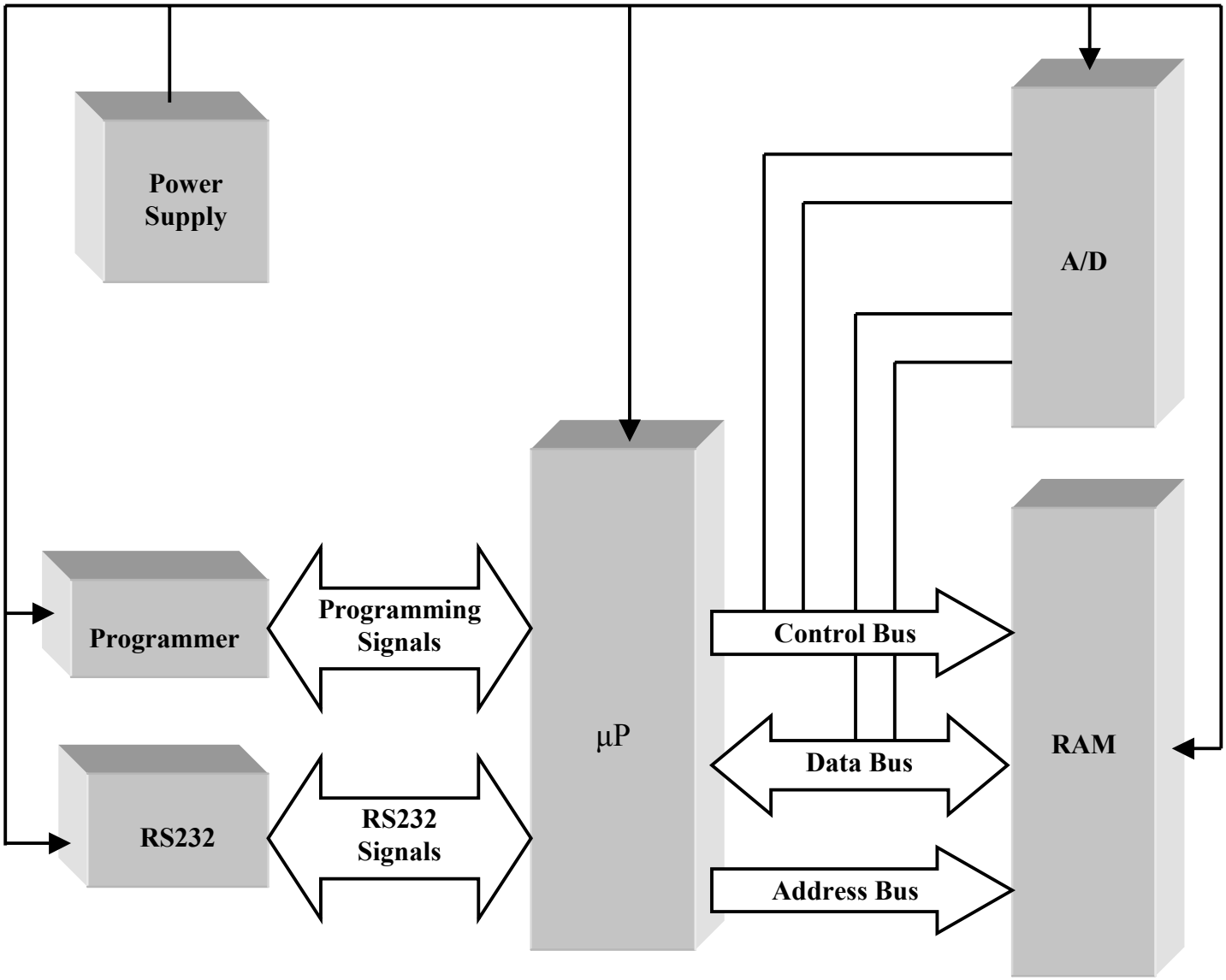
Αρχικά συζητήθηκε η χρήση του μικροϋπολογιστή AT90S8535 της οικογένειας AVR της ATMEL, ο οποίος διαθέτει ενσωματωμένο A/D converter 8 καναλιών με 10 bit ανάλυση. Η απαίτηση, όμως, για μεγαλύτερη ακρίβεια οδήγησε στην επιλογή ενός A/D converter 8 καναλιών με 12 bit ανάλυση και η ανάγκη για ταχύτητα στη δειγματοληψία και στην επικοινωνία μεταξύ A/D και CPU στην επιλογή παράλληλου A/D converter έναντι σειριακού. Τελικά, αποφασίστηκε η χρήση του A/D converter MAX 197 της εταιρίας Maxim.

Αναζητώντας το κατάλληλο αποθηκευτικό μέσο, στο οποίο θα αποθηκεύονταν τα δεδομένα των μετρήσεων, καθώς επίσης και η ώρα λήψης τους, αρχικά αποφασίστηκε η χρήση της σειριακής μνήμης EEPROM AT24C512 της Atmel χωρητικότητας 64 Kbytes και του real time clock DS1307 της Dallas Semiconductor. Τα πλεονεκτήματα αυτής της λύσεως ήταν η διατήρηση δεδομένων παρά την απώλεια τροφοδοσίας και η απλότητα στο σχεδιασμό (interface μέσω του I2C με τη χρήση διαύλου δύο καλωδίων). Οι απαιτήσεις, όμως, του συστήματος για συχνή επανεγγραφή της μνήμης και για υψηλή ταχύτητα εγγραφής και προσπέλασής της καθιστούν τη λύση αυτή αδύνατη, αφού οι μνήμες EEPROM καταστρέφονται μετά από καθορισμένο αριθμό εγγραφών και ο σειριακός δίαυλος I2C είναι συγκριτικά αργός. Για τους παραπάνω λόγους χρησιμοποιήθηκε η παράλληλη non volatile μνήμη RAM 32Kbytes με ενσωματωμένο real time clock DS1644 της DALLAS Semiconductor.

Τέλος, σαν κεντρική μονάδα επεξεργασίας (CPU) του datalogger χρησιμοποιήθηκε ο μικροεπεξεργαστής AT90S8515 της οικογένειας AVR της ATMEL που διαθέτει address bus των 16bit και data bus των 8bit και έχει τη δυνατότητα προσπέλασης της εξωτερικής μνήμης σε τρεις κύκλους ρολογιού.

## 4.2. Δομικό διάγραμμα

Στο παρακάτω σχήμα παρατηρούμε το γενικό διάγραμμα βαθμίδων της διάταξης.




**Σχήμα 4.2.1.** Διάγραμμα βαθμίδων

Η μονάδα τροφοδοσίας (power supply) τροφοδοτεί όλες τις επιμέρους μονάδες με τάση +5V. Η μνήμη RAM και ο μετατροπέας A/D ανταλλάσσουν δεδομένα με τον μικροεπεξεργαστή μέσω του διαύλου δεδομένων (data bus) και διευθυνσιοδοτούνται από αυτόν μέσω του διαύλου διευθύνσεων (address bus). Ο μικροεπεξεργαστής αναλαμβάνει τη διαχείριση των δύο αυτών μονάδων μέσω των σημάτων ελέγχου από τον αντίστοιχο δίαυλο (control bus). Τα σήματα handshaking που ανταλλάσσονται μεταξύ μικροεπεξεργαστή και μονάδας RS232 χρησιμοποιούνται για την οδήγηση εξωτερικού modem ή για την απευθείας σύνδεση με τη σειριακή θύρα του Η/Υ. Τέλος ο προγραμματιστής (programmer)

χρησιμοποιείται για τον επί τόπου προγραμματισμό του μικροεπεξεργαστή, με τον οποίο ανταλλάσσουν τα σήματα προγραμματισμού μέσω του αντίστοιχου διαύλου.

Στην ενότητα που ακολουθεί παρουσιάζονται αναλυτικά τα επιμέρους τμήματα του datalogger με τις λεπτομέρειες τόσο της σχεδίασης όσο και της λειτουργίας τους.

## 4.3. Κυκλωματικό διάγραμμα

### 4.3.1. Κύκλωμα τροφοδοσίας

Στο παρακάτω σχήμα 4.3.1.α φαίνεται το δομικό διάγραμμα του τμήματος του datalogger που χρησιμοποιείται για την τροφοδοσία της διάταξης. Η τάση τροφοδοσίας πρέπει να είναι 9-12V DC, η οποία μπορεί να δοθεί από μπαταρία ή από οποιοδήποτε pack τροφοδοτικό με δυνατότητα παροχής ρεύματος τουλάχιστον 150mA.

Η έξοδος του κυκλώματος είναι +5V DC σταθεροποιημένη τάση, με την οποία τροφοδοτείται το σύνολο της διάταξης. Πρέπει να αναφέρουμε ότι λόγω της μικρής απαίτησης της διάταξης σε ρεύμα, δεν κρίθηκε απαραίτητη η χρήση ψύκτρας στο σταθεροποιητή για την απαγωγή της θερμότητας.

Για την πιστοποίηση της ύπαρξης τάσης τροφοδοσίας στη συσκευή υπάρχει ενδεικτική πράσινη λυχνία (green led). Σε συστήματα unattended π.χ σε απομακρυσμένους σταθμούς μέτρησης, που το datalogger τροφοδοτείται από μπαταρίες και ηλιακά, η ενδεικτική λυχνία μπορεί να παραλείπεται για μεγαλύτερη εξοικονόμηση ενέργειας.

Το βύσμα που πρέπει να χρησιμοποιηθεί για την τροφοδότηση του συστήματος είναι θηλυκό jack, με το θετικό πόλο στο εσωτερικό του και τον αρνητικό στο εξωτερικό. Ιδιαίτερη προσοχή πρέπει να δοθεί στην παραπάνω πολικότητα καθώς η απουσία προστατευτικής διόδου θέτει σε κίνδυνο το σύστημα όταν αυτό τροφοδοτηθεί ανάστροφα.

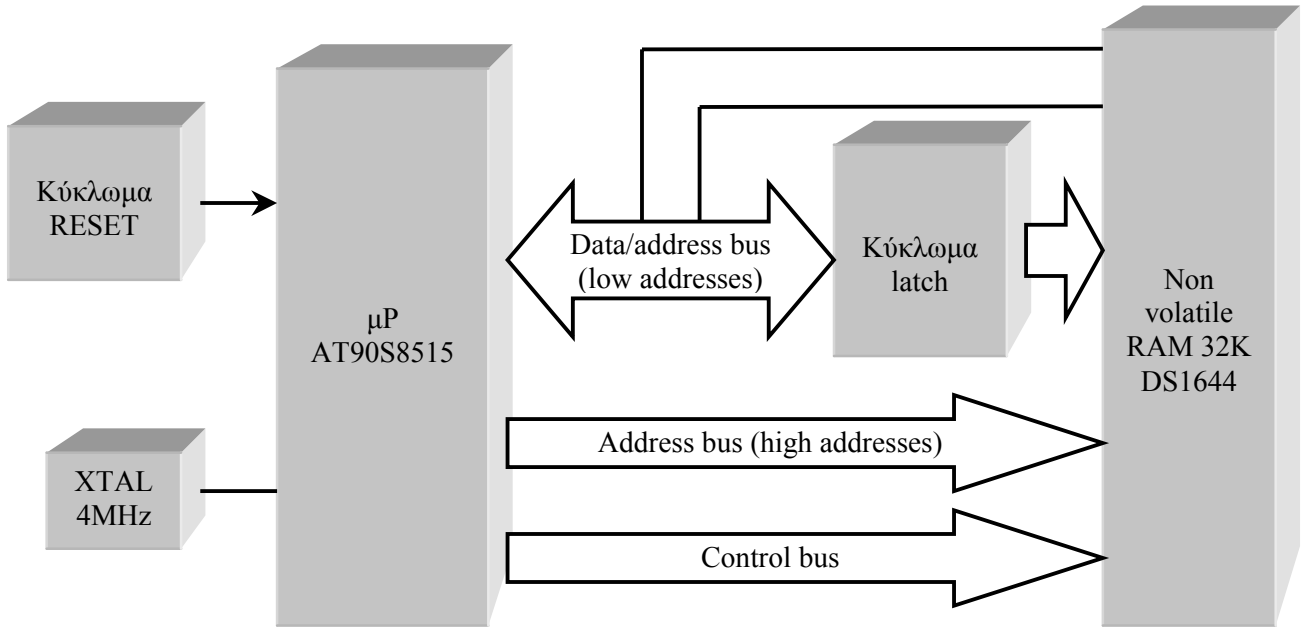

**Σχήμα 4.3.1.α** Δομικό διάγραμμα κυκλώματος τροφοδοσίας.

### 4.3.2. Βασικό κύκλωμα

Το δομικό διάγραμμα του βασικού κυκλώματος της διάταξης φαίνεται στο σχήμα 4.3.2.α. Όπως εύκολα παρατηρούμε η καρδιά του κυκλώματος είναι ο μικροεπεξεργαστής της ATMEL AT90S8515. Ο μικροεπεξεργαστής τροφοδοτείται από τάση +5V και χρονίζεται μέσω κρυστάλλου 4MHz. Με το χρονισμό του μικροϋπολογιστή με κρύσταλλο της παραπάνω συχνότητας, κάθε κύκλος ρολογιού διαρκεί 125nsec.

Για την επανατοποθέτηση του μικροεπεξεργαστή χρησιμοποιείται επιτηρητής τάσης, ο οποίος παράγει σήμα reset, σε περίπτωση, που η τάση τροφοδοσίας πέσει κάτω από ένα ορισμένο κατώφλι. Επίσης, είναι δυνατή και η χειροκίνητη επανατοποθέτησή του μέσω διακόπτη απλής επαφής, που βρίσκεται επί της συσκευής.

Στο 16 bit address bus το οποίο διαθέτει ο μικροεπεξεργαστής (Πόρτα A χαμηλές θέσεις μνήμης και 8 bit data bus, Πόρτα C υψηλές θέσεις μνήμης) προσαρμόστηκε η μνήμη RAM 32Kbytes DS1644 και ο A/D converter MAX197. Επειδή οι χαμηλές θέσεις μνήμης του address bus είναι ταυτόχρονα και data bus χρησιμοποιήθηκε κατάλληλο κύκλωμα latch (μανταλωτή) προκειμένου να κλειδώνουν οι χαμηλές διευθύνσεις μνήμης.




**Σχήμα 4.3.2.α** Δομικό διάγραμμα βασικού κυκλώματος.

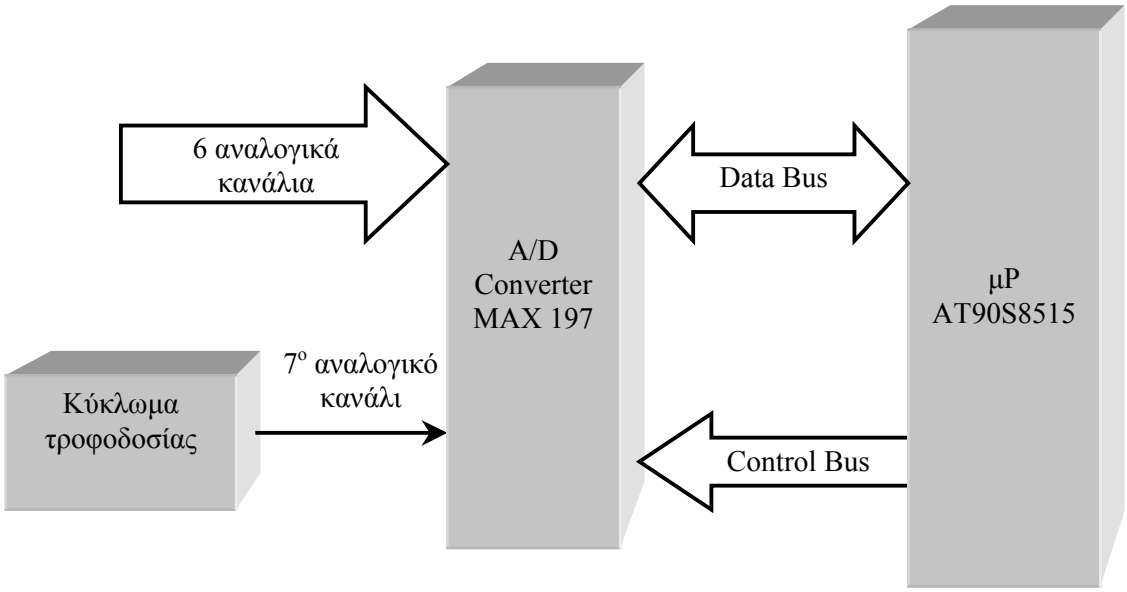
### 4.3.3. Κύκλωμα A/D converter

Στο σχήμα 4.3.3.α βλέπουμε το δομικό διάγραμμα του κυκλώματος του A/D converter. Όπως παρατηρούμε, ο A/D converter μπορεί να λαμβάνει σήμα από 8 αναλογικά κανάλια Ο χρονισμός του γίνεται από το εσωτερικό του clock στα 1.56 MHz με τη βοήθεια εξωτερικού πυκνωτή. Η ψηφιακή έξοδος που παρέχει (8+4 bit) συνδέεται στο data bus και τα control σήματά του σε αντίστοιχο αριθμό ψηφιακών I/O του μικροεπεξεργαστή.

Στη συγκεκριμένη συσκευή χρησιμοποιούνται τα 7 από τα 8 κανάλια του A/D converter. Τα έξι από αυτά χρησιμοποιούνται για τη λήψη, από εξωτερικές πηγές, αναλογικού σήματος στάθμης 0-5V. Το έβδομο κανάλι δειγματοληπτεί την τάση τροφοδοσίας της συσκευής και κατ' επέκταση του συνόλου των συσκευών του σταθμού υπαίθρου, καθώς αυτή η πληροφορία είναι απαραίτητη για τον έλεγχο της καλής λειτουργίας του και της ορθότητας των λαμβανομένων δεδομένων.

Προκειμένου να μετρηθεί η τάση τροφοδοσίας του datalogger (+12V) χρησιμοποιείται διαιρέτης τάσης για τον υποβιβασμό της, ώστε να βρίσκεται εντός της περιοχής των σταθμών σήματος (0-5 V) που είναι ρυθμισμένος να δέχεται ο A/D converter.

Επειδή η εξωτερική μνήμη RAM τοποθετήθηκε στις υψηλότερες θέσεις μνήμης του συστήματος, στη διευθυνσιοδότησή του επιλέχθηκε η τοποθέτηση του A/D converter σε μία από τις χαμηλές θέσεις μνήμης.

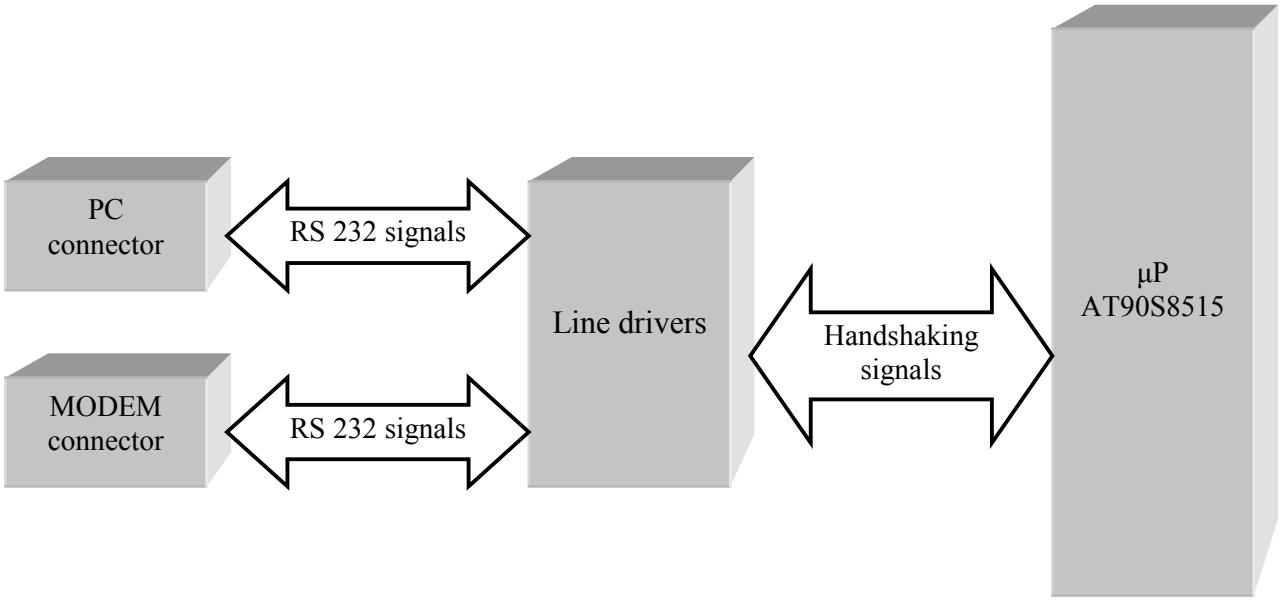



**Σχήμα 4.3.3.α** Δομικό διάγραμμα του κυκλώματος του A/D converter

### 4.3.4. Κύκλωμα επικοινωνίας

Στο σχήμα 4.3.4.α βλέπουμε το δομικό διάγραμμα του κυκλώματος που χρησιμοποιείται για την επικοινωνία του datalogger. Η σχεδίαση του συστήματος δίνει τη δυνατότητα σύνδεσής του, τόσο με εξωτερικό modem για εφαρμογές σε απομακρυσμένους σταθμούς μέτρησης, όσο και με τη σειριακή θύρα του υπολογιστή για εφαρμογές τοπικής συλλογής των δεδομένων. Με τη χρήση line drivers επιτυγχάνουμε τη δημιουργία πλήρους πρωτοκόλλου RS232 μετατρέποντας τις στάθμες TTL σε στάθμες RS232 και το αντίστροφο.

Με κατάλληλη χρήση κάποιων εισόδων και εξόδων του μικροεπεξεργαστή επιτυγχάνεται η δημιουργία όλων των σημάτων handshaking για να είναι εφικτός ο απαραίτητος έλεγχος της ροής των δεδομένων κατά την επικοινωνία. Ο έλεγχος αυτός δίνει τη δυνατότητα να πραγματοποιείται επικοινωνία και σε ανώτερους ρυθμούς, αφού η ροή δεδομένων, μπορεί να ανακόπτεται αποσοβώντας τον κίνδυνο απώλειας δεδομένων λόγω υπερχείλισης του buffer του modem.




**Σχήμα 4.3.4.α** Δομικό διάγραμμα κυκλώματος RS232

### 4.3.5. Κύκλωμα προγραμματιστή

Προκειμένου να εξασφαλίσουμε τον προγραμματισμό του μικροεπεξεργαστή του datalogger επί της διατάξεως (on-board programming) σχεδιάστηκε κατάλληλο κύκλωμα προγραμματισμού του, απ' ευθείας από τον υπολογιστή. Το πλεονέκτημα είναι ότι δεν απαιτείται η χρήση ξεχωριστής συσκευής προγραμματισμού γεγονός που καθιστά την συσκευή οικονομικότερη, πιο ευέλικτη και αυτοδύναμη.

Το κύκλωμα προγραμματιστή χωρίζεται σε δύο μέρη. Το τμήμα που βρίσκεται επί του datalogger και το τμήμα που βρίσκεται σε ξεχωριστή πλακέτα, η οποία προσαρμόζεται απ' ευθείας στην παράλληλη θύρα του Η/Υ.

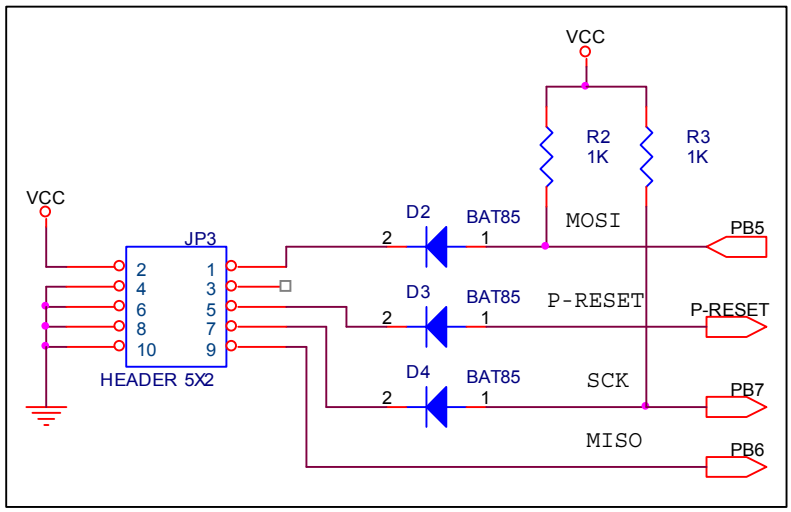
Οι λόγοι που οδήγησαν στο διαχωρισμό του κυκλώματος σε δύο μέρη είναι:

α) η οικονομία σε χώρο στο τυπωμένο κύκλωμα του datalogger.

β) η χαμηλότερη κατανάλωση ενέργειας της συσκευής και

γ) η ασφάλεια, αφού μόνο ο κάτοχος του προγραμματιστή μπορεί να προγραμματίσει τη συσκευή

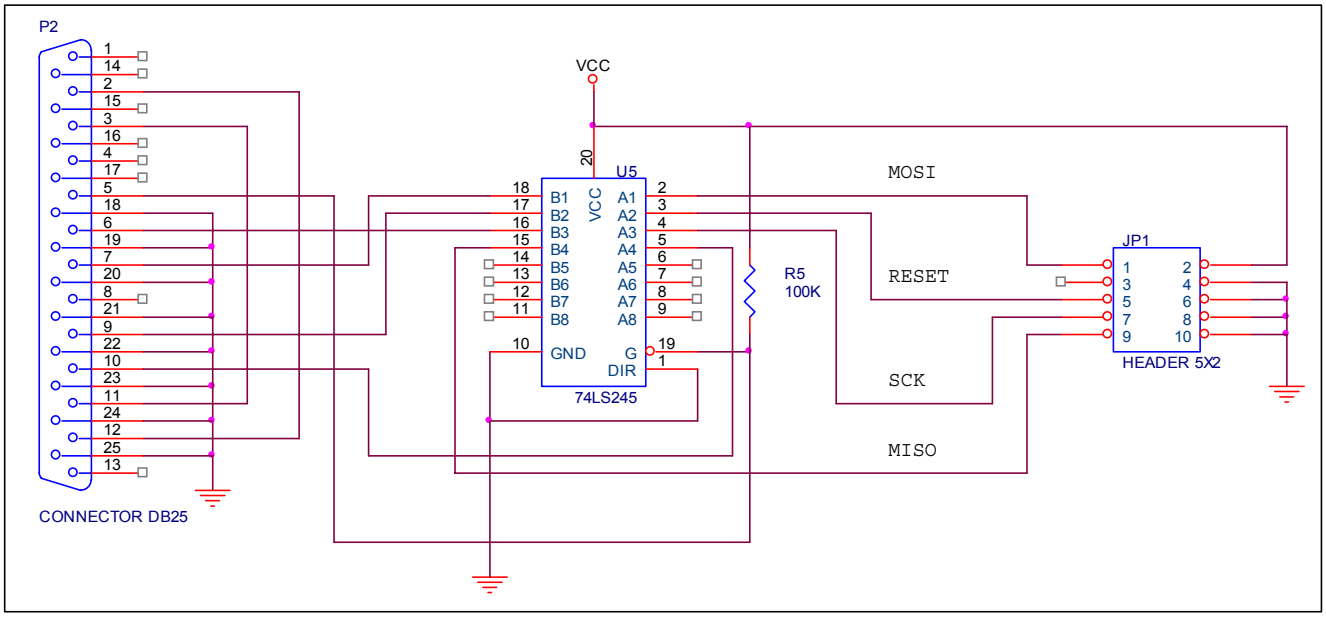
Το κύκλωμα του προγραμματιστή επί του datalogger φαίνεται στο σχήμα 4.3.5.α




**Σχήμα 4.3.5.α** Κύκλωμα προγραμματιστή επί του datalogger

Οι τρεις δίοδοι BAT85 χρησιμοποιούνται για την απομόνωση του μικροεπεξεργαστή από τη θύρα για να αποφεύγονται προβλήματα από σήματα που παρουσιάζονται στη θύρα κατά τη διάρκεια που δεν χρησιμοποιείται ο προγραμματιστής. Οι αντιστάσεις R2 και R3 είναι αντιστάσεις pull-up και χρησιμοποιούνται επειδή οι έξοδοι του μικροεπεξεργαστή είναι σε συνδεσμολογία open collector.

Το κύκλωμα της ξεχωριστής πλακέτας του προγραμματιστή φαίνεται στο σχήμα 4.3.5.β



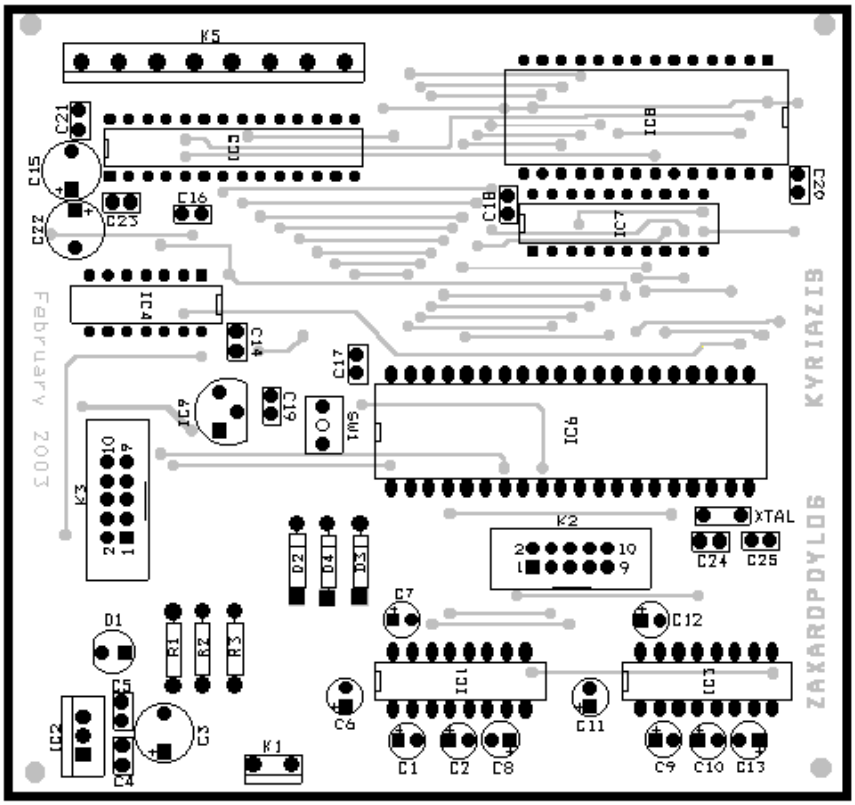

**Σχήμα 4.3.5.β.** Κύκλωμα προγραμματιστή εκτός datalogger

Το ολοκληρωμένο 74LS245 είναι ένας απομονωτής (buffer) που χρησιμοποιείται για την προστασία της θύρας καλύπτοντας την απαίτηση του υπόλοιπου κυκλώματος του προγραμματιστή σε ρεύμα, το οποίο αν δώσει η θύρα κινδυνεύει να καταστραφεί.

## 4.4. Τυπωμένα Κυκλώματα

### 4.4.1 Τυπωμένο κύκλωμα του datalogger

Η πλακέτα του datalogger σχεδιάστηκε σε δύο επίπεδα. Στο σχήμα που ακολουθεί παρατίθεται η διάταξη των εξαρτημάτων επί της πλακέτας, σε πραγματικές διαστάσεις (κλίμακα1:1).




**Σχήμα 4.4.1.α** Τοποθέτηση των εξαρτημάτων στην πλακέτα

#### 4.4.1.1. Κατάλογος εξαρτημάτων του datalogger

Αντιστάσεις:
R1=180Ω
R2=1KΩ
R3=1KΩ

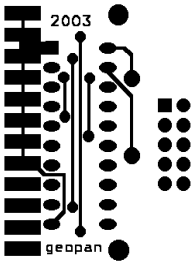
Πυκνωτές:
C1, C2, C6-C13=1μF ταντάλιου
C3=220μF / 25V
C4, C5, C14, C16-C18, C20=100nF κεραμικοί
C15, C22= 4.7μF / 12V
C19= 1nF κεραμικός
C21= 10nF κεραμικοί
C23= 100pF
C24, C25= 22pF κεραμικοί

Ημιαγωγοί:
D1= πράσινο LED
D2-D4= BAT 85
IC1, IC3= MAX232
IC2= LM7805
IC4= LS7404
IC5= MAX197
IC6= AT90S8515 (προγραμματισμένος)
IC7= 74HC573
IC8= DS1644
IC9= DS1233

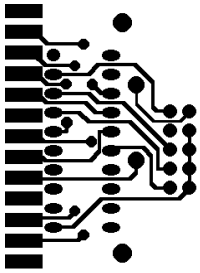
Διάφορα:
K1= 1 διπλή κλέμα
K2, K3 = Αρσενικός connector 2x5 για data καλώδιο κατάλληλος για πλακέτα
K5= 4 διπλές κλέμες
SW1= Διακόπτης απλής επαφής για πλακέτα
XTAL= Κρύσταλλος 4MHz
1 βάση για ολοκληρωμένο 14 pin
2 βάσεις για ολοκληρωμένο 16 pin
1 βάση για ολοκληρωμένο 20 pin
2 βάσεις για ολοκληρωμένο 28 pin
1 βάση για ολοκληρωμένο 40 pin

### 4.4.2. Τυπωμένο κύκλωμα του προγραμματιστή

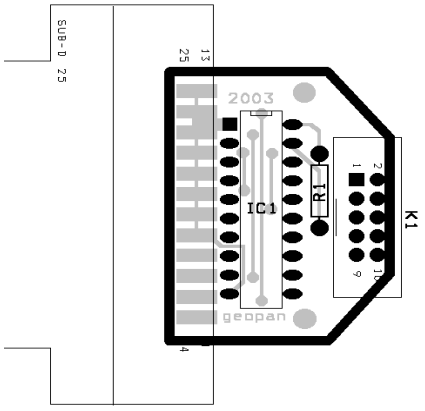
Το τυπωμένο κύκλωμα του προγραμματιστή που αποτελεί ξεχωριστή πλακέτα φαίνεται παρακάτω σε φυσικό μέγεθος.




**Σχήμα 4.4.2.α** Top layer



**Σχήμα 4.4.2.β** Bottom layer




**Σχήμα 4.4.2.γ** Τοποθέτηση εξαρτημάτων στο κύκλωμα του προγραμματιστή

#### 4.4.2.1.Κατάλογος εξαρτημάτων του προγραμματιστή

Αντιστάσεις:
R1=100KΩ

Ημιαγωγοί:
IC1= 74LS245

Διάφορα:
SUB-D 25= Βύσμα sub-D αρσενικό 25-πολικό για τοποθέτηση σε πλακέτα
K1= Αρσενικός connector 2x5 για data καλώδιο κατάλληλος για πλακέτα

# ΚΕΦΑΛΑΙΟ 5$^{Ο}$ : SOFTWARE ΤΟΥ DATALOGGER

Στο παρόν κεφάλαιο, θα αναλυθεί η δομή του λογισμικού το οποίο χρησιμοποιείται στον κεντρικό σταθμό και στο σταθμό υπαίθρου.

## 5.1. Λογισμικό του κεντρικού σταθμού

Προκειμένου να αναπτυχθεί το σύστημα του κεντρικού σταθμού πρέπει να χρησιμοποιηθεί η παρακάτω λογική.

Δημιουργούμε ένα αρχείο station_name.dli, το οποίο περιέχει τον τηλεφωνικό αριθμό κλήσης του σταθμού υπαίθρου, τον αριθμό της σειριακής θύρας του υπολογιστή που θα χρησιμοποιηθεί για την επικοινωνία, καθώς επίσης και το επιθυμητό baud rate της επικοινωνίας. Με τη δημιουργία αυτού του αρχείου καταχωρούνται όλες οι απαραίτητες παράμετροι, τις οποίες χρησιμοποιεί ο κεντρικός σταθμός για να επικοινωνεί με κάθε σταθμό υπαίθρου ξεχωριστά.

Για την επίτευξη της επικοινωνίας μεταξύ του κεντρικού σταθμού και του σταθμού υπαίθρου χρησιμοποιείται κατάλληλο λογισμικό, το οποίο διαβάζει τις παραμέτρους που υπάρχουν στο αρχείο station_name.dli και σύμφωνα με αυτές πραγματοποιεί τη σύνδεση μεταξύ κεντρικού σταθμού και σταθμού υπαίθρου για την λήψη των δεδομένων.

Προκειμένου, λοιπόν, να πραγματοποιούνται τα παραπάνω από τον κεντρικό σταθμό αναπτύχθηκαν τα εξής προγράμματα:

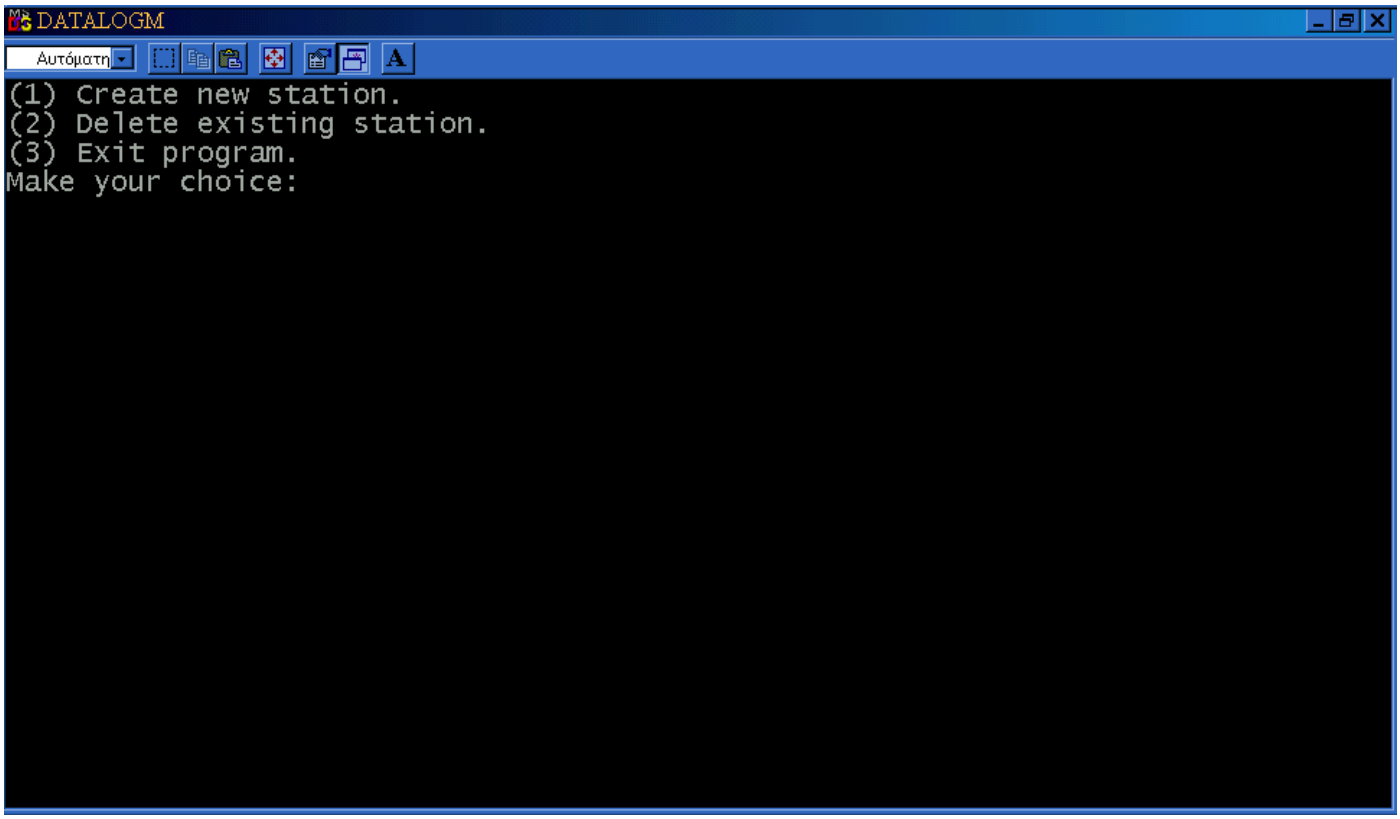
α) Datalogm.exe. Το συγκεκριμένο πρόγραμμα δημιουργεί τα αρχεία dli.

β) Dataloge.exe. Με τη χρήση αυτού του προγράμματος μπορούμε να αλλάξουμε τις παραμέτρους επικοινωνίας που υπάρχουν σε ένα αρχείο dli.

γ) Datalogc.exe. Το πρόγραμμα αυτό είναι υπεύθυνο για την επικοινωνία με το σταθμό υπαίθρου, κατά τη διάρκεια της οποίας πραγματοποιείται έλεγχος καλής λειτουργίας του σταθμού υπαίθρου, λήψη δεδομένων απ' αυτόν και αποστολή τρέχουσας ώρας για συγχρονισμό των δύο σταθμών.

### 5.1.1. Datalogm.exe

Με τη χρήση του συγκεκριμένου προγράμματος δημιουργούμε το αρχείο με επέκταση dli που περιέχει τις παραμέτρους επικοινωνίας. Η κεντρική οθόνη του προγράμματος φαίνεται στην εικόνα 5.1.1.α



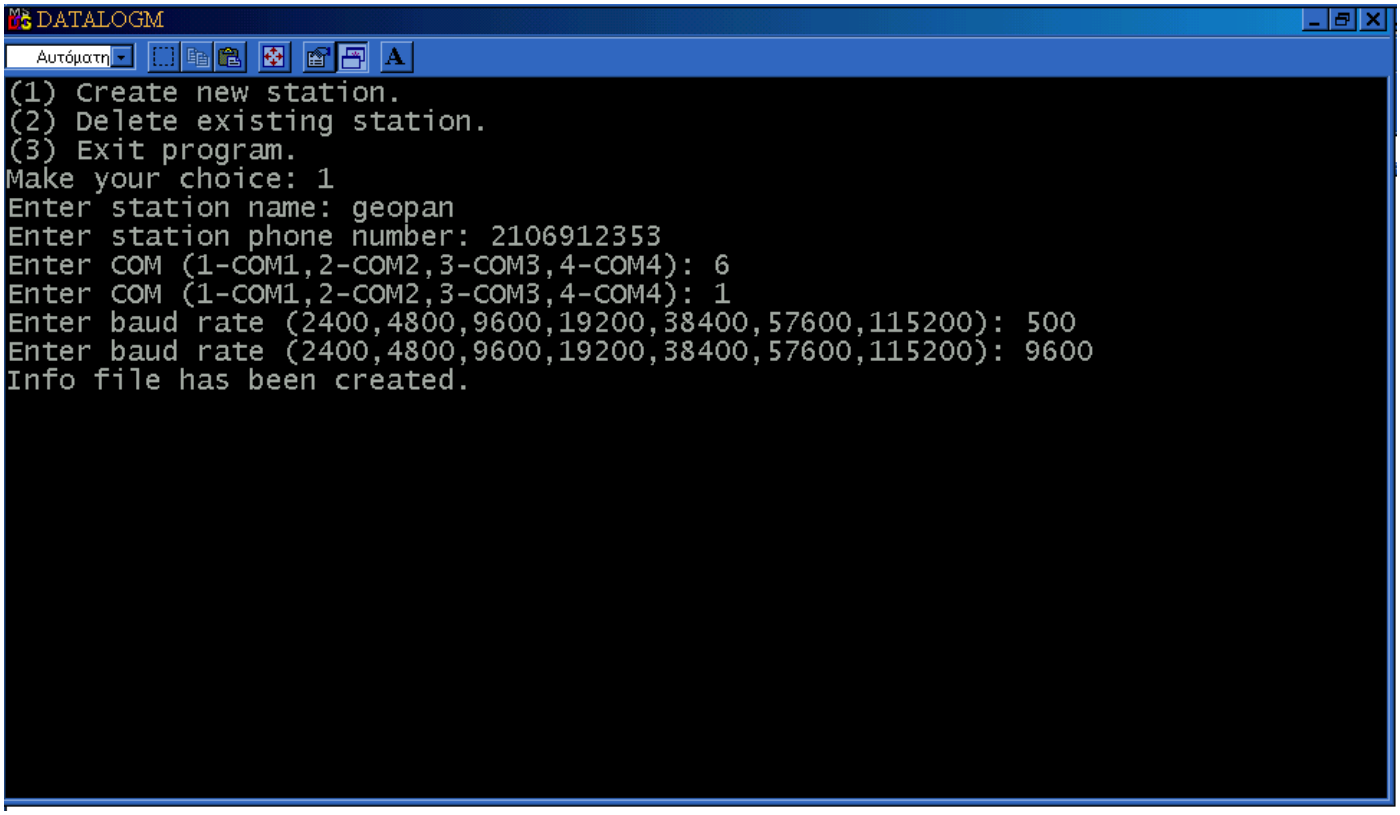

**Εικόνα 5.1.1.α.** Κεντρική οθόνη του προγράμματος datalogm

Όπως παρατηρούμε από την παραπάνω εικόνα, το πρόγραμμα μας δίνει τρεις επιλογές, μια για δημιουργία νέου σταθμού, μια για κατάργηση υπάρχοντος σταθμού και μια για έξοδο από το πρόγραμμα.

Κάνοντας χρήση της πρώτης επιλογής, το πρόγραμμα προτρέπει το χρήστη να εισάγει το όνομα του νέου σταθμού, τον αριθμό τηλεφώνου του σταθμού, τον αριθμό της σειριακής θύρας του υπολογιστή από την οποία θα γίνει η επικοινωνία και το baud rate της επικοινωνίας. Η παραπάνω διαδικασία φαίνεται στην εικόνα 5.1.1.β.

Όπως παρατηρούμε από την εικόνα 5.1.1.β. αρχικά ο χρήστης εισάγει το όνομα του νέου σταθμού υπαίθρου. Το όνομα αυτό δε θα πρέπει να ξεπερνά τους οκτώ χαρακτήρες. Εν συνεχεία ο χρήστης εισάγει τον αριθμό τηλεφώνου του σταθμού υπαίθρου, ο οποίος μπορεί να περιέχει μέχρι δέκα ψηφία. Κατόπιν το πρόγραμμα προτρέπει το χρήστη να εισάγει την επιθυμητή θύρα επικοινωνίας. Τέλος ο χρήστης εισάγει το επιθυμητό baud rate με το οποίο θα γίνει η επικοινωνία. Με το πέρας της παραπάνω διαδικασίας, εμφανίζεται μήνυμα που ενημερώνει για τη δημιουργία του αρχείου.

Όπως παρατηρούμε από την εικόνα 5.1.1.β. σε περίπτωση που εισαχθεί λανθασμένη τιμή είτε στο πεδίο της σειριακής θύρας, είτε στο πεδίο του baud rate τότε ζητείται από το χρήστη η εκ νέου εισαγωγή τιμής.

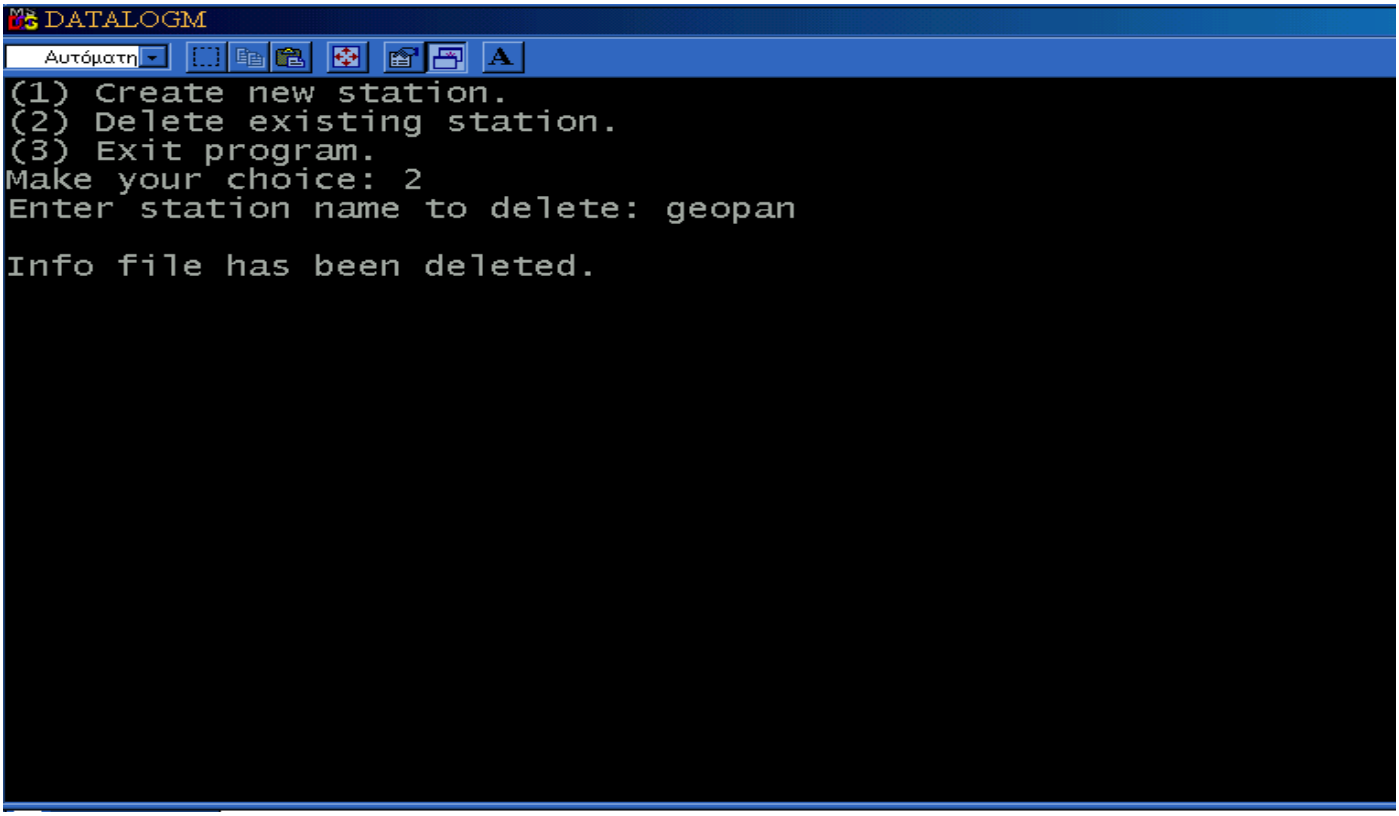



**Εικόνα 5.1.1.β.** Διαδικασία δημιουργίας σταθμού

Κάνοντας χρήση της δεύτερης επιλογής το πρόγραμμα προτρέπει το χρήστη να εισάγει το όνομα του σταθμού, τον οποίο θέλει να καταργήσει. Μετά την κατάργηση του σταθμού εμφανίζεται σχετικό μήνυμα για την επιβεβαίωση της κατάργησης του σταθμού (διαγραφή του αρχείου dli). Η παραπάνω διαδικασία φαίνεται στην εικόνα 5.1.1.γ.

Τέλος, κάνοντας χρήση της τρίτης και τελευταίας επιλογής η λειτουργία του προγράμματος τερματίζεται.

Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι η χρήση του προγράμματος datalogm.exe είναι απαραίτητη, αφού χωρίς την ύπαρξη αρχείων dli δεν είναι δυνατή η επικοινωνία του κεντρικού σταθμού με τους σταθμούς υπαίθρου.



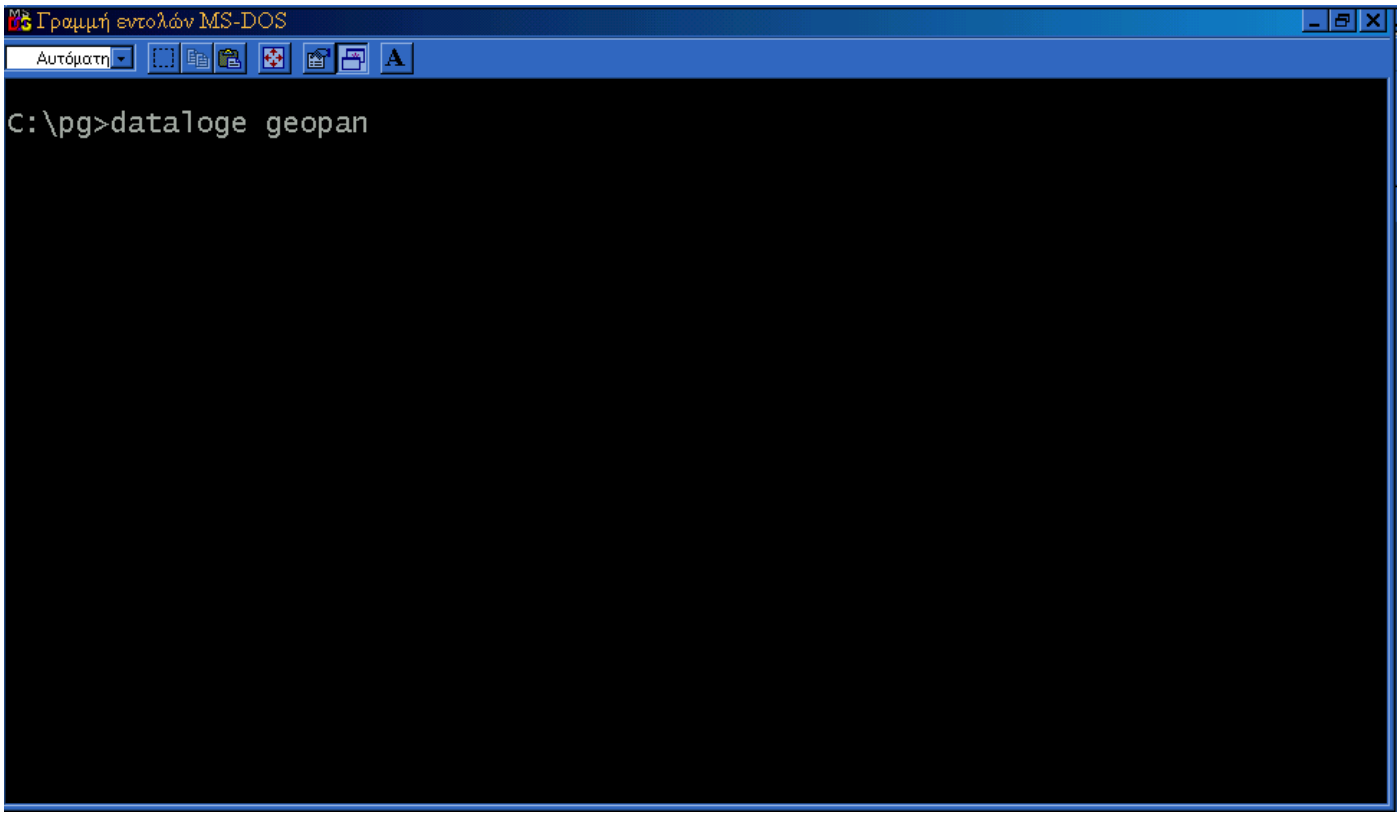

**Εικόνα 5.1.1.γ.** Διαδικασία κατάργησης υπάρχοντος σταθμού

### 5.1.2. Dataloge.exe

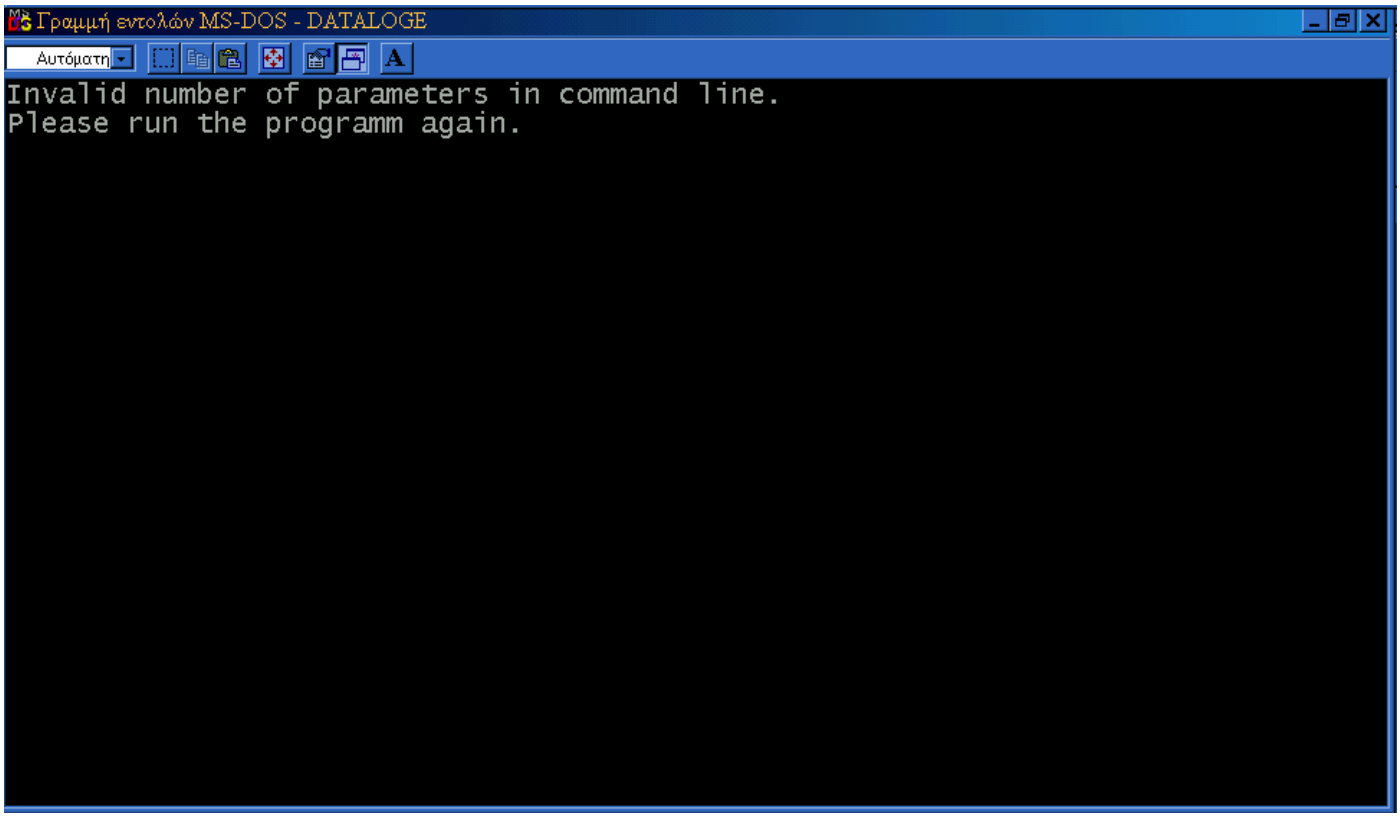
Με τη χρήση αυτού του προγράμματος μπορούμε να δούμε τον τηλεφωνικό αριθμό κλήσης ενός υπάρχοντος σταθμού υπαίθρου, καθώς επίσης και τις παραμέτρους επικοινωνίας του κεντρικού σταθμού με αυτόν. Επιπλέον, το πρόγραμμα μας δίνει τη δυνατότητα να αλλάξουμε τις παραπάνω παραμέτρους.

Προκειμένου να καλέσουμε το πρόγραμμα, στην γραμμή εντολών του MS-DOS γράφουμε: dataloge όνομα_σταθμού (εικόνα 5.1.2.α). Για τη σωστή λειτουργία του προγράμματος θα πρέπει τα αρχεία dataloge.exe και όνομα_σταθμού.dli να βρίσκονται στο ίδιο directory. Στην περίπτωση που η κλήση είναι ελλιπής, λείπει δηλαδή το όνομα του σταθμού, εμφανίζεται μήνυμα λάθους (εικόνα 5.1.2.β ) και το πρόγραμμα τερματίζεται. Στην αντίθετη περίπτωση που η κλήση γίνει σωστά εμφανίζεται η κεντρική οθόνη του προγράμματος (εικόνα 5.1.2.γ).

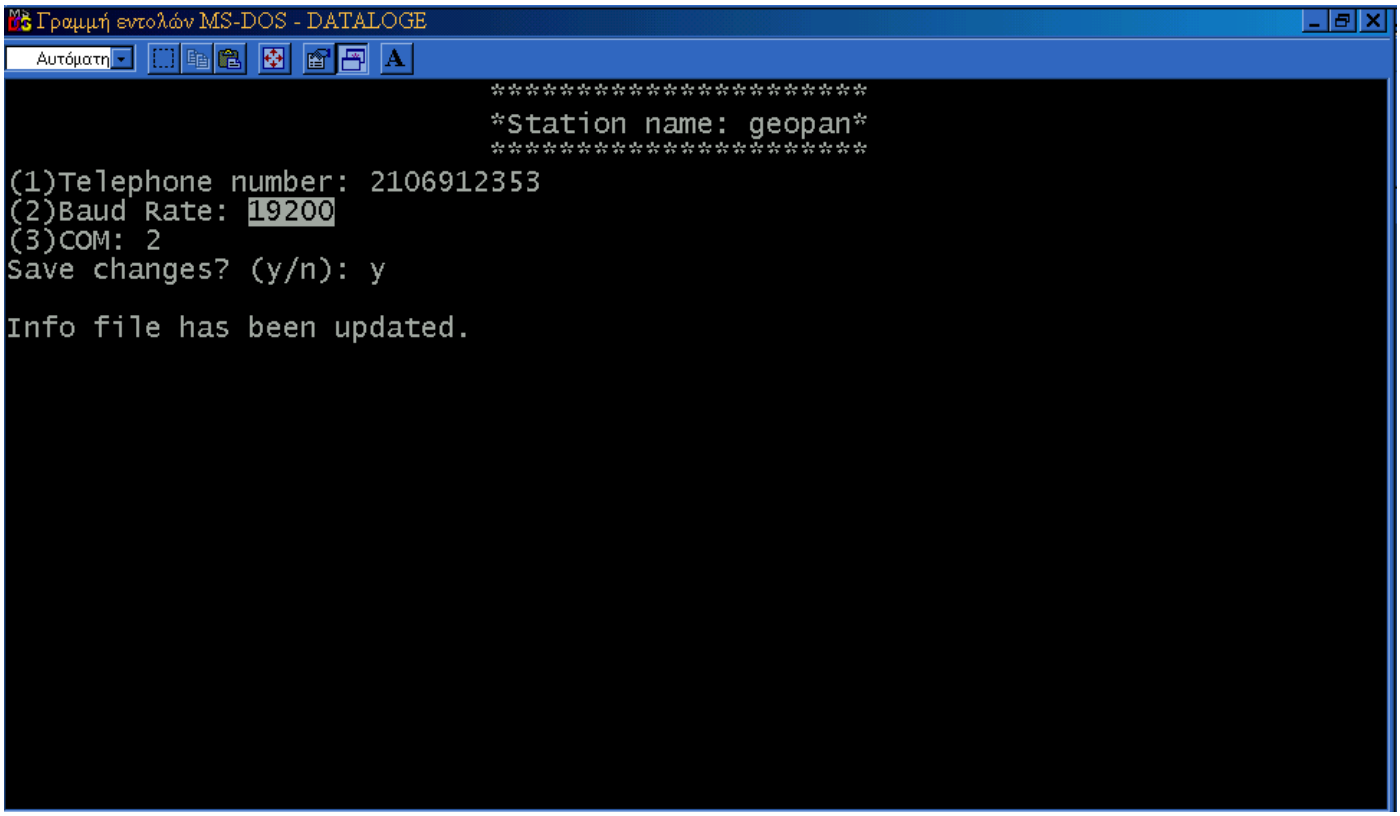
Όπως παρατηρούμε, στο πάνω μέρος της οθόνης εμφανίζεται το όνομα του σταθμού και από κάτω το τηλέφωνο κλήσης του σταθμού και οι παράμετροι




**Εικόνα 5.1.2.α.** Τρόπος κλήσης του προγράμματος στη γραμμή εντολών




**Εικόνα 5.1.2.β.** Μήνυμα λάθους σε ελλιπή κλήση του προγράμματος dataloge




**Εικόνα 5.1.2.γ.** Κεντρική οθόνη του προγράμματος dataloge

επικοινωνίας. Προκειμένου να κάνουμε κάποια αλλαγή στις παραμέτρους, αρχικά χρησιμοποιούμε το TAB για να μετακινηθούμε στην παράμετρο της επιλογής μας και εν συνεχεία με τα πλήκτρα + και – μεταβάλλουμε την τιμή της παραμέτρου που μας ενδιαφέρει. Για την έξοδο από το πρόγραμμα χρησιμοποιούμε το πλήκτρο Esc. Στην περίπτωση που έχουν γίνει αλλαγές στις παραμέτρους, εμφανίζεται μήνυμα το οποίο μας προτρέπει να τις αποθηκεύσουμε. Εάν θέλουμε να αποθηκευθούν πληκτρολογούμε y, εάν όχι n. Στην περίπτωση που επιλέξουμε να αποθηκευθούν οι αλλαγές μετά το πέρας της αποθήκευσης εμφανίζεται σχετικό μήνυμα και κατόπιν το πρόγραμμα τερματίζεται.

### 5.1.3. Datalogc.exe

Με τη χρήση αυτού του προγράμματος, γίνεται η επικοινωνία με οποιοδήποτε σταθμό υπαίθρου, ο έλεγχος της καλής λειτουργίας του, η λήψη των δεδομένων και η αποθήκευσή τους σε κατάλληλο αρχείο για περαιτέρω επεξεργασία (αρχείο dat). Επίσης γίνεται και ανανέωση της ώρας του σταθμού υπαίθρου για το συγχρονισμό του με τον κεντρικό σταθμό.

Προκειμένου να καλέσουμε το πρόγραμμα, στην γραμμή εντολών του MS-DOS γράφουμε: datalogc όνομα_σταθμού. Για τη σωστή λειτουργία του προγράμματος θα πρέπει τα αρχεία datalogc.exe και όνομα_σταθμού.dli να βρίσκονται στο ίδιο directory. Στην περίπτωση που η κλήση είναι ελλιπής, λείπει δηλαδή το όνομα του σταθμού, εμφανίζεται μήνυμα λάθους και το πρόγραμμα τερματίζεται. Το πρόγραμμα εκτελείτε αυτόματα χωρίς να υπάρχει interface με το χρήστη και εξάγει το αρχείο των δεδομένων του σταθμού υπαίθρου. Για τον παραπάνω λόγο δεν εμφανίζονται στην οθόνη μηνύματα.

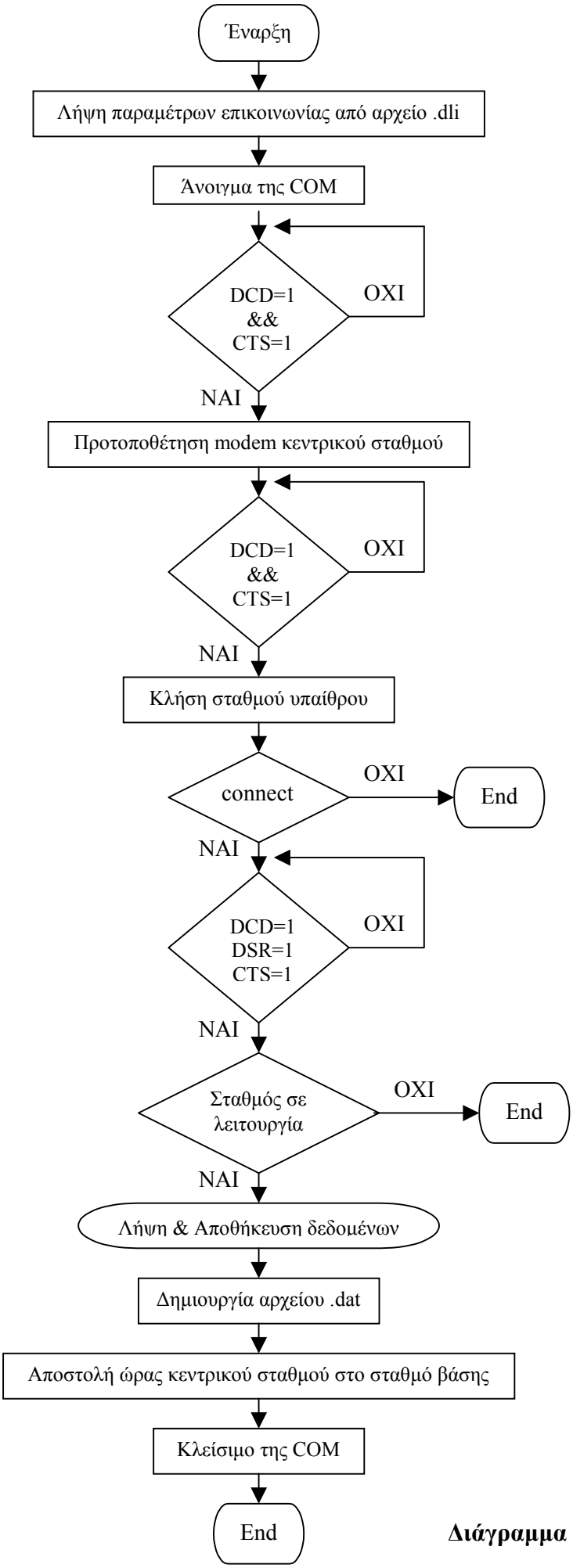
Η εκτέλεση του προγράμματος γίνεται σύμφωνα με το διάγραμμα ροής 5.1.3.α. Αρχικά το πρόγραμμα λαμβάνει τις παραμέτρους επικοινωνίας από το αντίστοιχο αρχείο .dli και με βάση αυτές ανοίγει την σειριακή θύρα του υπολογιστή του κεντρικού σταθμού και προτοποθετεί το modem. Η προτοποθέτηση του modem γίνεται με AT commands. Πιο συγκεκριμένα το modem προγραμματίζεται για λειτουργία echo off, με hardware handshaking, DSR συνεχώς ενεργοποιημένο και DCD ενεργοποιημένο μόνο κατά τη διάρκεια της σύνδεσης. Σε περίπτωση πτώσης του DTR το modem κλείνει τη γραμμή και επιστρέφει σε command state. Για την επίτευξη των παραπάνω στέλνουμε στο modem τις παρακάτω εντολές ***ATE0&S0&C1&D2&K3*** ↵.

Ελέγχοντας πάντα τα σήματα handshaking καλούμε το σταθμό υπαίθρου, συνδεόμαστε με αυτόν και ελέγχουμε την καλή λειτουργία του. Στη συνέχεια λαμβάνουμε τα δεδομένα και τα αποθηκεύουμε σε αρχείο dat σύμφωνα το με συγκεκριμένο format των προδιαγραφών, το οποίο επιτρέπει στο χρήστη την απεικόνιση τους από ήδη υπάρχοντα προγράμματα.

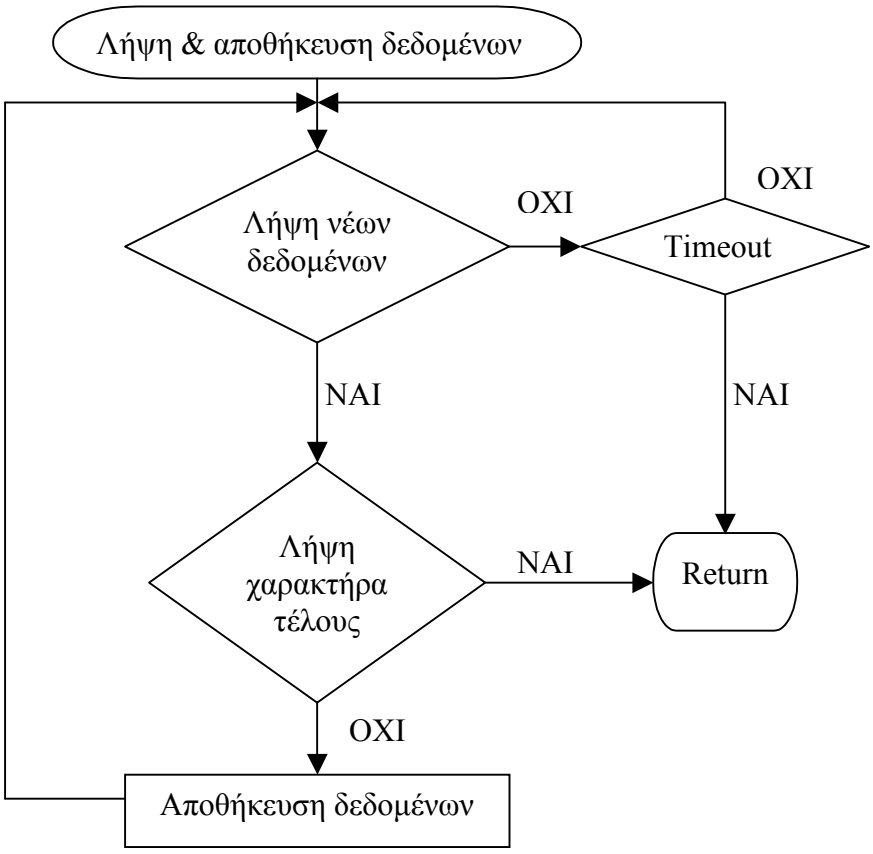
Με την ολοκλήρωση της λήψης των δεδομένων, ο κεντρικός σταθμός αποστέλλει την τρέχουσα ώρα στο σταθμό υπαίθρου ώστε να διορθωθεί το όποιο πιθανό σφάλμα έχει παρουσιαστεί στο ρολόι πραγματικού χρόνου του σταθμού.

Τέλος, το πρόγραμμα αποδεσμεύει τη σειριακή θύρα COM για να είναι δυνατή η προσπέλασή της από οποιοδήποτε άλλο πρόγραμμα.

Ιδιαίτερη αναφορά πρέπει να γίνει στο τμήμα του προγράμματος που είναι υπεύθυνο για τη λήψη και αποθήκευση των δεδομένων. Η αναλυτικότερη παρουσίαση της δομής του γίνεται στο διάγραμμα ροής 5.1.3.β.




**Διάγραμμα ροής 5.1.3.α.**




**Διάγραμμα ροής 5.1.3.β.** Λήψη και αποθήκευση δεδομένων.

Σύμφωνα με το παραπάνω διάγραμμα ροής γίνεται η λήψη και η αποθήκευση δεδομένων στον κεντρικό σταθμό. Παρατηρούμε ότι αρχικά ελέγχεται η λήψη κάποιου χαρακτήρα από τη σειριακή θύρα, ο οποίος αποθηκεύεται, εφόσον δεν είναι χαρακτήρας τέλους. Η έξοδος πραγματοποιείται είτε στην περίπτωση της λήψης χαρακτήρα τέλους είτε στην περίπτωση που εκπνεύσει ο χρόνος timeout χωρίς τη λήψη νέου χαρακτήρα.

## 5.2. Λογισμικό σταθμού υπαίθρου

Το λογισμικό του σταθμού υπαίθρου πραγματοποιεί τον έλεγχο της λειτουργίας του datalogger και την αποστολή των δεδομένων στον κεντρικό σταθμό.

Για την πραγματοποίηση των παραπάνω από το σταθμό υπαίθρου αναπτύχθηκε ο εξής κώδικας:

α) init.c :κώδικας για την αρχικοποίηση της μνήμης του συστήματος του datalogger.

β) datalogs.c: κώδικας υπεύθυνος για το σύνολο των λειτουργιών του συστήματος του datalogger.

### 5.2.1. Init.c

Το πρόγραμμα init.c χρησιμοποιείται σε κάθε νέα εγκατάσταση συσκευής datalogger και σε κάθε αντικατάσταση της μνήμη RAM μιας ήδη εγκαταστημένης συσκευής. Η λειτουργία του προγράμματος αποσκοπεί στην αρχικοποίηση της μνήμης, καθώς και της ημερομηνίας και της ώρας του RTC που είναι ενσωματωμένο σ' αυτή. Πριν αναλύσουμε τη δομή του συγκεκριμένου προγράμματος θα κάνουμε μια αναφορά στη διάρθρωση της μνήμης.

Η μνήμη που μπορεί να διαχειριστεί ο μικροϋπολογιστής, που διαθέτει 16 bit address bus, είναι 64K, τα οποία έχουν τη δομή που φαίνεται στον πίνακα 1. Αναλυτικότερα στις χαμηλότερες θέσεις μνήμης ($0000 - $001F) βρίσκονται 32 καταχωρητές γενικής χρήσης και στις αμέσως υψηλότερες θέσεις μνήμης ($0020 - $005F) ακολουθούν 64 καταχωρητές εισόδου-εξόδου με τη βοήθεια των οποίων ελέγχονται οι επιμέρους λειτουργίες του μικροϋπολογιστή. Τα επόμενα 512 bytes ($0060 - $025F) καταλαμβάνονται από εσωτερική μνήμη SRAM η οποία μπορεί να χρησιμοποιηθεί για οποιοδήποτε σκοπό. Στις θέσεις μνήμης που απομένουν ($0260 - $FFFF) μπορεί να τοποθετηθεί εξωτερική μνήμη αναλόγως με τις απαιτήσεις της εφαρμογής.

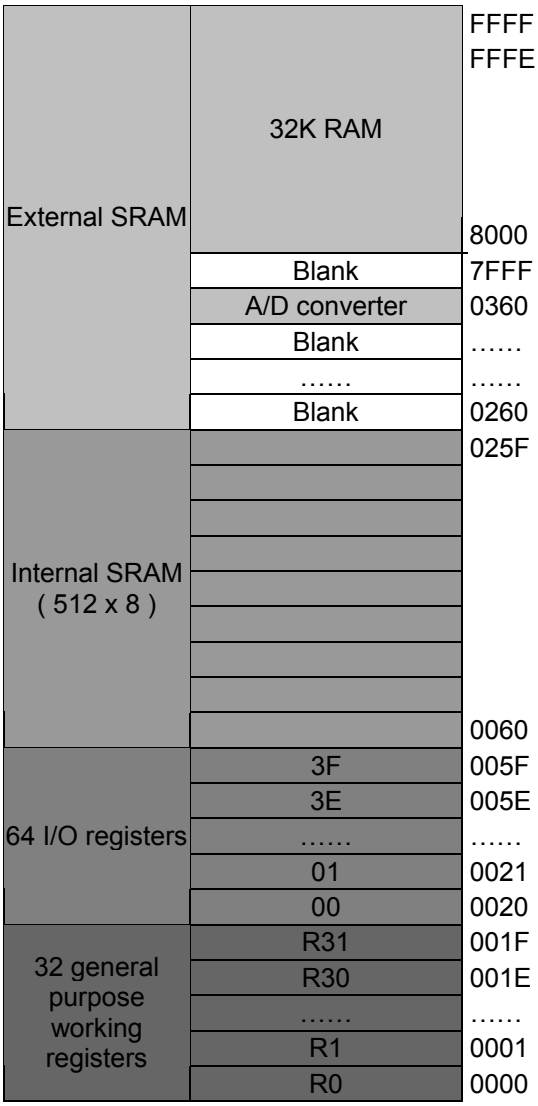
Στην παρούσα εφαρμογή που έχει αναπτυχθεί με τη χρήση του AVR code vision C compiler η διαχείριση της εσωτερικής μνήμης ($0000 - $025F) γίνεται από τον compiler. Η εξωτερική θέση μνήμης $0360 χρησιμοποιείται για την επικοινωνία του μικροϋπολογιστή με τον A/D converter. Στις ανώτερες θέσεις μνήμης ($8000-$FFFF) έχει τοποθετηθεί εξωτερική non volatile μνήμη RAM 32K για την αποθήκευση των δεδομένων που συλλέγονται από το σύστημα.

Η διάρθρωση της εξωτερικής μνήμης των 32K φαίνεται στον πίνακα 2. Αποτελείται από τις εξής τρεις επιμέρους περιοχές : α) περιοχή δεικτών , β) περιοχή δεδομένων, γ) περιοχή RTC.

Στην περιοχή των δεικτών αποθηκεύονται τιμές δεικτών απαραίτητες για την ορθή λειτουργία του συστήματος. Πιο αναλυτικά, στις θέσεις $8002 και $8003 αποθηκεύεται ο ram pointer που είναι υπεύθυνος για τη διαχείριση της μνήμης κατά την αποθήκευση και την αποστολή δεδομένων στον κεντρικό σταθμό. Στις επόμενες δύο θέσεις μνήμης, $8004 και $8005, τοποθετείται ο communication pointer που

υποδεικνύει στο σύστημα το σημείο της μνήμης δεδομένων από το οποίο ξεκινά η αποστολή τους προς τον κεντρικό σταθμό. Ο overflow pointer καταλαμβάνει τη θέση μνήμης $800A και παρακολουθεί την περίπτωση υπερκάλυψης της μνήμης δεδομένων. Τέλος ο temp ram pointer αποθηκεύεται στις θέσεις $8006 και $8007 και αποθηκεύει προσωρινά την τρέχουσα τιμή του ram pointer κάθε φορά που γίνεται αίτηση για αποστολή δεδομένων.

***Data address space***




**Πίνακας 1.** Διάρθρωση της μνήμης του συστήματος.

Η μνήμη δεδομένων περιέχει 32740 σημεία αποθήκευσης τα οποία έχουν οργανωθεί σε block των 20 byte. Ανά λεπτό αποθηκεύεται στην μνήμη και νέο block δεδομένων, το οποίο περιέχει τις μετρήσεις από τα 7 κανάλια καθώς και την ημερομηνία και ώρα λήψης των δεδομένων. Στις δύο πρώτες θέσεις κάθε block

***Block Diagram RAM memory***

| | |
|---|---|
| Year | FFFF |
| Month | FFFE |
| Date | FFFD |
| Day | FFFC |
| Hour | FFFB |
| Min | FFFA |
| Sec | FFF9 |
| Control byte RTC | FFF8 |
| Blank | FFF7 |
| ….. | ….. |
| Blank | FFF1 |
| Data | FFF0 |
| …. | ….. |
| …. | ….. |
| Data | 8020 |
| Blank | 801F |
| Blank | 801E |
| 7th channel (high byte) | 801D |
| 7th channel (low byte) | 801C |
| 6th channel (high byte) | 801B |
| 6th channel (low byte) | 801A |
| 5th channel (high byte) | 8019 |
| 5th channel (low byte) | 8018 |
| 4th channel (high byte) | 8017 |
| 4th channel (low byte) | 8016 |
| 3rd channel (high byte) | 8015 |
| 3rd channel (low byte) | 8014 |
| 2nd channel (high byte) | 8013 |
| 2nd channel (low byte) | 8012 |
| 1st channel (high byte) | 8011 |
| 1st channel (low byte) | 8010 |
| Minute | 800F |
| Hour | 800E |
| Ioulian day (high byte) | 800D |
| ioulian day (low byte) | 800C |
| Blank | 800B |
| Overflow pointer | 800A |
| Blank | 8009 |
| Blank | 8008 |
| temp ram pointer (high byte) | 8007 |
| temp ram pointer (low byte) | 8006 |
| Communication pointer (HB) | 8005 |
| Communication pointer (LB) | 8004 |
| ram pointer (high byte) | 8003 |
| ram pointer (low byte) | 8002 |

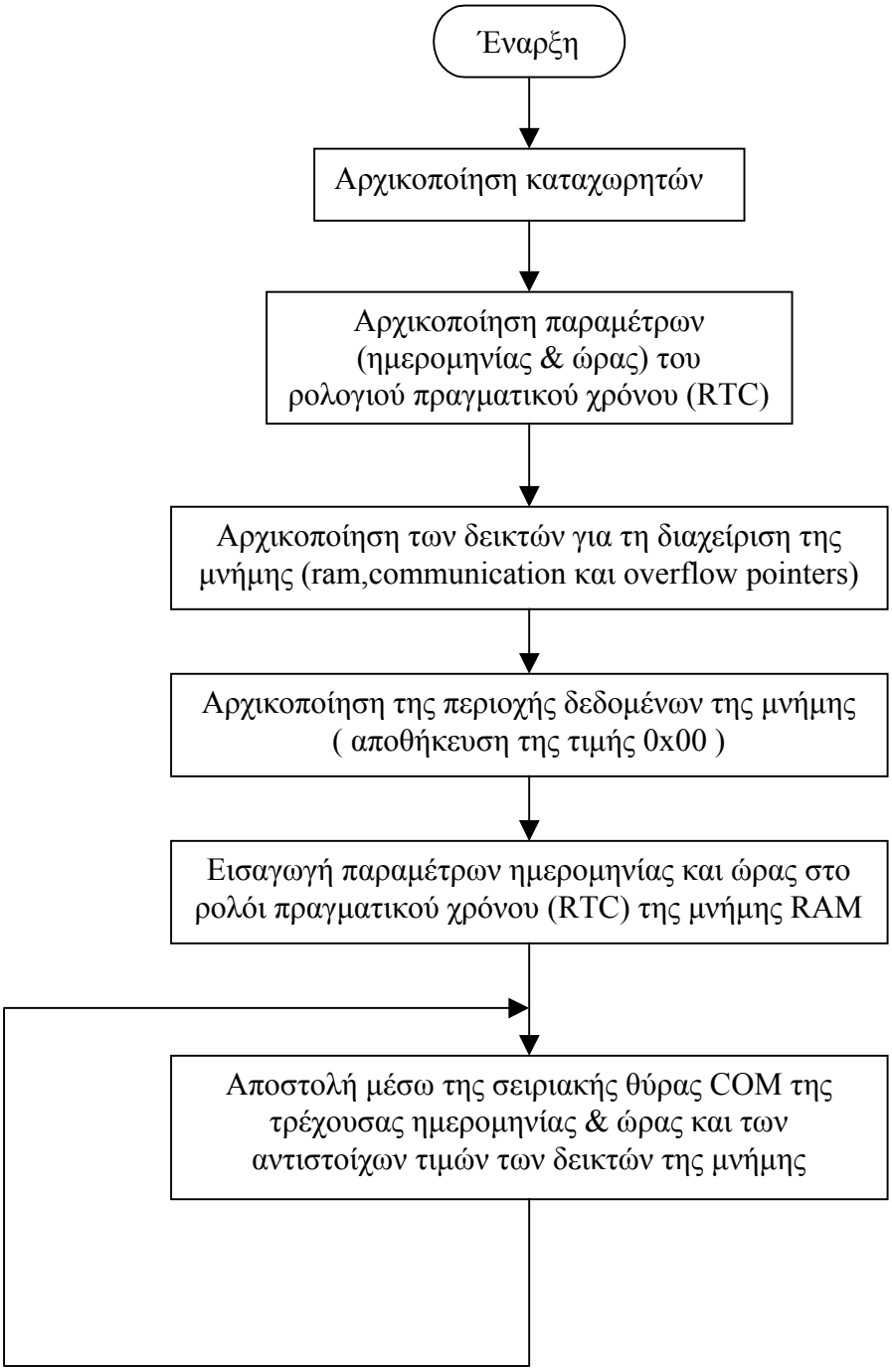
**Πίνακας 2.** Διάρθρωση της εξωτερικής μνήμης RAM.

αποθηκεύεται η Ιουλιανή ημέρα, στην τρίτη και τέταρτη θέση ακολουθούν η ώρα και το λεπτό της μέτρησης αντίστοιχα και στις επόμενες 14 θέσεις μνήμης ακολουθούν ανά δύο bytes (low και high byte αντίστοιχα) οι μετρήσεις καθενός από τα 7 αναλογικά κανάλια. Τα δύο τελευταία bytes του block είναι ελεύθερα για μελλοντική χρήση σε περίπτωση που απαιτηθεί η αποθήκευση δεδομένων από το 8ο κανάλι που παρέχει ο A/D converter.

Η τρίτη περιοχή της μνήμης ($FFF7 - $FFFF) είναι κατειλημμένη από τους καταχωρητές που χρησιμοποιεί το ενσωματωμένο, στη μνήμη RAM, real time clock. Οι παραπάνω καταχωρητές είναι read / write registers επιτρέποντας, τόσο την τοποθέτηση της τρέχουσας ημερομηνίας και ώρας, όποτε αυτό απαιτείται, όσο και την ανάγνωση των παραπάνω παραμέτρων για αποθήκευσή τους μαζί με τα αντίστοιχα δεδομένα.

Στο διάγραμμα ροής 5.2.1.α παρουσιάζεται η δομή του προγράμματος init.c. Στην πρώτη φάση του προγράμματος, αρχικοποιούνται οι καταχωρητές που σχετίζονται με τη διαχείριση των πόρων του μικροϋπολογιστή. Ακολουθεί η καταχώρηση αρχικών τιμών στις μεταβλητές που εν συνεχεία αποθηκεύονται στους καταχωρητές του RTC. Η καταχώρηση αυτή γίνεται από το χρήστη σύμφωνα με τη φόρμα που υπάρχει στο πρόγραμμα (παράρτημα 2). Στη συνέχεια αρχικοποιούνται οι δείκτες διαχείρισης της μνήμης, ram pointer, communication pointer και overflow pointer, και αποθηκεύονται στη μνήμη δεδομένων μηδενικά. Οι παράμετροι της ημερομηνίας και ώρας αποθηκεύονται στο real time clock και εκκινείται η λειτουργία του.

Προκειμένου να ελεγχθεί η ορθότητα της παραπάνω διαδικασίας, αποστέλλονται, ανά δευτερόλεπτο, μέσω της σειριακής θύρας COM του datalogger, οι τιμές των pointers, καθώς και το σύνολο των παραμέτρων της ημερομηνίας και ώρας.




**Διάγραμμα ροής 5.2.1.α.** Αρχικοποίηση της εξωτερικής μνήμης του datalogger.

### 5.2.2. Datalogs.c

Το πρόγραμμα αυτό είναι υπεύθυνο για την εύρυθμη λειτουργία του σταθμού υπαίθρου. Αναλυτικότερα εκτελεί τις παρακάτω λειτουργίες :

1) Δειγματοληψία 7 αναλογικών καναλιών μέσω του A/D converter με ρυθμό δειγματοληψίας 1 (sample/min)/channel.

2) Αποθήκευση αποτελεσμάτων της δειγματοληψίας καθώς και Ιουλιανής ημέρας ώρας και λεπτού στην εξωτερική μνήμη RAM του datalogger.

3) Προτοποθέτηση του modem και υλοποίηση πρωτοκόλλου επικοινωνίας για την επικοινωνία με τον κεντρικό σταθμό.

4)Αποστολή δεδομένων με hardware flow control.

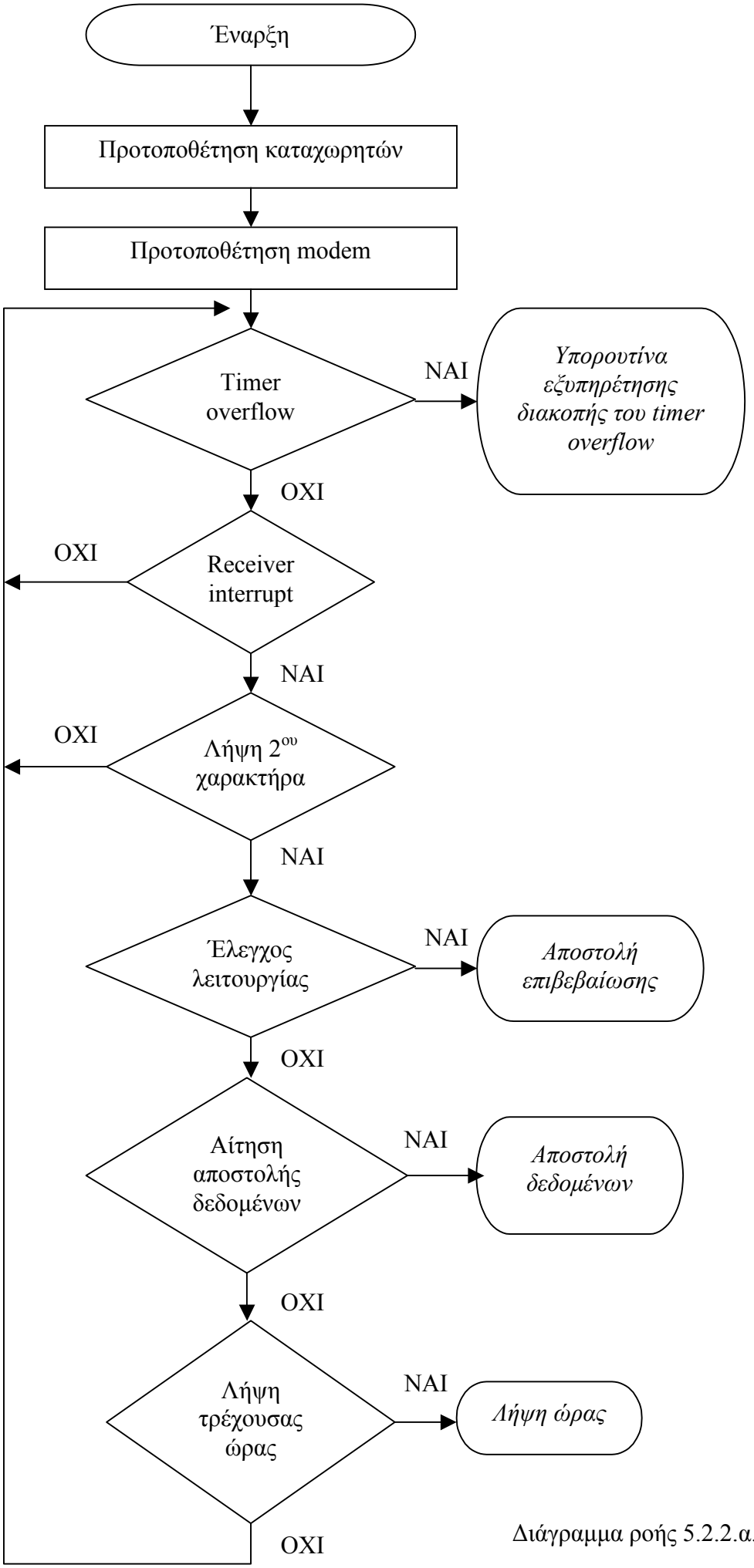
Στο διάγραμμα ροής 5.2.2.α. φαίνεται το κεντρικό loop του κυρίως προγράμματος. Αρχικά γίνεται η προτοποθέτηση κάποιων καταχωρητών ελέγχου του μικροελεγκτή (ports, timer overflow, UART receiver … ) και η αρχικοποίηση τοπικών και καθολικών μεταβλητών. Στη συνέχεια γίνεται προτοποθέτηση του modem ώστε να απαντά αυτόματα στις κλήσεις του κεντρικού σταθμού μετά από δύο κουδουνισμούς, να αγνοεί το σήμα handshaking DTR και να λειτουργεί σε echo off mode, με την εντολή

***AT&D0E0S0=2↵***

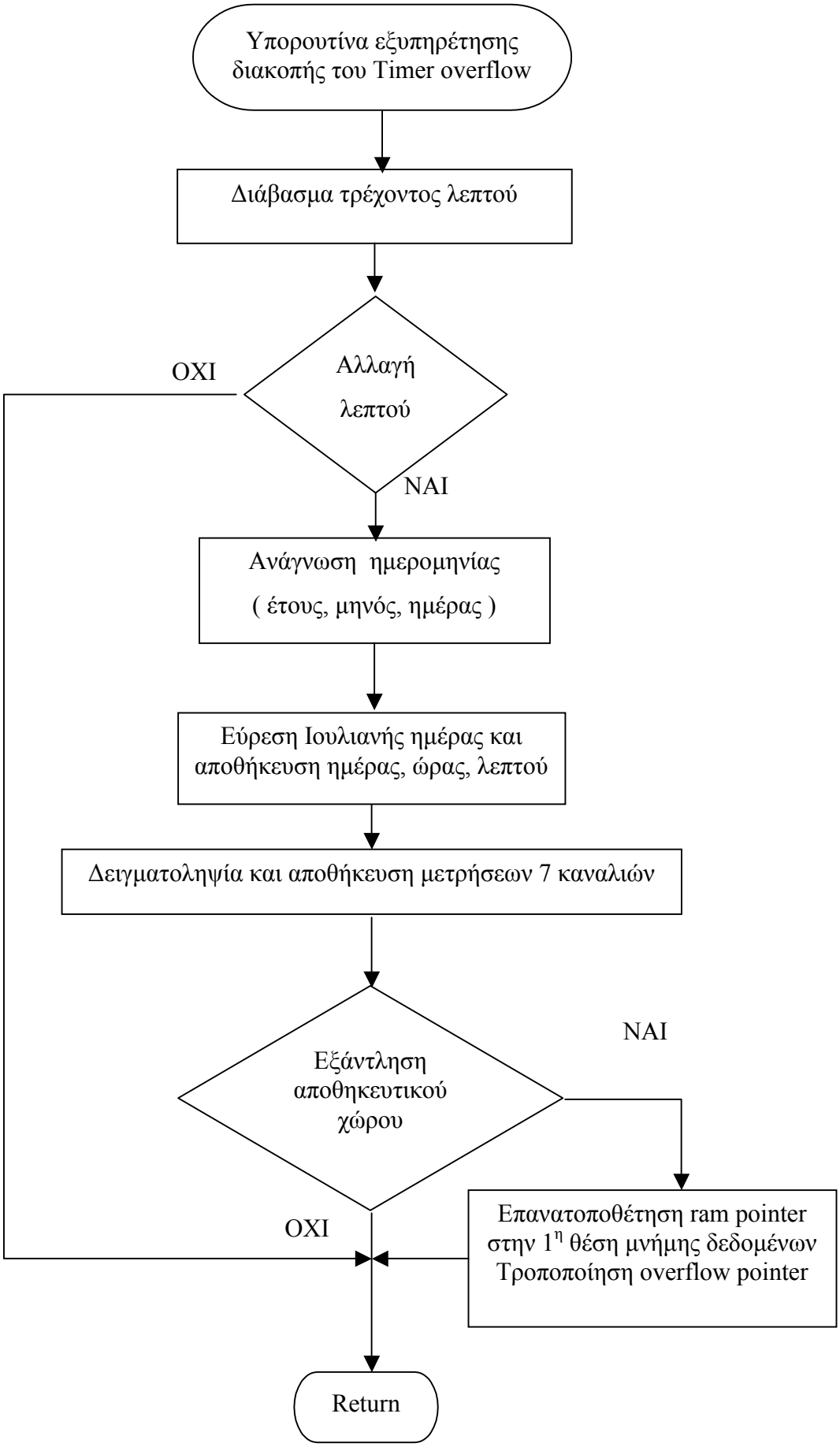
Στο κεντρικό loop του προγράμματος ο μικροϋπολογιστής περιμένει εντολές από τον κεντρικό σταθμό από τις οποίες εξαρτάται ποια από τις ακόλουθες λειτουργίες θα εκτελέσει:

1) Αποστολή επιβεβαίωσης καλής λειτουργίας (με την εντολή RU)
2) Αποστολή δεδομένων (με την εντολή SD)
3) Αναμονή για λήψη νέας ώρας από τον κεντρικό σταθμό (με την εντολή TM)

Κατά τη διάρκεια της συνεχούς εκτέλεσης όλων των παραπάνω, κάθε 1 sec στο πρόγραμμα πραγματοποιείται διακοπή, η υπορουτίνα εξυπηρέτησης της οποίας φαίνεται στο διάγραμμα ροής 5.2.2.β.




Διάγραμμα ροής 5.2.2.α.




**Διάγραμμα ροής 5.2.2.β.** Υπορουτίνα εξυπηρέτησης διακοπής του timer overflow.

Η συγκεκριμένη υπορουτίνα, όπως φαίνεται και στο παραπάνω διάγραμμα, ελέγχει την αλλαγή του λεπτού και στην περίπτωση που αυτή η συνθήκη αληθεύει διαβάζει την ημερομηνία και την ώρα, υπολογίζει και αποθηκεύει την Ιουλιανή ημέρα, την τρέχουσα ώρα και το λεπτό, δειγματοληπτεί το αναλογικό σήμα των 7 καναλιών και αποθηκεύει τα δεδομένα στη μνήμη RAM. Με το πέρας της δειγματοληψίας του 7ου καναλιού, ο A/D converter προγραμματίζεται σε λειτουργία standby μέχρι την επόμενη μέτρηση. Η συγκεκριμένη λειτουργία μειώνει αισθητά την κατανάλωση ισχύος του συστήματος. Στην περίπτωση που ο αποθηκευτικός χώρος της μνήμης εξαντληθεί, συνεχίζουμε την αποθήκευση από την πρώτη θέση της μνήμης δεδομένων. Πιο συγκεκριμένα θέτουμε την τιμή του ram pointer ίση με 0x800C και ενημερώνουμε τον overflow pointer για την υπερκάλυψη της μνήμης δεδομένων.

# ΚΕΦΑΛΑΙΟ 6$^{Ο}$ : ΜΕΤΡΗΣΕΙΣ

Μετά την ολοκλήρωση της ανάπτυξης του συστήματος, τόσο σε επίπεδο hardware, όσο και σε επίπεδο software, ακολούθησε ο έλεγχος της καλής και αξιόπιστης λειτουργίας του. Πιο συγκεκριμένα πριν τη δοκιμή του συστήματος σε πραγματικές μετρήσεις μεταβολών ηλεκτρομαγνητικού πεδίου, δοκιμάστηκε η απόδοσή του με τη μέτρηση προτύπου ημιτονικού σήματος. Με τη χρήση του datalogger, λήφθηκαν μετρήσεις από την ψηφιακή γεννήτρια Synthesized Generator DSG1 της Farnell, με δυνατότητα ρύθμισης του σήματος σε πολύ χαμηλές συχνότητες, της τάξεως των δεκάτων του mHz ( $10^{-4}$ Hz ). Το πλάτος του ημιτονοειδούς σήματος που μετρήθηκε ήταν 6Vp-p και η περίοδος που επιλέχτηκε ήταν :

***T=1/v => T=(1/ $10^{-4}$)sec => T=$10^{4}$sec => T=166min 40sec => T=2h 46min 40sec***

Επειδή το εύρος των μετρούμενων τάσεων, όπως αυτό ρυθμίστηκε για τις ανάγκες του τηλεμετρικού δικτύου είναι 0-5 V, η αρνητική ημιπερίοδος του ημιτόνου ελήφθη από το datalogger σαν μηδέν. Επομένως στον πίνακα που ακολουθεί παρατίθενται μόνο τις μετρήσεις της θετικής ημιπεριόδου του σήματος, που διήρκεσε ***t=T/2=>t=83min 20sec***

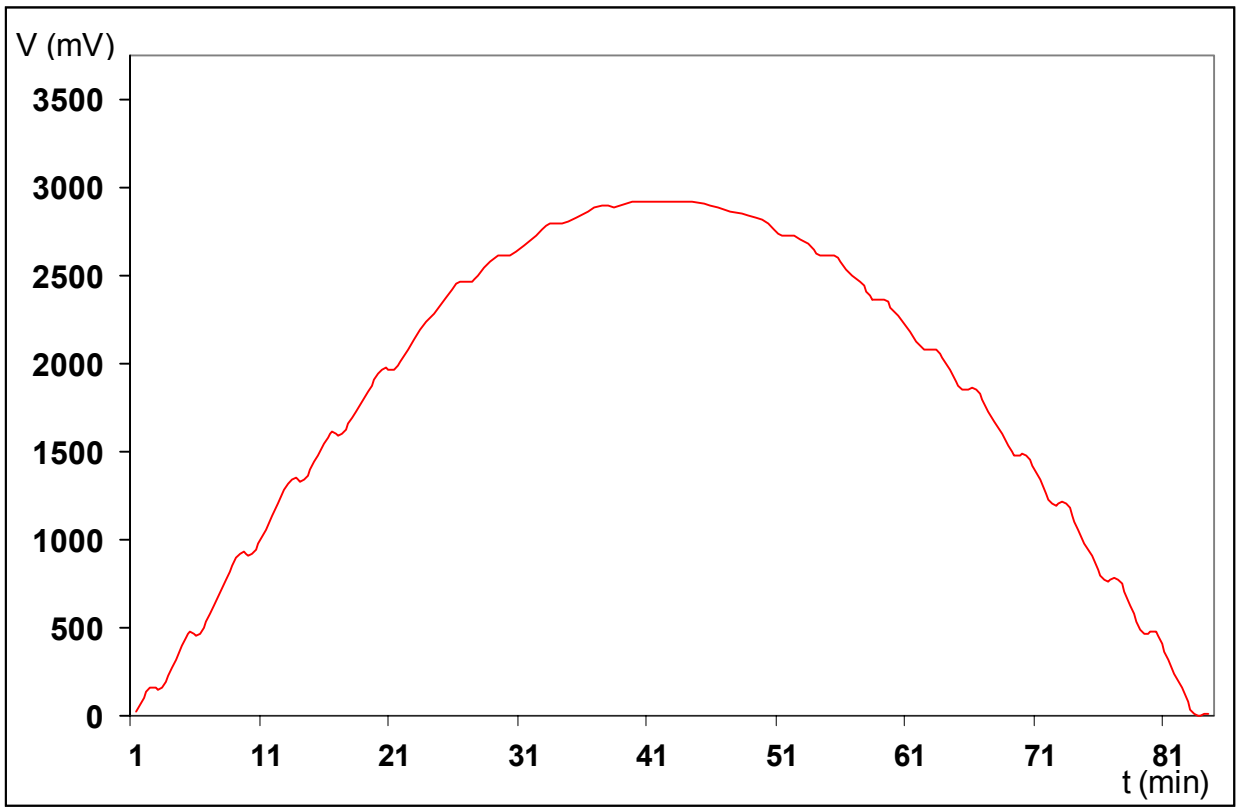
***Πίνακας Μετρήσεων***

| Time (min) | Voltage(mV) | Time (min) | Voltage(mV) |
|---|---|---|---|
| 1 | 20,8 | 43 | 2926,0 |
| 2 | 158,7 | 44 | 2915,0 |
| 3 | 162,4 | 45 | 2912,6 |
| 4 | 323,5 | 46 | 2890,6 |
| 5 | 471,2 | 47 | 2860,1 |
| 6 | 471,2 | 48 | 2857,7 |
| 7 | 623,8 | 49 | 2829,6 |
| 8 | 775,2 | 50 | 2794,2 |
| 9 | 916,8 | 51 | 2729,5 |
| 10 | 915,5 | 52 | 2729,5 |
| 11 | 1057,1 | 53 | 2677,0 |
| 12 | 1202,4 | 54 | 2609,9 |
| 13 | 1344,0 | 55 | 2611,1 |
| 14 | 1340,3 | 56 | 2537,8 |
| 15 | 1481,9 | 57 | 2465,8 |
| 16 | 1604,0 | 58 | 2364,5 |
| 17 | 1605,2 | 59 | 2365,7 |

| | | | |
|---|---|---|---|
| 18 | 1732,2 | 60 | 2277,8 |
| 19 | 1845,7 | 61 | 2187,5 |
| 20 | 1969,0 | 62 | 2074,0 |
| 21 | 1970,2 | 63 | 2076,4 |
| 22 | 2076,4 | 64 | 1967,8 |
| 23 | 2188,7 | 65 | 1849,4 |
| 24 | 2280,3 | 66 | 1846,9 |
| 25 | 2369,4 | 67 | 1731,0 |
| 26 | 2463,4 | 68 | 1606,4 |
| 27 | 2460,9 | 69 | 1481,9 |
| 28 | 2541,5 | 70 | 1481,9 |
| 29 | 2612,3 | 71 | 1344,0 |
| 30 | 2614,7 | 72 | 1203,6 |
| 31 | 2674,6 | 73 | 1202,4 |
| 32 | 2728,3 | 74 | 1053,5 |
| 33 | 2794,2 | 75 | 913,1 |
| 34 | 2797,9 | 76 | 771,5 |
| 35 | 2825,9 | 77 | 771,5 |
| 36 | 2858,9 | 78 | 621,3 |
| 37 | 2894,3 | 79 | 471,2 |
| 38 | 2891,8 | 80 | 473,6 |
| 39 | 2913,8 | 81 | 322,3 |
| 40 | 2922,4 | 82 | 159,9 |
| 41 | 2923,6 | 83 | 13,4 |
| 42 | 2922,4 | 84 | 11,0 |

Με βάση τις μετρήσεις που παραθέτουμε στον παραπάνω πίνακα προκύπτει το διάγραμμα που ακολουθεί, στο οποίο σαφώς διακρίνεται η ημιτονική μορφή του λαμβανομένου σήματος. Το σήμα ωστόσο θα μπορούσε να ήταν αρκετά ομαλότερο αν είχε επιλεγεί ανώτερος από 1 sample/min ρυθμός δειγματοληψίας για τις μετρήσεις. Η ανάλυση που εξασφαλίζει ο A/D converter είναι 12 bit, δηλαδή κβαντίζει το σήμα με 4096 στάθμες, οι οποίες για μετρήσεις όπως οι παραπάνω σε εύρος 0-5 V μας εξασφαλίζουν ακρίβεια 1.22mV. Το σύνολο των μετρήσεων, που έχει ακρίβεια πρώτου δεκαδικού ψηφίου του mV, επεξεργάστηκε στο λογισμικό πακέτο excel και απεικονίζεται στην κυματομορφή που ακολουθεί.

Η οδοντωτή μορφή του ημιτονικού σήματος δικαιολογείται από το γεγονός ότι η γεννήτρια παράγει βηματικά σήμα κβαντισμένης μορφής.




*Κυματομορφή σήματος από γεννήτρια πρότυπου ημιτονικού σήματος*

# ΚΕΦΑΛΑΙΟ 7$^{Ο}$ : ΕΠΕΚΤΑΣΙΜΟΤΗΤΑ ΚΑΙ ΑΝΑΒΑΘΜΙΣΕΙΣ

Το σύστημα τηλεμετρίας που παρουσιάστηκε σε επίπεδο δομής, software και hardware στα προηγούμενα κεφάλαια, είναι η πρώτη έκδοση ενός συστήματος που θα αποτελέσει βάση για την υπό σχεδίαση, νέα, αναβαθμισμένη, δεύτερη έκδοση.

Στο νέο σύστημα πρόκειται να συμπεριληφθούν τα εξής χαρακτηριστικά και βελτιώσεις.

1) Ανώτερος ρυθμός δειγματοληψίας και διαδικασία εύρεσης του μέσου όρου του μετρούμενου σήματος για πιο αξιόπιστα αποτελέσματα.
2) Επέκταση της μνήμης του συστήματος με την τοποθέτηση ισοδύναμων RAM μνημών, παράλληλα μεταξύ τους.
3) Περαιτέρω εκμετάλλευση της ήδη υπάρχουσας μνήμης, είτε με καλύτερη διαχείρισή της, είτε με την αποθήκευση συμπιεσμένων δεδομένων και την αποστολή αυτών για αποσυμπίεση και επεξεργασία στον κεντρικό σταθμό. Η εν λόγω διαδικασία αναμένεται να αυξήσει στο τριπλάσιο την αποθηκευτική ικανότητα του σταθμού και επομένως και την αυτονομία του.
4) Αντικατάσταση του υπάρχοντος A/D converter με άλλο αντίστοιχης αξιοπιστίας, αλλά μεγαλύτερης διακριτικής ικανότητας (14 ή 16 bits resolution) για μεγαλύτερης ακρίβειας μετρήσεις.
5) Αναβάθμιση του προγράμματος επικοινωνίας του κεντρικού σταθμού για την υπό συνθήκη και προαιρετική ανανέωση της ώρας του σταθμού βάσης.
6) Από απόσταση προγραμματισμός του datalogger, ώστε να είναι δυνατός ο επαναπρογραμματισμός του σταθμού υπαίθρου από το σταθμό βάσης.
7) Λειτουργία του datalogger σε τοπικό επίπεδο για συλλογή δεδομένων σε PC με ασύρματη σειριακή επικοινωνία με modules που υλοποιούν half duplex επικοινωνία σε ελεύθερη συχνότητα. Τα modules που έχουν ήδη επιλεγεί, ανήκουν στην εταιρεία aurel και επιτρέπουν datalogging από απόσταση έως και 50m, χωρίς τη χρήση ενισχυτή.
8) Έλεγχος του modem για λειτουργία μόνο κατά την επικοινωνία για εξοικονόμηση ενέργειας τόσο από το datalogger όσο και από το ίδιο το modem. Η ρύθμιση αυτή θα αυξήσει την αυτονομία σε ενέργεια του σταθμού υπαίθρου.

9) Χρήση πολυπλέκτη στις αναλογικές εισόδους του datalogger για δειγματοληψία περισσοτέρων καναλιών εφ' όσον αυτό είναι επιθυμητό σε κάποια εφαρμογή.

Οι παραπάνω επεκτάσεις αναμένεται να αναβαθμίσουν το datalogger και να το καταστήσουν πιο ανταγωνιστικό στην αγορά, αφού τα νέα του χαρακτηριστικά θα ανταποκρίνονται σε μεγαλύτερο πλήθος εφαρμογών.

# ΚΕΦΑΛΑΙΟ 8$^{Ο}$ : ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην ενότητα που ακολουθεί, θα αναφερθούμε στα σημαντικότερα συμπεράσματα και θα κάνουμε μια γενική ανασκόπηση και αποτίμηση του project.

Η πρώτη, ίσως και ουσιαστικότερη, αιτία ανάπτυξης ενός τηλεμετρικού δικτύου με βάση νέα dataloggers, ήταν το πολύ υψηλό κόστος αγοράς και συντήρησης των dataloggers της αγοράς και πιο συγκεκριμένα της εταιρείας Campbell Scientific, της οποίας τα dataloggers χρησιμοποιούνται στο υπάρχον τηλεμετρικό δίκτυο για την ανίχνευση των μεταβολών του ηλεκτρομαγνητικού πεδίου της γης πριν τους σεισμούς. Η ανάπτυξη λοιπόν των νέων dataloggers έγινε με γνώμονα το χαμηλό κόστος κατασκευής και συντήρησής τους αν λάβουμε υπ' όψιν ότι το κόστος των εξαρτημάτων και του κουτιού ανέρχεται στα 128 Euro ή 43.800δρχ όταν τα dataloggers της Campbell με αντίστοιχα τεχνικά χαρακτηριστικά κοστίζουν 20 με 25 φορές περισσότερο.

Αντικείμενο ιδιαίτερης μέριμνας αποτέλεσε η αξιοπιστία της συσκευής καθώς το datalogger έχει προδιαγραφές για να λειτουργεί 24 ώρες το 24ωρο 365 μέρες το χρόνο κάτω από τις αντίξοες συνθήκες ενός σταθμού υπαίθρου. Η παράμετρος αυτή είναι ουσιώδης αν αναλογιστεί κανείς πόσο κοστίζει η επίσκεψη τεχνικού στο σταθμό υπαίθρου για συντήρηση του hardware και του software. Η αξιοπιστία του τηλεμετρικού συστήματος βασίζεται στα εξαιρετικής ποιότητας εξαρτήματα που χρησιμοποιήθηκαν, στην τήρηση των κανόνων σχεδίασης που προτείνουν οι κατασκευαστές των εξαρτημάτων και οι σχεδιαστές αντιστοίχων συστημάτων και ο κατά το δυνατόν συμπαγής και πλήρης προγραμματισμός των σταθμών βάσης και υπαίθρου. Αξίζει να σημειωθεί η καλή συμπεριφορά του datalogger τόσο σε θέματα ηλεκτρομαγνητικής συμβατότητας όσο και σε θέματα εκπομπών (electromagnetic immunity & compatibility).

Το συγκεκριμένο σύστημα τηλεμετρίας έχει εξ' ολοκλήρου αναπτυχθεί από τους συγγραφείς και επομένως είναι εφικτή οποιαδήποτε αλλαγή και αναβάθμιση του software και hardware του συστήματος, διαδικασία που έχει ήδη ξεκινήσει μετά την πρώτη έκδοση του συστήματος. Εξέχουσας σημασίας είναι και η απόκτηση τεχνογνωσίας σε ένα αντικείμενο, για το οποίο οι εταιρείες προσφέρουν ελάχιστες πληροφορίες σε τεχνικό επίπεδο, καθώς η τηλεμετρία θεωρείται από τις πλέον αξιόλογες και προσοδοφόρες τεχνολογίες της Ηλεκτρονικής και της Πληροφορικής.

Σημείο σημαντικό που εξηγεί ένα μεγάλο μέρος των χαρακτηριστικών της σχεδίασης και προγραμματισμού του συστήματος είναι οι προδιαγραφές βάσει των οποίων υλοποιήθηκε. Κρίνοντας συνολικά, αλλά και σημείο προς σημείο το σύστημα, μπορούμε να πούμε ότι βρίσκεται εντός των απαιτούμενων προδιαγραφών. Πιο συγκεκριμένα επετεύχθησαν τα ακόλουθα:

α) Η απρόσκοπτη επικοινωνία του κεντρικού σταθμού με τον σταθμό υπαίθρου ανά 12 ώρες, ο ταυτόχρονος έλεγχος της καλής λειτουργίας του και η λήψη των δεδομένων που έχουν αποθηκευτεί στο χρονικό διάστημα από την προηγούμενη επικοινωνία μ' αυτόν.

β) Η αμφίδρομη επικοινωνία μεταξύ κεντρικού σταθμού και σταθμών μετρήσεων που επιτρέπει την ανταλλαγή κάποιων σημαντικών δεδομένων, όπως για παράδειγμα της τρέχουσας ώρας, που λαμβάνει ο σταθμός υπαίθρου, ώστε να είναι πάντα συγχρονισμένος με τον κεντρικό σταθμό.

γ) Η επικοινωνία του κεντρικού σταθμού με απεριόριστο αριθμό σταθμών υπαίθρου καθώς το τηλεμετρικό δίκτυο έχει απεριόριστες δυνατότητες επέκτασης με νέους σταθμούς υπαίθρου.

δ) Η επικοινωνία και η μεταφορά δεδομένων με ελάχιστη πιθανότητα απώλειας ή εσφαλμένης λήψης αυτών. Πιο συγκεκριμένα, η ανάπτυξη ενός πρωτοκόλλου για την επικοινωνία μεταξύ των σταθμών και η χρήση του πρωτοκόλλου επικοινωνίας RS232 με πλήρη σήματα handshaking, μειώνουν το ενδεχόμενο του λάθους στο ελάχιστο. Βαρύτητα επίσης δόθηκε στην περίπτωση που ο κεντρικός σταθμός δεν καταφέρει να επικοινωνήσει για περισσότερο από δύο 12ωρα με κάποιον σταθμό υπαίθρου, οπότε η υπερκάλυψη της μνήμης οδηγεί σε απώλεια δεδομένων. Η απώλεια αυτή μειώθηκε στο ελάχιστο.

ε) Η αξιόπιστη κατασκευή και ο συμπαγής προγραμματισμός του datalogger, ώστε να ελαχιστοποιηθούν οι πιθανότητες βλάβης και κατ' επέκταση οι επισκέψεις τεχνικού στους σταθμούς υπαίθρου για συντήρηση του συστήματος.

στ) Η χαμηλή κατανάλωση του συστήματος που επιτρέπει την τροφοδότηση του από μπαταρίες και φωτοβολταϊκά στοιχεία για μεγάλο χρονικό διάστημα ακόμα και σε περιοχές με περιορισμένη ηλιοφάνεια. Το σύστημα υλοποιήθηκε με τη χρήση ολοκληρωμένων τεχνολογίας CMOS με τη δυνατότητα προγραμματισμού σε λειτουργία power down, ώστε να είναι δυνατή η επιπλέον εξοικονόμηση ενέργειας, όταν δεν απαιτείται τα ολοκληρωμένα να είναι εν πλήρη λειτουργία. Η χαμηλή κατανάλωση του συστήματος μειώνει και το θόρυβο που εισάγεται στις μετρήσεις, ο

οποίος πρέπει να παραμένει κατά το δυνατόν χαμηλότερος. Η παράμετρος του θορύβου, άλλωστε, είναι που δίνει συγκριτικό πλεονέκτημα στη λύση ενός συστήματος βασισμένο σε dataloggers σε σχέση με ένα pc-based σύστημα του οποίου η κατανάλωση και ο θόρυβος είναι αισθητά μεγαλύτεροι.

ζ) Διατήρηση των δεδομένων στη μνήμη του datalogger ακόμα και σε περίπτωση απώλειας της τροφοδοσίας. Η συγκεκριμένη ικανότητα του datalogger επετεύχθη με τη χρήση non-volatile μνήμης RAM η οποία περιέχει ενσωματωμένο επιτηρητή τάσης και μπαταρία λιθίου, ώστε να διατηρεί τα δεδομένα για 10 χρόνια χωρίς τροφοδοσία.

η) Η επανατοποθέτηση του συστήματος (reset) με τη χρήση του watchdog timer που διαθέτει ο AT90S8515, όταν για οποιοδήποτε λόγο το πρόγραμμα κολλήσει, ή η τάση τροφοδοσίας πέσει κάτω από ορισμένο κατώφλι.

θ) Η δημιουργία εύχρηστων προγραμμάτων, τα οποία ζητήθηκε να γραφούν σε περιβάλλον MS-DOS για να μπορούν να τρέχουν στον κεντρικό σταθμό και μάλιστα σε unattended-mode κατά τις βραδινές ώρες που το κόστος επικοινωνίας είναι μικρότερο και οι γραμμές του σταθερού δικτύου τηλεφωνίας λιγότερο κατειλημμένες.

ι) Τέλος το datalogger σχεδιάστηκε για δύο mode λειτουργίας. Το πρώτο mode είναι για τοπική συλλογή δεδομένων και επικοινωνία μέσω της σειριακής θύρας του PC, και το δεύτερο είναι για συλλογή δεδομένων από απομακρυσμένο σταθμό και αποστολή τους με τη χρήση εξωτερικού modem στον κεντρικό σταθμό για περαιτέρω επεξεργασία. Η συγκεκριμένη συσκευή μπορεί να φανεί χρήσιμη και στην περίπτωση που απαιτείται περαιτέρω ασφάλεια σε ένα σύστημα τηλεμετρίας πραγματικού χρόνου. Πιο συγκεκριμένα σε ένα σύστημα τηλεμετρίας πραγματικού χρόνου που για οποιοδήποτε λόγο χαθεί η επικοινωνία μεταξύ κεντρικού σταθμού και σταθμού υπαίθρου, το datalogger αποθηκεύει τα τρέχοντα δεδομένα και τα αποστέλλει στον κεντρικό σταθμό όταν αποκατασταθεί η επικοινωνία μειώνοντας έτσι τις απώλειες στο ελάχιστό.

Κεντρικός σκοπός της σχεδίασης ήταν να μειωθούν έως και εξαλειφθούν τα προβλήματα που παρουσιάζει το υπάρχον σύστημα τηλεμετρίας με τα dataloggers της Campbell. Πιο αναλυτικά, δόθηκε λύση στο πρόβλημα των περιορισμένων σε αριθμό baud rate επικοινωνίας, αφού το νέο datalogger έχει τη δυνατότητα να υποστηρίξει μια ευρεία ποικιλία ρυθμών επικοινωνίας. Με τη χρήση των σημάτων handshaking, έγινε επίσης εφικτή η επικοινωνία και σε ρυθμούς επικοινωνίας ανώτερους των 1200kbps ακόμα και όταν η γραμμή δεν είναι ιδιαιτέρως καλή, δυνατότητα που δεν

έχει το υπάρχον σύστημα. Με τη χρήση του πλήρους πρωτοκόλλου RS232 και των αντίστοιχων σταθμών RS232 των σημάτων είναι δυνατή η σύνδεση του datalogger με οποιοδήποτε τυπικό smart modem της αγοράς. Αξίζει να σημειωθεί ότι η Campbell έχει modem δικής της κατασκευής με στάθμες σήματος TTL και με πρωτόκολλο επικοινωνίας της εταιρείας, όχι μόνο για τεχνικούς λόγους (χαμηλή κατανάλωση ισχύος) αλλά και για προφανείς εμπορικούς λόγους.

Στο επόμενο και τελικό στάδιο αξιολόγησης του συστήματος που είναι αυτό της παράλληλης λειτουργίας των δύο συστημάτων για την επιβεβαίωση της καλής και αξιόπιστης λειτουργίας του νέου συστήματος θα μπορέσουμε να αποφανθούμε συνολικά για το βαθμό επιτυχίας του έργου. Ωστόσο πρέπει να παρατηρήσουμε ότι οι μετρήσεις εντός του εργαστηρίου είναι ιδιαιτέρως ενθαρρυντικές και μέχρι στιγμής όλες οι δοκιμές έχουν στεφθεί με επιτυχία. Με άλλα λόγια τα αποτελέσματα είναι θετικά και ανταποκρίνονται στους αρχικούς σχεδιασμούς και στόχους του project, ιδιαιτέρως αν συνυπολογιστεί η δρομολογημένη βελτίωση του συστήματος με μια σειρά από αλλαγές που ήδη σχεδιάζονται.

# ΠΑΡΑΡΤΗΜΑ 1 : ΤΕΧΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ DATALOGGER

Στο παράρτημα που ακολουθεί αναπτύσσονται τα επί μέρους χαρακτηριστικά του datalogger, όπως αυτά προέκυψαν από μια σειρά μετρήσεις με ψηφιακά όργανα ακριβείας και σε κάποιες περιπτώσεις από τα χαρακτηριστικά των ολοκληρωμένων, όπως δίνονται από τους κατασκευαστές και από τα οποία αποτελείται η συσκευή.

***Κατανάλωση Datalogger***

Στον παρακάτω πίνακα παρουσιάζονται οι μετρήσεις που σχετίζονται με την κατανάλωση του datalogger σε τέσσερις διαφορετικές περιπτώσεις. Στην περίπτωση που στο datalogger έχει συνδεσμολογηθεί η συσκευή του προγραμματιστή και είναι σε σύνδεση και η ενδεικτική πράσινη λυχνία και το modem, τότε η κατανάλωση είναι μέγιστη και φτάνει τα 118 mA. Ενώ στη συνηθισμένη κατάσταση της συσκευής που κανένα από τα παραπάνω δεν έχει συνδεθεί, η κατανάλωση είναι ελάχιστη και ανέρχεται σε 40 mA. Οι παραπάνω μετρήσεις έγιναν με ψηφιακό τροφοδοτικό HP E3631A (0-6V στα 5A, ±25V στο 1A) με ψηφιακή ένδειξη ρεύματος σε τάση λειτουργίας +12V.

| Περίπτωση | programmer | Led | modem | Συνολική κατανάλωση |
|---|---|---|---|---|
| $1^{η}$ | • | • | • | 118mA |
| $2^{η}$ | | • | • | 66mA |
| $3^{η}$ | | | • | 51mA |
| $4^{η}$ | | | | 40mA |

***Αναλογικές είσοδοι***

Το datalogger διαθέτει 8 αναλογικά κανάλια εισόδου single-ended.

Το εύρος του σήματος που μπορεί να δέχεται και η διακριτική ικανότητα της συσκευής είναι προγραμματιζόμενες παράμετροι του συστήματος. Πιο συγκεκριμένα:

- Για εύρος 0-5 V η διακριτική ικανότητα του datalogger είναι 1.2207 mV
- Για εύρος 0-10 V η διακριτική ικανότητα του datalogger είναι 2.4414mV

Η αντίσταση εισόδου των καναλιών είναι 10kΩ για λειτουργίες normal και standby power down.

Οι αναλογικές είσοδοι του datalogger μπορούν να δεχθούν υπερτάσεις χωρίς πρόβλημα σε εύρος ±16.5 V

***Ψηφιακές I/O***

Το σύστημα διαθέτει 5 ψηφιακές εισόδους / εξόδους γενικής χρήσης που μπορούν να χρησιμοποιηθούν για τη δημιουργία μονάδας ελέγχου.

***CPU και Interface***

Επεξεργαστής : AT90S8515

Μνήμη προγράμματος : 4K/8K Bytes προγραμματιζόμενη μνήμη Flash

Μνήμη δεδομένων : 32K Bytes (περίπου 32760 σημεία αποθήκευσης)

Interface για περιφερειακά :

α) 9pin D τύπου connector για σύνδεση με smart modem και οποιοδήποτε περιφερειακό υποστηρίζει πλήρες πρωτόκολλο RS232

β) 9pin D τύπου connector για σύνδεση του datalogger απ' ευθείας με Η/Υ

Baud rates : Υποστηρίζει 2400, 4800, 9600, 19200, 57600, 115200 bps με ASCII πρωτόκολλο , 1 start bit, 1 stop bit, 8 data bits και καμμία ισοτιμία (no parity).

Ακρίβεια ρολογιού : ±1 λεπτό το μήνα

***Φυσικές διαστάσεις***

Διαστάσεις ηλεκτρονικής πλακέτας 10.5cm x 11.5cm x 4.5cm

Διαστάσεις συσκευής datalogger 14.0cm x 15.0cm x 6.0cm

Το συνολικό βάρος της συσκευής είναι 350 gr.

Τα χαρακτηριστικά της συσκευής προκύπτουν κατά βάση από τα χαρακτηριστικά των ολοκληρωμένων από τα οποία αποτελείται. Για τον παραπάνω λόγο κρίθηκε απαραίτητη η παράθεση των χαρακτηριστικών των ολοκληρωμένων, όπως αυτά δίνονται από τους κατασκευαστές τους.

*Χαρακτηριστικά ολοκληρωμένων*

**AT90S8515 8bit AVR Microcontroller**

•Χρησιμοποιεί την αρχιτεκτονική RISC της AVR

• AVR – Υψηλής απόδοσης και χαμηλής ισχύος Αρχιτεκτονική RISC

– 118 ισχυρές εντολές εκτελέσιμες ως επί το πλείστον σε ένα κύκλο ρολογιού

– 32 καταχωρητές γενικής χρήσης των 8 bits

– ταχύτητα επεξεργασίας πάνω από 8 MIPS στα 8 MHz

• Μνήμες δεδομένων και προγράμματος (Nonvolatile)

– 4K/8K Bytes προγραμματιζόμενη μνήμη Flash

1,000 κύκλων εγγραφής / σβησίματος

– 256 / 512 Bytes μνήμη SRAM

– 256 / 512 Bytes προγραμματιζόμενη μνήμη EEPROM

100,000 κύκλων εγγραφής / διαγραφής

– Προγραμματιζόμενο κλείδωμα για το πρόγραμμα της Flash και ασφάλεια των δεδομένων της EEPROM

• Χαρακτηριστικά περιφερειακών

– 1 χρονιστής/μετρητής των 8 bit

– 1 χρονιστής/μετρητής των 16 bit με δυνατότητα PWM διαμόρφωσης

– ενσωματωμένος αναλογικός συγκριτής

– Προγραμματιζόμενο Watchdog Timer με ενσωματωμένο ταλαντωτή

– Προγραμματιζόμενο σειριακό UART

– Master/Slave σειριακό interface SPI

• Ειδικά χαρακτηριστικά του Μικροϋπολογιστή

– Λειτουργίες χαμηλής ισχύος και ελάχιστης κατανάλωσης

– Εξωτερικές και εσωτερικές πηγές διακοπών

• Χαρακτηριστικά

– Χαμηλή ισχύς, υψηλής ταχύτητας CMOS τεχνολογία επεξεργασίας

– Πλήρως στατική λειτουργία

• Κατανάλωση ισχύος στα 4 MHz, 3V, 25°C

– Ενεργό: 3.0 mA

– Λειτουργία χωρίς φορτίο: 1.0 mA

– Εκτός λειτουργίας : <1 μA

• Είσοδοι / έξοδοι και συσκευασίες

– 32 προγραμματιζόμενες γραμμές ελέγχου

– 40 ακίδων PDIP, 44 ακίδων PLCC and TQFP

• Τάσεις λειτουργίας

– 2.7 - 6.0V (AT90S4414-4 και AT90S8515-4)

– 4.0 - 6.0V (AT90S4414-8 και AT90S8515-8)

• Ταχύτητα

– 0 - 4 MHz (AT90S4414-4 και AT90S8515-4)

– 0 - 8 MHz (AT90S4414-8 και AT90S8515-8)

**DS1644 Nonvolatile Timekeeping Ram**

•Ολοκληρωμένη NV SRAM, με real time clock,
κρύσταλλο, κύκλωμα ελέγχου πτώσης της τάσης και πηγή τάσεως λιθίου

•οι καταχωρητές του ρολογιού προσπελαύνονται ιδιαιτέρως στη στατική μνήμη RAM. Αυτοί οι καταχωρητές είναι μονίμως στις 8 ανώτερες θέσεις της μνήμης RAM

• Απολύτως ευσταθής με πάνω από 10 χρόνια λειτουργίας εκτός τάσεως

• έτος, μήνας, ημερομηνία, ημέρα, ώρες, λεπτά, και δευτερόλεπτα κωδικοποιημένα σε BCD με υπολογισμό των δίσεκτων ετών που ισχύει έως και το 2100

• Προστασία από διακυμάνσεις ±10% της τάσης τροφοδοσίας VCC

• DS1644 DIP συσκευασία

•Συμβατότητα προς τα πάνω με την DS1643 Timekeeping RAM

•πρότυπο pin out στατικής μνήμης RAM 32k x 8 JEDEC

**MAX197 Multi-range 12bit A/D Converter**

• 12 bit ανάλυση, ½ LSB γραμμικότητα

• Τάση λειτουργίας +5V

• Επιλεγόμενες μέσω λογισμικού περιοχές σήματος εισόδου :
±10V, ±5V , 0V έως 10V και 0V έως 5V.

• Πολυπλέκτης προστασίας εισόδου ±16.5 V

• 8 κανάλια αναλογικής εισόδου

• Χρόνος μετατροπής 6 μs

• Ρυθμός δειγματοληψίας 100 ksps

• Έλεγχος λήψης εξωτερικά ή εσωτερικά

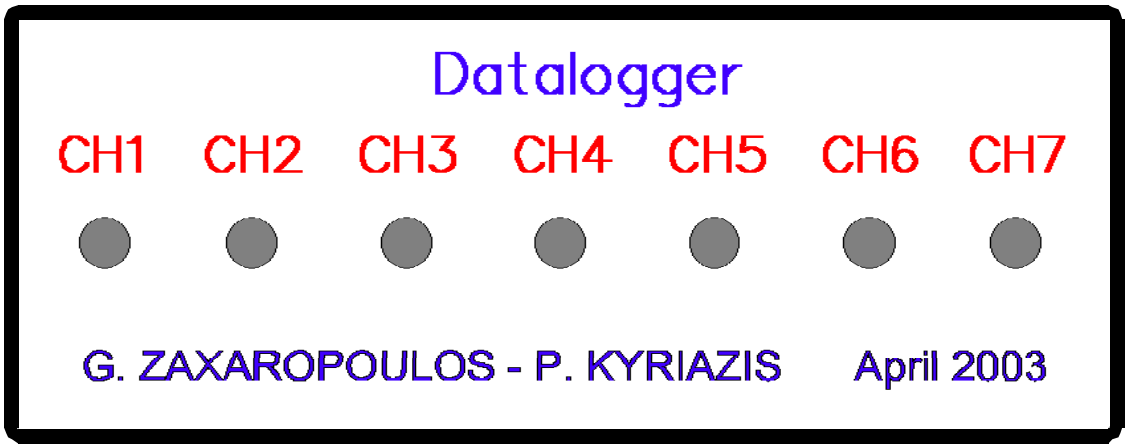
• Εσωτερική τάση αναφοράς 4.096 V ή εξωτερική αναφορά

• Δύο λειτουργίες μειωμένης κατανάλωσης

• εσωτερικό και εξωτερικό ρολό ι

***Τοπολογία πλακέτας datalogger και προσόψεις κατασκευής***

Στα σχήματα 1,2,3 και 4 που ακολουθούν παρουσιάζονται τα σχέδια των δύο μεταλλικών όψεων του κουτιού, που χρησιμοποιήθηκε για τον εγκιβωτισμό του datalogger.

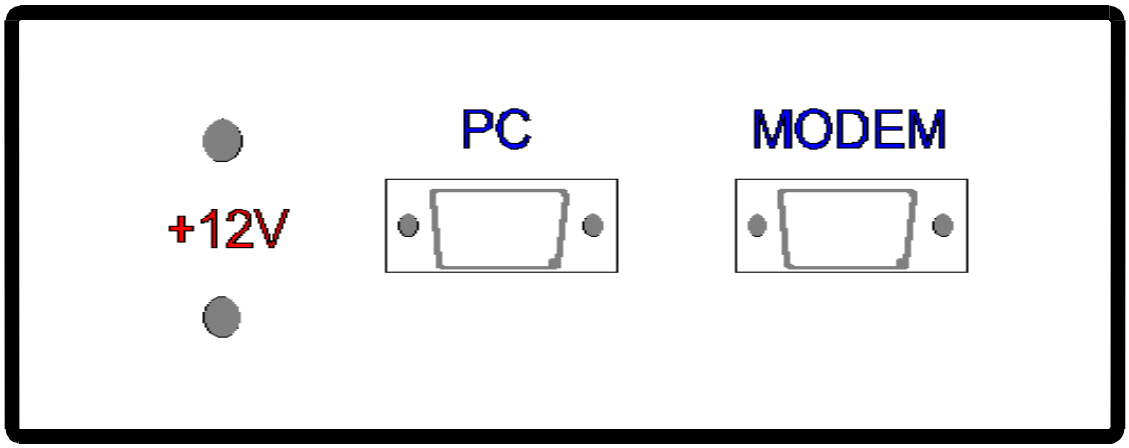



**Σχήμα 1.** Σχέδιο πρόσοψης κουτιού

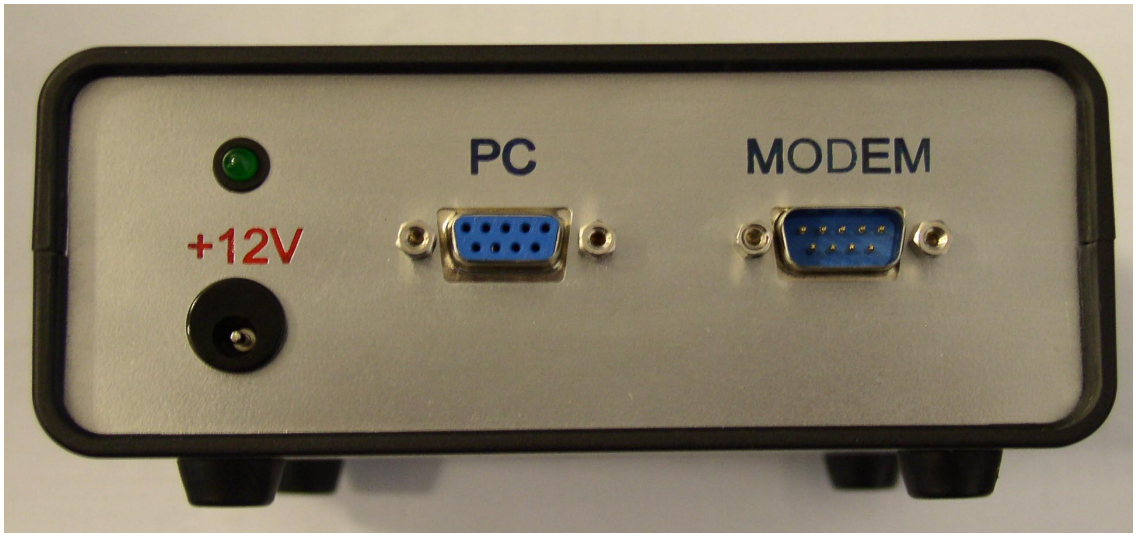

**Σχήμα 2.** Πρόσοψη κουτιού

Όπως φαίνεται στο παραπάνω σχήμα στην πρόσοψη της συσκευής υπάρχουν 7 οπές για ίσο αριθμό αναλογικών καναλιών στις οποίες έχουν τοποθετηθεί βύσματα RCA και στην ουσία είναι οι είσοδοι της συσκευής και το interface με τον αναλογικό κόσμο.

Στο σχήμα που ακολουθεί φαίνεται η πίσω όψη της συσκευής, όπου διακρίνονται οι δύο οπές για την τροφοδοσία με jack η μία και για την ενδεικτική πράσινη λυχνία, που πιστοποιεί τη λειτουργία της συσκευής η άλλη. Επίσης υπάρχουν θέσεις για δύο D-connectors, προκειμένου να συνδέεται η συσκευή με modem και PC αντίστοιχα

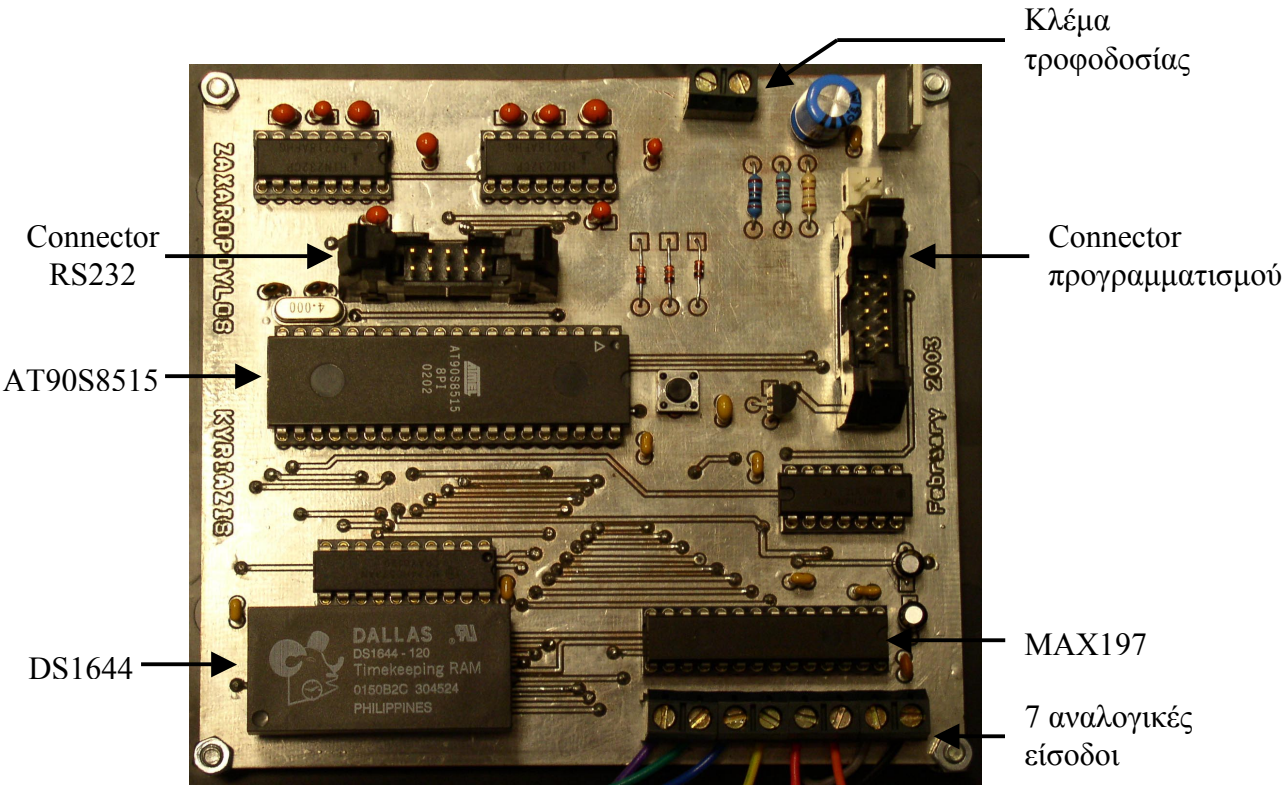



**Σχήμα 3.** Σχέδιο πίσω όψης του κουτιού




**Σχήμα 4.** Πίσω όψη του κουτιού

Στην εικόνα που ακολουθεί φαίνεται το εσωτερικό του datalogger, και τα επιμέρους σημεία της πλακέτας για την ευκολία του συντηρητή της συσκευής σε περίπτωση βλάβης ή του χειριστή στην περίπτωση που θέλει να επαναπρογραμματίσει τη συσκευή από τον αντίστοιχο connector που βρίσκεται επί της πλακέτας.




**Σχήμα 5.** Τοπολογία πλακέτας datalogger

## ΠΑΡΑΡΤΗΜΑ 2 : ΠΗΓΑΙΟΣ ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΩΝ

```
/*
Filename: datalogm.cpp
Project : Datalogger
Version : 1.0
Date    : 16/4/2003
Authors : G.Zaxaropoulos P.Kyriazis
Comments: Program to make .dli files
*/

#include<fstream.h>
#include<conio.h>
#include<string.h>
#include<process.h>

#define TRUE 1
#define FALSE 0

//main function
void main()
{

//functions declaration
int menu();
int new_station();
int delete_station();

//variables declaration
int exit=TRUE;

while(exit)
{
 clrscr();
 switch(menu())
 {
  case 1:  new_station();    getch(); break;
  case 2:  delete_station(); getch(); break;
  case 3:  exit=FALSE;               break;
  default: cout<<"\nWrong button. Please try again.";
getch();
 }//Close switch
}//Close while

}//Close main

/*
Function which displays the menu
and returns user's choice
*/
int menu()
{
```

```
int resp;

cout<<"(1) Create new station."<<endl;
cout<<"(2) Delete existing station."<<endl;
cout<<"(3) Exit program."<<endl;
cout<<"Make your choice: ";cin>>resp;

return resp;
}

/*
Function to create a new station
returns 0 if the process is successful
returns 1 if error
*/
int new_station()
{
//variables declaration
char station_name[13],phone_number[11];
unsigned int COM_number;
unsigned long baud_rate;


cout<<"Enter station name: ";cin>>station_name;
cout<<"Enter station phone number: ";cin>>phone_number;

do
{
 cout<<"Enter COM (1-COM1,2-COM2,3-COM3,4-COM4):
";cin>>COM_number;
}
while(COM_number!=1 && COM_number!=2 && COM_number!=3 &&
COM_number!=4);

do
{
 cout<<"Enter baud rate
(2400,4800,9600,19200,38400,57600,115200):
";cin>>baud_rate;
}
while(baud_rate!=2400 && baud_rate!=4800
&&baud_rate!=9600 &&
baud_rate!=19200 && baud_rate!=38400 && baud_rate!=57600
&& baud_rate!=115200);

ofstream fout(strcat(station_name,".dli"));

if(!fout)
{
  cout<<"Unable to create file "<<station_name<<".\n";
  return 1;
```

```
}
fout<<phone_number<<" "<<COM_number<<" "<<baud_rate;

fout.close();

cout<<"Info file created.";
return 0;
}


//Function to delete a station
int delete_station()
{
char station_name[13],command[18]="del ";

cout<<"Enter station name to delete: ";cin>>station_name;

strcat(station_name,".dli");
strcat(command,station_name);

system(command);
return 0;
}

/*
Filename: dataloge.cpp
Project : Datalogger
Version : 1.0
Date    : 16/4/2003
Authors : G.Zaxaropoulos P.Kyriazis
Comments: Program to edit .dli files
*/

#include<conio.h>
#include<fstream.h>
#include<iomanip.h>
#include<stdio.h>
#include<dos.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>

#define TRUE 1
#define FALSE 0

//global variables declaration
char phone_number[]="2106912358",station_name[8];
unsigned long current_baud=9600;
int current_COM=1;
int COM_counter=0,Baud_Rate_counter=0;
```

```
//main function
int main(int argc,char **argv)
{
clrscr();
//variables declaration
int c,TAB_counter=0,save_state=0;

//functions declaration
int read_current_spec(char filename[]);
void show_title(char station_name[]);
void show_menu();
void set_counters();
void refresh_screen(int TAB_counter);
void refresh_choises();
void change_option(int TAB_counter,int mode);
int decide_save();
int save_changes(char filename[],int mode);

if(argc!=2)
{
 cout<<"Invalid number of parameters in command line.\n";
 cout<<"Please run the programm again.";
 getch();
 return 1;
}
strcpy(station_name,argv[1]);

if(read_current_spec(argv[1])==1)
return 1;

set_counters();
show_title(station_name);
show_menu();
refresh_choises();
refresh_screen(0);
refresh_screen(2);
refresh_screen(0);

if(save_changes(argv[1],0)==1)
return 1;

do
{
 if(kbhit())
 {
   c=getch();
   if(c==9) //if TAB is pressed
   {
     TAB_counter++;

      if(TAB_counter==3)
```

```
       TAB_counter=0;

      refresh_screen(TAB_counter);
   }
   else if(c==43)// if '+' is pressed
   {
     save_state=1;
     change_option(TAB_counter,0);
     refresh_choises();
     refresh_screen(TAB_counter);
   }
   else if(c==45)// if '-' is pressed
   {
     save_state=1;
     change_option(TAB_counter,1);
     refresh_choises();
     refresh_screen(TAB_counter);
   }//close else if
 }//close if
}//close do while
while(c!=27); //wait for ESC

if(save_state==1)
{
  if(decide_save()==0)
    if(save_changes(argv[1],1)==1)
        return 1;
}
else
gotoxy(1,7);
getch();
return 0;
}

/*
Function which reads the current
specifications from the .dli file
and initializes the phone_number,
current_COM and current_baud.
*/
int read_current_spec(char filename[])
{
char current_station_name[13];

strcpy(current_station_name,filename);
ifstream ifile(strcat(current_station_name,".dli"));

if(!ifile)
{
 cout<<"Unable to open file
"<<current_station_name<<".\n";
```

```
 return 1;
}

ifile>>phone_number>>current_COM>>current_baud;

ifile.close();
return 0;
}

/*
Function which shows the menu
the first time we run the program.
*/
void show_menu()
{
char temp[8];
void set_text(char text[],int mode);

cout<<"\n(1)Telephone number: ";
set_text(phone_number,0);

cout<<"\n(2)Baud Rate: ";
set_text(ultoa(current_baud,temp,10),1);

cout<<"\n(3)COM: ";
set_text(itoa(current_COM,temp,10),1);
}

/*
Function to display the text[] on the screen
mode 0: Black text and white background.
mode 1: Light gray text and black background.
*/
void set_text(char text[],int mode)
{
char ch[15];
if(mode==0)
{
textcolor(BLACK);
textbackground(WHITE);
}
else
{
textcolor(LIGHTGRAY);
textbackground(BLACK);
}
sprintf(ch,text);
cprintf("%s",ch);
}

/*
```

```
Function which reads the current position of the
menu and moves the cursor to the next point.
*/
void refresh_screen(int TAB_counter)
{
char temp[8];
void set_text(char text[],int mode);
void clear(int x,int y);

 switch(TAB_counter)
 {
   case 0:
    clear(9,6);
    set_text(itoa(current_COM,temp,10),1);
    clear(22,4);
    set_text(phone_number,0);
   break;

   case 1:
    clear(22,4);
    set_text(phone_number,1);
    clear(15,5);
    set_text(ultoa(current_baud,temp,10),0);
   break;

   case 2:
    clear(15,5);
    set_text(ultoa(current_baud,temp,10),1);
    clear(9,6);
    set_text(itoa(current_COM,temp,10),0);
   break;
 }
}

/*
Function which clears the x line on the
screen from the y column declared to the end
*/
void clear(int x,int y)
{
gotoxy(x,y);
textbackground(BLACK);
clreol();
}

/*
Function which initializes Baud_Rate_counter
and COM_counter in relation to the Baud rate
and the COM that we have read from .dli file.
*/
void set_counters()
```

```
{
 switch(current_baud)
 {
   case 2400:
    Baud_Rate_counter=0;
   break;

   case 4800:
    Baud_Rate_counter=1;
   break;

   case 9600:
    Baud_Rate_counter=2;
   break;

   case 19200:
    Baud_Rate_counter=3;
   break;

   case 38400:
    Baud_Rate_counter=4;
   break;

   case 57600:
    Baud_Rate_counter=5;
   break;

   case 115200:
    Baud_Rate_counter=6;
   break;

   default:
   Baud_Rate_counter=2;

 }

 switch(current_COM)
 {
   case 1:
    COM_counter=0;
   break;

   case 2:
    COM_counter=1;
   break;

   case 3:
    COM_counter=2;
   break;

   case 4:
```

```
    COM_counter=3;
   break;

   default:
    COM_counter=0;

 }
}

/*
Function which increases or decreases
Baud_Rate_counter and COM_counter takes
account of the TAB_counter and what we have
pressed on the keyboard (+ or -).
*/
void change_option(int TAB_counter,int mode)
{
 switch(TAB_counter)
 {
   case 1:
    if(mode==0)
    {
     Baud_Rate_counter++;

      if(Baud_Rate_counter==7)
        Baud_Rate_counter=0;
     }
    else
    {
     Baud_Rate_counter--;

      if(Baud_Rate_counter==-1)
         Baud_Rate_counter=6;
     }
   break;

   case 2:
    if(mode==0)
    {
     COM_counter++;
     if(COM_counter==4)
        COM_counter=0;
     }
     else
     {
      COM_counter--;

       if(COM_counter==-1)
          COM_counter=3;
      }
   break;
```

```
 }
}

/*
Function to refresh the current_baud
and current_COM according to the
Baud_Rate_counter and the COM_counter.
*/
void refresh_choises()
{
 switch(Baud_Rate_counter)
 {
  case 0:
   current_baud=2400;
  break;

  case 1:
   current_baud=4800;
  break;

  case 2:
   current_baud=9600;
  break;

  case 3:
   current_baud=19200;
  break;

  case 4:
   current_baud=38400;
  break;

  case 5:
   current_baud=57600;
  break;

  case 6:
   current_baud=115200;
  break;
 }

 switch(COM_counter)
 {
  case 0:
   current_COM=1;
  break;

  case 1:
   current_COM=2;
  break;
```

```
  case 2:
   current_COM=3;
  break;

  case 3:
   current_COM=4;
  break;
 }
}

/*
Function which shows the name of the
station in the middle of the screen.
*/
void show_title(char station_name[])
{
int length,i;
char temp[25];

length=strlen(station_name);
gotoxy((35-length),1);

for(i=0;i<=(15+length);i++)
 cout<<"*";

gotoxy((35-length),2);
sprintf(temp,"*Station name: %s*",station_name);
cout<<temp;
gotoxy((35-length),3);

for(i=0;i<=(15+length);i++)
 cout<<"*";
}

/*
Function to save new specifications
if it is desired so by the user.
*/
int decide_save()
{
char save_char;
int exit=TRUE,state,y=7;

 while(exit)
 {
   gotoxy(1,y);
   cout<<"Save changes? (y/n): ";cin>>save_char;
   save_char=(char)toupper(save_char);

   switch(save_char)
   {
```

```
    case 'Y':
     state=0;
     exit=FALSE;
    break;

    case 'N':
     state=1;
     exit=FALSE;
    break;
   }
   y++;
 }
return state;
}

/*
Function which saves the current
specifications to the .dli file
before exit the program.
returns 0 :when ok
returns 1 :when error
*/
int save_changes(char filename[],int mode)
{
char current_station_name[13];

strcpy(current_station_name,filename);
ofstream ofile(strcat(current_station_name,".dli"));

if(!ofile)
{
 cout<<"Unable to open file "<<current_station_name<<"to
save changes\n";
 return 1;
}

ofile<<phone_number<<" "<<current_COM<<" "<<current_baud;

ofile.close();

if(mode==1)
 cout<<"\nInfo file has been updated.";

return 0;
}
```

```
/*
Filename: datalogc.cpp
Project : Datalogger
Version : 1.0
Date    : 2/5/2003
Authors : G.Zaxaropoulos P.Kyriazis
Comments: Program to communicate and receive data
*/

#include <fstream.h>
#include <dos.h>
#include <stdio.h>
#include <conio.h>
#include <bios.h>
#include <dtlg.h>
#include <string.h>
#include <time.h>
#include <stdlib.h>

#define SETTINGS ( 0xE0 | 0x00 | 0x00 | 0x03)

#ifdef __cplusplus
    #define __CPPARGS ...
#else
    #define __CPPARGS
#endif

//global variables declaration
int bufferin = 0;
int bufferout = 0;
char ch;
char buffer[1025];
int PORT1=0x3F8;

char mdm_cmd1[]="ATE0&S0&C1&D2&K3\r";
char mdm_cmd2[25]="ATX3DT";

char
mdm_rsp1[]="OK",mdm_rsp2[]="ERROR",mdm_rsp3[]="BUSY";
char mdm_rsp4[]="NO CARRIER",mdm_rsp5[]="NO ANSWER" ;

char avr_cmd1[]="RU*",avr_cmd2[]="SD*",avr_cmd3[]="TM*";

void interrupt (*oldport1isr)(__CPPARGS);

//Interrupt Service Routine (ISR) for PORT1
void interrupt PORT1INT(__CPPARGS)
{
int c;
do
{
```

```
 c=inportb(PORT1+5);
 if(c&1)
 {
  buffer[bufferin]=inportb(PORT1);
  bufferin++;

  if(bufferin==1024)
  {
   bufferin=0;
  }
 }
}
while(c&1);

outportb(0x20,0x20);
}


int main(int argc, char **argv)
{
clrscr();
//local variables declaration
char filename[9],station_name[9],phone_number[11];
int COM_number;
unsigned long baud_rate;

int c;
int COM_base_address,COM_IVE,COM_baud_rate;
int set_PIC,turn_off_PIC;

int timeout;
int cnt1=0;
char cur_mdm_rsp[15];
int mode=0;

//functions declaration
int read_current_spec(char filename[],char
*phone_number,int& COM_number,unsigned long& baud_rate);
void init_port_values(int COM_number,unsigned long
baud_rate,int& COM_base_address,int& COM_IVE,int&
COM_baud_rate,int &set_PIC,int& turn_off_PIC);
void open_com(int COM_number,int COM_base_address,int
COM_IVE,int COM_baud_rate,int set_PIC);
void close_com(int COM_IVE,int turn_off_PIC);
void send_command(char command[]);
int wait_answer(char *modem_response,int timeout);
int convert_to_dat(char filename[],int mode);
void set_AVR_time();

if(argc!=2)
{
```

```
 cout<<"Invalid number of parameters in command line.\n";
 cout<<"Please run the programm again.";
 getch();
 return 1;
}

strcpy(station_name,argv[1]);
strcpy(filename,station_name);

if(read_current_spec(filename,&phone_number[0],COM_number
,baud_rate)==1)
return 1;


init_port_values(COM_number,baud_rate,COM_base_address,CO
M_IVE,COM_baud_rate,set_PIC,turn_off_PIC);

open_com(COM_number,COM_base_address,COM_IVE,COM_baud_rat
e,set_PIC);

//check handshaking signals ( CTS & DSR )
while(!inportb(COM_base_address+6)&0x30)
{
  ;
}
send_command(mdm_cmd1); //initialize modem

if(wait_answer(&cur_mdm_rsp[0],1000))
 return 1;

strcat(mdm_cmd2,phone_number);
strcat(mdm_cmd2,"\r");

//check handshaking signals ( CTS & DSR )
while(!inportb(COM_base_address+6)&0x30)
{
  ;
}
send_command(mdm_cmd2); //dial command
cout<<"\rWait while connecting...";
sleep(60); // wait while modems try to connect

if(wait_answer(&cur_mdm_rsp[0],500))
{
  cout<<"\nConnection failed.";
  close_com(COM_IVE,turn_off_PIC); //close COM
  return 1;
}
```

```
if(strstr(cur_mdm_rsp,mdm_rsp3)!=NULL ||
strstr(cur_mdm_rsp,mdm_rsp4)!=NULL ||
strstr(cur_mdm_rsp,mdm_rsp5)!=NULL)
{
  cout<<"\nNo carrier or busy or no answer.";
  close_com(COM_IVE,turn_off_PIC); //close COM
  return 1;
}

cout<<"\nConnection established.";

//check handshaking signals ( CTS & DSR & DCD)
while(!inportb(COM_base_address+6)&0xB0)
{
  ;
}
send_command(avr_cmd1); //send RU
delay(2000);

if(wait_answer(&cur_mdm_rsp[0],2000))
{
  cout<<"\nDatalogger communication error.";
  close_com(COM_IVE,turn_off_PIC); //close COM
  return 1;
}


ofstream ofile("temp.txt");
if(!ofile)
 return 1;

delay(1500);

//check handshaking signals ( CTS & DSR & DCD)
while(!inportb(COM_base_address+6)&0xB0)
{
  ;
}
send_command(avr_cmd2); //send SD

cout<<"\nDownloading data.";

cnt1=0;

do
{
 if (bufferin != bufferout)
 {
  cnt1=0;
  ch = buffer[bufferout];
  bufferout++;
```

```
  if (bufferout == 1024)
   bufferout = 0;

  if(ch!='S'|| ((int)ch>=0x30 && (int)ch<=0x39) ||
(int)ch==0x0A)
  {
 //cout<<ch;
   ofile<<ch;
  }
 }
 else
 {
  delay(500);
  cnt1++;
 }

 if(cnt1>=20)
 {
  mode=1;
  break;
 }
}
while(ch!='S');

ofile.close();
cout<<"\nFile has been transfered.";
convert_to_dat(station_name,mode); //make .dat file

send_command(avr_cmd3); //send TM
delay(2000);

if(wait_answer(&cur_mdm_rsp[0],2000))
{
 close_com(COM_IVE,turn_off_PIC); //close COM
 return 1;
}

set_AVR_time(); //send central station current time
cout<<"\nDatalogger time has been changed.";
close_com(COM_IVE,turn_off_PIC); //close COM

return 0;
}

/*
Function which returns appropriate values of
a)COM base address
b)Interrupt vector entry IVE
c)Baud rate
d)Value to set Programmable the Interrupt Controller
set_PIC
```

```
e)Value to turn off interrupts
to ititialize port.
To open port after this function the open_com function
has to be called.
*/
void init_port_values(int COM_number,unsigned long
baud_rate,int& COM_base_address,int& COM_IVE,int&
COM_baud_rate,int &set_PIC,int& turn_off_PIC)
{

switch(COM_number)
{
 case 1:
  PORT1=0x3F8;
  COM_base_address=0x3F8;
  COM_IVE=0x0C;
  set_PIC=0xEF;
  turn_off_PIC=0x10;
 break;

 case 2:
  PORT1=0x2F8;
  COM_base_address=0x2F8;
  COM_IVE=0x0B;
  set_PIC=0xF7;
  turn_off_PIC=0x08;
 break;

 case 3:
  PORT1=0X3E8;
  COM_base_address=0x3E8;
  COM_IVE=0x0C;
  set_PIC=0xEF;
  turn_off_PIC=0x10;
 break;

 case 4:
  PORT1=0X2E8;
  COM_base_address=0x2E8;
  COM_IVE=0x0B;
  set_PIC=0xF7;
  turn_off_PIC=0x08;
 break;
}

switch (baud_rate)
{
 case 2400:
  COM_baud_rate=0x30;
 break;
```

```
 case 4800:
  COM_baud_rate=0x18;
 break;

 case 9600:
  COM_baud_rate=0x0C;
 break;

 case 19200:
  COM_baud_rate=0x06;
 break;

 case 57600:
  COM_baud_rate=0x02;
 break;

 case 115200:
  COM_baud_rate=0x01;
 break;
}
}

/*
Function to open the port.
Before calling this function,
init_port_values function must be called
*/
void open_com(int COM_number,int COM_base_address,int
COM_IVE,int COM_baud_rate,int set_PIC)
{
bioscom(0,SETTINGS,(COM_number-1));
outportb(COM_base_address+1,0);  //Turn off interrupts -
Port1

oldport1isr=getvect(COM_IVE); //Save old Interrupt Vector
for later recovery

setvect(COM_IVE,PORT1INT);   // Set Interrupt Vector
Entry
                                  // COM1 - 0x0C
                                  // COM2 - 0x0B
                                  // COM3 - 0x0C
                                  // COM4 - 0x0B

 //PORT 1 - Communication Settings

outportb(COM_base_address+3,0x80);          // SET DLAB
ON
outportb(COM_base_address+0,COM_baud_rate); // Set Baud
rate - Divisor Latch Low Byte
                           // Default 0x03 =  38,400 BPS
```

```
                                //          0x01 = 115,200 BPS
                                //          0x02 =  57,600 BPS
                                //          0x06 =  19,200 BPS
                                //          0x0C =   9,600 BPS
                                //          0x18 =   4,800 BPS
                                //          0x30 =   2,400 BPS

outportb(COM_base_address+1,0x00);  // Set Baud rate -
Divisor Latch High Byte
outportb(COM_base_address+3,0x03);  // 8 Bits, No Parity,
1 Stop Bit
outportb(COM_base_address+2,0xC7);  // FIFO Control
Register
outportb(COM_base_address+4,0x0B);  // Turn on DTR, RTS,
and OUT2

outportb(0x21,(inportb(0x21)&set_PIC)); // Set
Programmable Interrupt Controller
                                    // COM1 (IRQ4) - 0xEF
                                    // COM2 (IRQ3) - 0xF7
                                    // COM3 (IRQ4) - 0xEF
                                    // COM4 (IRQ3) - 0xF7

outportb(COM_base_address+1,0x01);       // Interrupt
when data received
}

/*
Function which closes the COM
and returns the interrupt vectors
*/
void close_com(int COM_IVE,int turn_off_PIC)
{
outportb(PORT1+4,0x00);    // Turn off DTR, RTS, and OUT2
outportb(PORT1+1,0);       // Turn off interrupts - Port1
outportb(0x21,inportb(0x21) | turn_off_PIC);  // MASK IRQ
using PIC
                                       // COM1 (IRQ4) - 0x10
                                       // COM2 (IRQ3) - 0x08
                                       // COM3 (IRQ4) - 0x10
                                       // COM4 (IRQ3) - 0x08
setvect(COM_IVE, oldport1isr); // Restore old interrupt
vector
}

/*
Function wich takes high and low
byte of any measurement and returns
the value in mV
*/
float convert_data(int low,int high)
```

```
{
int tmp;
tmp=((256*high)+low);

return (float)tmp*1.220703;
}

/*********************************************
This program was produced by the
CodeWizardAVR V1.0.2.1b Standard
Automatic Program Generator
© Copyright 1998-2001
Pavel Haiduc, HP InfoTech S.R.L.
http://infotech.ir.ro
e-mail:dhptechn@ir.ro , hpinfotech@xnet.ro

Filename:init.c
Project :Datalogger
Version :1.0
Date    :20/4/2003
Author  :Zaxaropoulos G. - Kyriazis P.
Comments:Program to initialize external memory and RTC


Chip type           : AT90S8515
Clock frequency     : 4,000000 MHz
Memory model        : Small
Internal SRAM size  : 512
External SRAM size  : 32K
Data Stack size     : 128
*********************************************/

#include <90s8515.h>
#include <mem.h>
#include <bcd.h>
#include <delay.h>
#include <stdio.h>

// Declare your global variables here
void main(void)
{
// Declare your local variables here
unsigned int i;

/*** Date and time must be imported here to initialize
the real time clock***/

//Import the current year
unsigned char year=3;  //***example:for year 2003 type
3***
```

```
//Import the current month
unsigned char month=2; //***example:for May type 5***

//Import the current date
unsigned char date=3;  //***example:for the 16th day of a
month type 16***

//Import the current day
unsigned char day=1;   //***example:for Monday type 1***

//Import the current hour
unsigned char hour=9;  //***example:for 10pm type 22***

//Import the current minute
unsigned char min=0;   //***example:for the 38th minute
of an hour type 38***

//Import the current second
unsigned char sec=0;   //***example:for the  54th second
of a minute type 54***


// Input/Output Ports initialization
// Port A
PORTA=0x00;
DDRA=0x00;


// Port B
PORTB=0x00;
DDRB=0xFD;

// Port C
PORTC=0x00;
DDRC=0x00;

// Port D
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Output Compare
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
```

```
// Mode: Output Compare
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
GIMSK=0x00;
MCUCR=0x80;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;

// UART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// UART Receiver: On
// UART Transmitter: On
// UART Baud rate: 9600
UCR=0x18;
UBRR=0x19;

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



delay_ms(500);

pokew(0x8002,0x800C);                  //Initialization
of ram pointer

printf("%-u ",(unsigned int)peekw(0x8002));  //Ram
pointer points at the
                                              //beginning
of data memory
putchar(0x0A);

delay_ms(500);
```

```
pokew(0x8004,0x800C);                  //Initialization
of communication pointer

printf("%-u ",(unsigned int)peekw(0x8004));
//Communication pointer points at
                                                 //the
beginning of data memory
putchar(0x0A);

delay_ms(500);



for(i=0x8008;i<=0xFFF0;i++)    // Loop to fill data
memory with zeros (0)
{                               //to be ready to store new
data
  pokeb(i,0);
}


pokeb(0xFFF8,0x80);           //Stop clock

pokeb(0xFFFF,bin2bcd(year));
pokeb(0xFFFE,bin2bcd(month)); //Initialize all the
parameters of RTC
pokeb(0xFFFD,bin2bcd(date));
pokeb(0xFFFC,bin2bcd(day));
pokeb(0xFFFB,bin2bcd(hour));
pokeb(0xFFFA,bin2bcd(min));
pokeb(0xFFF9,bin2bcd(sec));

pokeb(0xFFF8,0x00);           //Start clock

delay_ms(2000);

/*
for(i=0x8000;i<=0xFFF7;i++)    //Loop to send through a
serial port of the PC
{                               //the content of the data
memory to verify it is
 printf("%-i ",(int)peekb(i)); //cleared (filled with
zeros)
 if(i%20==0)                   // (NOT COMPULSORY)
 putchar(0x0A);
} */
```

```
putchar(0x0A);

while (1)
{
// Place your code here


pokeb(0xFFF8,0x40);          //Stop clock

year=bcd2bin(peekb(0xFFFF)); //Read date and time from
the RTC
month=bcd2bin(peekb(0xFFFE));//and store the parameters
in variables
date=bcd2bin(peekb(0xFFFD));
hour=bcd2bin(peekb(0xFFFB));
min=bcd2bin(peekb(0xFFFA));
sec=bcd2bin(peekb(0xFFF9));

pokeb(0xFFF8,0x00);          //Start clock


printf("%-i %-i %-i %-i %-i %-
i",(int)year,(int)month,(int)date,(int)hour,(int)min,(int
)sec);

putchar(0x0A);// send the parameters of time and date
through serial port

printf("%-u ",(unsigned int)peekw(0x8002));  //send the
value of ram pointer

printf("%-u ",(unsigned int)peekw(0x8004));  //send the
value of

//communication pointer

putchar(0x0A);

delay_ms(1000);


};
}
```

```
/*********************************************
This program was produced by the
CodeWizardAVR V1.0.2.1b Standard
Automatic Program Generator
© Copyright 1998-2001
Pavel Haiduc, HP InfoTech S.R.L.
http://infotech.ir.ro
e-mail:dhptechn@ir.ro , hpinfotech@xnet.ro

Filename: datalogs.c
Project : Datalogger
Version : 1.0
Date    : 18/4/2003
Author  : G.Zaxaropoulos P.Kyriazis
Comments: Program for remote datalogger


Chip type           : AT90S8515
Clock frequency     : 4,000000 MHz
Memory model        : Small
Internal SRAM size  : 512
External SRAM size  : 32K
Data Stack size     : 128
*********************************************/
/*
*******RS-232*******
*hanshaking signals*

PINB2  =CTS
PORTB3 =RTS
PINB4  =DCD

PORTD0 =RXD
PORTD1 =TXD
PIND4  =DSR
PORTD5 =DTR
*/

#include <90s8515.h>
#include <mem.h>
#include <bcd.h>
#include <string.h>
#include <delay.h>
#pragma used+

// UART Receiver buffer
char rx_buffer[8];
volatile unsigned char
rx_wr_index,rx_rd_index,rx_counter;

// This flag is set on UART Receiver buffer overflow
```

```
bit rx_buffer_overflow;
#pragma used-

// UART Receiver interrupt service routine
#pragma savereg-

interrupt [UART_RXC] void uart_rx_isr(void)
{
#asm
    .equ __rx_buffer_size=8

    push r26
    in   r26,sreg
    push r26
    push r27
    push r30
    push r31
#endasm
#ifdef _MODEL_TINY_
#asm
    ldi  r26,_rx_buffer
    lds  r30,_rx_wr_index
    add  r26,r30
    clr  r27
#endasm
#endif
#ifdef _MODEL_SMALL_
#asm
    ldi  r26,low(_rx_buffer)
    ldi  r27,high(_rx_buffer)
    lds  r30,_rx_wr_index
    clr  r31
    add  r26,r30
    adc  r27,r31
#endasm
#endif
#asm
    in   r30,udr
    st   x,r30
    lds  r30,_rx_wr_index
    inc  r30
    cpi  r30,__rx_buffer_size
    brlo __uart_rx_isr0
    clr  r30
__uart_rx_isr0:
    sts  _rx_wr_index,r30
    lds  r30,_rx_counter
    inc  r30
    sts  _rx_counter,r30
    cpi  r30,__rx_buffer_size+1
    brlo __uart_rx_isr1
```

```
#endasm
rx_buffer_overflow=1;
#asm
__uart_rx_isr1:
    pop  r31
    pop  r30
    pop  r27
    pop  r26
    out  sreg,r26
    pop  r26
#endasm
}
#pragma savereg+

#ifndef _DEBUG_TERMINAL_IO_
#include <uart.h>
// Get a character from the UART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma warn-
#pragma used+
char getchar(void)
{
#asm
    lds  r30,_rx_counter
    tst  r30
    breq _getchar
#endasm
#ifdef _MODEL_TINY_
#asm
    ldi  r26,_rx_buffer
    lds  r30,_rx_rd_index
    add  r26,r30
    clr  r27
#endasm
#endif
#ifdef _MODEL_SMALL_
#asm
    ldi  r26,low(_rx_buffer)
    ldi  r27,high(_rx_buffer)
    lds  r30,_rx_rd_index
    clr  r31
    add  r26,r30
    adc  r27,r31
#endasm
#endif
#asm
    inc  r30
    cpi  r30,__rx_buffer_size
    brlo __getchar0
    clr  r30
__getchar0:
```

```
    sts  _rx_rd_index,r30
    ld   r30,x
    cli
    lds  r26,_rx_counter
    dec  r26
    sts  _rx_counter,r26
    sei
#endasm
}
#pragma used-
#ifdef _WARNINGS_ON_
#pragma warn+
#endif
#endif

// Standard Input/Output functions
#include <stdio.h>

unsigned char new_min,old_min;
unsigned int find_ioul(unsigned char year,unsigned char
month,unsigned char day);

// Timer 1 overflow interrupt service routine
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
//Values declaration
unsigned char i,year,month,day;
unsigned int ioulian_day,temp_pointer;

// Reinitialize Timer's 1 value
TCNT1H=0xC2;
TCNT1L=0xF6;


pokeb(0xFFF8,0x40);     //Set read bit in the control
register of the RTC to stop the clock

new_min=bcd2bin(peekb(0xFFFA)); //Read current minute

pokeb(0xFFF8,0x00);     //Reset read bit in the control
register of the RTC to restart the clock

if(new_min!=old_min)    //Check the change of the minute
{
 pokeb(0xFFF8,0x40);    //Set read bit in the control
register of the RTC to stop the clock

 year=bcd2bin(peekb(0xFFFF));  //Read the parameters of
date from RTC
 month=bcd2bin(peekb(0xFFFE));
 day=bcd2bin(peekb(0xFFFD));
```

```
 ioulian_day=find_ioul(year,month,day);//Calculate the
ioulian day

 temp_pointer=peekw(0x8002);           //Save ram pointer
in temp variable

 pokew(temp_pointer,ioulian_day);      //Store ioulian
day in memory

 temp_pointer+=2;                      //Increase temp
pointer to jump to the next word of memory

 pokeb(temp_pointer,bcd2bin(peekb(0xFFFB))); //Store the
current hour in RAM

 temp_pointer++;                       //Increase temp
pointer to jump to the next byte of memory

 pokeb(temp_pointer,bcd2bin(peekb(0xFFFA))); //Store the
current minute in RAM

 temp_pointer++;                       //Increase temp
pointer to jump to the next byte of memory

 pokeb(0xFFF8,0x00);    //Reset read bit in the control
register of the RTC to restart the clock


 for(i=0;i<=6;i++)
 {

  if(i<6)

   pokeb(0x0360,(0x40|i));   //Control byte for the first
6 channels

  else

   pokeb(0x0360,(0x80|i));   //Control byte for the 7th
channel

  while(PINB.1==1);            //Wait for /INT

  PORTB.0=0;                  //Ask for the low byte of
the current channel

  pokeb(temp_pointer,peekb(0x0360));  //Read and store
the low byte of the current channel
```

```
  temp_pointer++;                      //Increase ram
pointer

  PORTB.0=1;                 //Ask for the high byte of
the current channel

  pokeb(temp_pointer,peekb(0x0360));  //Read and store
the high byte of the current channel

  temp_pointer++;                      //Increase ram
pointer

 }


 while(temp_pointer%20!=0)     // Ensure that temp
pointer points at a new block of 20 bytes

  temp_pointer++;              // Ignore the last two
blank bytes of the block



 do
 {
  pokew(0x8002,temp_pointer);
 }
 while(peekw(0x8002)!=temp_pointer);

 if(temp_pointer==0xFFF0)      // Memory reached to the
end of storage space
 {

  pokew(0x8002,0x800C);         // Start storing again
from the beginning

  if(peekb(0x800A)==0x00)      // Check for the 1st
overflow of the memory

   pokeb(0x800A,0x01);         // Set the overflow
pointer

  else if(peekb(0x800A)==0x01) // Check for more
overflows of the memory

   pokeb(0x800A,0x02);         // Set the overflow
pointer
 }

old_min=new_min;               //Renew the value of the
old minute
```

```
}

}

//Global functions declaration
void send_ok();
void get_time();
void send_data_rout();
void send_all();
void send_pointers();
int  send_data_subrout(unsigned int start_addr,unsigned
int end_addr);

//Global variables declaration
flash char
tst1[]="RU*",tst2[]="TM*",tst3[]="SD*",tst4[]="GO*",tst5[
]="PN*";
flash char  mdm_init[]="AT&D0E0Q1S0=2\r";

void main(void)
{

//Local variables declaration
unsigned char bufcnt,i;
char ch[20];
unsigned int tmp_poin;
bufcnt=rx_counter;
i=0;

// Input/Output Ports initialization
// Port A
PORTA=0x00;
DDRA=0x00;

// Port B
PORTB=0x00;
DDRB=0xE9;

// Port C
PORTC=0x00;
DDRC=0x00;

// Port D
PORTD=0x00;
DDRD=0x20;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Output Compare
```

```
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 15,625 kHz
// Mode: Output Compare
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1A=0x00;
TCCR1B=0x04;
TCNT1H=0xC2;
TCNT1L=0xF6;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
GIMSK=0x00;
MCUCR=0xC0;   //or 0x80

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x80;

// UART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// UART Receiver: On
// UART Transmitter: On
// UART Baud rate: 9600
UCR=0x98;
UBRR=0x19;

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

pokeb(0xFFF8,0x40);          //Set read bit in the
control register of the RTC to stop the clock

new_min=bcd2bin(peekb(0xFFFA)); //Read current minute

pokeb(0xFFF8,0x00);     //Reset read bit in the control
register of the RTC to restart the clock
```

```
old_min=new_min;        //Renew the value of the old
minute


tmp_poin=peekw(0x8002);

while(tmp_poin%20!=0)             //Decrease ram pointer
{
 tmp_poin--;                      //Ensure that the block
is appropriate
}

do
{
 pokew(0x8002,tmp_poin);
}
while(peekw(0x8002)!=tmp_poin);


tmp_poin=peekw(0x8004);

while(tmp_poin%20!=0)             //Decrease ram pointer
{
 tmp_poin--;                      //Ensure that the block
is appropriate
}

do
{
 pokew(0x8004,tmp_poin);
}
while(peekw(0x8004)!=tmp_poin);


delay_ms(4000);
putsf(mdm_init);       //Initialization AT commands for
the MODEM

PORTD.5=0; //DTR on
PORTB.3=0; //RTS on

// Global enable interrupts
#asm("sei")

while (1)
{
// Main code

 if(bufcnt!=rx_counter)
 {
```

```
  ch[i]=getchar();

  i++;

 }


 if(ch[i-1]=='*')
 {
  ch[i]='\0';

  if(strstrf(ch,tst1)!=NULL)  //Case RU*
   send_ok();

  if(strstrf(ch,tst2)!=NULL)  //Case TM*
   get_time();

  if(strstrf(ch,tst3)!=NULL)  //Case SD*
   send_data_rout();

  if(strstrf(ch,tst4)!=NULL)  //Case GO*
   send_all();

  if(strstrf(ch,tst5)!=NULL)  //Case PN*
   send_pointers();

  i=0;
 }

};
}

void send_ok()
{

 printf("\r\nOK\r\n");

}

//Function to send all memory in a different format
//to check the performance of the remote station
void send_all()
{
unsigned int i;

putchar(0x0A);

printf("Communication pointer=%-u\r",peekw(0x8004));

printf("RAM pointer= %-u\r",peekw(0x8002));
```

```
for(i=0x800C;i<=0xFFF0;i++)
{
 if(i%20==0)

 printf("%-u ",(unsigned int)i);

 #asm("cli")

 printf("%-i ",(int)peekb(i));

 #asm("sei")

 if((i+1)%20==0)

 putchar(0x0A);
}

}

//Function to send the current values of the pointers at
the central station

void send_pointers()
{

printf("\r\nCommunication pointer=%-u\r",peekw(0x8004));

printf("RAM pointer= %-u\r",peekw(0x8002));

printf("Overflow pointer= %-u\r",peekw(0x800A));

}

//Function to calculate the ioulian day
unsigned int find_ioul(unsigned char year,unsigned char
month,unsigned char day)
{
unsigned int x,y,yearf,monthf,dayf;

yearf=(unsigned int)year;

monthf=(unsigned int)month;

dayf=(unsigned int)day;

yearf+=2000;

y=monthf;
y--;
x=y*31;
```

```
if(monthf>=2)
{
 x-=(y/2);

 if(monthf>=8 && monthf%2!=0)
  x++;

 if (monthf>2)
 {
  if ((yearf%4==0 && yearf%100!=0) || (yearf%400==0))
   x--;
  else
   x-=2;
 }
 return (x+dayf);
}

else
{
 return dayf;
}
}
```

**Βιβλιογραφία**


1. Τηλεμετρία :Κ.Νομικός, Ινστιτούτο Τεχνολογικής Εκπαίδευσης ©Αθήνα 1995
2. Mastering Serial Communication : Peter W. Gofton, SYBEX ©Berkley 1986
3. C Programmer's Guide to Serial Communications :Joe Campbell, Sams Publishing ©Indianapolis 1994
4. Προγραμματίζοντας τον μικροελεγκτή AVR :Dhananjay V. Gadre ©
5. Microprocessors and Digital Systems Second Edition : Douglas V. Hall, Mc Graw Hill, ©Portland 1983
6. Microprocessor Technology : j.S.Anderson © 1994
7. Datasheet Multirange (±10V, ±5V, +10V,+5V), Single +5V, 12-Bit DAS with 8+4 Bus Interface MAX 197, MAXIM ©USA 1996
8. Datasheet DS1644/DS1644P Nonvolatile Timekeeping RAM, Dallas Semiconductor © 2001
9. Datasheet 8-bit AVR Microcontroller with 8K Bytes In-System Programmable Flash AT90S8515, ATMEL Corporation © 2001
10. Instruction manual of 21X and CR10 Micrologger , Campbell Scientific ©USA 1990.
11. User manual of Code Vision AVR V1.0.1.8, HP InfoTech S.R.L © 1998-2001