Ελληνικό Μεσογειακό Πανεπιστήμιο
Τμήμα Ηλεκτρονικών Μηχανικών

Σχεδίαση συστήματος μέτρησης και καταγραφής θερμοκρασιών ψυγείων.

Μεταπτυχιακή Διπλωματική Εργασία
του
Ζήση Μακρή

Επιβλέπων : Εμμανουήλ Αντωνιδάκης Καθηγητής Ελληνικό Μεσογειακό Πανεπιστήμιο

Περίληψη

Στην εργασία αυτή σχεδιάστηκε και αναπτύχθηκε ένα σύστημα μέτρησης και καταγραφής θερμοκρασιών ψυγείων και ψυκτικών θαλάμων. Η λύση που προτείνεται μπορεί να είναι ζωτικής σημασίας σε όλες τις ιατρικές εγκαταστάσεις, ειδικά κατά την αποθήκευση υλικών που εξαρτώνται από τη θερμοκρασία αλλά και σε εγκαταστάσεις διανομής φαρμάκων, ξενοδοχειακές μονάδες και γενικά σε ότι έχει να κάνει με τη διατήρηση και την παρακολούθηση της θερμοκρασίας. Χρησιμοποιήθηκε το πρωτόκολλο RS-485 για τη διασύνδεση των αισθητήρων θερμοκρασίας με το καταγραφικό των τιμών, και μέσω κώδικα αποθηκεύονταν οι τιμές των θερμοκρασιών. Έγινε κατασκευή κυκλωμάτων με τον επεξεργαστή Msp430 και γράφτηκε κώδικας σε C και Assembly.

Λέξεις κλειδιά : Μέτρηση, Καταγραφή, Θερμοκρασίες, Ψυγεία, Υλικά εξαρτώμενα από τη θερμοκρασία, Παρακολούθηση θερμοκρασίας, Πρωτόκολλο RS-485, Επεξεργαστής Msp430.

Abstract

In this thesis are proposed and developed the architecture of a system for measuring and recording temperatures of refrigerators and cold rooms. The proposed solution can be of vital importance in all medical facilities, especially when storing temperature-dependent materials but also in drug distribution facilities, hotel units and generally in everything related to maintaining and monitoring temperature. The RS-485 protocol was used to interface the temperature sensors with the value logger, and the temperature values were stored via code. Built circuits with the Msp430 processor and wrote code in C# and Assembly.

Keywords : Measuring, Recording, Temperatures, Refrigerators, Temperature-dependent materials, Temperature monitoring, RS-485 protocol, Msp430 processor.

Περιεχόμενα

1. Εισαγωγή - Σκοπός Εργασίας

Συστήματα μέτρησης και καταγραφής θερμοκρασίας χρησιμοποιούνται για την παρακολούθηση της θερμοκρασίας σε πολλούς ψυκτικούς θαλάμους. Τέτοιου είδους συστήματα είναι απαραίτητα για τη διατήρηση των προϊόντων σε σωστή θερμοκρασία και για τη διάγνωση προβλημάτων σε περίπτωση που η θερμοκρασία δεν είναι σωστή.

Υπάρχουν διάφορες λύσεις γι' αυτό το πρόβλημα, αλλά μια από τις πιο δημοφιλείς είναι η χρήση αισθητήρων θερμοκρασίας. Οι αισθητήρες αυτοί μπορούν να τοποθετηθούν σε διάφορες θέσεις στους ψυκτικούς θαλάμους και να μετρήσουν τη θερμοκρασία. Τέλος, τα δεδομένα μπορούν να καταγράφουν και να αποθηκεύουν σε μια βάση δεδομένων, ώστε να είναι δυνατή η παρακολούθησή τους

Σκοπός της εργασίας είναι η μέτρηση και θερμοκρασίας πολλών σημείων. Σήμερα υπάρχουν ευαίσθητα στην θερμοκρασία υλικά, όπως τρόφιμα και φάρμακα, τα οποία δεν πρέπει να καταναλωθούν αν η θερμοκρασία τους βγει εκτός ορίων για κάποιο χρονικό διάστημα. Ιδιαίτερο ενδιαφέρον παρουσιάζει η μέτρηση της θερμοκρασίας διατήρησης εμβολίων για παιδιά.
Στην συγκεκριμένη εργασία γίνεται σχεδίαση και η κατασκευή δικτύου αισθητήρων θερμοκρασίας με καταγραφή. Σε συνδυασμό τις νέες τεχνολογίες με οικονομικά και μικρά εξαρτήματα, ώστε να είναι ευέλικτη και εύκολη στην χρήση, άλλα και οικονομική στην κατασκευή.
Η ευκολία έγκειται στο γεγονός ότι ο χρήστης την χειρίζεται από υπολογιστή ο οποίος μέσω πρωτοκόλλου παίρνει από το καταγραφικό (modem) τις θερμοκρασίες. Πάνω στο καταγραφικό συνδέονται τερματικά (Terminal) modules, με 2 αισθητήρες θερμοκρασίας το κάθε ένα. Σε κάθε καταγραφικό μπορούν να συνδεθούν μέχρι 256 τερματικά (Terminal) modules.

2. Συστήματα Παρακολούθησης Θερμοκρασιών Ψυγείων

Όλα τα εργαστήρια και οι ιατρικές εγκαταστάσεις εγκαθιστούν ασφαλή αποθήκευση εμβολίων και φαρμάκων σε ψυχρή αλυσίδα για να αποτρέψουν την απώλεια της αποτελεσματικότητάς τους από λανθασμένα εύρη θερμοκρασίας. Από τη στιγμή που αυτά τα φάρμακα χάσουν την αποτελεσματικότητά τους, δεν υπάρχει τρόπος να αποκατασταθούν, με αποτέλεσμα τη σπατάλη δισεκατομμυρίων.

Για να διασφαλιστεί η αποτελεσματικότητα αυτών των ευαίσθητων στη θερμοκρασία φαρμάκων και εμβολίων, τα συστήματα παρακολούθησης της θερμοκρασίας του ψυγείου συμβάλλουν στη διατήρηση ενός κατάλληλου εύρους θερμοκρασίας. Χωρίς αυτά τα μέτρα, χάνετε σημαντικό χρηματικό ποσό σε υλικά.

Η σημασία των συστημάτων παρακολούθησης θερμοκρασίας ψυγείου

Η διατήρηση της διαδικασίας της ψυχρής αλυσίδας είναι ζωτικής σημασίας σε όλες τις ιατρικές εγκαταστάσεις, ειδικά κατά την αποθήκευση υλικών που εξαρτώνται από τη θερμοκρασία. Για να διατηρήσετε μια πρακτική και λειτουργική αλυσίδα ψύξης, πρέπει να έχετε ακριβή διαχείριση αποθεμάτων και καλά εκπαιδευμένο προσωπικό. Εκτός από αυτό, χρειάζεστε επίσης αξιόπιστη τεχνολογία παρακολούθησης θερμοκρασίας.

Εάν χρειάζεται να αποθηκεύσετε τα εμβόλια, πρέπει να τα διατηρήσετε στο σωστό εύρος θερμοκρασίας για να διασφαλίσετε την αποτελεσματικότητα και τη διάρκεια ζωής. Τα εμβόλια που βρίσκονται εκτός του αποδεκτού εύρους θερμοκρασίας δεν είναι αποτελεσματικά. Οι ασθενείς που λαμβάνουν δόσεις από αυτά τα εμβόλια θα χρειαστούν εκ νέου εμβολιασμοί.

Δεν υπάρχει τυπική αποδεκτή θερμοκρασία για όλα τα εμβόλια και άλλες βιολογικές ουσίες. Αυτό συμβαίνει επειδή τα συστατικά του φαρμάκου έχουν το δικό τους σύνολο συνθέσεων και απαιτήσεις αποθήκευσης.

Ορισμένες ουσίες πρέπει να καταψύχονται, ενώ άλλες μπορεί να απαιτούν συγκεκριμένες θερμοκρασίες. Για τα μέλη του ιατρικού προσωπικού, η αυστηρή συμμόρφωση μπορεί να είναι μια πρόκληση.

Λύσεις παρακολούθησης θερμοκρασίας ψυγείου

Επειδή η αυστηρή συμμόρφωση μπορεί να είναι πολύπλοκη για τους επαγγελματίες του ιατρικού τομέα, είναι διαθέσιμα συγκεκριμένα συστήματα παρακολούθησης της θερμοκρασίας του ψυγείου. Η παρακολούθηση εμβολίων και δειγμάτων με εξοπλισμό που μπορεί να καταγράφει δεδομένα, όπως μια συσκευή παρακολούθησης θερμοκρασίας, μπορεί να σας παρέχει πραγματικές μετρήσεις θερμοκρασίας.

Οι συσκευές παρακολούθησης θερμοκρασίας, γνωστές και ως TMD, μπορούν να καταγράφουν την ελάχιστη και μέγιστη θερμοκρασία των αντικειμένων μέσα στις μονάδες ψύξης. Εάν η θερμοκρασία στο εσωτερικό της μονάδας ψύξης φτάσει σε ένα ανεπιθύμητο εύρος, θα ειδοποιήσει αμέσως το ιατρικό προσωπικό ώστε να μπορέσει να διορθώσει την κατάσταση.

Το προσωπικό του γραφείου σας μπορεί να επικεντρωθεί περισσότερο στα βασικά του καθήκοντα με αυτά τα μέτρα. Εάν το φάρμακο χρειάζεται την άμεση προσοχή τους, οι συναγερμοί θα τους ενημερώσουν.

Ηλεκτρονικοί καταγραφείς θερμοκρασίας 30 ημερών

Αυτή η συσκευή τοποθετείται μαζί με τα αντικείμενα μέσα στη μονάδα ψύξης. Θα καταγράφει τη θερμοκρασία του ψυγείου τουλάχιστον μία φορά κάθε δέκα λεπτά ή λιγότερο για 30 συνεχόμενες ημέρες.

Ο ηλεκτρονικός καταγραφέας θερμοκρασίας 30 ημερών, γνωστός και ως 30 DTR (Digital Temperature Recorder), θα καταγράφει επίσης τριάντα ημέρες τυχόν διακοπές συναγερμού παγώματος ή υψηλής θερμοκρασίας που συνέβησαν. Αυτές οι συσκευές συνιστώνται ιδιαίτερα για ψυγεία εμβολίων αλλά όχι για καταψύκτες εμβολίων.

Ηλεκτρονικοί δείκτες παγώματος

Οι ηλεκτρονικοί δείκτες παγώματος είναι το αντίθετο των 30 DTR επειδή λειτουργούν καλύτερα σε καταψύκτες αντί για ψυγεία. Εάν δεν υπάρχουν διαθέσιμα 30 DTR, μπορείτε να χρησιμοποιήσετε αυτές τις ηλεκτρονικές ενδείξεις παγώματος. αλλά δεν συνιστάται.

Οι ηλεκτρονικοί δείκτες παγώματος είναι μικροί ψηφιακοί αισθητήρες θερμοκρασίας που δείχνουν εάν τα αντικείμενα στο ψυγείο έχουν εκτεθεί σε θερμοκρασίες παγώματος. Μόλις χτυπήσει το ξυπνητήρι, δεν μπορείτε να χρησιμοποιήσετε ξανά τη συσκευή. Θα χρειαστεί να τα πετάξετε αμέσως.

Ψηφιακό καταγραφικό δεδομένων

Τα ψηφιακά καταγραφικά δεδομένων, γνωστά και ως DDL (Digital Data Loggers), κοστίζουν περισσότερο από τα τυπικά θερμόμετρα, αλλά σας παρέχουν συνεχή παρακολούθηση και αποθήκευση δεδομένων. Σε αντίθεση με τα χειροκίνητα θερμόμετρα, τα ψηφιακά καταγραφικά δεδομένων διαθέτουν ανιχνευτές που μπορούν να μετρήσουν με ακρίβεια και άμεσα τη θερμοκρασία των εμβολίων ή άλλων ιατρικών ουσιών.

Μπορούν επίσης να εμφανίσουν δεδομένα για το χρόνο λειτουργίας μιας μονάδας ψύξης εκτός του προκαθορισμένου εύρους θερμοκρασίας. Για να διασφαλίσετε ότι το ψηφιακό καταγραφικό σας λειτουργεί όπως θα έπρεπε, θα έχει μια έγκυρη αναφορά βαθμονόμησης.

Βασικά Συστήματα Παρακολούθησης Θερμοκρασίας

Η παρακολούθηση της θερμοκρασίας των ιατρικών ουσιών σας ανεξάρτητα από την εσωτερική θερμοκρασία του καταψύκτη σας παρέχει έναν τρόπο να βεβαιωθείτε ότι η θερμοκρασία είναι πάντα εντός ενός συγκεκριμένου εύρους.
Υπάρχουν τρεις βασικές παράμετροι που πρέπει να έχει το ψυγείο σας για καλύτερα αποτελέσματα:

1. Συστήματα παρακολούθησης πίνακα συναγερμού καταψύκτη
2. Παρακολούθηση διακοπής ρεύματος
3. Ανεξάρτητη παρακολούθηση θερμοκρασίας

Οι περισσότεροι καταψύκτες και ψυγεία εμπορικής ποιότητας διαθέτουν συναγερμούς που θα ειδοποιούν το προσωπικό σας εάν η θερμοκρασία αλλάξει σημαντικά, αλλά τι συμβαίνει όταν δεν υπάρχει κανένας; Εάν ο καταψύκτης σας διαθέτει λειτουργία συναγερμού, υπάρχει μεγάλη πιθανότητα να διαθέτει και πίνακα συναγερμού. Όταν συνδέετε αυτόν τον πίνακα συναγερμού σε μια συσκευή απομακρυσμένης παρακολούθησης, το σύστημα θα ειδοποιήσει έναν χρήστη εάν ένας συναγερμός χτυπήσει μετά τις ώρες γραφείου.

Βασικές αιτίες της αύξησης της θερμοκρασίας

Η βασική αιτία της αύξησης της θερμοκρασίας στον καταψύκτη είναι οι διακοπές ρεύματος. Όταν εγκαταστήσετε τη σωστή τεχνολογία αισθητήρα θερμοκρασίας, θα παρακολουθεί για διακοπές ρεύματος και θα σας ειδοποιεί όταν χάνετε ρεύμα. Οι περισσότερες αξιόπιστες τεχνολογίες αισθητήρων θερμοκρασίας διαθέτουν ενσωματωμένες εφεδρικές μπαταρίες, παρέχοντάς σας 24ωρη παρακολούθηση.

Συναγερμοί συσκευών παρακολούθησης θερμοκρασίας

Δεν χρειάζεται πολύς χρόνος για να επηρεάσουν αρνητικά οι αλλαγές θερμοκρασίας τα ιατρικά προϊόντα και τα εμβόλια για το κρυολόγημα. Όπως αναφέρθηκε προηγουμένως, όταν η θερμοκρασία στη μονάδα ψύξης ή κατάψυξης αρχίσει να ξεφεύγει από το προ-προγραμματισμένο όριο, θα σβήσει ένας συναγερμός.

Ανάλογα με το σύστημά σας, μπορείτε να στείλετε αυτές τις πληροφορίες μέσω email, τηλεφωνικής κλήσης ή μηνύματος κειμένου. Έχετε την ευκαιρία να προγραμματίσετε την αποστολή του συναγερμού σε συγκεκριμένο προσωπικό ταυτόχρονα ή σε κλιμακούμενες βαθμίδες. Όταν κάποιος αναγνωρίσει τον συναγερμό και διορθώσει την κατάσταση, ένα αρχείο καταγραφής ελέγχου θα σας παρέχει αυτές τις πληροφορίες.

Πληροφορίες καταγραφής δεδομένων

Τα TMD (Temperature Monitoring Devices) μπορούν να καταγράφουν δεδομένα θερμοκρασίας σε καθορισμένα διαστήματα και μπορείτε να τα ρυθμίσετε ώστε να καταγράφει συχνότερα μετά από ένα ανησυχητικό περιστατικό. Η χρήση των ικανοτήτων καταγραφής δεδομένων αυτού του συστήματος μπορεί να αποδείξει τη συμμόρφωση της εταιρείας σας με τις ρυθμιστικές αρχές. Μπορείτε να ελέγξετε γραφήματα και άλλες πληροφορίες παρακολούθησης από απόσταση, εάν χρειάζεται.

Πώς λειτουργούν τα συστήματα απομακρυσμένης παρακολούθησης θερμοκρασίας ψυγείου;

Υπάρχουν πολλοί διαφορετικοί τύποι ασύρματων αισθητήρων θερμοκρασίας που μπορείτε να χρησιμοποιήσετε. Για παράδειγμα, υπάρχουν αισθητήρες θερμοκρασίας με αισθητήρες, αδιάβροχοι αισθητήρες ή αισθητήρες υψηλής ακρίβειας. Αυτοί οι ασύρματοι αισθητήρες έχουν τη δυνατότητα να διαβάζουν τις παραμέτρους θερμοκρασίας τουλάχιστον μία φορά κάθε πέντε λεπτά. Τα δεδομένα από αυτούς τους αισθητήρες παραδίδονται σε ένα ηλεκτρονικό σύστημα αναφοράς

μέσω μιας πύλης δεδομένων Διαδικτύου. Η πύλη δεδομένων Διαδικτύου διαθέτει εσωτερική μνήμη για δημιουργία αντιγράφων ασφαλείας για την αποφυγή διαρροής δεδομένων σε περίπτωση προβλημάτων σύνδεσης στο Διαδίκτυο.

Πώς να εγκαταστήσετε συστήματα παρακολούθησης θερμοκρασίας

Τα συστήματα παρακολούθησης θερμοκρασίας ψυγείου και καταψύκτη που βασίζονται σε cloud δεν είναι δύσκολο να εγκατασταθούν. Παρόλο που αυτά τα συστήματα εκτελούν πολύπλοκες λειτουργίες, είναι σχετικά εύκολο να εγκατασταθούν.

Απλά πρέπει να συνδέσετε έναν σε σύνδεση στο Διαδίκτυο και πολλούς άλλους αισθητήρες θερμοκρασίας στο σύστημα παρακολούθησης. Μόλις ρυθμιστεί όλος ο φυσικός εξοπλισμός του συστήματος, μπορείτε να προγραμματίσετε τις συσκευές σας μέσω ενός φιλικού προς τον χρήστη ιστότοπου.

Εάν αγοράζετε ασύρματες οθόνες και συναγερμούς, δεν θα χρειαστεί να αντιμετωπίσετε φυσικές ενσύρματες συνδέσεις. Αντίθετα, κάθε μονάδα θα χρησιμοποιεί μπαταρίες για να τις τροφοδοτεί. Οι μονάδες έχουν τη δυνατότητα απευθείας σύνδεσης στο διαδίκτυο.

Σταθεροποίηση θερμοκρασίας

Όταν εγκαθιστάτε ένα πρόσφατα επισκευασμένο ή ολοκαίνουργιο ιατρικό ψυγείο, συνήθως χρειάζονται περίπου δύο έως επτά ημέρες για να σταθεροποιηθεί η θερμοκρασία. Προτού αποθηκεύσετε οτιδήποτε σε αυτή τη νέα μονάδα, φροντίστε να ελέγξετε και να καταγράψετε τις ελάχιστες και μέγιστες θερμοκρασίες της μονάδας για περίπου δύο έως έξι ημέρες.

Εάν δεν μπορείτε να καταγράψετε τα δεδομένα ψηφιακά, μπορείτε να τα ελέγχετε χειροκίνητα τουλάχιστον δύο φορές την ημέρα. Μόλις η θερμοκρασία στο εσωτερικό της μονάδας μετρηθεί εντός του συνιστώμενου εύρους για δύο συνεχόμενα, η μονάδα είναι σταθερή και έτοιμη για χρήση.

Γιατί είναι απαραίτητες οι λύσεις παρακολούθησης της θερμοκρασίας του ψυγείου;
Ένας από τους μεγαλύτερους λόγους για τους οποίους είναι ζωτικής σημασίας η παρακολούθηση της θερμοκρασίας των ιατρικών ουσιών είναι για λόγους ασφαλείας. Τα ευαίσθητα στη θερμοκρασία φάρμακα μπορεί να γίνουν δηλητηριώδη ή αναποτελεσματικά εάν παραμείνουν εκτός της συνιστώμενης θερμοκρασίας τους για πάρα πολύ καιρό. Η παρακολούθηση της θερμοκρασίας ορισμένων ουσιών διασφαλίζει τη σωστή αποθήκευση και επιτρέπει στα μέλη του ιατρικού σας προσωπικού να απορρίπτουν υλικά που ενέχουν πιθανούς κινδύνους.

Δυνατότητα αναπαραγωγής

Τα εργαστήρια που κατασκευάζουν προϊόντα ή διεξάγουν έρευνα πρέπει να έχουν αναπαραγόμενα αποτελέσματα. Όλα τα προϊόντα πρέπει να είναι ομοιόμορφα και τυποποιημένα για να διασφαλίζεται η αποτελεσματικότητα.

Εάν το υλικό αφεθεί έξω σε περιβάλλον που μπορεί να βλάψει τα εξαρτήματά του και κανείς δεν το προσέξει, το ιατρικό προσωπικό μπορεί να καταγράψει ανακριβείς πληροφορίες. Καθώς αυτά τα ανακριβή δεδομένα μετακινούνται προς τα κάτω, τα δεδομένα που αναλύονται με αυτά τα αναποτελεσματικά φάρμακα θα βλάψουν την αποτελεσματικότητά τους.

Έρευνα Κόστους-Αποτελεσματικότητας

Τα συστήματα παρακολούθησης της θερμοκρασίας του ψυγείου είναι οικονομικά. Θα ειδοποιήσουν τα μέλη του προσωπικού σας για τυχόν διακυμάνσεις της θερμοκρασίας σε περιοχές όπου βρίσκονται οι ιατρικές σας ουσίες. Στη συνέχεια, μπορούν να επέμβουν και να προστατεύσουν αυτές τις ελεγχόμενες από το κλίμα ουσίες πριν η θερμοκρασία υπερβεί την ασφαλή ζώνη θερμοκρασίας.

Χωρίς αυτό το είδος παρέμβασης, πρέπει να απορρίψετε τυχόν υπερθερμασμένα αντικείμενα. Η αποτροπή αυτού του τύπου ζημιών μπορεί να εξοικονομήσει την εταιρεία σας εκατομμύρια, αν όχι δισεκατομμύρια. Με τα πρόσφατα λανσαρισμένα φαρμακευτικά προϊόντα ψυχρής αλυσίδας να κυκλοφορούν στην αγορά, η ανάγκη για κατάλληλα πρωτόκολλα ψυχρής αλυσίδας είναι πιο σημαντική από ποτέ. Υπολογίζεται ότι μέχρι το 2024, θα υπάρχουν προϊόντα ψυχρής αλυσίδας αξίας 341 δισεκατομμυρίων.

Είναι σημαντικό να φροντίζετε σωστά τις ιατρικές σας ουσίες σύμφωνα με τους κανόνες και τους κανονισμούς που ορίζονται από κρατικούς φορείς. Όταν χρησιμοποιείτε ένα αξιόπιστο και αξιόπιστο σύστημα παρακολούθησης θερμοκρασίας, μπορείτε να αποφύγετε τυχαία μη συμμόρφωση.

Το CDC, γνωστό και ως Κέντρα Ελέγχου Νοσημάτων, επιβάλλει εντολές για την πρόληψη του κινδύνου ή της εξάπλωσης τραυματισμού ή ασθένειας. Αυτός ο οργανισμός δημοσίευσε εργαστηριακά εγχειρίδια με διαδικασίες για τον τρόπο χειρισμού ορισμένων ασθενειών και φαρμάκων. Αυτά τα εγχειρίδια είναι εξαιρετικά χρήσιμα για εργαστήρια που εργάζονται με ευαίσθητα στη θερμοκρασία υλικά.

Οργανισμός τροφίμων και φαρμάκων (ΟΤΦ)

Σε διάφορες χώρες υπάρχει Οργανισμός Τροφίμων και Φαρμάκων που ρυθμίζει ιατρικές πρακτικές, ουσίες και συσκευές. Όλα τα εργαστήρια που χειρίζονται ιατρικό ή φαρμακευτικό εξοπλισμό πρέπει να ακολουθούν εγκεκριμένες από τον ΟΤΦ διαδικασίες και οδηγίες.
Η χρήση διαλυμάτων παρακολούθησης θερμοκρασίας ψυγείου είναι εγκεκριμένη και συνιστάται από τον ΟΤΦ. Όταν τα χρησιμοποιείτε στα εργαστήριά σας, συμμορφώνεστε και πληροίτε τα πρότυπα του ΟΤΦ.

Η Υπηρεσία Τροφίμων και Φαρμάκων ενημερώνει τακτικά τις Καλές Εργαστηριακές Πρακτικές της, συμπεριλαμβανομένων των κατάλληλων πρωτοκόλλων παρακολούθησης της θερμοκρασίας. Μπορείτε να χρησιμοποιήσετε αυτό το έγγραφο για να διευκρινίσετε τυχόν ανησυχίες σχετικά με τους εργαστηριακούς κανονισμούς του ΟΤΦ.

Βέλτιστες πρακτικές παρακολούθησης θερμοκρασίας

Για να διασφαλίσετε ότι το εργαστήριό σας αποκομίζει όλα τα οφέλη της τεχνολογίας παρακολούθησης θερμοκρασίας, υπάρχουν ορισμένες βέλτιστες πρακτικές που μπορείτε να ακολουθήσετε. Για παράδειγμα, πρέπει να βεβαιωθείτε ότι όλα τα όργανα σας είναι σωστά βαθμονομημένα. Με την πάροδο του χρόνου, αυτά τα εργαλεία μέτρησης χάνουν την ακρίβειά τους.

Τα εργαστήρια θα πρέπει να προγραμματίσουν τη βαθμονόμηση για να επαναφέρουν τα όργανα τους και να τα συντονίσουν στο πρότυπο ακρίβειας. Για να παραμείνετε συμβατοί και να εξοικονομήσετε χρόνο, μπορεί να θέλετε να αναθέσετε τις ανάγκες βαθμονόμησής σας σε έναν ειδικό. Αυτό θα επιτρέψει στο προσωπικό του εργαστηρίου να εστιάσει τον χρόνο του σε άλλες βασικές εργασίες.

Μετριασμός ορισμένων περιβαλλοντικών παραγόντων

Ορισμένοι περιβαλλοντικοί παράγοντες μπορούν να επηρεάσουν τη βαθμονόμηση των αντικειμένων σας. Η υποβάθμιση της στοιχειακής πίεσης και της ροής αέρα μπορεί να επηρεάσει την ακρίβεια του μετρικού σας εργαλείου.

Για να αποφευχθεί η στοιχειακή έκθεση ή οι αλλαγές στην πίεση, τα εργαστήρια θα πρέπει να λαμβάνουν μέτρα για να ελαχιστοποιούν αυτές τις αλλαγές. Για παράδειγμα, ένα εργαστήριο μπορεί να εφαρμόσει αυτοματοποιημένες πόρτες, να επιλέξει φλάντζες υψηλής πρόσφυσης, να τηρήσει ένα πρόγραμμα ρουτίνας καθαρισμού και να αποτρέψει την υπερβολική γέμιση ψυγείων.

Πρωτόκολλο RS-485

Το πρωτόκολλο RS-485 είναι ένα στάνταρ επικοινωνίας σε σειριακή μορφή που χρησιμοποιείται κυρίως για τη μετάδοση δεδομένων μεταξύ διαφορετικών συσκευών μέσω ενός κοινού δικτύου. Το RS-485 είναι μια βιομηχανική προδιαγραφή που ορίζει τα ηλεκτρικά χαρακτηριστικά, το φυσικό μέσο μετάδοσης και το πρωτόκολλο επικοινωνίας για συσκευές που επικοινωνούν μεταξύ τους μέσω σειριακής σύνδεσης.

Ορισμένα χαρακτηριστικά του πρωτοκόλλου RS-485 περιλαμβάνουν:

1. Ισορροπημένη γραμμή: Το RS-485 χρησιμοποιεί μια ισορροπημένη γραμμή, η οποία επιτρέπει τη μετάδοση δεδομένων σε μεγάλες αποστάσεις χωρίς σημαντική απώλεια σήματος.

2. Πολλαπλή σύνδεση: Μπορεί να συνδεθούν πολλές συσκευές στο ίδιο δίκτυο RS-485, με κάθε συσκευή να έχει μοναδική διεύθυνση για αναγνώριση.

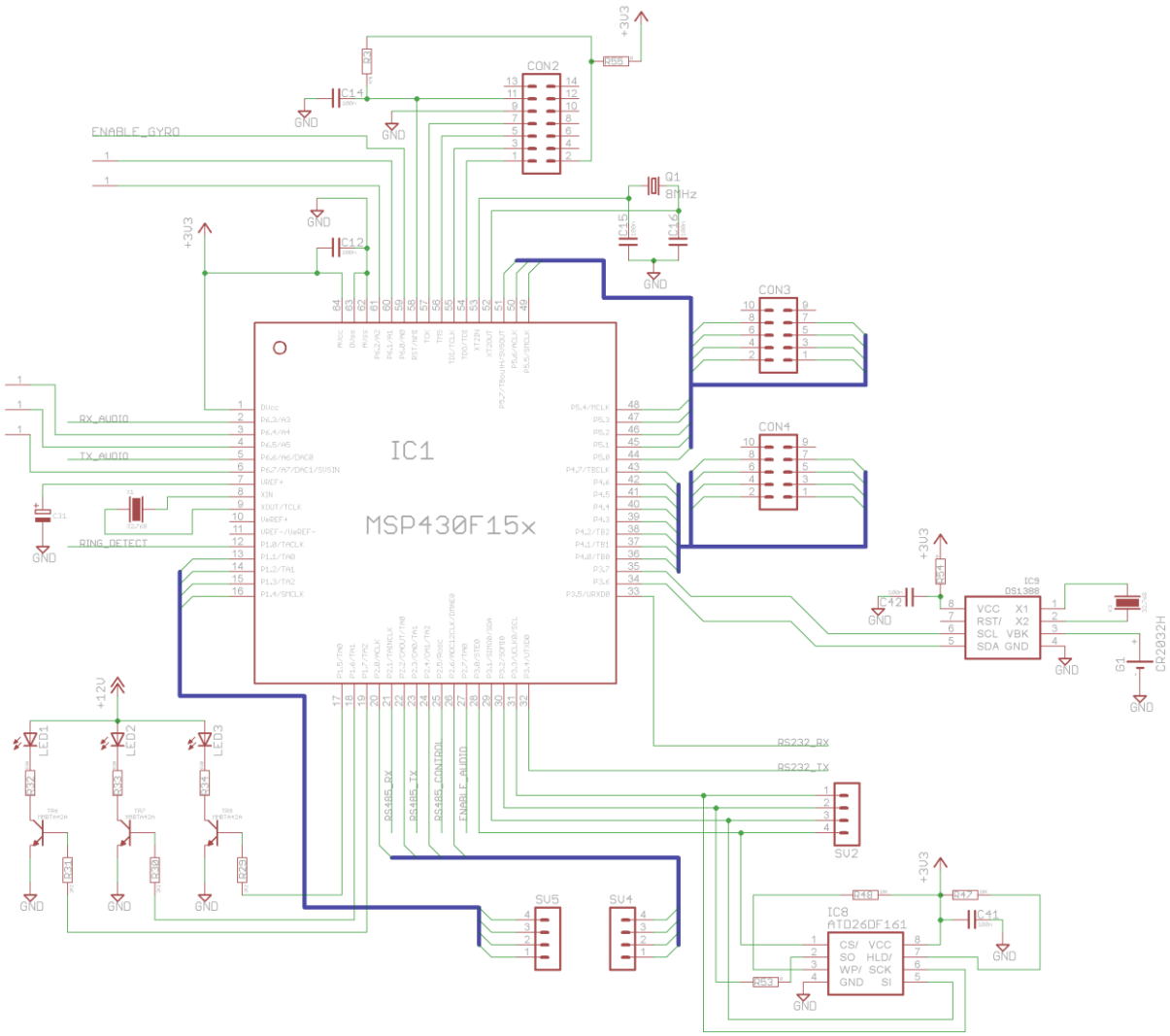
3. Πλήρης διπλή διεύθυνση: Το RS-485 υποστηρίζει πλήρη διπλή διεύθυνση επικοινωνίας, δηλαδή μπορεί να γίνει μετάδοση και λήψη δεδομένων ταυτόχρονα.

4. Ανθεκτικότητα σε θόρυβο: Είναι σχεδιασμένο για να αντέχει στον ηλεκτρομαγνητικό θόρυβο και τις παρεμβολές στο περιβάλλον βιομηχανικής εφαρμογής.

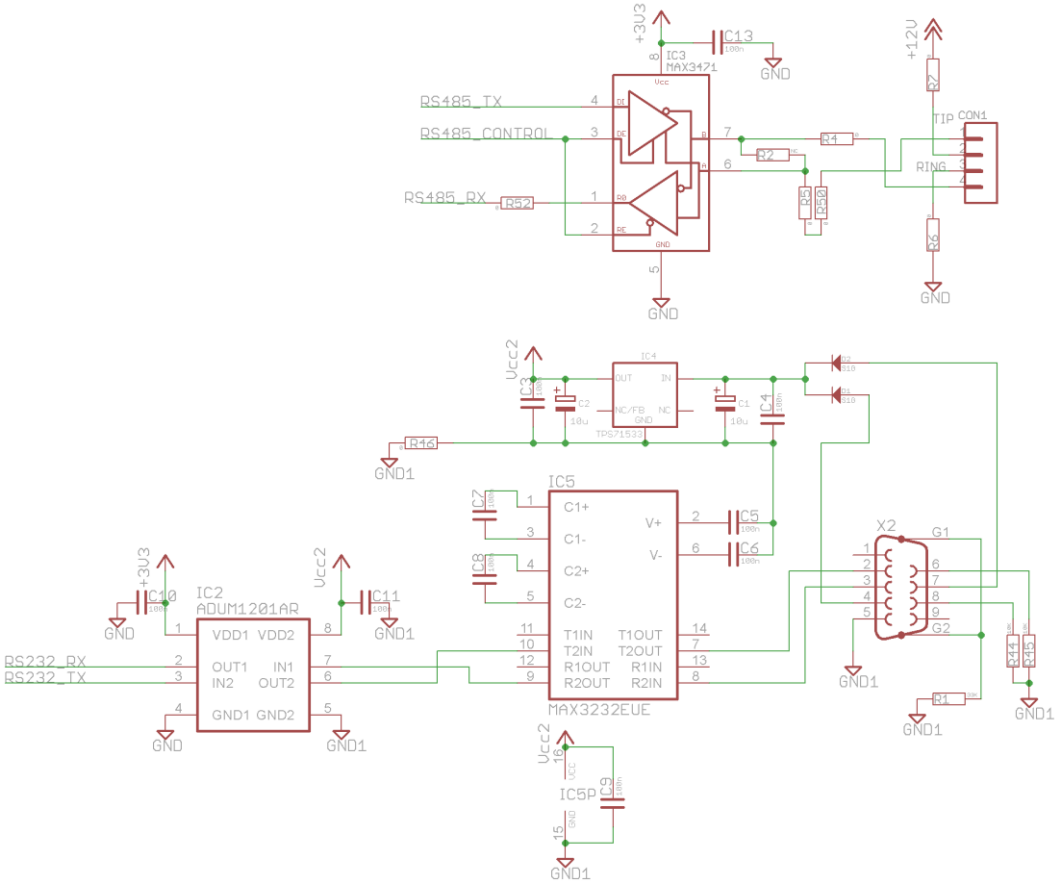
Το RS-485 είναι διαδεδομένο σε βιομηχανικές εφαρμογές όπως συστήματα ελέγχου και αυτοματισμού, συστήματα ασφαλείας, συστήματα επικοινωνίας μεταξύ συσκευών και πολλές άλλες εφαρμογές όπου απαιτείται αξιόπιστη μετάδοση δεδομένων σε μεγάλες αποστάσεις.

3. Σχεδίαση Καταγραφικού

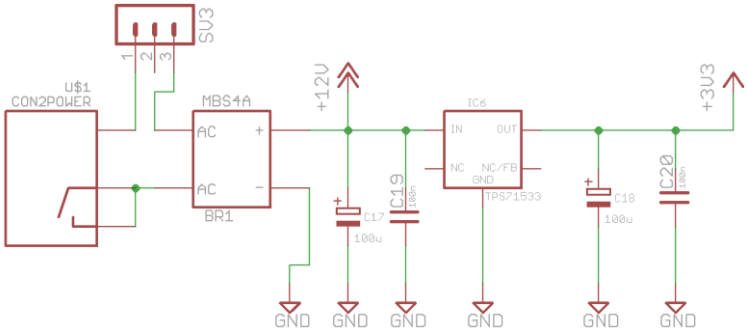
MCU



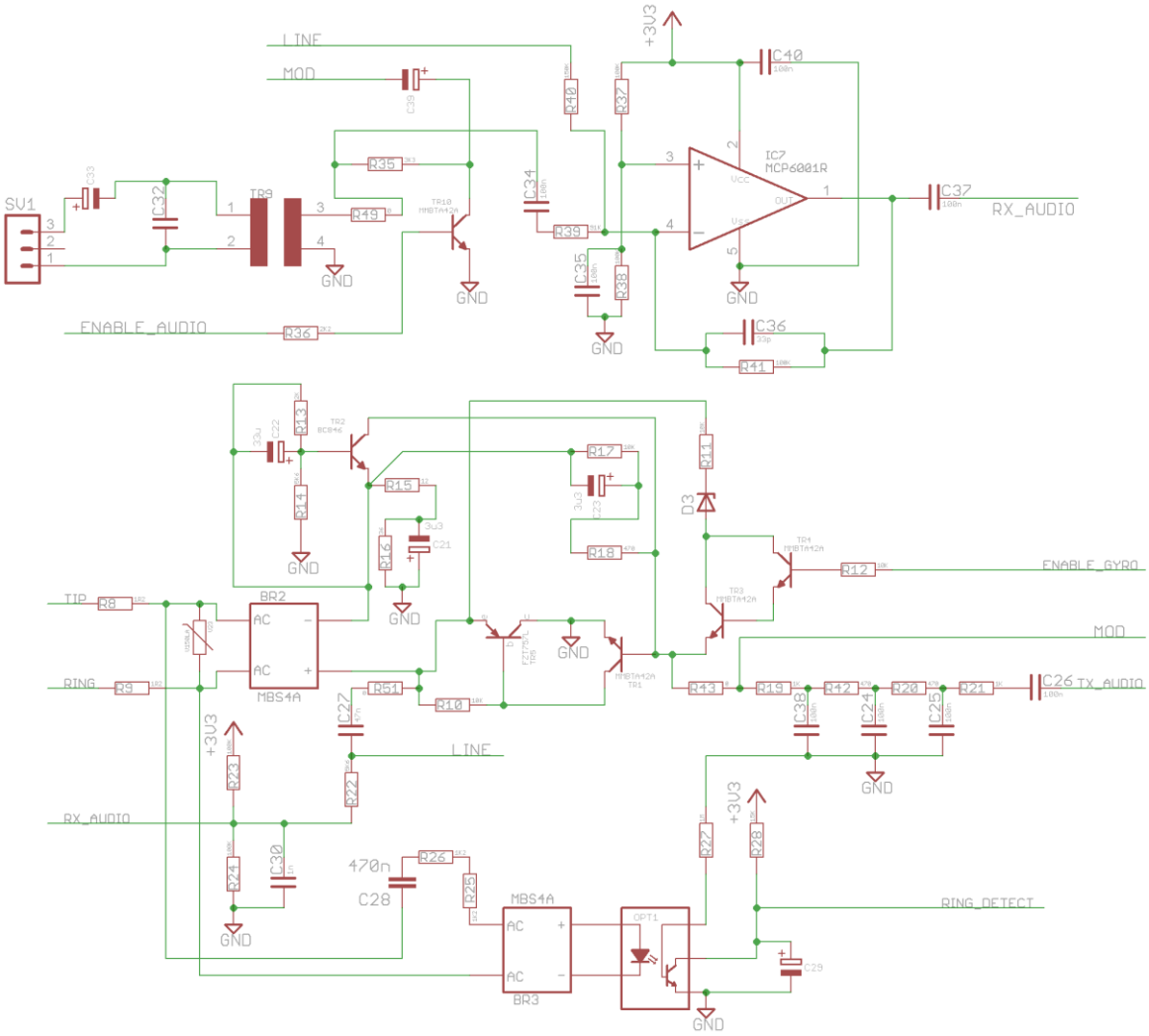
4. Interface – RS485 BUS για διασύνδεση αισθητήρων θερμοκρασίας



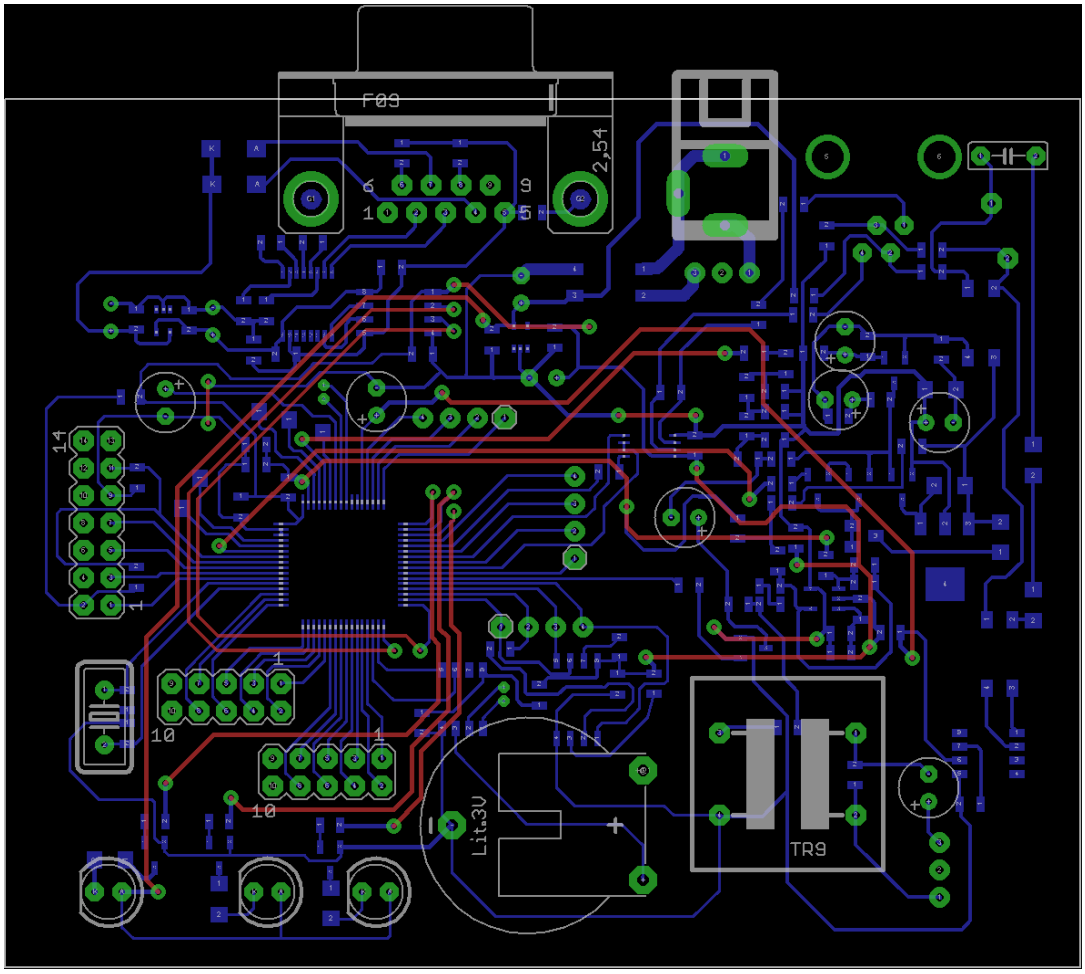
5. Τροφοδοτικό



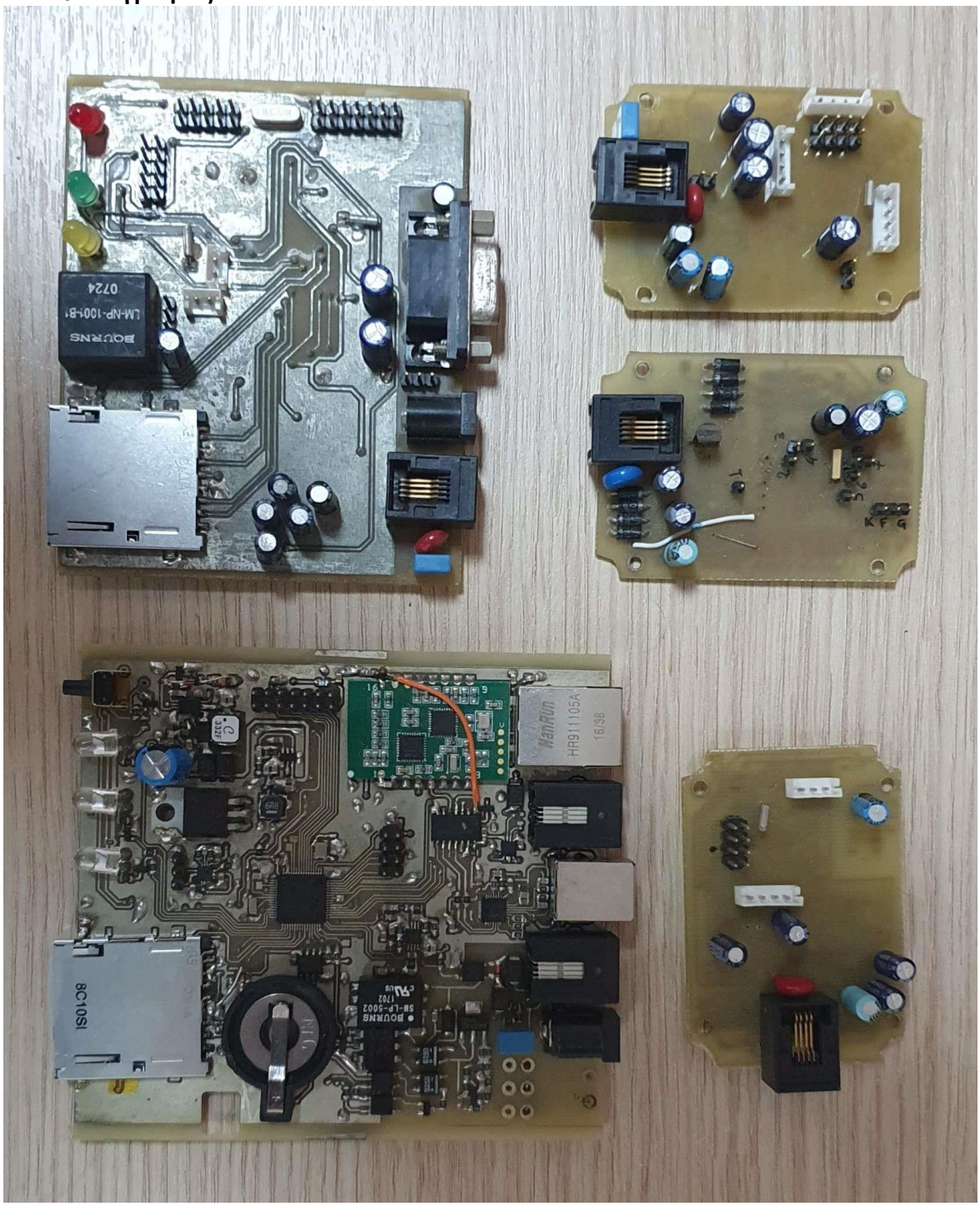
6. Interface Τηλεφωνικής Γραμμής

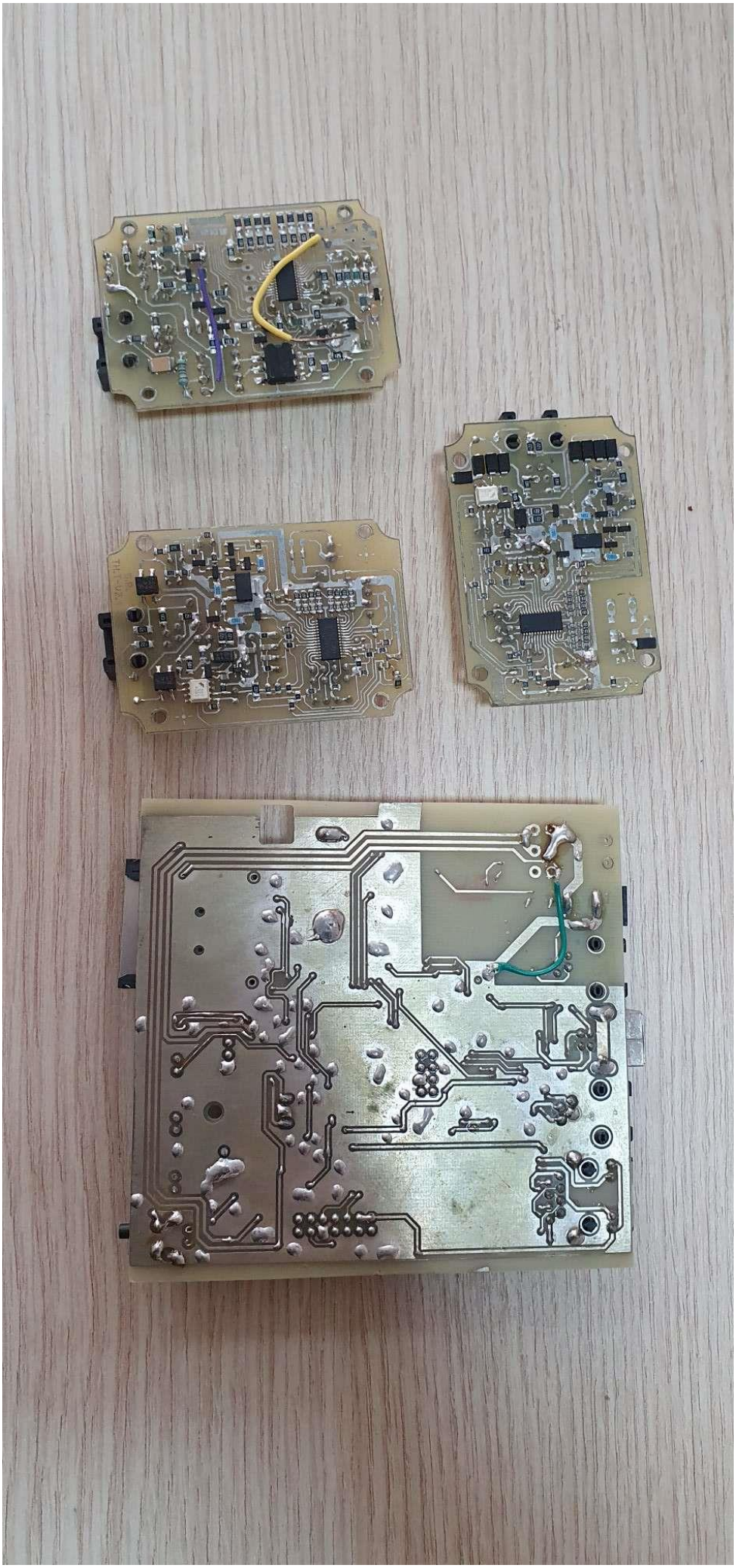


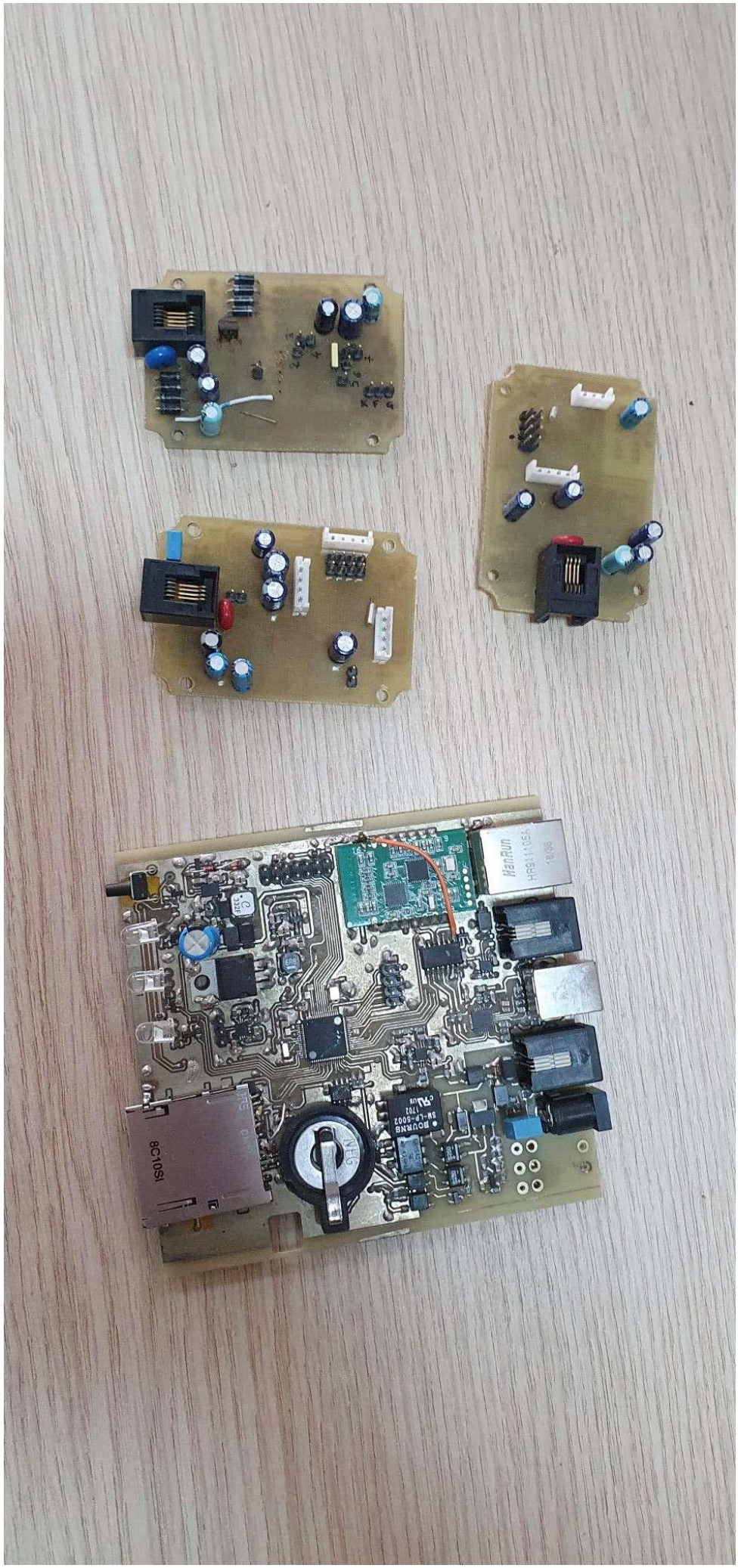
7. PCB

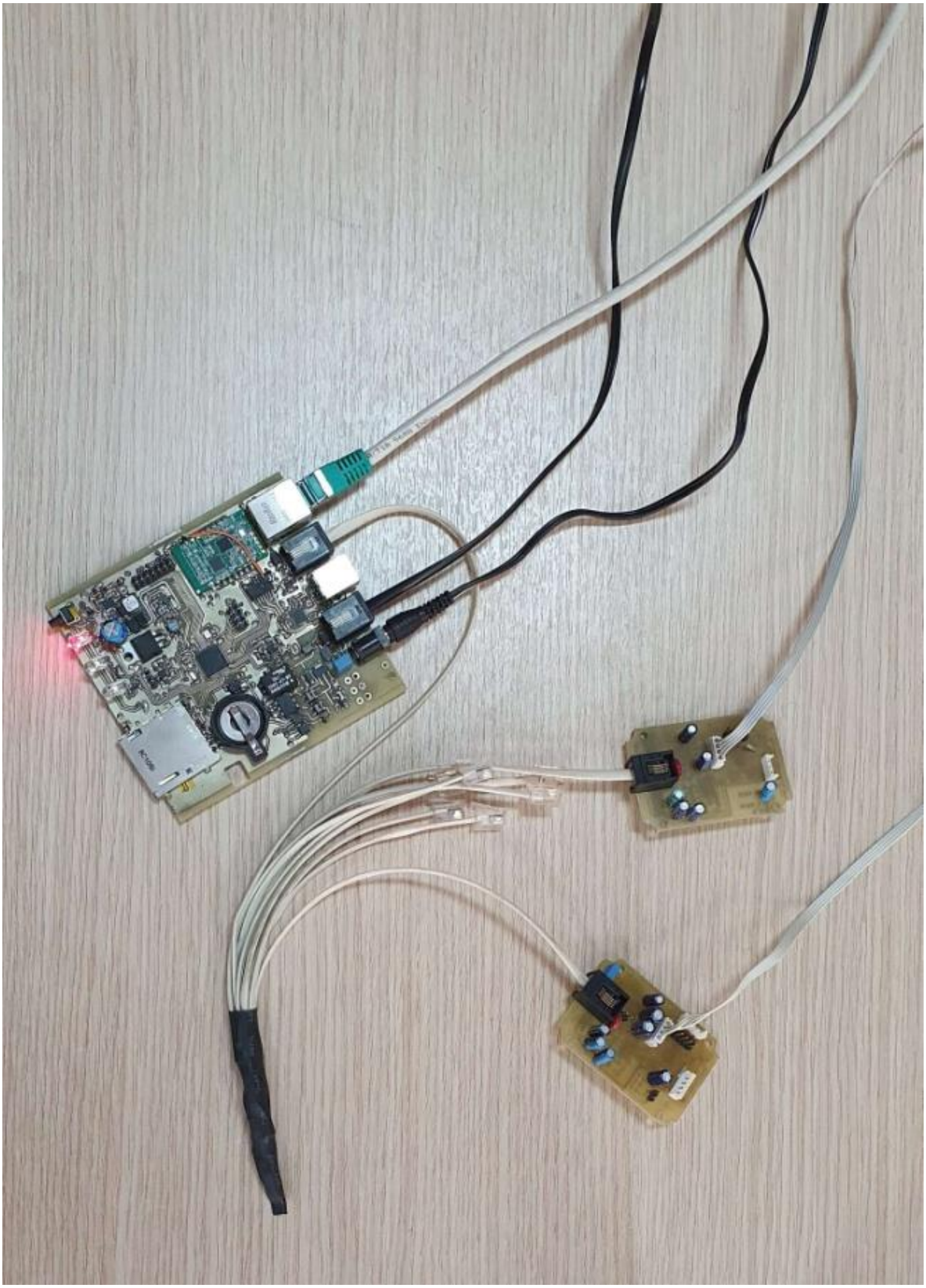


8. Φωτογραφίες






MeanRun
ROURNS
8C10SI



9. Πρωτόκολλο επικοινωνίας υπολογιστή με συσκευή καταγραφικού

Δομή των πακέτων εντολών

Έλεγχος ώρας

Η ώρα στη συσκευή είναι σε UTC
Σε κάθε επικοινωνία ρωτάει ο Υπολογιστής την ώρα
TC#@[CHECKsum](CRLF)

Το Καταγραφικό στέλνει την ώρα
T#Χρονολογία(4 χαρακτήρες)/μήνας(2 χαρακτήρες)/μέρα(2 χαρακτήρες)-ώρα(24ώρο 2 χαρακτήρες):λεπτά(2 χαρακτήρες):δευτερόλεπτα(2 χαρακτήρες)**@[CHECKsum](CRLF)**
T#2017**/**01**/**17**-**19**:**06**:**09**@[CHECKsum](CRLF)**

Όταν λάβει ο Υπολογιστής την ώρα στέλνει **V#OK[CHECKsum](CRLF)**
Ελέγχει και αν η διαφορά ώρας μας είναι μεγαλύτερη από 1 λεπτό , τότε ρωτάει το χρήστη αν θέλει να στείλει καινούργια

Την νέα ώρα την στέλνει με
NT#2017**/**01**/**17-19:06:09**@[CHECKsum](CRLF)**
Και περιμένει
V#OK[CHECKsum](CRLF)

Διαχείριση αρχείων

Για να πάρει ο Υπολογιστής την λίστα με τα αρχεία που βρίσκονται στην συσκευή του Καταγραφικό.
Dir#@[CHECKsum](CRLF)

Το Καταγραφικό αποκρίνεται με το παρακάτω
LF#ΌνομαΦακέλου1**$**αρχείο1**,**αρχείο2**,**.....**&**ΌνομαΦακέλου2**$**αρχείο1**,**αρχείο2**,**.....**&**...**@[CHECKsum](CRLF)**
Όταν λάβει την λίστα στέλνει **V#OK[CHECKsum](CRLF)**

Αποστολή Περιεχόμενου Αρχείου

Για να πάρει ένα αρχείο από τη συσκευή
FG#ΌνομαΦακέλου**$**Όνομααρχείου**@[CHECKsum](CRLF)**
Περιμένει
FZ#μέγεθος αρχείου **$**Όνομα Φακέλου**,** Όνομα αρχείου**@[CHECKsum](CRLF)**
Στέλνει
V#FOK@[CHECKsum](CRLF)
Ή
V#FFALSE@[CHECKsum](CRLF)
Παίρνει τα δεδομένα του αρχείου
FD#Δεδομένα αρχείου**@[CHECKsum](CRLF)**
Στέλνει
V#FDOK@[CHECKsum](CRLF)

Για να πάρει το τελευταίο αρχείο
FL#@[CHECKsum](CRLF)
Περιμένει
FZ#μέγεθος αρχείου **$** Όνομα Φακέλου**,** Όνομα αρχείου**@[CHECKsum](CRLF)**
Στέλνει
V#FOK@[CHECKsum](CRLF)
Παίρνει τα δεδομένα του αρχείου
FD#Δεδομένα αρχείου**@[CHECKsum](CRLF)**
Η Εντολή FD φτιάχνεται όλη μαζί , με όλα τα δεδομένα και το checksum στο τέλος απλός δεν στέλνονται όλα μαζί στο buffer αλλά ανά 1000
Στέλνει
V#FDOK@[CHECKsum](CRLF)

Δεδομένα Αρχείου

Module ID, Password, Chamber 1 Name, Chamber 1 Condition, Chamber 1 Temperature, Electricity 1 status (0:Electricity OFF, 1: Electricity ON), Chamber 1 Possible Reason, Chamber 2 Name, Chamber 2 Condition, Chamber 2 Temperature, Electricity 2 status (0:Electricity OFF, 1: Electricity ON), Chamber 2 Possible Reason, Date, Time**(CRLF) …….@[CHECKsum](CRLF)**

Αποστολή Ρυθμίσεων

Για να στείλει ρυθμίσεις αρχείο Settings.txt στέλνει
SFS#μεγεθός αρχείου**@[CHECKsum](CRLF)**
Περιμένει
V#SFSOK@[CHECKsum](CRLF)
Στέλνει
SSP#ρυθμίσεις**@[CHECKsum](CRLF)**
Περιμένει
V#SPOK@[CHECKsum](CRLF)
Περιμένει
V#SOK@[CHECKsum](CRLF)

Ρυθμίσεις αρχείου Settings.txt

I, Module ID, UpdateTime, numberofChambers, FirstChamberID, FirstUsed (0:notused,1:Temperature,2: Humidity), FirstLowLimit, FirstHighLimit, FirstIntegralNewLimit,FirstIntegralCriticalLimit,FirstIntegralBadLimit, SecondChamberID, SecondUsed (0:not used,1:Temperature,2: Humidity), SecondLowLimit, SecondHighLimit, SecondIntegralNewLimit,SecondIntegralCriticalLimit,SecondIntegralBadLimit***(CRLF).....@[CHECKsum](CRLF)**

Για να στείλει ρυθμίσεις αρχείο Telsettings.txt στέλνει
STS#ρυθμίσεις**@[CHECKsum](CRLF)**
Περιμένει
V#SOK@[CHECKsum](CRLF)
Ρυθμίσεις αρχείου Telsettings.txt
T1,Telephon**(CRLF)**
PR,Prefix**(CRLF)**
TR,(Call Retries 1-9)**(CRLF)**

AN, (0: Wait confirm from 1 telephone, 1: Wait confirm from all Telephones)**@[CHECKsum](CRLF)**

Checksum

Αλγόριθμος

1. Μετατρέπει το string σε byte
2. Προσθέτει τα byte
3. Κάνει λογική πράξη AND του αθροίσματος με το FF
4. Επιστρέφει σε δεκαεξαδικό τα 2 τελευταία byte

```
private string CalculateChecksum(string dataToCalculate) {

byte[] byteToCalculate = Encoding.ASCII.GetBytes(dataToCalculate);

int checksum = 0;

foreach (byte chData in byteToCalculate) {

 checksum += chData; }

checksum &= 0xff;

return checksum.ToString("X2"); }
```

Αναγνωριστικό USB συσκευής
Διαδρομή εμφάνισης συσκευής
USB\VID_10C4&PID_EA60\2675

ElectricityString
0 "Electricity OFF"
1 "Electricity ON”

Condition
"1" "Processing”
"2" "New”
"5" "Critical"
"4" "Bad"
"3" "Fixed”
"6" "Periodical "

ΚΩΔΙΚΑΣ

ΚΑΤΑΓΡΑΦΙΚΟ ΚΩΔΙΚΑΣ C DEFINITIONS

```
#define     mSecondsFL    0x0001
#define     new_char      0x0002
#define     accept_char   0x0004
#define     RxTimeFL      0x0008
#define     TxEnable      0x0010
#define     RxEnable      0x0020
#define     PointDegreeA  0x0040
#define     VoiceMode     0x0080
#define     WordFinishedFL  0x0100
#define     serialRequest  0x0200
#define     receivingSetting  0x0400



#define     alarmBufferSize  50


#define     RS485ControlH()  P4OUT |= BIT2;
#define     RS485ControlL()  P4OUT &= ~BIT2;
#define     RxLedON()     P1OUT |= BIT6;
#define     RxLedOFF()    P1OUT &=~BIT6;
#define     PowerLedON()P1OUT |= BIT7;
#define     PowerLedOFF()  P1OUT &=~BIT7;
#define     TxLedON()     P1OUT |= BIT5;
#define     TxLedOFF()    P1OUT &=~BIT5;

#define     BSCL   BIT0;
#define     BSDA   BIT1;

// DTMF Filter Variables //
#define     THRE   400
#define     LENGTH    8

//Voice Messages
#define     Ena                  1
#define     OneCelsiusDegree          181
#define     CelsiousDegrees           149
#define     chamberTemperature        100
#define     Einai                101
#define     VWelcome                  102
#define     Negative                  106
#define     HighWindowTemp            107
#define     YouHaveAlarmSignalFromChamber  108
#define     VEndOfConnection          110
#define     Diesi                111
```

```
#define VPliktrologisate 112
#define VEnterInstruction 113
#define VUnprogrammed 114
#define VChamberLowLimit 115
#define Open 116
#define VChamberHighLimit 117
#define Closed 118
#define Vphone 119
#define Door 120
#define AndFromChamber 121
#define VWrongPassword 122
#define VCentralPrefix 123
#define VTemperatureWindow 96
#define VRingCounts 124
#define LowWindowTemp 125
#define VTime 126
#define VConfirmationDiesis 127
#define Asteraki 128
#define Tria 3
#define Tessera 4
#define PointFive 129
#define Stored 130
#define FromFridgeTalkerDev 131
#define InstrError 132
#define Minuits 133
#define AndFrom 121
#define YouHaveAlarmMessageFrom 108
#define Chamber 148
#define YouHaveAlarmSignalFromDoor 147
#define AndFromDoor 146
#define newAlarm 182
#define criticalAlarm 183
#define badAlarm 184
#define fixedAlarm 185
#define signalCondition 186
```

ΚΩΔΙΚΑΣ

2. ΚΑΤΑΓΡΑΦΙΚΟ ΚΩΔΙΚΑΣ C

```
#include "ff.h"
#include "ff.c"
#include "msp430.h"

#include "stdio.h"
#include "string.h"
#include "math.h"
#include "mmcbb.c"
#include "zisis.h"
#include <stdbool.h>
#include <stdlib.h>
//#include "Filter.s43"
//#############################################################################
// settings.txt format:
//#UID,FirstChamberID,UpdateTime,FirstUsed,FirstLowLimit,FirstHighLimit,FirstNewLimit,FirstCritical
Limit,FirstBadLimit,SecondChamberID,SecondUsed,SecondLowLimit,SecondHighLimit,SecondNewLi
mit,SecondCriticalLimit,SecondBadLimit
//I,28001,10,2,1,1,-20,20,200,300,400,2,1,-20,20,200,300,400*
//I,28008,10,2,1,1,-20,20,200,300,400,2,1,-20,20,200,300,400*
//I,28009,10,2,1,1,-20,20,200,300,400,2,1,-20,20,200,300,400*
//I,28010,10,2,1,1,-20,20,200,300,400,2,1,-20,20,200,300,400*
//I,28011,10,2,1,1,-20,20,200,300,400,2,1,-20,20,200,300,400*
//I,28012,10,2,1,1,-20,20,200,300,400,2,1,-20,20,200,300,400*
//#############################################################################

// This version Sends settings and store incoming signals in "FAM.alm"


FATFS Fatfs;            /* File system object */
FIL Fil;                /* File object */
FRESULT rc;             /* Result code */
FRESULT res;            /* Result code */
DIR dir;
FILINFO fno;

FIL ReadFile;
char Lfname[_MAX_LFN+1];
unsigned int usBytesWrite;
unsigned int usBytesRead;
// ************  DTMF filter Variables  ************************************
unsigned int    COUNTb;
unsigned int    ROW_CMP;
unsigned int    COL_CMP;
unsigned char input_char;
unsigned int    Seconds_hook;
```

```
int             T2_1633;
int             T1_1633;
int             T2_1477;
int             T1_1477;
int             T2_1336;
int             T1_1336;
int             T2_1209;
int             T1_1209;
int             T2_941;
int             T1_941;
int             T2_852;
int             T1_852;
int             T2_770;
int             T1_770;
int             T2_697;
int             T1_697;
unsigned int    MyFlags;
unsigned int    msTimer;
unsigned int    LCOUNTb;
int             inb;
int             MAXLOb;
int             MAXHIb;
unsigned int    ROWb;
unsigned int    COLb;
int             OUTPUTb;

unsigned int    X_TMP;
unsigned int    X_LUTAEXT;
unsigned int    X_SWA;
unsigned int    X_LUTBEXT;
unsigned int    X_SWB;
unsigned int    OCRA;
unsigned int    OCRB;
unsigned int    TEMP;
unsigned char   sendDigit;
unsigned int    SpeechBytesCounter;
unsigned int    StartAddress;
unsigned int    DelayCount;
unsigned int    speechIndex;

unsigned int    VdataInL;
unsigned int    VdataInH;
unsigned int    VdataOutL;
unsigned int    VdataOutH;
unsigned int    Vcounter;

unsigned long int       fileSize;

unsigned char nextCallTime = 1;
```

```
unsigned char numberOfRetries = 3;
bool          tel1Programmed = true;
bool          tel2Programmed = false;
bool          tel3Programmed = false;
bool          getConfirmFromAll = false;
unsigned char AlarmID = 1;
int           Year;
unsigned char Month;

unsigned int  serialInChecksum = 0;
unsigned int  fileChecksum = 0;

unsigned int  R4B,R5B,R6B,R7B,R8B,R9B,R10B,R11B,R12B,R13B,R14B,R15B;

#define       enableGyro()  P6OUT |= BIT0;
#define       disableGyro()  P6OUT &= ~BIT0;

void enableTransmitter(void);
void disableTransmitter(void);
void enableReceiver(void);
void disableReceiver(void);

unsigned char SPI_TX_BYTE,SPI_RX_BYTE,SPI_Counter,SPI_Help;
unsigned char checksum,inchar;
unsigned int mSeconds,TimemS,MyFlags,ptr;
unsigned char NumberOfModules=0;
extern void Send_SPI(void);
extern void Get_SPI(void);
extern void Clocks_Init(void);
extern void DTMF_FILTER(void);
extern void SendAudio(void);
extern void SendDtmf(void);
extern void Speech(void);
extern void Get5sChar(void);
char makeCall( char *phoneNumber, char *prefix);


void TimerA0Init (void);
void TimerB0Init(void);
void SendByte485 (char ByteToSend);
void UARTA1init(void);
void StartCondition(void);
void StopCondition(void);
void I2Cdelay(void);
void WaitAck(void);
void SendI2C(char Sdata);
char GetI2C(void);
void GetRTC(void);
void StoreRTC(void);
```

```
void getPhoneSettings(void);

const unsigned char conditionTable[8] = {1,1,182,185,184,183,1,1};
unsigned char alarmPointer=0;
bool minFL = false;

char I2CCounter,I2COutData,I2CInData;
unsigned int Seconds,Minuets, TimeToUpdate=0;
const char Months[36] = "JanFebMarAprMayJunJulAugSepOctNovDec";
const char asciiTab[]="0123456789ABCDEF";

unsigned char proccess=0;


char GetRS485Byte(void);
char GetRS485TimedByte(void);
unsigned char HelpBuffer[24];
char TempBuffer[512];
int  TempPointer=0;
char fileToSend[16];
char ClockBuffer[24];
char RxBuffer[150];
char IDbuffer[512];
int  IDpointer=0;
char lineBuffer[256];
char fieldBuffer[128];
char currentID[8];

char Telephone1[16]="";
char Telephone2[16]="";
char Telephone3[16]="";
char Prefix[8] = "";
char serialInBuffer[1024];
int  serialInPointer=0;
char serialOutBuffer[1100];
char serialRequests=0xfe;


void SendRS485String(char *RS485Buffer);
void SendConfirmToModule(unsigned char *RS485Buffer);
void SendRS485Binary(char *RS485Buffer, int len);
char RS485GetString(void);
unsigned char sendSettings(char *buf);
void getField(char *buff, int field);
void decodeIncomingSignal(char *buf);
void callHandler(void);
unsigned char shortAlarms(void);
bool speechAlarms(unsigned char tel);
void SpeechTemperature (float temp);
```

```
float getLineVoltage(void);
char storeToFile(void);
void serialInit(void);
void sendSerialString(void);
void sendSerialTime(void);
void sendFileTreeToPC(void);
char * getDirectoryFromBuffer(void);
void sendFileSizeToPC(void);
void sendSerialFilePart(void);
void changeTime(void);
char countFields(char *buf);

typedef struct {
  unsigned char alarmID;
  unsigned char chamberName;
  unsigned char Condition;
  float Temperature;
  unsigned char Reason;
  unsigned int Retries;

  unsigned char door;
  unsigned char nextCallTimer_1;
  unsigned char nextCallTimer_2;
  unsigned char nextCallTimer_3;
} Alarm;
Alarm alarmBuffer[alarmBufferSize];

enum serialRequests{
  sendTime,
  sendTimeConfirmation,
  sendFileTree,
  sendFileSize,
  sendFile,
  sendFileParts,
  settingsReceived,
  telSettingsReceived
};
enum Conditions {
  Processing=1,
  New=2,
  Fixed=3,
  Bad=4,
  Critical=5,
  Periodical=6
};
//*****************************************************************************
//*************************** Main Program  ***********************************
//*****************************************************************************
int main( void )
```

```
{
  WDTCTL = WDTPW + WDTHOLD;
  P1DIR &= ~(BIT3+BIT4);
  P1DIR |= BIT5 + BIT6 + BIT7;
  P1OUT &= ~(BIT5+BIT6+BIT7);
  P4DIR |=BIT2;
  P6DIR |= BIT0;
  P6OUT &= ~BIT0;


  alarmPointer = shortAlarms();
  P4OUT |= BIT7;
  P4DIR |= BIT7; /
  P4DIR |= BIT0+BIT1;
  P4OUT |= BIT0+BIT1;

  P6DIR &= ~BIT2;

  P5OUT = 0x00;
  P5DIR = 0xff;
  P5OUT |= (BIT1 + BIT3 + BIT2 + BIT6 + BIT7);
  P5DIR &= ~BIT0;
  MyFlags=0;

  Clocks_Init();
  TimerA0Init();
  UARTA1init();
  serialInit();
  __bis_SR_register(GIE);

        PowerLedON();
          TimerB0Init();
          __delay_cycles(1000000*16);
          f_mount(&Fatfs,"",0); // Mount SD Card
        StoreRTC();



          PowerLedOFF();
          IFG1=0;
          IFG2=0;
          P1IFG=0;
          P2IFG=0;
           __delay_cycles(16000000);



           GetRTC();
```

```
 __bis_SR_register(GIE);       // Interrupts enabled
   __delay_cycles(16000000);
   PowerLedON();
Minuets=150;
   TimeToUpdate=1;

   for (int i=0; i < alarmBufferSize; i++)
   {
    alarmBuffer[i].nextCallTimer_1 = 0xff;
    alarmBuffer[i].nextCallTimer_2 = 0xff;
    alarmBuffer[i].nextCallTimer_3 = 0xff;
   }

   getPhoneSettings();

 f_open (&ReadFile,"settings.txt", FA_READ|FA_WRITE);
   int ptr=0;
   while (f_gets(lineBuffer, sizeof lineBuffer, &ReadFile))
   {
    if (lineBuffer[0] == 'i')
    {
     IDbuffer[ptr++] = lineBuffer[2];
     IDbuffer[ptr++] = lineBuffer[3];
     IDbuffer[ptr++] = lineBuffer[4];
     IDbuffer[ptr++] = lineBuffer[5];
     IDbuffer[ptr++] = lineBuffer[6];
     NumberOfModules++;
    }
    else if (lineBuffer[0] == 'I')
    {
     IDbuffer[ptr++] = lineBuffer[2];
     IDbuffer[ptr++] = lineBuffer[3];
     IDbuffer[ptr++] = lineBuffer[4];
     IDbuffer[ptr++] = lineBuffer[5];
     IDbuffer[ptr++] = lineBuffer[6];
     currentID[0] = lineBuffer[2];
     currentID[1] = lineBuffer[3];
     currentID[2] = lineBuffer[4];
     currentID[3] = lineBuffer[5];
     currentID[4] = lineBuffer[6];
     currentID[5] =0;
     NumberOfModules++;

    }
    else if (lineBuffer[0] == 'T')
    {
     TimeToUpdate=atoi(lineBuffer+1);
    }
    else if (lineBuffer[0] == '#')
```

```
          {
          }
          else
          {
          }
        }
        f_close(&ReadFile);
//*****************************************************************************
//*************************** main Loop ***************************************
//*****************************************************************************
        while(1)
        {
          for(int i=0;i<5;i++) currentID[i] = IDbuffer[IDpointer++];
          if (currentID[0] != 0)
          {
            strcpy(TempBuffer,"*\x67,");
            strcat(TempBuffer,currentID);
            strcat(TempBuffer,"#\0");
            TxLedON();
            __delay_cycles(80000);
            SendRS485String(TempBuffer);
            RS485ControlL();
            TxLedOFF();
            if (RS485GetString()==0)
            {
              RxLedON();
              SendConfirmToModule("*\x29,#\0");
              decodeIncomingSignal(RxBuffer);

              if (storeToFile() == 0)
              {
                RxLedOFF();
                TxLedOFF();
              }
              else
              {
                TxLedON();
              }
            }
            if (minFL == true)
            {
              callHandler();
              minFL = false;
            }
          }
          else
          {
            IDpointer = 0;
          }
```

```
if ((MyFlags & serialRequest) !=0)
{
 if (serialRequests == sendTime)
 {
  sendSerialTime();
  MyFlags &= ~serialRequest;
  serialRequests=0;
 }
 if (serialRequests == sendTimeConfirmation)
 {
  changeTime();
  strcpy(serialOutBuffer,"V#OK");
  sendSerialString();
  MyFlags &= ~serialRequest;
  serialRequests=0;
 }
 if (serialRequests == sendFileTree)
 {
  sendFileTreeToPC();
  MyFlags &= ~serialRequest;
  serialRequests=0;
 }
 if (serialRequests == sendFileSize)
 {
  sendFileSizeToPC();
  MyFlags &= ~serialRequest;
  serialRequests=0;
 }
 if (serialRequests == sendFile)
 {
  f_open(&Fil, fileToSend, FA_READ);
       while (!(IFG2&UCA0TXIFG));
       UCA0TXBUF = 'F';
       while (!(IFG2&UCA0TXIFG));
       UCA0TXBUF = 'D';
       while (!(IFG2&UCA0TXIFG));
       UCA0TXBUF = '#';
       while (!(IFG2&UCA0TXIFG));
       fileChecksum = 'F' + 'D' + '#';
  while (fileSize > 997)
  {
f_read(&Fil,serialOutBuffer,997,&usBytesRead);
    serialOutBuffer[usBytesRead]=0;
    fileSize -= 997;
    serialRequests=0;
    sendSerialFilePart();

  }
```

```
  if (fileSize != 0)
  {
   f_read(&Fil,serialOutBuffer,fileSize,&usBytesRead);
   serialOutBuffer[usBytesRead]=0;
   serialRequests=0;
   sendSerialFilePart();


  }
  char chL,chH;
  UCA0TXBUF = '@';
       while (!(IFG2&UCA0TXIFG));
       chL = asciiTab[fileChecksum & 0x000f];
       chH = asciiTab[(fileChecksum & 0x00f0) >> 4];
       UCA0TXBUF = chH;
       while (!(IFG2&UCA0TXIFG));
       UCA0TXBUF = chL;
       while (!(IFG2&UCA0TXIFG));
        UCA0TXBUF = 0x0d;
        while (!(IFG2&UCA0TXIFG));
        UCA0TXBUF = 0x0a;
        while (!(IFG2&UCA0TXIFG));

  MyFlags &= ~serialRequest;
  f_close(&Fil);
}
if (serialRequests == settingsReceived)
{
  f_open(&Fil, "settings.txt", FA_CREATE_ALWAYS | FA_WRITE);
  int i=0;
  while(serialInBuffer[i] != '@')
  {
   serialOutBuffer[i] = serialInBuffer[i+3];
   i++;
  }
  serialOutBuffer[i]=0;
  f_write(&Fil,serialOutBuffer,strlen(serialOutBuffer),&usBytesWrite);
  f_close(&Fil);
  strcpy(serialOutBuffer,"V#SOK");
  sendSerialString();
  MyFlags &= ~serialRequest;
  serialRequests = 0;
}
if (serialRequests == telSettingsReceived)
{
f_open(&Fil, "telSettings.txt", FA_CREATE_ALWAYS | FA_WRITE);
  int i=0;
  while(serialInBuffer[i] != '@')
  {
```

```
                serialOutBuffer[i] = serialInBuffer[i+4];
                i++;
              }
              serialOutBuffer[i]=0;
              f_write(&Fil,serialOutBuffer,strlen(serialOutBuffer),&usBytesWrite);
              f_close(&Fil);
              strcpy(serialOutBuffer,"V#SOK");
              sendSerialString();
              MyFlags &= ~serialRequest;
              serialRequests = 0;
            }

          }
         }
}
//*****************************************************************************
//***************** End of Main Loop ******************************************
//*****************************************************************************
void changeTime(void)          //20/03/17 00:40:46
{
 StartCondition();
 SendI2C(0xd0);
 WaitAck();
 SendI2C(0x00);
 WaitAck();
 SendI2C((serialInBuffer[21] & 0x0f) | ((serialInBuffer[20] & 0x0f) * 0x10)); // seconds
 WaitAck();
 SendI2C((serialInBuffer[18] & 0x0f) | ((serialInBuffer[17] & 0x0f) * 0x10)); // mins
 WaitAck();
 SendI2C((serialInBuffer[15] & 0x0f) | ((serialInBuffer[14] & 0x0f) * 0x10)); // hours
 WaitAck();
 SendI2C(0x00);
 WaitAck();
 SendI2C((serialInBuffer[12] & 0x0f) | ((serialInBuffer[11] & 0x0f) * 0x10)); // day
 WaitAck();
 SendI2C((serialInBuffer[9] & 0x0f) | ((serialInBuffer[8] & 0x0f) * 0x10)); // month
 WaitAck();
 SendI2C((serialInBuffer[6] & 0x0f) | ((serialInBuffer[5] & 0x0f) * 0x10)); // year
 WaitAck();
 StopCondition();
}
//*****************************************************************************
void sendSerialFilePart(void)
{
 int i=0;



 while (serialOutBuffer[i] !=0)
```

```
 {
  fileChecksum = fileChecksum + serialOutBuffer[i];
  UCA0TXBUF = serialOutBuffer[i];
  while (!(IFG2&UCA0TXIFG));
  i++;
 }

}
//*****************************************************************************

void sendFileSizeToPC(void)
{
 //FG#Y2017$Mar.alm@7F
 char path[32]={'/'};
 char fileNM[32]={'$'};
 int i=0;
 int j=1;
 int k=1;
 while (serialInBuffer[i++] != '#');
 i--;
 while (serialInBuffer[i++] != '$')
 {
  path[j++] = serialInBuffer[i];
  fileNM[k++] = serialInBuffer[i];
 }
 path[j-1]= '/';
 fileNM[k-1] = ',';
 //j++;
 i--;
 while (serialInBuffer[i++] != '@')
 {
  path[j++] = serialInBuffer[i];
  fileNM[k++]=serialInBuffer[i];
 }
 path[j-1]=0;
 fileNM[k-1]=0;
 strcpy(fileToSend,path);
 res=f_stat(path,&fno);
 fileSize = fno.fsize;
 sprintf(TempBuffer,"%li",fileSize);
 strcpy(serialOutBuffer,"FZ#");
 strcat(serialOutBuffer,TempBuffer);
 strcat(serialOutBuffer,fileNM);
 sendSerialString();
 __no_operation();


}
//*****************************************************************************
```

```
void sendFileTreeToPC(void)
{
 strcpy(serialOutBuffer,"LF#");
 FRESULT res;
 char name[16];
 char path[16] = '/';
 char inpath[16] = '/';
 DIR dir;
 UINT i;
 static FILINFO fno;
 res = f_opendir(&dir, path);
 TempBuffer[0]=0;
 while ((f_readdir(&dir, &fno) == FR_OK) && (fno.fname[0] != 0))
 {

  if ((fno.fattrib & AM_DIR) && (fno.fname[0] == 'Y') && (strlen(fno.fname)==5))
  {
   strcat(TempBuffer,fno.fname);
   strcat(TempBuffer,",");

  }
 }
 TempPointer=0;
 for(;;)
 {
 strcpy(path,getDirectoryFromBuffer());
 if (path[0] == 0) break;
 strcat(serialOutBuffer,path);
 strcat(serialOutBuffer,"$");

 strcpy(inpath,"/");
 strcat(inpath,path);
 f_opendir(&dir, inpath);
 res = f_readdir(&dir, &fno);
 while((res == FR_OK) && (strlen(fno.fname) > 3))
    {
     strcat(serialOutBuffer,fno.fname);
     strcat(serialOutBuffer,",");
     res = f_readdir(&dir,&fno);
    }
 serialOutBuffer[strlen(serialOutBuffer)-1] =0;
 strcat(serialOutBuffer,"&");
 }
 f_closedir(&dir);
 serialOutBuffer[strlen(serialOutBuffer)-1] =0;
 sendSerialString();


}
```

```
char * getDirectoryFromBuffer(void)
{
 if (TempBuffer[TempPointer] == 0) return "";
 int i=0;
 char hbuf[8]={0};
 while((TempBuffer[TempPointer] != ',') && (i < 8)) hbuf[i++] = TempBuffer[TempPointer++];
 hbuf[i]=0;
 TempPointer++;
 return hbuf;
}

//*****************************************************************************
void sendSerialTime(void)
{
 GetRTC();
 serialOutBuffer[0] = 'T';
 serialOutBuffer[1] = '#';
 serialOutBuffer[2] = ClockBuffer[6];
 serialOutBuffer[3] = ClockBuffer[7];
 serialOutBuffer[4] = ClockBuffer[8];
 serialOutBuffer[5] = ClockBuffer[9];
 serialOutBuffer[6] = '/';
 serialOutBuffer[7] = ClockBuffer[3];
 serialOutBuffer[8] = ClockBuffer[4];
 serialOutBuffer[9] = '/';
 serialOutBuffer[10] = ClockBuffer[0];
 serialOutBuffer[11] = ClockBuffer[1];
 serialOutBuffer[12] = '-';
 serialOutBuffer[13] = ClockBuffer[11];
 serialOutBuffer[14] = ClockBuffer[12];
 serialOutBuffer[15] = ClockBuffer[13];
 serialOutBuffer[16] = ClockBuffer[14];
 serialOutBuffer[17] = ClockBuffer[15];
 serialOutBuffer[18] = ClockBuffer[16];
 serialOutBuffer[19] = ClockBuffer[17];
 serialOutBuffer[20] = ClockBuffer[18];
 serialOutBuffer[21] = 0;
 sendSerialString();

//ClockBuffer: 09/02/2017,16:45:17
}
char storeToFile(void)
{
 char name[32];
 char tmp[8];
 GetRTC();
 strcpy(name,"/");
 //name[0]=0;
 sprintf(tmp,"%i",Year);
```

```
 strcat(name,"Y");
 strcat(name,tmp);
 strcat(name,"/");
 strncat(name,Months+(Month-1)*3,3);
 strcat(name,".alm");
 if (f_open(&Fil, name, FA_OPEN_EXISTING | FA_WRITE) == FR_OK)
 {
  f_lseek(&Fil, f_size(&Fil));
  if (f_write(&Fil, RxBuffer, strlen(RxBuffer), &usBytesWrite)==FR_OK)
  {

    f_write(&Fil,ClockBuffer,strlen(ClockBuffer),&usBytesWrite);
    f_close (&Fil);
    RxLedOFF();
    TxLedOFF();
    return 0;
  }
  else return 0xff;
 }
 else
 {
  sprintf(tmp,"Y%i",Year);
  if (f_opendir(&dir, tmp) != FR_OK)
  {
    res = f_mkdir(tmp);
    f_closedir(&dir);
  }
  f_closedir(&dir);
  if (f_open(&Fil, name, FA_CREATE_ALWAYS | FA_WRITE) == FR_OK)
  {
    if (f_write(&Fil, RxBuffer, strlen(RxBuffer), &usBytesWrite)==FR_OK)
    {

      f_write(&Fil,ClockBuffer,strlen(ClockBuffer),&usBytesWrite);
      f_close (&Fil);
      RxLedOFF();
      TxLedOFF();
      return 0;
    }
  }
  else return 0xff;
 }
 return 0xff;
}
// ClockBuffer: 09/02/2017,16:45:17
//*****************************************************************************
void getPhoneSettings(void)
{
 if (f_open(&Fil, "telSettings.txt", FA_READ) == FR_OK)
```

```
{
  f_read(&Fil,TempBuffer,200,&usBytesRead);
  f_close(&Fil);

  char *ret;
  char buz;
  //
  ret = strstr(TempBuffer,"AN,");
  if (ret!=0) buz = (unsigned char)atoi(ret+strlen("AN,"));
  if(buz == 1)
  {
    getConfirmFromAll = true;
  }
  else
  {
    getConfirmFromAll = false;
  }

  ret = strstr(TempBuffer,"TR,");
  if (ret!=0) numberOfRetries = (unsigned char)atoi(ret+strlen("TR,"));

  ret = strstr(TempBuffer,"PR,");
  if (ret!=0)
  {
    ret = ret + strlen("PR,");
    int i=0;
    while((ret[0] >= '0') && (ret[0] <= '9'))
    {
        Prefix[i++] = ret[0];
        ret++;
    }
  }

  ret = strstr(TempBuffer,"T1,");
  if (ret!=0)
  {
    ret = ret + strlen("T1,");
    int i=0;
    while((ret[0] >= '0') && (ret[0] <= '9'))
    {
        Telephone1[i++] = ret[0];
        ret++;
    }
  }

  ret = strstr(TempBuffer,"T2,");
  if (ret!=0)
  {
    ret = ret + strlen("T2,");
```

```
      int i=0;
      while((ret[0] >= '0') && (ret[0] <= '9'))
      {
          Telephone2[i++] = ret[0];
          ret++;
      }
     }

     ret = strstr(TempBuffer,"T3,");
     if (ret!=0)
     {
      ret = ret + strlen("T3,");
      int i=0;
      while((ret[0] >= '0') && (ret[0] <= '9'))
      {
          Telephone3[i++] = ret[0];
          ret++;
      }
     }
   }
  tel1Programmed = true; tel2Programmed = true;tel3Programmed = true;

  if ((strlen(Telephone1) < 1) || strlen(Telephone1) > 16) tel1Programmed = false;
  if ((strlen(Telephone2) < 1) || strlen(Telephone2) > 16) tel2Programmed = false;
  if ((strlen(Telephone3) < 1) || strlen(Telephone3) > 16) tel3Programmed = false;
}
//*****************************************************************************
void callHandler(void)
{
  char result;
  int retr=0;
  for (int i=0;i<alarmBufferSize;i++)
  {
   if (alarmBuffer[i].alarmID != 0)
   {
    if (alarmBuffer[i].nextCallTimer_1 != 0xff) alarmBuffer[i].nextCallTimer_1 += 1;
    if (alarmBuffer[i].nextCallTimer_2 != 0xff) alarmBuffer[i].nextCallTimer_2 += 1;
    if (alarmBuffer[i].nextCallTimer_3 != 0xff) alarmBuffer[i].nextCallTimer_3 += 1;
   }
  }
  if (tel1Programmed == true)
  {
   for (int i=0; i < alarmBufferSize; i++)
   {
    if ((alarmBuffer[i].alarmID != 0) && (alarmBuffer[i].nextCallTimer_1 > nextCallTime) &&
(alarmBuffer[i].nextCallTimer_1 != 0xff))
    {
     result = makeCall(Telephone1, Prefix);
     if (result == 0)
```

```
      {
        if (speechAlarms(1) == true)
        {
          if (getConfirmFromAll == true)
          {
            for (int i=0; i<alarmBufferSize;i++) alarmBuffer[i].nextCallTimer_1 = 0xff;
          }
          else
          {
            for (int i=0; i<alarmBufferSize;i++) alarmBuffer[i].alarmID = 0;
          }
        }
        else
        {
          for (int i=0; i < alarmBufferSize; i++)
          {
            alarmBuffer[i].nextCallTimer_1 = 0;
            retr = alarmBuffer[i].Retries;
            alarmBuffer[i].Retries &= 0xfff0;
            retr &= 0x000f;
            retr--;
            if (retr != 0)
            {
              alarmBuffer[i].Retries |= retr;
            }
            else
            {
              alarmBuffer[i].nextCallTimer_1 = 0xff;
            }
          }
        }
        disableGyro();
      }
      else
      {
        alarmBuffer[i].nextCallTimer_1 = nextCallTime - 1;

      }
    }
  }
 }

  if (tel2Programmed == true)
  {
    for (int i=0; i < alarmBufferSize; i++)
    {
      if ((alarmBuffer[i].alarmID != 0) && (alarmBuffer[i].nextCallTimer_2 > nextCallTime) &&
(alarmBuffer[i].nextCallTimer_2 != 0xff))
      {
```

```
    result = makeCall(Telephone2, Prefix);
    if (result == 0)
    {
        if (speechAlarms(2) == true)
        {
         if (getConfirmFromAll == true)
         {
          for (int i=0; i<alarmBufferSize;i++) alarmBuffer[i].nextCallTimer_2 = 0xff;
         }
         else
         {
          for (int i=0; i<alarmBufferSize;i++) alarmBuffer[i].alarmID = 0;
         }
        }
        else
        {
         for (int i=0; i < alarmBufferSize; i++)
         {
          alarmBuffer[i].nextCallTimer_2 = 0;
          retr = alarmBuffer[i].Retries;
          alarmBuffer[i].Retries &= 0xff0f;
          retr &= 0x00f0;
          retr = retr >> 4;
          retr--;
          if (retr != 0)
          {
           alarmBuffer[i].Retries |= retr << 4;
          }
          else
          {
           alarmBuffer[i].nextCallTimer_2 = 0xff;
          }
         }
        }
        disableGyro();
    }
    else
    {

        alarmBuffer[i].nextCallTimer_2 = nextCallTime - 1;
    }
   }
  }
}

if (tel3Programmed == true)
{
  for (int i=0; i < alarmBufferSize; i++)
  {
```

```
    if ((alarmBuffer[i].alarmID != 0) && (alarmBuffer[i].nextCallTimer_3 > nextCallTime) &&
(alarmBuffer[i].nextCallTimer_3 != 0xff))
    {
      result = makeCall(Telephone3, Prefix);
      if (result == 0)
      {
          if (speechAlarms(3) == true)
          {
            if (getConfirmFromAll == true)
            {
              for (int i=0; i<alarmBufferSize;i++) alarmBuffer[i].nextCallTimer_3 = 0xff;
            }
            else
            {
              for (int i=0; i<alarmBufferSize;i++) alarmBuffer[i].alarmID = 0;
            }
          }
          else
          {
            for (int i=0; i < alarmBufferSize; i++)
            {
              alarmBuffer[i].nextCallTimer_3 = 0;
              retr = alarmBuffer[i].Retries;
              alarmBuffer[i].Retries &= 0xf0ff;
              retr &= 0x0f00;
              retr = retr >> 8;
              retr--;
              if (retr != 0)
              {
                alarmBuffer[i].Retries |= retr << 8;
              }
              else
              {
                alarmBuffer[i].nextCallTimer_3 = 0xff;
              }
            }
          }
          disableGyro();
      }
      else
      {

          alarmBuffer[i].nextCallTimer_3 = nextCallTime - 1;
      }
    }
  }
 }

 for (int i=0; i < alarmBufferSize; i++)
```

```
  {
    if ((alarmBuffer[i].Retries & 0x0fff) == 0) alarmBuffer[i].alarmID = 0;       /
  }
  for (int i=0; i < alarmBufferSize; i++)
  {
    if ((alarmBuffer[i].nextCallTimer_1 == 0xff) && (alarmBuffer[i].nextCallTimer_2 == 0xff) &&
(alarmBuffer[i].nextCallTimer_3 == 0xff)) alarmBuffer[i].alarmID = 0;          /
  }
}
//*************************************************************************
void decodeIncomingSignal(char *buf)
{
  char numberOfFields = countFields(buf);
  getField(buf, 2);
  int t=atoi(fieldBuffer);
  if ((t==New) || (t==Critical) || (t==Bad))
  {
    getField(buf, 1);
    unsigned char name = atoi(fieldBuffer);
    for (int i=0;i<alarmBufferSize;i++)
    {
      if(alarmBuffer[i].chamberName == name)
      {
        alarmBuffer[i].alarmID = 0;
        alarmBuffer[i].chamberName = 0;
      }
    }
    alarmPointer = shortAlarms();
    if (alarmPointer >= alarmBufferSize)
    {
      for (unsigned char i=0;i<(alarmBufferSize-1);i++)
      {
        alarmBuffer[0] = alarmBuffer[1];
        alarmPointer = alarmBufferSize - 1;
      }
    }
    alarmBuffer[alarmPointer].chamberName = atoi(fieldBuffer);
    alarmBuffer[alarmPointer].alarmID = AlarmID;
    AlarmID++;
    if (AlarmID == 0) AlarmID = 1;
    getField(buf, 2);
    alarmBuffer[alarmPointer].Condition = atoi(fieldBuffer);
    getField(buf, 4);
    alarmBuffer[alarmPointer].Reason = atoi(fieldBuffer);
    getField(buf, 3);
    alarmBuffer[alarmPointer].Temperature = atof(fieldBuffer);
    alarmBuffer[alarmPointer].Retries = 0;
    if (tel1Programmed == true)
    {
```

```
    alarmBuffer[alarmPointer].nextCallTimer_1 = 0;
    alarmBuffer[alarmPointer].Retries = alarmBuffer[alarmPointer].Retries | 0x1000 |
numberOfRetries;
   }
   else alarmBuffer[alarmPointer].nextCallTimer_1 = 0xff;
   if (tel2Programmed == true)
   {
    alarmBuffer[alarmPointer].nextCallTimer_2 = 0;
    alarmBuffer[alarmPointer].Retries = alarmBuffer[alarmPointer].Retries | 0x2000 |
numberOfRetries << 4;
   }
   else alarmBuffer[alarmPointer].nextCallTimer_2 = 0xff;
   if (tel3Programmed == true)
   {
    alarmBuffer[alarmPointer].nextCallTimer_3 = 0;
    alarmBuffer[alarmPointer].Retries = alarmBuffer[alarmPointer].Retries | 0x4000 |
numberOfRetries << 8;
   }
   else alarmBuffer[alarmPointer].nextCallTimer_3 = 0xff;
  }
  //---------------------------------------------------------------------------
  if (numberOfFields > 10)
  {
   getField(buf,7);
   t=atoi(fieldBuffer);
   if ((t==New) || (t==Critical) || (t==Bad))
   {
    getField(buf, 6);
    unsigned char name = atoi(fieldBuffer);
    for (int i=0;i<alarmBufferSize;i++)
    {
     if(alarmBuffer[i].chamberName == name)
     {
         alarmBuffer[i].alarmID = 0;
         alarmBuffer[i].chamberName = 0;
     }
    }
    alarmPointer = shortAlarms();
    if (alarmPointer >= alarmBufferSize)
    {
     for (unsigned char i=0;i<(alarmBufferSize-1);i++)
     {
         alarmBuffer[0] = alarmBuffer[1];
         alarmPointer = alarmBufferSize - 1;
     }
    }
    alarmBuffer[alarmPointer].chamberName = atoi(fieldBuffer);
    alarmBuffer[alarmPointer].alarmID = AlarmID;
    AlarmID++;
```

```
    if (AlarmID == 0) AlarmID = 1;
    getField(buf, 7);
    alarmBuffer[alarmPointer].Condition = atoi(fieldBuffer);
    getField(buf, 9);
    alarmBuffer[alarmPointer].Reason = atoi(fieldBuffer);
    getField(buf, 8);
    alarmBuffer[alarmPointer].Temperature = atof(fieldBuffer);
    alarmBuffer[alarmPointer].Retries = 0;
    if (tel1Programmed == true)
    {
      alarmBuffer[alarmPointer].nextCallTimer_1 = 0;
      alarmBuffer[alarmPointer].Retries = alarmBuffer[alarmPointer].Retries | 0x1000 |
numberOfRetries;
    }
    else alarmBuffer[alarmPointer].nextCallTimer_1 = 0xff;
    if (tel2Programmed == true)
    {
      alarmBuffer[alarmPointer].nextCallTimer_2 = 0;
      alarmBuffer[alarmPointer].Retries = alarmBuffer[alarmPointer].Retries | 0x2000 |
numberOfRetries << 4;
    }
    else alarmBuffer[alarmPointer].nextCallTimer_2 = 0xff;
    if (tel3Programmed == true)
    {
      alarmBuffer[alarmPointer].nextCallTimer_3 = 0;
      alarmBuffer[alarmPointer].Retries = alarmBuffer[alarmPointer].Retries | 0x4000 |
numberOfRetries << 8;
    }
    else alarmBuffer[alarmPointer].nextCallTimer_3 = 0xff;
   }
  }

}
//*****************************************************************************
char countFields(char *buf)
{
  char fields = 0;
  int i=0;
  while (buf[i] != 0)
  {
    if (buf[i++] == ',') fields++;
    if (i > 128) return fields;
  }
  return fields;
}
//*****************************************************************************
unsigned char shortAlarms(void)
{
  unsigned char i=0;
```

```
 unsigned char j=0;
 unsigned char lastFreePosition=0;
 for (i=0; i<alarmBufferSize;i++)
 {
  if (alarmBuffer[i].alarmID == 0)
  {
   j=i+1;
   while (alarmBuffer[j].alarmID == 0)
   {
    if (j >= alarmBufferSize-1)
    {
        return i;
    }
    else
    {
        j++;
    }
   }
   alarmBuffer[i] = alarmBuffer[j];
   alarmBuffer[j].alarmID=0;
   lastFreePosition = i + 1;
  }
  else
  {
   lastFreePosition = i + 1;
  }
 }
  return lastFreePosition;
}
//*****************************************************************************
// Speech alarms and returns true if answered and false if not answered
bool speechAlarms(unsigned char tel) //tel: first, second or third telephone number
{
 bool start = true;
 __delay_cycles(16000000 * 5);
 speechIndex = input_char;
 TACCTL1 = CCIE;
 TACCR1 = TAR + 4;
 DAC12_0CTL = DAC12IR + DAC12AMP_5 + DAC12ENC;
 int j=0;
 for (j=0; j < 3; j++)
 {
  TACCTL1 = CCIE;
  TACCR1 = TAR + 4;
  DAC12_0CTL = DAC12IR + DAC12AMP_5 + DAC12ENC;
  speechIndex = YouHaveAlarmSignalFromChamber;
  Speech();
  if (tel == 1)
  {
```

```
 for (int i = 0; i < alarmBufferSize; i++)
 {
  if ((alarmBuffer[i].alarmID != 0) && (alarmBuffer[i].nextCallTimer_1 != 0xff))
  {
      if (start == false)
      {
        speechIndex = AndFromChamber;
        Speech();
        __delay_cycles(4000000);
      }
      speechIndex = alarmBuffer[i].chamberName;
      Speech();
      __delay_cycles(4000000);
      speechIndex = chamberTemperature;
      Speech();
      __delay_cycles(2000000);
      speechIndex = Einai;
      Speech();
      __delay_cycles(4000000);
      SpeechTemperature(alarmBuffer[i].Temperature);
      __delay_cycles(4000000);
      speechIndex = signalCondition;
      Speech();
      __delay_cycles(4000000);
      speechIndex = alarmBuffer[i].Condition;
      speechIndex = conditionTable[speechIndex];
      Speech();
      __delay_cycles(8000000);
      start = false;
  }
 }
}
else if (tel == 2)
{
 for (int i = 0; i < alarmBufferSize; i++)
 {
  if ((alarmBuffer[i].alarmID != 0) && (alarmBuffer[i].nextCallTimer_3 != 0xff))
  {
      if (start == false)
      {
        speechIndex = AndFromChamber;
        Speech();
        __delay_cycles(4000000);
      }
      speechIndex = alarmBuffer[i].chamberName;
      Speech();
      __delay_cycles(4000000);
      speechIndex = chamberTemperature;
      Speech();
```

```
        __delay_cycles(2000000);
        speechIndex = Einai;
        Speech();
        __delay_cycles(4000000);
        SpeechTemperature(alarmBuffer[i].Temperature);
        __delay_cycles(4000000);
        speechIndex = signalCondition;
        Speech();
        __delay_cycles(4000000);
        speechIndex = alarmBuffer[i].Condition;
        speechIndex = conditionTable[speechIndex];
        Speech();
        __delay_cycles(8000000);
        start = false;
    }
  }
}
else if (tel == 3)
{
  for (int i = 0; i < alarmBufferSize; i++)
  {
    if ((alarmBuffer[i].alarmID != 0) && (alarmBuffer[i].nextCallTimer_3 != 0xff))
    {
        if (start == false)
        {
          speechIndex = AndFromChamber;
          Speech();
          __delay_cycles(4000000);
        }
        speechIndex = alarmBuffer[i].chamberName;
        Speech();
        __delay_cycles(4000000);
        speechIndex = chamberTemperature;
        Speech();
        __delay_cycles(2000000);
        speechIndex = Einai;
        Speech();
        __delay_cycles(4000000);
        SpeechTemperature(alarmBuffer[i].Temperature);
        __delay_cycles(4000000);
        speechIndex = signalCondition;
        Speech();
        __delay_cycles(4000000);
        speechIndex = alarmBuffer[i].Condition;
        speechIndex = conditionTable[speechIndex];
        Speech();
        __delay_cycles(8000000);
        start = false;
    }
```

```
    }
   }
   speechIndex = VConfirmationDiesis;
   Speech();
   enableReceiver();
   MyFlags |= RxEnable;
   Get5sChar();
   disableReceiver();
   start = true;
   if (input_char == '#') return true;
  }
  return false;
}
//------------------------------------------------------------------------------
void SpeechTemperature (float temp)
{
  int tmp=(int)temp;
  if (tmp < 0)
  {
   speechIndex = Negative;
   Speech();
   tmp = abs(tmp);
   __delay_cycles(4000000);
  }

  if (tmp == 3) tmp = 150;
  else if (tmp == 4) tmp = 151;
  else if (tmp == 13) tmp = 152;
  else if (tmp == 14) tmp = 153;
  else if (tmp == 23) tmp = 154;
  else if (tmp == 24) tmp = 155;
  else if (tmp == 33) tmp = 156;
  else if (tmp == 34) tmp = 157;
  else if (tmp == 43) tmp = 158;
  else if (tmp == 44) tmp = 159;
  else if (tmp == 53) tmp = 160;
  else if (tmp == 54) tmp = 161;
  else if (tmp == 63) tmp = 162;
  else if (tmp == 64) tmp = 163;
  else if (tmp == 73) tmp = 164;
  else if (tmp == 74) tmp = 165;
  else if (tmp == 83) tmp = 166;
  else if (tmp == 84) tmp = 167;
  else if (tmp == 93) tmp = 168;
  else if (tmp == 94) tmp = 169;
  else tmp = tmp;
  speechIndex = tmp;
   Speech();
   __delay_cycles(4000000);
```

```
  tmp = (temp - (int)temp) * 10;
  if (tmp != 0)
  {
    speechIndex = PointFive;
    Speech();
    __delay_cycles(4000000);
  }
  speechIndex = CelsiousDegrees;
  Speech();
  __delay_cycles(8000000);
}
//*****************************************************************************
//*****************************************************************************
unsigned char sendSettings(char *buf)
{
 int length=0;
 unsigned char Checksum='#';

 for (int i=0; i<sizeof(TempBuffer); i++) TempBuffer[i]=0;
 TempBuffer[0] = '*';
 TempBuffer[1] = 0x85;
 TempBuffer[2] = ',';
 getField(lineBuffer,0);
 strcat(TempBuffer,fieldBuffer);       // user ID
 strcat(TempBuffer,"\v9999\v9999\v0\v0\v0\v0\v0\v0\v0\v0\v0\v");
 getField(lineBuffer,1);               // InformTime
 strcat(TempBuffer,fieldBuffer);
 strcat(TempBuffer,"\v");
 getField(lineBuffer,2);               // Number of Chambers
 strcat(TempBuffer,fieldBuffer);
 strcat(TempBuffer,"\v");

 getField(lineBuffer,3);               // first chamber code
 strcat(TempBuffer,fieldBuffer);
 strcat(TempBuffer,"\v");
 getField(lineBuffer,4);               // first Used?
 strcat(TempBuffer,fieldBuffer);
 strcat(TempBuffer,"\v");
 getField(lineBuffer,5);               // first Low Limit?
 if (fieldBuffer[0] == '-') fieldBuffer[0] = 0x0e;
 strcat(TempBuffer,fieldBuffer);
 strcat(TempBuffer,"\v");
 getField(lineBuffer,6);               // first High Limit?
 if (fieldBuffer[0] == '-') fieldBuffer[0] = 0x0e;
 strcat(TempBuffer,fieldBuffer);
 strcat(TempBuffer,"\v\v0\v");                 // + Window width
 getField(lineBuffer,7);               // first NEW
 strcat(TempBuffer,fieldBuffer);
 strcat(TempBuffer,"\v");
```

```
getField(lineBuffer,8);                 // first CRITICAL
strcat(TempBuffer,fieldBuffer);
strcat(TempBuffer,"\v");
getField(lineBuffer,9);                 // first BAD
strcat(TempBuffer,fieldBuffer);
strcat(TempBuffer,"\v1\v");             // + processing Methode

getField(lineBuffer,10);                // second chamber code
strcat(TempBuffer,fieldBuffer);
strcat(TempBuffer,"\v");
getField(lineBuffer,11);                // second Used?
strcat(TempBuffer,fieldBuffer);
strcat(TempBuffer,"\v");
getField(lineBuffer,12);                // second Low Limit?
if (fieldBuffer[0] == '-') fieldBuffer[0] = 0x0e;
strcat(TempBuffer,fieldBuffer);
strcat(TempBuffer,"\v");
getField(lineBuffer,13);                // second High Limit?
if (fieldBuffer[0] == '-') fieldBuffer[0] = 0x0e;
strcat(TempBuffer,fieldBuffer);
strcat(TempBuffer,"\v\v0\v");                   // + Window width
getField(lineBuffer,14);                // second NEW
strcat(TempBuffer,fieldBuffer);
strcat(TempBuffer,"\v");
getField(lineBuffer,15);                // second CRITICAL
strcat(TempBuffer,fieldBuffer);
strcat(TempBuffer,"\v");
getField(lineBuffer,16);                // second BAD
strcat(TempBuffer,fieldBuffer);
strcat(TempBuffer,"\v1\v");             // + processing Methode
strcat(TempBuffer,"96\v#");
int i=3;
while (TempBuffer[i] != '#') TempBuffer[i] = TempBuffer[i++] & 0x0f;
length=i+2;
i=0;
while (TempBuffer[i] != '#') Checksum += TempBuffer[i++];
TempBuffer[length-1]=0xff - Checksum;
__no_operation();
TxLedON();
__delay_cycles(80000);
SendRS485Binary(TempBuffer,length);
RS485ControlL();
TxLedOFF();
if (RS485GetString()==0)
{
  if (strncmp(currentID,RxBuffer,5) == 0)
  {
    return 0;
  }
```

```
   else return 1;
  }
  else return 1;
}
//-------------------------------------------------------------------------
// Gets the field n from buff and it stored in fieldBuffer (seperators is * and ,)
void getField(char *buff, int field)
{
  char *p;
  int i;
  p = strchr(buff+1,',');
  if (field == 0)
  {
  }
  else if (field > 0)
  {
    while (field > 0)
    {
      p = strchr(p+1,',');
      field--;
    }
  }
  i=0;
  p++;
  while ((p[i] != ',') && (p[i] != '*'))
  {
    fieldBuffer[i] = p[i++];
    if (i>127) break;
  }
  fieldBuffer[i]=0;
}
//*****************************************************************************
void StoreRTC(void)
{
  if (f_open(&Fil, "Time.txt", FA_OPEN_EXISTING | FA_READ)==FR_OK)
  {
    f_read(&Fil,HelpBuffer,17,&usBytesRead);
    f_close(&Fil);
    if (usBytesRead > 16)
    {
      StartCondition();
      SendI2C(0xd0);
      WaitAck();
      SendI2C(0x00);
      WaitAck();
      SendI2C((HelpBuffer[16] & 0x0f) | ((HelpBuffer[15] & 0x0f) * 0x10)); // seconds
      WaitAck();
      SendI2C((HelpBuffer[13] & 0x0f) | ((HelpBuffer[12] & 0x0f) * 0x10)); // mins
      WaitAck();
```

```
    SendI2C((HelpBuffer[10] & 0x0f) | ((HelpBuffer[9] & 0x0f) * 0x10)); // hours
    WaitAck();
    SendI2C(0x00);
    WaitAck();
    SendI2C((HelpBuffer[1] & 0x0f) | ((HelpBuffer[0] & 0x0f) * 0x10)); // day
    WaitAck();
    SendI2C((HelpBuffer[4] & 0x0f) | ((HelpBuffer[3] & 0x0f) * 0x10)); // month
    WaitAck();
    SendI2C((HelpBuffer[7] & 0x0f) | ((HelpBuffer[6] & 0x0f) * 0x10)); // year
    WaitAck();
    StopCondition();
    f_unlink("Time.txt");
   }
  }
}
//*****************************************************************************
void GetRTC(void)
{
  char mbuf[8];
  StartCondition();
  SendI2C(0xd0);
  WaitAck();
  StartCondition();
  SendI2C(0);
  WaitAck();
  StopCondition();
  StartCondition();
  SendI2C(0xd1);
  WaitAck();
  for (int i=0; i<7; i++) TempBuffer[i] = GetI2C();
  StopCondition();
  ClockBuffer[0] = ((TempBuffer[4] & 0xf0) >> 4)| 0x30;
  ClockBuffer[1] = (TempBuffer[4] & 0x0f)| 0x30;
  ClockBuffer[2] = '/';
  ClockBuffer[3] = ((TempBuffer[5] & 0xf0) >> 4)| 0x30;
  ClockBuffer[4] = (TempBuffer[5] & 0x0f)| 0x30;
  ClockBuffer[5] = '/';
  ClockBuffer[6] = '2';
  ClockBuffer[7] = '0';
  ClockBuffer[8] = ((TempBuffer[6] & 0xf0) >> 4)| 0x30;
  ClockBuffer[9] = (TempBuffer[6] & 0x0f)| 0x30;
  ClockBuffer[10] = ',';
  ClockBuffer[11] = ((TempBuffer[2] & 0xf0) >> 4)| 0x30;
  ClockBuffer[12] = (TempBuffer[2] & 0x0f)| 0x30;
  ClockBuffer[13] = ':';
  ClockBuffer[14] = ((TempBuffer[1] & 0xf0) >> 4)| 0x30;
  ClockBuffer[15] = (TempBuffer[1] & 0x0f)| 0x30;
  ClockBuffer[16] = ':';
  ClockBuffer[17] = ((TempBuffer[0] & 0xf0) >> 4)| 0x30;
```

```
  ClockBuffer[18] = (TempBuffer[0] & 0x0f)| 0x30;
  ClockBuffer[19] = 0x0d;
  ClockBuffer[20] = 0x0a;
  ClockBuffer[21] = 0;
  mbuf[0] = '2';
  mbuf[1] = '0';
  mbuf[2] = ((TempBuffer[6] & 0xf0) >> 4)| 0x30;
  mbuf[3] = (TempBuffer[6] & 0x0f)| 0x30;
  mbuf[4] = 0;
  Year = atoi(mbuf);
  mbuf[0] = ((TempBuffer[5] & 0xf0) >> 4)| 0x30;
  mbuf[1] =  (TempBuffer[5] & 0x0f)| 0x30;
  mbuf[2] = 0;
  Month = atoi(mbuf);

}
//*************************************************************************
void SendI2C(char Sdata)
{
  P4OUT &= ~BIT1;
  P4DIR |= BIT1;
  unsigned char mask=0x80;
  for (int i=0;i<8;i++)
  {
    if((Sdata & mask)!= 0) P4OUT |= BIT1;
    else P4OUT&=~BIT1;
    P4OUT |=BIT0;
    __delay_cycles(10);
    P4OUT &=~BIT0;
    mask/=2;
  }
  P4OUT &=~BIT1;
  P4DIR &=~BIT1;
}
//*************************************************************************
char GetI2C(void)
{
  char Sdata;
  P4DIR &=~BIT1;
  for (int i=0;i<8;i++)
  {
    P4OUT |= BIT0;
    __no_operation();
    if (P4IN & BIT1) Sdata = Sdata*2+1;
    else Sdata = Sdata*2;
    P4OUT &= ~BIT0;
    __delay_cycles(10);
  }
  P4DIR |= BIT1;
```

```
  P4OUT &=~BIT1;
  P4OUT |= BIT0;
  __delay_cycles(10);
  P4OUT &=~ BIT0;
  P4DIR |=  BIT1;

  return Sdata;
}
//*****************************************************************************
char RS485GetString(void)
{
  RS485ControlL();
  checksum = 0;
  for (ptr=0;inchar!='#';ptr++)
  {
    inchar=GetRS485Byte();
    if (inchar == 0xff) return 0xff;
    checksum+= inchar;
    RxBuffer[ptr]=inchar;
  }
  inchar=GetRS485Byte();
  checksum+=inchar;
  if (checksum==0)
  {

    RxBuffer[ptr-1]=0;
    int i=strlen(RxBuffer);
      for (int j=0;j<i;j++)
           {
        RxBuffer[j]=RxBuffer[j+3];
        if (RxBuffer[j]=='*') RxBuffer[j]='1';
           }
    return 0;
  }
  else return 0xff;
}
//*****************************************************************************
void SendRS485String(char *RS485Buffer)
{
          checksum=0 ;
          RS485ControlH();
            __delay_cycles(1280000);

            __delay_cycles(1280000);
          for (int i=0;RS485Buffer[i] !=0;i++)
          {
          checksum+= RS485Buffer[i];
          SendByte485(RS485Buffer[i]);
          }
```

```
          checksum +=1;
          SendByte485(0-checksum);
          __delay_cycles(320000);
        RS485ControlL();
}
//*****************************************************************************
void SendConfirmToModule(unsigned char *RS485Buffer)
{
          checksum=0 ;
          RS485ControlH();
           __delay_cycles(2560000);

          for (int i=0;RS485Buffer[i] !=0;i++)
          {
          checksum+= RS485Buffer[i];
          SendByte485(RS485Buffer[i]);
          }
          SendByte485(0-checksum);
          __delay_cycles(320000); /
        RS485ControlL();
}

//*****************************************************************************
void SendRS485Binary(char *RS485Buffer, int len)
{
          RS485ControlH();
           __delay_cycles(2560000);
          for (int i=0;i<len;i++)
          {
          SendByte485(RS485Buffer[i]);
          }
          __delay_cycles(320000);
        RS485ControlL();
}
//*****************************************************************************
DWORD get_fattime (void)
{
 char mbuf[8];
 GetRTC();
 mbuf[0] = ClockBuffer[0];
 mbuf[1] = ClockBuffer[1];
 mbuf[2] = 0;
 unsigned char Day = atoi(mbuf);
 mbuf[0] = ClockBuffer[11];
 mbuf[1] =  ClockBuffer[12];
 mbuf[2] = 0;
 unsigned char Hour = atoi(mbuf);
 mbuf[0] = ClockBuffer[14];
 mbuf[1] = ClockBuffer[15];
```

```
  mbuf[2] = 0;
  unsigned char Min = atoi(mbuf);
  mbuf[0] = ClockBuffer[17];
  mbuf[1] = ClockBuffer[18];
  mbuf[2] = 0;
  unsigned char Sec = atoi(mbuf);


        return          ((DWORD)(Year - 1980) << 25)        /* Year = 2012 */
                        | ((DWORD)Month << 21)              /* Month = 1 */
                        | ((DWORD)Day << 16)                /* Day_m = 1*/
                        | ((DWORD)Hour << 11)               /* Hour = 0 */
                        | ((DWORD)Min << 5)          /* Min = 0 */
                        | ((DWORD)Sec >> 1);         /* Sec = 0 */
}
//09/02/2017,16:45:17
//*****************************************************************************
void TimerA0Init (void)
{
  CCTL0 = CCIE;
  CCR0 = 32;
  TACTL = TASSEL_1 + MC_2;
}
//*****************************************************************************
void TimerB0Init (void)
{
  TBCCTL0 = CCIE;
  TBCCR0 = 32768;
  TBCTL = TBSSEL_1 + MC_2;
}

//*****************************************************************************
void UARTA1init(void)
{
  P3SEL |= 0xC0;
  UCA1CTL1 |= UCSSEL_1;
  UCA1BR0 = 0x1A;
  UCA1BR1 = 0x00;
  UCA1CTL1 &= ~UCSWRST;
}
//*****************************************************************************
void SendByte485 (char ByteToSend)
{
  while (!(UC1IFG&UCA1TXIFG));
  UCA1TXBUF = ByteToSend;
}
//*****************************************************************************
char GetRS485Byte(void)
{
```

```
  mSeconds=1000;
  return GetRS485TimedByte();
}
//---------------------------------------------------------------------
char GetRS485TimedByte(void)
{
  do
  {
    if (UC1IFG & UCA1RXIFG) return UCA1RXBUF;
  }while(mSeconds > 0);

  return 0xFF;
}
//*********************************************************************************
void StartCondition(void)
{
  P4DIR |= BIT1;
  P4OUT &= ~BIT1;
  I2Cdelay();
  P4OUT &= ~BIT0;
}
//*********************************************************************************
void StopCondition(void)
{
  P4OUT |= BIT0;
  I2Cdelay();
  P4DIR |= BIT1;
  P4OUT |= BIT1;
}
//*********************************************************************************
void I2Cdelay(void)
{
  __delay_cycles(10);
}
//*********************************************************************************
void WaitAck(void)
{
  P4DIR &= ~BIT1;
  P4OUT |= BIT0;
  while(P4IN & BIT1){}
  P4OUT &= ~BIT0;
  P4DIR |= BIT1;
}
//*********************************************************************************
char makeCall( char *phoneNumber, char *prefix)
{
  float volt = getLineVoltage();
  if (volt > 14)
  {
```

```
   enableGyro();
   __delay_cycles(2000000 * 16);
   enableTransmitter();
   if (prefix[0] != 0)
   {
    for (int i=0;i<strlen(prefix); i++)
    {
     sendDigit = prefix[i];
     sendDigit &= 0x0f;
     MyFlags |= TxEnable;
     __delay_cycles(250000 * 16);
     MyFlags &= ~TxEnable;
     __delay_cycles(500000 * 16);
    }
    __delay_cycles(2000000 * 16);
   }
   for (int i=0;i<strlen(phoneNumber); i++)
   {
    sendDigit = phoneNumber[i];
    sendDigit &= 0x0f;
    MyFlags |= TxEnable;
    __delay_cycles(250000 * 16);
    MyFlags &= ~TxEnable;
    __delay_cycles(500000 * 16);
   }
   disableTransmitter();
   return 0;
  }
  else
   return 0xff;
}
//*************************************************************************************
void enableTransmitter(void)
{
 MyFlags &= ~TxEnable;
 TACCR1 = TAR + 3;
 CCTL1 = CCIE;
 //CCTL0 = 0;
 ADC12CTL0 &= ~ENC;
 ADC12CTL0 = REF2_5V + REFON;
 DAC12_0CTL = DAC12IR + DAC12AMP_5 + DAC12ENC;
}
//-------------------------------------------------------------------------
void disableTransmitter(void)
{
 CCTL1 = 0;
}

//*************************************************************************************
```

```
void enableReceiver(void)
{
 TACCR2 = TAR + 9;
 CCTL2 = CCIE;
 CCTL1=0;
 MyFlags |= RxEnable;
 COUNTb = 0;
 LCOUNTb = 0;

 ADC12CTL0 &= ~ENC;
 ADC12CTL0 = ADC12ON + SHT0_5 + REF2_5V + REFON;
 ADC12CTL1 = SHP;
 ADC12IE = 0x01;
 P6SEL |= BIT3;
 ADC12MCTL0 = 0x03;
 ADC12CTL0 |= ENC;
}
//-------------------------------------------------------------------------
void disableReceiver(void)
{
 CCTL2 = 0;
}

//*************************************************************************************
float getLineVoltage(void)
{
 ADC12CTL0 &= ~ENC;
 ADC12CTL0 = ADC12ON + SHT0_5 + REF2_5V + REFON;
 ADC12CTL1 = SHP;
 P6SEL |= BIT2;
 ADC12MCTL0 = SREF_1 + 0x02;
 ADC12CTL0 |= ENC;
 __delay_cycles(2000000);
 ADC12CTL0 |= ADC12SC;
 __delay_cycles(1600000);
 int volt = ADC12MEM0;
 return volt / 38.70233446;
}
//*************************************************************************************
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A0 (void)
{
 TACCR0 += 32;
 mSeconds-=1;
 if (mSeconds==0) MyFlags |= mSecondsFL;
}
//-------------------------------------------------------------------------
#pragma vector=TIMERA1_VECTOR
__interrupt void Timer_A1(void)
```

```
{
 switch( TAIV )
 {
 case  2:
         if ((MyFlags & TxEnable) !=0)
         {
          TACCR1 += 3;
          SendDtmf();
         }
         else
         {
          TACCR1 += 4;
          SendDtmf();
         }

        break;
 case  4: CCR2 = TAR + 9;
         msTimer -=1;
         if (msTimer == 0) MyFlags |= RxTimeFL;
         ADC12CTL0 |= ADC12SC;

        break;
 case 10: P1OUT ^= 0x01;
        break;
 }
}
//*************************************************************************
#pragma vector=TIMERB0_VECTOR
__interrupt void TIMERB0v (void)
{
 TBCCR0 += 32768;
 Seconds +=1;
 if (Seconds > 59)
 {
  Seconds = 0;
  Minuets +=1;
  minFL = true;
 }
}
//*************************************************************************
void serialInit(void)
{
 P3SEL |= 0x30;
 UCA0CTL1 |= UCSSEL_2;
 UCA0BR0 = 17;
 UCA0BR1 = 0;
 UCA0MCTL = UCBRS1 + UCBRS0;
 UCA0CTL1 &= ~UCSWRST;
 IE2 |= UCA0RXIE;
```

```
}
//*****************************************************************************
//*****************************************************************************
//*****************************************************************************
//NT#201@A8cl

#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCIAB0RXv (void)
{
  char inByte, chH, chL;
  inByte = UCA0RXBUF;

  serialInBuffer[serialInPointer++] = inByte;
  if ((serialInBuffer[serialInPointer-3] == 'S') && (serialInBuffer[serialInPointer-2] == 'S') &&
(serialInBuffer[serialInPointer-1] == '#')) MyFlags |= receivingSetting;
  if ((serialInBuffer[serialInPointer-4] == 'S') && (serialInBuffer[serialInPointer-3] == 'T') &&
(serialInBuffer[serialInPointer-2] == 'S')&& (serialInBuffer[serialInPointer-1] == '#')) MyFlags |=
receivingSetting;
  if ((inByte == '\n') && ((MyFlags & receivingSetting) == 0))
  {
    serialInChecksum = 0;
    for (int i = 0; i < serialInPointer-5; i++) serialInChecksum += serialInBuffer[i];
    chL = asciiTab[serialInChecksum & 0x000f];
    chH = asciiTab[(serialInChecksum & 0x00f0) >> 4];
    if (chH == serialInBuffer[serialInPointer-4] && chL == serialInBuffer[serialInPointer-3])
    {
      if ((serialInBuffer[0] == 'T') && (serialInBuffer[1] == 'C'))
      {
        __no_operation();
        MyFlags |= serialRequest;
        serialRequests = sendTime;
      }
      else if ((serialInBuffer[0] == 'N') && (serialInBuffer[1] == 'T') && (serialInBuffer[2] == '#'))
      {
        __no_operation();
        MyFlags |= serialRequest;
        serialRequests = sendTimeConfirmation;
      }
      else if ((serialInBuffer[0] == 'V') && (serialInBuffer[1] == '#') && (serialInBuffer[2] == 'O') &&
(serialInBuffer[3] == 'K'))
      {

        __no_operation();
      }
      else if ((serialInBuffer[0] == 'D') && (serialInBuffer[1] == 'I') && (serialInBuffer[2] == 'R'))
      {
        MyFlags |= serialRequest;
        serialRequests = sendFileTree;
        __no_operation();
```

```
    }
    else if ((serialInBuffer[0] == 'F') && (serialInBuffer[1] == 'G'))
    {

        MyFlags |= serialRequest;
        serialRequests = sendFileSize;
        __no_operation();
        proccess=1;
    }
    else if ((serialInBuffer[0] == 'V') && (serialInBuffer[1] == '#') && (serialInBuffer[2] == 'F') &&
(serialInBuffer[3] == 'O'))
    {
      MyFlags |= serialRequest;
      serialRequests = sendFile;
      __no_operation();
    }
    else if ((serialInBuffer[0] == 'V') && (serialInBuffer[1] == '#') && (serialInBuffer[2] == 'F') &&
(serialInBuffer[3] == 'D') && (serialInBuffer[4] == 'O') && (serialInBuffer[5] == 'K') )
    {
      MyFlags |= serialRequest;
      serialRequests = sendFileParts;
      __no_operation();
    }
    else if ((serialInBuffer[0] == 'S') && (serialInBuffer[1] == 'S') && (serialInBuffer[2] == '#'))
    {
      MyFlags |= serialRequest;
      serialRequests = settingsReceived;
      __no_operation();
    }
    else if ((serialInBuffer[0] == 'S') && (serialInBuffer[1] == 'T') && (serialInBuffer[2] == 'S') &&
(serialInBuffer[3] == '#'))
    {
      MyFlags |= serialRequest;
      serialRequests = telSettingsReceived;
      __no_operation();
    }

   }
   serialInPointer = 0;
  }
  else if ((MyFlags & receivingSetting) != 0 )
  {
   if (inByte == '@')
   {
    MyFlags &= ~receivingSetting;
   }


  }
```

```
}
//*****************************************************************************
void sendSerialString(void)
{
 int i=0;
 int checksum = 0;
 char chL,chH;
 while (serialOutBuffer[i] !=0)
 {
  checksum = checksum + serialOutBuffer[i];
  UCA0TXBUF = serialOutBuffer[i];
  while (!(IFG2&UCA0TXIFG));
  i++;
 }
 UCA0TXBUF = '@';
 while (!(IFG2&UCA0TXIFG));
 chL = asciiTab[checksum & 0x000f];
 chH = asciiTab[(checksum & 0x00f0) >> 4];
 UCA0TXBUF = chH;
 while (!(IFG2&UCA0TXIFG));
 UCA0TXBUF = chL;
 while (!(IFG2&UCA0TXIFG));
 UCA0TXBUF = 0x0d;
 while (!(IFG2&UCA0TXIFG));
 UCA0TXBUF = 0x0a;
 while (!(IFG2&UCA0TXIFG));
}
//*****************************************************************************
//*****************************************************************************
//*****************************************************************************
#pragma vector=DAC12_VECTOR
__interrupt void DAC12v (void)
{
 PowerLedON();
 while(1)
 {
 }
}
#pragma vector=DMA_VECTOR
__interrupt void DMAv (void)
{
 PowerLedON();
 while(1)
 {
 }
}
#pragma vector=USCIAB1TX_VECTOR
__interrupt void USCIAB1TXv (void)
{
```

```
  PowerLedON();
  while(1)
  {
  }
}
#pragma vector=USCIAB1RX_VECTOR
__interrupt void USCIAB1RXv (void)
{
  PowerLedON();
  while(1)
  {
  }
}
#pragma vector=PORT1_VECTOR
__interrupt void PORT1v (void)
{
  PowerLedON();
  while(1)
  {
  }
}
#pragma vector=PORT2_VECTOR
__interrupt void PORT2v (void)
{
  PowerLedON();
  while(1)
  {
  }
}
#pragma vector=ADC12_VECTOR
__interrupt void ADC12v (void)
{
  DTMF_FILTER();
}
#pragma vector=USCIAB0TX_VECTOR
__interrupt void USCIAB0TXv (void)
{
  PowerLedON();
  while(1)
  {
  }
}

#pragma vector=WDT_VECTOR
__interrupt void WDTv (void)
{
  PowerLedON();
  while(1)
  {
```

```
  }
}
#pragma vector=TIMERB1_VECTOR
__interrupt void TIMERB1v (void)
{
  PowerLedON();
  while(1)
  {
  }
}

#pragma vector=NMI_VECTOR
__interrupt void NMIv (void)
{
  PowerLedON();
  while(1)
  {
  }
}
```

ΚΩΔΙΚΑΣ

3. ΑΙΣΘΗΤΗΡΙΟ ΚΩΔΙΚΑΣ DEFINITIONS

```
;_____________________________________________
; Constants/Constant Variables
;_____________________________________________
ElectricityAlg=0
RelaysAlg=0
PT100=0
password_length     EQU   4       ; length of password sequence
Up_window    EQU    3       ;
Down_window         EQU   3       ;
Hook_Time    EQU    600     ; Maximum OffHook Duration (seconds)
watch_time   EQU    30      ; Maximum OFF-HOOK Time Without Activity (Seconds)
next_call_time      EQU   120     ; time for next mesure after a call (seconds)
LENGTH              EQU   7       ; (Length+1) * 5.48ms=Signallength
THRE         EQU    200     ; threshold for noise of Input DTMF signal

#define      ENABLE_POWER              BIT0,&p1out
#define      DTMF_SENSOR_POWER               BIT5,&p2out
#define      ENABLE_GYRO        BIT3,&p2out
#define      ENABLE_POWER_dir          BIT0,&p1dir
#define      DTMF_SENSOR_POWER_dir  BIT5,&p2dir
#define      ENABLE_GYRO_dir           BIT3,&p2dir
```

```
;#define     RS485_Control               BIT6,&p3out   ; first version
#define     RS485_Control               BIT3,&p3out   ; Second Version
#define     RelayA              BIT7,&P1out
#define     RelayB              BIT5,&P1out
RING_PIN        equ         BIT3
#define     Door1Dir                    Bit2,&p1dir
#define     Door2Dir                    Bit1,&p1dir
#define     Door1               Bit2,&p1in
#define     Door2               Bit1,&p1in

;_____________________________________________________________________________
_____________________________________
; Software Definitions
;_____________________________________________________________________________
_____________________________________
#define     RS485TxPointer      R5
#define     RS485RxPointer      R6
#define     CompAR7
#define     CompBR8

#define     counter             R15
#define     LowProduct  R10
#define     Multiplicant  R13
#define     HighProduct  R9

#define     left_remainterr10
#define     right_remainter     r13
#define     divisor     r14
#define     Divident            r13

;----- &MY_FLAGS BIT DEFINITIONS --------------------
#define Hook_mode           0x0001,&My_Flags
#define ElectricityAlarmFL  0x0002,&My_Flags
#define Rx_enable           0x0004,&My_Flags
#define ElectricityAlarm    0x0008,&My_Flags
#define ElectricityHelpFL   0x0010,&My_Flags
#define AskInF          0x0020,&My_Flags
#define Door1FL                 0x0040,&My_Flags
#define Door2FL                 0x0080,&My_Flags
#define SerialHelpFL    0x0100,&My_Flags
#define help_flag           0x0200,&My_Flags
#define AskFromModemFL      0x0400,&My_Flags
#define ChamberA_Informs    0x0800,&My_Flags
#define ChamberB_Informs    0x1000,&My_Flags
#define call_from_sensors   0x2000,&My_Flags
#define UidPasswordFL           0x4000,&My_Flags
#define FirstUsedFL         0x8000,&My_Flags
```

```
;-------- &My_FLAGS1 DEFINITIONS ----------------------
#define Help_FlagA              0x0001,&My_Flags1
#define SecondUsedFL                0x0002,&My_Flags1
#define Help_FlagC          0x0004,&My_Flags1
#define CallWait                0x0008,&My_Flags1
#define FirstChamberUsedFl  0x0010,&My_Flags1
#define SecondChamberUsedFl         0x0020,&My_Flags1
#define TerminalUsedFl              0x0040,&My_Flags1
#define ChamberA_Informed 0x0080,&My_Flags1
#define ChamberB_Informed 0x0100,&My_Flags1
#define Confirmation_Flag     0x0200,&My_Flags1
#define HumidityUsedFL              0x0400,&My_Flags1
#define AnswerFL                0x0800,&My_Flags1
#define AnswerModeFL                0x1000,&My_Flags1
#define TxTFL                   0x2000,&My_Flags1
#define RxTFL                   0x4000,&My_Flags1
#define HumFL                       0x8000,&My_Flags1

;----- &MY_ALARMS BIT DEFINITIONS -------------------
#define TempStatusA             0x0001,&My_Alarms
#define TempStatusB             0x0002,&My_Alarms
#define AoutFL                      0x0004,&My_Alarms
#define BoutFL                      0x0008,&My_Alarms
#define ChamberA_Alarm              0x0010,&My_Alarms
#define ChamberB_Alarm              0x0020,&My_Alarms
#define TemperatureA_alarm      0x0040,&My_Alarms
#define TemperatureB_alarm      0x0080,&My_Alarms
#define CriticalA_Alarm             0x0100,&My_Alarms       ;0 temperature is over high limit
#define CriticalB_Alarm             0x0200,&My_Alarms       ;1 temperature is under low limit
#define WindowA_Flag                0x0400,&My_Alarms
#define WindowB_Flag                0x0800,&My_Alarms
#define TimeToInformAlarm  0x1000,&My_Alarms  ;



#define _TempStatus             0x0001,&_Flags          ;
#define _OutFL                      0x0004,&_Flags      ;
#define _ChamberAlarm               0x0010,&_Flags      ;
#define _TemperatureAlarm  0x0040,&_Flags          ;
#define _CriticalAlarm              0x0100,&_Flags      ;
#define _WindowFlag             0x0400,&_Flags          ;
#define RS485CF                     0x0800,&_Flags      ;

InBufferSize    EQU     100
;------------------ Conditions --------------------
Processing      EQU     1
New             EQU     2
Fixed           EQU     3
```

```
Bad             EQU     4
Critical        EQU     5
Periodical      EQU     6
;----------------- Reasons ------------------------
Informing               EQU     '0'
Aknown                  EQU     '1'
Door            EQU     '2'
Electricity     EQU     '3'

;--------- Processing Methods -----------
TimeTemperatureIntegral     equ     1
WindowCalculation           equ     2
Level                   equ     3
HumidityIntegral            equ     4
HumidityWindow              equ     5
;-------- Incoming Instructions ---------
GetSettings             EQU     1
RequestInformations         EQU     2
SettingsAndInformations     EQU     3
;--------- Answer State -----------------
BadCRC                      EQU     1
BADUidPassword              EQU     2
Confirmed                   EQU     3
;------  Timeouts ------------------
;----------- Transmit --------------
t80ms           equ     873
t112ms                  equ     1222
t250ms                  equ     2730
t500ms                  equ     5461
t1s             equ     10922
t2s             equ     21845
t3s             equ     32768
t4s             equ     43689
t5s             equ     54611
t6s             equ     65535
;------------ Receive --------------
r80ms           equ     291
r250ms                  equ     910
r500ms                  equ     1820
r1500ms                 equ     5460
r1s             equ     3640
r2s             equ     7281
r3s             equ     10922
r4s             equ     14563
r5s             equ     18204
r6s             equ     21845
r7s             equ     25485
r8s             equ     29125
r9s             equ     32768
```

```
r10s            equ     36408
r11s            equ     40048
r12s            equ     43688
r13s            equ     47328
r14s            equ     50968
r15s            equ     54610
;----- Field Locations--------------------------------
UnitID                  EQU     0
PasswordOld             EQU     1
CurrentPassword                 EQU     2
TerminalUsed            EQU     3
TerminalPrefix          EQU     4
FirstCallingNumber              EQU     5
SecondCallingNumber             EQU     6
ThirdCallingNumber              EQU     7
RingCounts              EQU     8
ServiceTelephone                EQU     9
ServiceMobile           EQU     10
AutoCallService                 EQU     11
AutoCallUser            EQU     12
InformPeriod                    EQU      13
NumberOfChambers                EQU     14

FirstChamberCode                EQU     15
FirstChamberUsed                EQU     16
FirstLowLimit           EQU     17
FirstHighLimit          EQU     18
FirstIntegralFactor             EQU     19      ;
FirstWindowWidth                EQU     20      ;
FirstIntegralNew                EQU     21      ;
FirstIntegralCritical   EQU     22      ;
FirstIntegralBad                EQU     23      ; Methode (1:Temp Integral 2: Temp Window 3:Level
4:Humidity Integral 5:Humidity Window)
FirstProcessingMethode          EQU     24

SecondChamberCode               EQU     25
SecondChamberUsed               EQU     26
SecondLowLimit                  EQU     27
SecondHighLimit                 EQU     28
SecondIntegralFactor EQU        29
SecondWindowWidth               EQU     30
SecondIntegralNew               EQU     31
SecondIntegralCritical EQU      32
SecondIntegralBad               EQU     33
SecondProcessingMethode EQU     34

;-------- RAM Definitions ----------------

                ORG     0x200
```

```
        My_Flags:                   DS      2
        My_Flags1:          DS      2
        My_Alarms:          DS      2
        My_Alarms1:         DS      2
        TEMP:               DS      2
        TemperatureA_Timer:                 DS      2
        TemperatureB_Timer:                 DS      2
        HookTimer:          DS      2
        NextCallTimer:              DS      2
Field:          DS      2
        Value_hex:          DS      2
        Inform_Timer:       DS      2
        Inform_Period:              DS      2
        Delay:              DS      2
    IntegralA_LimitNew:     DS      2
        IntegralA_LimitCritical:    DS      2
        IntegralA_LimitBad:         DS      2
        IntegralB_LimitNew:         DS      2
        IntegralB_LimitCritical:    DS      2
        IntegralB_LimitBad:         DS      2
        IntegralA:          DS      2
        IntegralB:          DS      2       ;44 bytes
        _Integral:          DS      2
        _Integral_LimitNew:         DS      2
        _Integral_LimitCritical:    DS      2
        _Integral_LimitBad:         DS      2
        _Temperature_LowLimit:      DS      2
        _Temperature_HighLimit:     DS      2
        _Temperature:               DS      2
        _Condition:         DS      2
        _Flags:             DS      2
        _Temp:                      DS      2
        _Temperature_Timer:         DS      2
        _Window_Time:               DS      2
        _ChamberTime2:              DS      2
        _ChamberTime3:              DS      2
        _Window_Level:              DS      2
        _TemperatureAOld:           DS      2
        _TemperatureBOld:           DS      2
        temp_conv:          DS      2

        ElectricityTimer:           DS      2       ;82 Bytes

        Door1Integral:      DS      2
        Door2Integral:      DS      2
        TempDec:                    DS      2       ;88

        TemperatureA:               DS      2
        TemperatureA_LowLimit:      DS      2
```

```
        TemperatureA_HighLimit:     DS      2
        ChamberA_Procedure:                 DS      1
        WindowA_Level:              DS      2
        ChamberA_Condition:                 DS      1

        TemperatureB:               DS      2
        TemperatureB_LowLimit:      DS      2
        TemperatureB_HighLimit:     DS      2
        ChamberB_Procedure:                 DS      1
        WindowB_Level:              DS      2
        ChamberB_Condition:         DS      1

        Condition:              DS      1


        ByteToSend:   DS      1
        Checksum485:DS      1
        Seconds_ring: DS      1
        InputInstruction:       DS      1
        Seconds:                DS      1

        Send_digit:     DS      1
    FieldCounter:     DS      1
    Tmp:            DS      1
        Watch_Timer: DS      1

        GeneralCounter:         DS      1
        TempSeconds:DS      1       ;110 Bytes
        AnswerState:  DS      1       ;111 Bytes
        ChamberCode:            DS      1       ;116 Bytes
        Input_buffer:  DS      InBufferSize    ;216 bytes reserved finaly
        ValueBuffer:   DS      10



Temp_Buffer         EQU     Input_Buffer

TimeA_LimitNew              EQU     IntegralA_LimitNew
TimeA_LimitCritical         EQU     IntegralA_LimitCritical
TimeA_LimitBad              EQU     IntegralA_LimitBad
TimeB_LimitNew              EQU     IntegralB_LimitNew
TimeB_LimitCritical         EQU     IntegralB_LimitCritical
TimeB_LimitBad              EQU     IntegralB_LimitBad

        ORG     0x1000
Info_Table
    DB      0x00,0x00,0x0b      ;Uid
    DB      0x00,0x00,0xb0    ;PassOld
```

```
DB      0x00,0x0b       ;PassNow
DB      0x1b            ;TerminalUsed?
DB      0x0b,0x00               ;Prefix
DB      0x0b            ;FirstCN
DB      0x0b            ;SecondCN
DB      0x0b            ;ThirdCN
DB      0x5b            ;RingCounts
DB      0x0b            ;ServPhone
DB      0x0b            ;ServMobile
DB      0x0b            ;AutoCallService
DB      0x0b            ;AutoCallUser
DB      0x10,0x0b       ;InformPeriod
DB      0x2b            ;NoOfChambers

DB      0x1b            ;FirstChamberCode
DB      0x1b            ;FirstChamberUsed?
DB      0xe6,0x4b               ;FirstLowLimit
DB      0x63,0xb0               ;FirstHighLimit
DB      0x0b            ;FirstIntegralFactor
DB      0x0b            ;FirstWindowWidth
DB      0x10,0x0b       ;FirstIntegralNew
DB      0x20,0x0b       ;FirstIntegralCritical
DB      0x30,0x0b       ;FirstIntegralBad
DB      0x1b            ;FirstProcessingMethode

DB      0x2b            ;SecondChamberCode
DB      0x1b            ;SecondChamberUsed?
DB      0xe6,0x4b               ;SecondLowLimit
DB      0x63,0xb0               ;SecondHighLimit
DB      0x0b            ;SecondIntegralFactor
DB      0x0b            ;SecondWindowWidth
DB      0x40,0x0b       ;SecondIntegralNew
DB      0x50,0x0b       ;SecondIntegralCritical
DB      0x60,0x0b       ;SecondIntegralBad
DB      0x1b            ;SecondProcessingMethode
```

ΚΩΔΙΚΑΣ

4.TERMINAL ΚΩΔΙΚΑΣ ASSEMBLY

```
        #include  "msp430x12x2.h"
        #include  "zisisTerminal.h"
;Version 16 - Alarming Algorithm and send alarming info Included also Critical & Bad Included (OK)

;*******************************************************************************
******************
        ORG     0xE000
ROMUnitID   DB      '*',0x76
        DB      ',12345&',0     ;ROM Unit ID
main
RESET   dint
        mov     #0x300,sp
        mov     #WDTPW+WDTHOLD,&WDTCTL
      BIS.B     #ENABLE_POWER_DIR
        BIS.B   #ENABLE_POWER
        call    #delay_2sec
        call    #Get_Limits
      mov       #0,&my_flags
        mov     #0,&my_flags1
        mov     #0,&my_alarms
        mov     #0x0000,&IntegralA
        mov     #0x0000,&IntegralB
        clr     &Door1Integral
        clr     &Door2Integral
        mov     #0,&ElectricityTimer
      mov       #0,&TemperatureA_Timer
      mov       #0,&TemperatureB_Timer
        mov.b   #59,&TempSeconds
        mov     #0,&Inform_Timer
        mov.b   #0,&Seconds
        mov.b   #Fixed,&ChamberA_Condition
        mov.b   #Fixed,&ChamberB_Condition

        call    #ports_init
        bis.b   #DTMF_SENSOR_POWER
        call    #ReadTemperatures
        mov     &TemperatureA,&_TemperatureAOld
        mov     &TemperatureB,&_TemperatureBOld
        bic.b   DTMF_SENSOR_POWER
Start   dint
        bis.b   #ENABLE_POWER
        mov.b   #dco2+dco1+dco0,&dcoctl
        mov.b   #RSEL2+RSEL1+RSEL0+XT2OFF,&BCSCTL1
        mov     #0x300,sp
```

```
        call    #delay_1sec
        mov.b   #0,&Watch_Timer

        CALL    #timer_init
        bic.w    #hook_mode
        call    #Get_Limits
;*******************************************************************************
;       Main Loop Here
;*******************************************************************************
MainLp          mov    #0x300,sp
        call    #SetupSCI
        bic     #Hook_Mode
        bic.b   #RS485_Control
        eint
wmd     bit     #AskFromModemFL
        jz      wmd
        dint
        bic     #AskFromModemFL
        bic     #AskInF
        cmp.b   #0x67,&Input_Buffer
        jeq     AskStatus
        cmp.b   #0x85,&Input_Buffer
        jeq     SettAs
        cmp.b   #0xd7,&Input_Buffer
        jne     MainLp
        bis     #AskInF
        jmp     AskSt
SettAs  call    #SettingsReceived
        jmp     Reset
AskStatus       bis     #Hook_Mode
        bit     #ChamberA_Alarm
        jnz     AskSt
        bit     #ChamberB_Alarm
        jnz     AskSt
        bit     #TimeToInformAlarm
        jnz     AskSt
        jmp      MainLp
AskSt   bis     #Hook_Mode
        bis.b   #RS485_Control
        call    #delay80ms
        call    #delay80ms
        call    #Delay80ms
        call    #Delay80ms
        call    #delay80ms
        call    #SendToModem
        call    #delay80ms
        bic.b   #RS485_Control
        call    #delay80ms
```

```
        call    #GetConfirmFromModem
        bic     #Hook_Mode
        call    #delay80ms
        call    #delay80ms
      jmp     MainLp

;*******************************************************************************
;       End of Main Loop
;*******************************************************************************
GetConfirmFromModem
        call    #SetupSCINoInterrupt
        clr.b   &Checksum485
        mov     #Input_Buffer,RS485RxPointer
GtNc    call    #GetTimedSerialChar
        cmp.b   #'*',&RxBuf0
        jne     GtNc
        add.b   &RXBUF0,&Checksum485
        mov.b   &RxBuf0,0(RS485RxPointer)
        inc     RS485RxPointer
GetAlCh         call    #GetTimedSerialChar
        mov.b   &RxBuf0,0(RS485RxPointer)
        inc     RS485RxPointer
        add.b   &RXBUF0,&Checksum485
        cmp.b   #'#',&RxBuf0
        jeq     GetCH
        cmp     #Input_Buffer+10,Rs485RxPointer
        jhs     AlConfErr
        jmp     GetAlCh
GetCH   call    #GetTimedSerialChar
        add.b   &RXBUF0,&Checksum485
        cmp.b   #0x00,&Checksum485
        jne     AlConfErr
        cmp.b   #0x29,&Input_Buffer+1
        jne     AlConfErr
        if ElectricityAlg
        bit     #ElectricityHelpFL
        jz      NoSel
        bic     #ElectricityAlarm
        endif
NoSel   bit     #AskInF
        jnz     AlConfErr
        bit     #ChamberA_Alarm
        jnz     ClrAlrm
        bit     #ChamberB_Alarm
        jnz     ClrAlrm
        bic     #TimeToInformAlarm
        clr.b   &Seconds
        clr     &Inform_Timer
        clr     &Door1Integral
```

```
        clr     &Door2Integral
        ret
ClrAlrm         bic     #ChamberA_Alarm
        bic     #ChamberB_Alarm
        ret
AlConfErr
        ret
;*******************************************************************************

SettingsReceived
        call    #DecodeFrame
        call    #ram2info
        call    #Get_Limits
        bis.b   #RS485_Control
        call    #delay80ms
        call    #delay80ms
        call    #delay80ms
        call    #Delay80ms
        call    #Delay80ms
        call    #SendConfirmToModem
        ret
;*******************************************************************************
SendConfirmToModem      clr.b   &Checksum485
                call    #SetupSCI
                mov.b   #'*',&ByteToSend
                call    #SendSerialByte
                mov.b   #0x14,&ByteToSend
                call    #SendSerialByte
                mov.b   #',',&ByteToSend
                call    #SendSerialByte
                mov     #ROMUnitID+3,RS485TXPointer
SnbR485a                mov.b   @RS485TxPointer+,&ByteToSend
                call    #SendSerialByte
                cmp.b   #'&',0(RS485TxPointer)
                jne     SnbR485a
                mov.b   #',',&ByteToSend
                call    #SendSerialByte
                call    #Load_CurrentPassword
                mov     #Temp_Buffer,RS485TxPointer
SnbR485b                mov.b   @RS485TxPointer+,&ByteToSend
                bis.b   #0x30,&ByteToSend
                call    #SendSerialByte
                cmp.b   #'*',0(RS485TxPointer)
                jne     SnbR485b
                mov.b   #',',&ByteToSend
                call    #SendSerialByte
                mov.b   #'#',&ByteToSend
                call    #SendSerialByte
                push.b &temp
```

```
                mov.b  #0x00,&temp
                sub.b   &Checksum485,&temp
                mov.b  &Temp,&Checksum485
                pop.b  &Temp
                mov.b    &Checksum485,&ByteToSend
                call     #SendSerialByte
                call      #delay80ms
                ret

;*******************************************************************************
******************
SetupSCI
SetupP3         bis.b    #030h,&P3SEL
        bis.b    #UTXE0+URXE0,&ME2
        bis.b    #CHAR,&UCTL0
        bis.b    #SSEL0,&UTCTL0
        mov.b  #27,&UBR00
        mov.b  #0,&UBR10
        mov.b  #000h,&UMCTL0
        bic.b    #SWRST,&UCTL0
        bis.b    #URXIE0,&IE2
        bis.b    #bit6,&p3dir
        ret
SetupSCINoInterrupt
        bis.b    #030h,&P3SEL
        bis.b    #UTXE0+URXE0,&ME2
        bis.b    #CHAR,&UCTL0
        bis.b    #SSEL0,&UTCTL0
        mov.b  #27,&UBR00
        mov.b  #0,&UBR10
        mov.b  #000h,&UMCTL0
        bic.b    #SWRST,&UCTL0
        bic.b    #URXIE0,&IE2
        bis.b    #bit6,&p3dir
        ret
;-------------------------------------------------------------------------------------------------
SendToModem         call      #SetupSCI
                mov     #ROMUnitID,RS485TXPointer
                clr.b     &Checksum485
SnbR485                 mov.b  @RS485TxPointer,&ByteToSend
                inc       RS485TxPointer
                call      #SendSerialByte
                cmp.b  #'&',0(RS485TxPointer)
                jne       SnbR485
                mov.b  #',',&ByteToSend
                call      #SendSerialByte

                call      #Load_CurrentPassword
                mov     #Temp_Buffer,RS485TXPointer
```

```
chkEOP          cmp.b  #'*',0(RS485TxPointer)
                jne    ConTPn
                jmp    EoP
ConTPn          mov.b  @RS485TxPointer+,&ByteToSend
                bis.b  #0x30,&ByteToSend
                call   #SendSerialByte
                jmp    chkEOP
EoP             mov.b  #',',&ByteToSend
                call   #SendSerialByte
;Check_ChamberA
                bit    #FirstUsedFl
                jz     ChamberB
                mov.b  #'1',&ByteToSend
                bit    #AskInF
                jz     ToALa
                jmp    EopA
ToALa           bit    #ChamberA_Alarm
                jz     EopA
                mov.b  &ChamberA_Condition,&Condition
                jmp    EopB
EopA            mov.b    #Periodical,&Condition
EopB            mov    &TemperatureA,&Temp
                bic    #HumFL
                if ElectricityAlg
                bit    #ElectricityAlarm
                jz     Lda1
                mov.b  #Critical,&Condition
                bis    #ElectricityHelpFL
                endif
Lda1            call   #Send_Details
                call   #SendReasonA
;Check_ChamberB
ChamberB        bit      #SecondUsedFl
                jz     FinishData
                mov.b  #'2',&ByteToSend
                bit    #AskInF
                jz     ToALb
                jmp    EopC
ToALb           bit    #ChamberB_Alarm
                jz     EopC
                mov.b  &ChamberB_Condition,&Condition
                jmp    EopD
EopC            mov.b    #Periodical,&Condition
EopD            mov    &TemperatureB,&Temp
                bic    #HumFL
                bit    #HumidityUsedFL
                jz     ContUm

                bis    #HumFL
```

```
ContUm
                if ElectricityAlg
                bit     #ElectricityAlarm
                jz      Lda2
                mov.b  #Critical,&Condition
                bis     #ElectricityHelpFL
                endif
Lda2            call    #Send_Details
                call    #SendReasonB

; For More Chambers Enter Code Here
;Check_NextChamber
FinishData
                mov.b  #'#',&ByteToSend
                call    #SendSerialByte
                push.b &temp
                mov.b  #0x00,&temp
                sub.b   &Checksum485,&temp
                mov.b  &Temp,&Checksum485
                pop.b   &Temp
                mov.b    &Checksum485,&ByteToSend
                call    #SendSerialByte
                ret
;-------------------------------------------------------------------------------------------------
Send_Details
;Input: [Alarmed Chamber,Condition,Temperature,Reason,Description]
        call    #SendSerialByte
        mov.b  #',',&ByteToSend
        call    #SendSerialByte
        mov.b  &Condition,&ByteToSend
        bis.b   #0x30,&ByteToSend
        call    #SendSerialByte
        mov.b  #',',&ByteToSend
        call    #SendSerialByte
; Send Chamber Temperature
        call    #Value_Hex2Dec
        mov     #ValueBuffer,r9
LLght   cmp.b  #'@',0(r9)
        jeq     ContSndd
        inc     r9
        jmp     LLght
ContSndd        inc     r9
ContSndda       mov.b  @r9+,&ByteToSend
        call    #SendSerialByte
        cmp.b  #'*',0(r9)
        jne     ContSndda
Snd_Al_5        mov.b  #',',&ByteToSend
        call    #SendSerialByte
        ret
```

```
GetTimedSerialChar  mov.w #CCIE,&CCTL0
                    mov   &tar,&Taccr0
                    add   #33,&Taccr0
                    mov   #1500,&Delay
                    bic   #RxTFL
CheckSer                  bit.b   #URXIFG0,&IFG2
                    jz    CheckTim
                    mov.w #0,&CCTL0
                    ret
CheckTim                  bit   #RxTFL
                    jz    CheckSer
                    mov.w #0,&CCTL0
                    ret
;*******************************************************************************
******************
SCITX_ISR
                    reti

SCIRX_ISR                 bit   #SerialHelpFL
                    jz    CHStr
                    add.b &RXBUF0,&Checksum485
                    cmp.b #0xff,&Checksum485
                    jne   EX4
                    call  #CompareUID

                    jmp   EX4
CHStr               bit   #RS485CF
                    jz    CheckStart
                    mov.b &RXBUF0,0(RS485RxPointer)
                    inc   RS485RxPointer
                    add.b &RXBUF0,&Checksum485
                    cmp.b #'#',&RXBUF0
                    jne   SciExit
                    bis   #SerialHelpFL
                    jmp   SciExit
CheckStart          cmp.b #'*',&RXBUF0
                    jne   EX4
                    mov   #Input_Buffer,RS485RxPointer
                    bis   #RS485CF
                    clr.b &Checksum485
                    add.b &RXBUF0,&Checksum485
                    bic   #SerialHelpFL
SciExit             reti
EX4                 bic   #RS485CF
                    bic   #SerialHelpFL
                    reti
;-------------------------------------------------------------------------------------------------
CompareUID  cmp.b #0x85,&Input_Buffer
                    jeq   UIDsettings
```

```
                cmp.b  #0xd7,&Input_Buffer
                jeq    UIDsettings
                mov    #Input_Buffer+2,CompA
                mov    #ROMUnitID+3,CompB
NCUId           cmp.b  @CompA+,0(CompB)
                jne    UIDerr
                inc    CompB
                jmp    NCUId
UIDerr          cmp    #ROMUnitID+8,CompB
                jne    UIDer
                bis    #AskFromModemFL
                ret
UIDer           bic    #AskFromModemFL
                ret

UIDsettings     mov    #Input_Buffer+2,CompA
                mov    #ROMUnitID+3,CompB
NCUIda                 bis.b  #0x30,0(CompA)
                cmp.b  @CompA+,0(CompB)
                jne    UIDerra
                inc    CompB
                jmp    NCUIda
UIDerra                cmp    #ROMUnitID+8,CompB
                jne    UIDera
                bis    #AskFromModemFL
                ret
UIDera          bic    #AskFromModemFL
                ret

;*******************************************************************************
******************
DecodeFrame
        mov     #input_buffer+2,r4
          mov   #input_Buffer,r5
wait_cha        mov.b  @r4+,&tmp
        cmp.b   #'#',&tmp
          jeq   Get_CRC
        rlc.b   &tmp
        rlc.b   &tmp
        rlc.b   &tmp
        rlc.b   &tmp
          cmp.b #'#',0(r4)
          jeq   Get_CRC
          and.b #0xf0,&tmp
          and.b #0x0f,0(r4)
          bis.b @r4+,&tmp
        mov.b   &tmp,0(r5)
          inc   r5
          jmp   wait_cha
```

```
crc_error
        ret
;*******************************************************************************
******************
Get_Crc
       ret

CheckUID_Password

CuDv   cmp.b  #0x00,&Info_Table+3
       jne    CUiN1          ;
       cmp.b  #0x00,&Info_Table+4
       jne    CuiN1          ;
       mov    #Input_Buffer+70,r5

       mov.b  &Input_Buffer+3,&temp
       and.b  #0xf0,&temp
       cmp.b  #0xb0,&temp
       jeq    MaG
       ret

MaG    mov.b  @r5,2(r5)
       dec    r5
       cmp    #Input_Buffer+2,r5
       jne    MaG

       call   #CopyNewToOld

       ret

CUiN1  ret
       bic    #Help_FlagC
       mov    #Input_Buffer,r4
       mov    #Info_Table,r5
CUii   bit    #Help_FlagC
       jz     CUvi
       cmp.b  @r4,0(r5)
       jne    CUiv
CUvi   inc    r5
       inc    r4
       mov.b  -1(r5),&Temp
       and.b  #0x0f,&temp
       cmp.b  #0x0b,&Temp
       jeq    CUi
       mov.b  -1(r5),&temp
       and.b  #0xf0,&Temp
       cmp.b  #0xb0,&temp
       jeq    CUi
       jmp    CUii
```

```
CUi     bit     #Help_FlagC
        jnz     CUv
        bis     #Help_FlagC
        jmp     CUii
CUiii   mov.b   #BadUidPassword,&AnswerState
        ret
CUiv    bit     #Help_FlagC
        jz      CUiii
        mov.b   0(r5),&temp
        and.b   #0xf0,&Temp
        cmp.b   #0xb0,&temp
        jne     CUiii
CUv     cmp.b   #RequestInformations,&InputInstruction
        jeq     CUvE
        call    #CopyNewToOld
CUvE    ret

CopyNewToOld
        mov     #input_buffer,r5
        mov.b   5(r5),3(r5)
        mov.b   6(r5),4(r5)
        rlc.b   4(r5)
        rlc.b   3(r5)
        rlc.b   4(r5)
        rlc.b   3(r5)
        rlc.b   4(r5)
        rlc.b   3(r5)
        rlc.b   4(r5)
        rlc.b   3(r5)
        mov.b   7(r5),&temp
        rrc.b   &temp
        rrc.b   &temp
        rrc.b   &temp
        rrc.b   &temp
        and.b   #0x0f,&temp
        and.b   #0xf0,&Input_Buffer+4
        bis.b   &temp,&Input_Buffer+4
        ret
;_____________________________________________________________________________
_____________________
PORT_1_ISR
        bit.b   #Ring_Pin,&P1IFG
        jz      Not_RingInt
        reti
Not_RingInt
        RETI
;-------------------------------------------------------------------------------
;*******************************************************************************
******************
```

```
CCR2_ISR  ADD         #32768,&CCR2
          inc.b    &seconds
          cmp.b  #70,&Seconds
          jlo       CFHC
          clr.b    &Seconds
CFHC   ;eint
          bit       #Hook_Mode
          jz        ON_HOOK_MODE
OFF_HOOK_MODE
        inc         &HookTimer
        cmp       #hook_time,&HookTimer
        jhs         enof
          inc.b    &Watch_Timer
          cmp.b  #Watch_Time,&Watch_Timer
          jhs       enof
          reti
enof     mov    #0,&NextCallTimer
End_Of_Communication
          dint
          br        #start
;-------------------------------------------------------------------------------------------
ON_HOOK_MODE
          call      #InformAlgorithm
          if ElectricityAlg
          call      #ElectricityAlgorithm
          endif
          call      #Measure_and_Process_Algorithm
          if DoorsAlg
          call      #DoorsAlgorithm
          endif
          if RelaysAlg
          call      #Relays_Set
          endif
          reti
;*************************************************************************************
******************
          if DoorsAlg
DoorsAlgorithm
          bit.b    #door1
          jz        Lfll
          inc       &Door1Integral
          cmp     #0x270f,&Door1Integral
          jlo       Lfll
          mov     #0x0270e,&Door1Integral
Lfll      bic       #Door1FL
          cmp     #300,&Door1Integral
          jlo       Lfl2
          bis       #Door1FL
Lfl2      bit.b    #door2
```

```
        jz      Lflt
        inc     &Door2Integral
        cmp     #0x270f,&Door2Integral
        jlo     Lflt
        mov     #0x270e,&Door2Integral
Lflt    bic     #Door2FL
        cmp     #300,&Door2Integral
        jlo     Lflr
        bis     #Door2FL
Lflr    ret
        endif
;*******************************************************************************
******************
        if RelaysAlg
Relays_Set
        cmp.b  #fixed,&ChamberA_Condition
        jne     RelA
        bic.b   #RelayA
        jmp     sRelayB
RelA    bis.b   #RelayA
sRelayB         cmp.b  #fixed,&ChamberB_Condition
        jne     RelB
        bic.b   #RelayB
        jmp     RelExit
RelB    bis.b   #RelayB
RelExit ret
        endif
;--------------  Electricity Algorithm ----------------------------------------------------------
        if ElectricityAlg
ElectricityAlgorithm
        inc     &ElectricityTimer
        cmp     #0xfffe,&ElectricityTimer
        jne     EL_A
        dec     &ElectricityTimer
EL_A    call    #Read_ElectricityVoltage
        cmp     #770,&Temp              ;2.25V Volts
        jlo     LowVoltage
        bit     #ElectricityAlarmFL
        jz      NoELAL
        bis     #TimeToInformAlarm
NoELAL          bic     #ElectricityAlarmFL
        clr     &ElectricityTimer
        ret
LowVoltage
        cmp     #(3*60),&ElectricityTimer
        jlo     ELNoAL
        bit     #ElectricityAlarmFL
        jnz     ELNoAL
        bis     #ElectricityAlarm
```

```
        bis     #ElectricityAlarmFL
        bis     #TimeToInformAlarm
ELNoAL          ret
        endif
;-------------------------------------------------------------------------------------------------
Measure_and_Process_Algorithm
        mov.b   &TempSeconds,&tmp
        and.b   #0x07,&tmp
        cmp.b   #0x04,&tmp
        jne     MAPb
        call    #ReadTemperatures
        clr     r14
        clr     r15
        mov     &_TemperatureAOld,r14
        mov     &TemperatureA,r15
        add     r14,r15
        rra     r15
        mov     r15,&TemperatureA

        clr     r14
        clr     r15
        mov     &_TemperatureBOld,r14
        mov     &TemperatureB,r15
        add     r14,r15
        rra     r15
        mov     r15,&TemperatureB

MAPb    inc.b   &TempSeconds
        cmp.b   #60,&TempSeconds
        jlo     MAPA
        clr.b   &TempSeconds

        inc     &TemperatureA_Timer
        inc     &TemperatureB_Timer
        call    #Process_TemperatureA
        call    #Process_TemperatureB
        call    #ReadTemperatures
        mov     &TemperatureA,&_TemperatureAOld
        mov     &TemperatureB,&_TemperatureBOld
MAPA    inc     &NextCallTimer
        ret
;-------------------------------------------------------------------------------------------------
ReadTemperatures
        bis.b   #DTMF_SENSOR_POWER
        push    r15
        mov     #1000,r15
LLes    dec     r15
        jnz     LLes
        pop     r15
```

```
        call    #read_temperatureA
        bit     #HumidityUsedFL
        jz      RdTmp
RHum    call    #Read_Humidity
        jmp     RdExit
RdTmp   call    #read_temperatureB
RdExit  bis.b   #DTMF_SENSOR_POWER
        ret
;-------------------------------------------------------------------------------------------------
InformAlgorithm
        cmp     #0,&Inform_Period
        jeq     Ccr2_N3
        cmp     #0xffff,&Inform_Timer
        jeq     Cont_CCR23
        cmp.b   #60,&Seconds
        jlo     CCR2_N3
        clr.b   &Seconds
        inc     &Inform_Timer
Cont_CCR23
        cmp     &Inform_Period,&Inform_Timer
        jlo     ccr2_n3
        bis     #TimeToInformAlarm
        bit     #CallWait
        jnz     CCR2_N3
        mov     #0,&Inform_Timer
Ccr2_N3         ret
;-------------------------------------------------------------------------------------------------
delay80ms
        push    temp
        mov     #5,temp
t16ms_again
        mov.w #WDT_ADLY_16,&WDTCTL
Test_16fg
        bit.b   #WDTIFG,&IFG1
        jz      Test_16fg
        mov     #WDTPW+WDTHOLD+WDTCNTCL,&WDTCTL
        bic.b   #WDTIFG,&IFG1
        dec     temp
        jnz     t16ms_again
        pop     temp
        RET
;-------------------------------------------------------------------------------------------------
delay_2sec
        call    #Delay_1sec
        call    #Delay_1sec
        ret
;.....................................................................................
delay_1sec
        mov.w #WDT_ADLY_1000,&WDTCTL
```

```
Test_WDTIFG
        bit.b   #WDTIFG,&IFG1
        jz      Test_WDTIFG
        mov     #WDTPW+WDTHOLD+WDTCNTCL,&WDTCTL
        bic.b   #WDTIFG,&IFG1
        ret
FieldDec2Hex
      push    sr
      dint
        push   r14
        push   temp
        push   counter
      clr     &value_hex
      clr     &temp_conv
      mov     #temp_Buffer,r14
        bic     #help_flag
conv_start_t1
        cmp.b  #0x0e,0(r14)
        jne     cont_conv
        bis     #help_flag
        inc     r14
cont_conv       mov.b  @r14+,temp_conv
        cmp.b  #'*',0(R14)
        jeq     dec2end_t
        and     #0x000f,&temp_conv
        call    #mul_10
        jmp     conv_start_t1
dec2end_t       and     #0x000f,&temp_conv
        add     &temp_conv,&Value_hex
        bit     #help_flag
        jz      exit_conv
        inv     &Value_hex
        inc     &Value_hex
exit_conv       pop     counter
        pop     temp
        pop     r14
      pop     sr
        ret
mul_10          mov    #9,counter
        add     &temp_conv,&Value_hex
        mov    &Value_hex,&temp
mul_11          add     &temp,&Value_hex
        dec     counter
        jnz     mul_11
        ret
;*****************************************************************************
*****************

;-------------------------------------------------------------------------------
CLEAR_RAM
```

```
        MOV   #200H,R10
CR1     MOV   #0,0(R10)
        INCD  R10
        CMP   #2faH,R10
        JLO   CR1
        RET
;-------------------------------------------------------------------------------------------------
PORTS_INIT
        MOV.B #0FFH,&P3DIR
        MOV.B #0,&P3OUT
        BIS.B #ENABLE_POWER_dir
        BIS.B #ENABLE_POWER
        BIS.B #DTMF_SENSOR_POWER_dir
        BIS.B #DTMF_SENSOR_POWER
        BIS.B #ENABLE_GYRO_dir
        BIC.B #ENABLE_GYRO

        bis.b #BIT7+BIT5,&p1dir
        bic.b #BIT7+BIT5,&p1out
        bic.b #Ring_Pin,&P1DIR
        bis.b #Ring_Pin,&P1IES
        bic.b #Ring_Pin,&P1IFG
        bic.b #Door1Dir
        bic.b #Door2Dir

        RET

;_____________________________________________________________________________________
_____________________
flash_write ; Write to Flash new Settings
;_____________________________________________________________________________________
_____________________
ram2info        push    sr
      dint              ; Disable Interrupts
        call    #info_copy
        mov     #fwkey+fssel1+fn0+fn1+fn2+fn3,&fctl2
        mov     #fwkey,&fctl3
        mov     #fwkey+erase,&fctl1
        clr     0x1000
        mov     #fwkey+lock,&fctl3

        mov     #fwkey+fssel1+fn0+fn1+fn2+fn3,&fctl2
        mov     #fwkey,&fctl3
        mov     #fwkey+wrt,&fctl1

        mov     #0x1000,r11
        mov     #Input_Buffer,r10
flwr1a  mov.b   @r10+,0(r11)
        inc     r11
```

```
        cmp     #0x107f,r11
        jlo     flwr1a
flwrb1a         bit     #busy,&fctl3
        jnz     flwrb1a
        mov     #fwkey+lock,&fctl3
      pop     sr
        ret
;_____________________________________________________________________________
_____________________
info_copy
        mov     #fwkey+fssel1+fn0+fn1+fn2+fn3,&fctl2
        mov     #fwkey,&fctl3
        mov     #fwkey+erase,&fctl1
        clr     &0x1080
        mov     #fwkey+lock,&fctl3

        mov     #fwkey+fssel1+fn0+fn1+fn2+fn3,&fctl2
        mov     #fwkey,&fctl3
        mov     #fwkey+wrt,&fctl1

        mov     #0x1080,r11
        mov     #0x1000,r10
flwr1   mov.b  @r10+,0(r11)
        inc     r11
        cmp     #0x10ff,r11
        jlo     flwr1
flwrb1 bit     #busy,&fctl3
        jnz     flwrb1
        mov     #fwkey+lock,&fctl3
        ret
;-------------------------------------------------------------------------------------------
TIMER_INIT
SetupTA         mov.w #TASSEL0+TACLR+TAIE,&TACTL
SetupC0         mov.w #0,&CCTL0
SetupC1         mov.w #0,&CCTL1
SetupC2         mov.w #CCIE,&CCTL2
        mov.w #9,&CCR0
        mov.w #3,&CCR1
        mov.w #32768,&CCR2
        bis.w   #MC1,&TACTL
        RET
;-------------------------------------------------------------------------------------------
TA0_ISR;
;-------------------------------------------------------------------------------------------
        add.w  #32,&CCR0
exit_ta0
        dec     &Delay
        jnz     TA0L1R
        bis     #RxTFL
```

```
TA0L1R          reti
;-------------------------------------------------------------------------------------------
TAX_ISR                                 ;   Common ISR for CCR1-4 and overflow
;-------------------------------------------------------------------------------------------
        add.w  &TAIV,PC
        reti
        jmp     CCR1_ISR
        jmp     CCR2_ISRA
        reti
        reti
TA_over
        reti
CCR2_ISRA      br       #CCR2_ISR
CCR1_ISR
        reti

;*****************************************************************************
******************
Read_temperatureA
        call     #Adc_mesurment_init_Temperature1
        push    r15
        mov     #1500,r15
L$$D    dec     r15
        jnz     L$$D
        pop     r15
      bis.w    #ADC10SC,&ADC10CTL0
k_end_tst       bit.w    #adc10ifg,&adc10ctl0
      jz      k_end_tst
      mov      &adc10mem,r11
        rra      r11
      rra        r11
      and        #0FFH,R11
        clr      &TemperatureA
        mov.b  Temp_table(r11),&TemperatureA
        sxt      &TemperatureA
        ret
;-------------------------------------------------------------------------------------------
Read_temperatureB
        call     #Adc_mesurment_init_Temperature2
        push    r15
        mov     #1000,r15
L$$E    dec     r15
        jnz     L$$E
        pop     r15
      bis.w    #ADC10SC,&ADC10CTL0
f_end_tst bit.w    #adc10ifg,&adc10ctl0
      jz      f_end_tst
        if        PT100=0
        mov      &adc10mem,r12
```

```
        rra     r12
        rra     r12
        and     #0FFH,R12
        clr     &TemperatureB
        mov.b   Temp_table(r12),&TemperatureB
        sxt     &TemperatureB
        endif
        if      PT100=1
        cmp     #405,&Adc10mem
        jlo     Lpt1
        mov     &Adc10mem,r11
        jmp     Lpt2
Lpt1    mov     #405,r11
Lpt2    sub     #405,r11
        clr     &TemperatureB
        mov.b   PT100Table(r11),&TemperatureB
        sxt     &TemperatureB
        rla     &TemperatureB
        endif
        ret
;-------------------------------------------------------------------------------------------
Read_Humidity
        call    #Adc_mesurment_init_Humidity
        push    r15
        mov     #1000,r15
L$$F    dec     r15
        jnz     L$$F
        pop     r15
      bis.w   #ADC10SC,&ADC10CTL0
end_tsta        bit.w   #adc10ifg,&adc10ctl0
      jz      end_tsta
        cmp     #221,&Adc10mem
        jhs     Lkio
        mov     #221,Multiplicant
        jmp     Lkio1
Lkio    mov      &adc10mem,Multiplicant
Lkio1   mov     #12460,LowProduct
        call    #Mul_16x16
        rlc     LowProduct
        rlc     HighProduct
        sub     #84,HighProduct
        mov     HighProduct,&TemperatureB
        cmp     #200,&TemperatureB
        jhs     L$$G
        jmp     HumX
L$$G    mov     #199,&TemperatureB
HumX    ret          ;
;.....................................................................................
mul_16x16       ;170 cpu cycles
```

```
        mov.b  #16,counter
        clr    HighProduct
againmclrc
        bit    #bit0,LowProduct
        jz     shift_product
        add    Multiplicant,HighProduct
shift_product
        clrc
        rrc    HighProduct
        rrc    LowProduct
        dec.b  Counter
        jne    againm
        ret
;-------------------------------------------------------------------------------------------------
Div16_16       ;204 cpu cycles
        clr    left_remainter
        mov.b  #16,Counter
        clrc
        rlc    right_remainter
        rlc    left_remainter

againd sub     divisor,left_remainter
        tst    left_remainter
        jl     restored
        clrc
        rlc    right_remainter
        rlc    left_remainter
        bis    #bit0,right_remainter
        jmp    no_restd
restored       add    divisor,left_remainter
        clrc
        rlc    right_remainter
        rlc    left_remainter
        bic    #bit0,right_remainter
no_restd       dec.b  Counter
        jne    againd
        clrc
        rrc    left_remainter
        ret

;-------------------------------------------------------------------------------------------------
Read_ElectricityVoltage
        call   #Adc_mesurment_init_Electricity
        bic    #adc10ifg,&adc10ctl0
      bis.w   #ADC10SC,&ADC10CTL0
E_end_tst bit.w   #adc10ifg,&adc10ctl0
      jz     E_end_tst
        mov    &adc10mem,&temp
        ret
```

```
;*******************************************************************************
******************
;-------------------------------------------------------------------------------------------
Adc_mesurment_init_Temperature1
        bic     #enc,&adc10ctl0
        mov.b    #02h,&ADC10AE
        mov     #inch_1,&ADC10CTL1
        mov     #ADC10SHT_3+ADC10ON,&ADC10CTL0
        bis     #ENC,&ADC10CTL0
        ret
Adc_mesurment_init_Humidity
        bic     #enc,&adc10ctl0
        mov.b  #04h,&ADC10AE
        mov     #inch_2,&ADC10CTL1
        mov     #ADC10SHT_3+ADC10ON,&ADC10CTL0
        bis     #ENC,&ADC10CTL0
        ret
Adc_mesurment_init_Temperature2
        bic     #enc,&adc10ctl0
        mov.b  #04h,&ADC10AE
        mov     #inch_2,&ADC10CTL1
        mov     #ADC10SHT_3+ADC10ON,&ADC10CTL0
        bis     #ENC,&ADC10CTL0
        ret
Adc_mesurment_init_Electricity
        bic.b   #bit4,&p2dir
        bic.b   #bit4,&p2out
        bic     #enc,&adc10ctl0
        mov.b  #0x10,&ADC10AE
        mov     #inch_4,&ADC10CTL1
        mov     #ADC10SHT_1+ADC10ON,&ADC10CTL0
        bis     #ENC,&ADC10CTL0
        ret
;_____________________________________________________________________________________
_____________________
ADC10_ISR         ; A/D interrupt service routine
;_____________________________________________________________________________________
_____________________
        reti
;-------------------------------------------------------------------------------------------
;*******************************************************************************
******************
Process_TemperatureA
        bit     #FirstUsedFl
        jz      AExit
        bic     #HumFL
      mov       &TemperatureA,&_Temperature
        mov     &TemperatureA_LowLimit,&_Temperature_LowLimit
```

```
        mov     &TemperatureA_HighLimit,&_Temperature_HighLimit
        mov.b   &ChamberA_Condition,&_Condition
        mov     &WindowA_Level,&_Window_Level
        rla     &_temperature_LowLimit
        rla     &_Temperature_HighLimit
        rla     &_Window_Level

        cmp.b   #TimeTemperatureIntegral,&ChamberA_Procedure
        jne     Awindow
        mov     &IntegralA,&_integral
        mov     &IntegralA_LimitNew,&_Integral_LimitNew
        mov     &IntegralA_LimitCritical,&_Integral_LimitCritical
        mov     &IntegralA_LimitBad,&_Integral_LimitBad
        mov     &My_Alarms,&_temp
        bic     #BIT1+BIT3+BIT5+BIT7+BIT9+BITB+BITC+BITD+BITE+BITF,&_temp
        mov     &_temp,&_Flags
                call    #TimeTemperature
        mov     &_Integral,&IntegralA
        mov.b   &_Condition,&ChamberA_Condition
        bic     #BIT0+BIT2+BIT4+BIT6+BIT8+BITA,&My_Alarms
        bis     &_Flags,&My_Alarms
        jmp     AExit

Awindow         mov     &TemperatureA_Timer,&_Temperature_Timer
        mov     &TimeA_LimitNew,&_Window_Time
        mov     &TimeA_LimitCritical,&_ChamberTime2
        mov     &TimeA_LimitBad,&_ChamberTime3
        mov     &My_Alarms,&_temp
        bic     #BIT1+BIT3+BIT5+BIT7+BIT9+BITB+BITC+BITD+BITE+BITF,&_temp
        mov     &_temp,&_Flags
                call    #WindowCalc
        mov.b   &_Condition,&ChamberA_Condition
        bic     #BIT0+BIT2+BIT4+BIT6+BIT8+BITA,&My_Alarms
        bis     &_Flags,&My_Alarms
        mov     &_Temperature_Timer,&TemperatureA_Timer

AExit   ret

;-------------------------------------------------------------------------------------------------
Process_TemperatureB
        bit     #SecondUsedFl
        jz      Bexit
        mov     &TemperatureB,&_Temperature
        mov     &TemperatureB_LowLimit,&_Temperature_LowLimit
        mov     &TemperatureB_HighLimit,&_Temperature_HighLimit
        mov.b   &ChamberB_Condition,&_Condition
        mov     &WindowB_Level,&_Window_Level
        bit     #HumidityUsedFL
        jnz     IsHm
```

```
        rla     &_temperature_LowLimit
        rla     &_Temperature_HighLimit
        rla     &_Window_Level

IsHm    cmp.b   #WindowCalculation,&ChamberB_Procedure
        jeq     Bwindow
        bit     #HumidityUsedFL
        jz      NitH
IsHum   bis     #HumFL
        clrc
        rrc     &_Temperature

NitH    mov     &IntegralB,&_integral
        mov     &IntegralB_LimitNew,&_Integral_LimitNew
        mov     &IntegralB_LimitCritical,&_Integral_LimitCritical
        mov     &IntegralB_LimitBad,&_Integral_LimitBad
        mov     &My_Alarms,&_temp
        bic     #BIT0+BIT2+BIT4+BIT6+BIT8+BITA+BITC+BITD+BITE+BITF,&_temp
        rra     &_Temp
        mov     &_temp,&_Flags
                call    #TimeTemperature
        mov     &_Integral,&IntegralB
        mov.b   &_Condition,&ChamberB_Condition
        bic     #BIT1+BIT3+BIT5+BIT7+BIT9+BITB,&My_Alarms
        rla     &_Flags
        bis     &_Flags,&My_Alarms
        jmp     BExit

BWindow         mov     &TemperatureB_Timer,&_Temperature_Timer
        mov     &TimeB_LimitNew,&_Window_Time
        mov     &TimeB_LimitCritical,&_ChamberTime2
        mov     &TimeB_LimitBad,&_ChamberTime3
        mov     &My_Alarms,&_temp
        bic     #BIT0+BIT2+BIT4+BIT6+BIT8+BITA+BITC+BITD+BITE+BITF,&_temp
        rra     &_Temp
        mov     &_temp,&_Flags
                call    #WindowCalc
        mov.b   &_Condition,&ChamberB_Condition
        bic     #BIT1+BIT3+BIT5+BIT7+BIT9+BITB,&My_Alarms
        rla     &_Flags
        bis     &_Flags,&My_Alarms
        mov     &_Temperature_Timer,&TemperatureB_Timer
BExit   ret
;*******************************************************************************
******************
WindowCalc
        cmp     &_Temperature_LowLimit,&_Temperature
        jl      ANoNormal
      cmp       &_Temperature,&_Temperature_HighLimit
```

```
        jl      ANoNormal
;normal temperature here
      bit   #_TemperatureAlarm
        jnz     AL1
        mov   #0,&_Temperature_Timer
      jmp    _AEXIT  ;---------->
AL1    mov.b #Fixed,&_Condition
        bis     #_ChamberAlarm
      mov    #0,&_Temperature_Timer
      bic   #_TemperatureAlarm
        jmp    _AExit
ANoNormal
;out of normal temperature here
        bit     #_WindowFlag
        jnz     ACheckConditions
AFlagZero       bis     #_WindowFlag
        mov   #0,&_Temperature_Timer
        sub     &_Window_Level,&_Temperature_LowLimit
        cmp   &_Temperature_LowLimit,&_Temperature
        jl      AOutOfWindows
        add     &_Window_Level,&_Temperature_HighLimit
        cmp   &_Temperature,&_Temperature_HighLimit
        jl      AOutOfWindows
; Temperature inside windows here
ACheckConditions

ACheckNew cmp       &_Window_Time,&_Temperature_Timer
        jlo     _AExit
        cmp   &_ChamberTime2,&_Temperature_Timer
        jhs     ACheckCritical
; Here is new timed alarm
        cmp.b #New,&_Condition
        jeq     _AExit
        mov.b #New,&_Condition
        bis     #_TemperatureAlarm
        bis     #_ChamberAlarm
        jmp    _AExit
ACheckCritical
        cmp   &_ChamberTime3,&_Temperature_Timer
        jhs     ACheckBad
      cmp.b   #Critical,&_Condition
        jeq     _AExit
        mov.b #Critical,&_Condition
        bis     #_TemperatureAlarm
        bis     #_ChamberAlarm
        jmp    _AExit
ACheckBad cmp.b     #Bad,_Condition
        jeq     _AExit
        mov.b #Bad,&_Condition
```

```
        bis     #_TemperatureAlarm
        bis     #_ChamberAlarm
        jmp     _AExit

AOutOfWindows
; Temperature out of windows here
        bis     #_ChamberAlarm
        bis     #_TemperatureAlarm
        mov.b   #New,&_Condition
_Aexit
        ret
;*************************************************************************
******************
TimeTemperature
        cmp.b   #Bad,&_Condition
        jne     TTI1
        cmp     &_Temperature_LowLimit,&_Temperature            ;TemperatureA > Low Limit ?
        jl      TTI1
      cmp       &_Temperature,&_Temperature_HighLimit   ;TemperatureA < High Limit ?
        jl      TTI1
        jmp     TTI2
TTI1    clr     &temp
        cmp     #0x0000,&_Integral
        jne     Ain

        cmp     &_Temperature_LowLimit,&_Temperature            ;TemperatureA > Low Limit ?
        jl      Aout
      cmp       &_Temperature,&_Temperature_HighLimit   ;TemperatureA < High Limit ?
        jl      Aout1
TTI2    bic     #_OutFL                 ; Normal Temperature, clear Flag
        cmp.b   #Fixed,&_Condition
        jeq     Aend
        mov.b   #Fixed,&_Condition
        bis     #_ChamberAlarm
        bis     #_TemperatureAlarm
        mov     #0x0000,&_Integral
        jmp     AEnd
Aout    bis     #_TempStatus
        jmp     Aas
Aout1   bic     #_TempStatus
Aas     bis     #_OutFL                 ; Not Normal Temperature, set Flag
Ain     bit     #_TempStatus ; Previeus was higher or lower than limits?
        jnz     ALow

        cmp     &_Temperature,&_Temperature_HighLimit   ;TemperatureA < High Limit ?
        jl      Aoutc
        bic     #_OutFL
        jmp     AA1
Aoutc   bis     #_OutFL
```

```
AA1     sub     &_Temperature,&_Temperature_HighLimit
        bit     #bitf,&_Temperature_HighLimit
        jz      A1
        xor     #0xffff,&_Temperature_HighLimit
        inc     &_Temperature_HighLimit
A1      mov     &_Temperature_HighLimit,&Temp
        bit     #HumFL
        jnz     NtHmd
        rra     &temp
NtHmd   bit     #_OutFL
        jnz     Aad
        sub     &temp,&_Integral
        cmp     #65400,&_Integral
        jlo     Aend
        mov     #0x0000,&_Integral
        jmp     Aend
Aad     add     &temp,&_Integral
        cmp     #65400,&_Integral
        jlo     Aend
        mov     #65400,&_Integral
        jmp     Aend

ALow    cmp     &_Temperature_LowLimit,&_Temperature   ;TemperatureA < High Limit ?
        jl      Aoute
        bic     #_OutFL
        jmp     AA2
Aoute   bis     #_OutFL
AA2
        sub     &_Temperature,&_Temperature_LowLimit
        bit     #bit7,&_Temperature_LowLimit
        jz      AHC
        xor     #0xffff,&_Temperature_LowLimit
        inc     &_Temperature_LowLimit
AHC     mov     &_Temperature_LowLimit,&Temp
        bit     #HumFL
        jnz     NtHmda
        rra     &temp
NtHmda          bit     #_OutFL
        jnz     Aaf
        sub     &temp,&_Integral
        cmp     #65400,&_Integral
        jlo     Aend
        mov     #0x0000,&_Integral
        jmp     Aend
Aaf     add     &Temp,&_Integral
        cmp     #65400,&_Integral
        jlo     Aend
        mov     #65450,&_Integral
```

```
AMakeConditions
Aend


Acni    cmp    &_Integral_LimitNew,&_Integral
        jhs    ACN
;Normal Integral Here
        cmp.b  #Fixed,&_Condition
        jeq    AFExit
        bis    #_ChamberAlarm
        bis    #_TemperatureAlarm
        mov.b  #Fixed,&_Condition
        jmp    AFExit

ACN     cmp    &_Integral_LimitCritical,&_Integral
        jhs    ACC
        cmp.b  #Fixed,&_Condition
        jne    AFexit
        bis    #_ChamberAlarm
        bis    #_TemperatureAlarm
        mov.b  #New,&_Condition
        jmp    AFExit

ACC     cmp    &_Integral_LimitBad,&_Integral
        jhs    ACB
        cmp.b  #New,&_Condition
        jne    AFexit
        bis    #_ChamberAlarm
        bis    #_TemperatureAlarm
        mov.b  #Critical,&_Condition
        jmp    AFExit

ACB     cmp.b  #Critical,&_Condition
        jne    AFExit
ABD     bis    #_ChamberAlarm
        bis    #_TemperatureAlarm
        mov.b  #BAD,&_Condition
AFExit  ret
;*******************************************************************************
*****************

;*******************************************************************************
*****************
Get_Limits
        call    #Load_InformPeriod
        call   #FieldDec2Hex
        mov    &Value_Hex,&Inform_Period
```

```
        bic     #FirstUsedFL
        call    #Load_FirstChamberUsed
        cmp.b   #0x01,&Temp_Buffer
        jne     GLU1
        bis     #FirstUsedFL

        call    #Load_FirstLowLimit
      call    #FieldDec2Hex
      mov     &Value_Hex,&TemperatureA_LowLimit

      call    #Load_FirstHighLimit
      call    #FieldDec2Hex
      mov     &Value_Hex,&TemperatureA_HighLimit

        cmp     &TemperatureA_HighLimit,&TemperatureA_LowLimit
        jl      GL1
        mov     &TemperatureA_LowLimit,&temp
        mov     &TemperatureA_HighLimit,&TemperatureA_LowLimit
        mov     &Temp,&TemperatureA_HighLimit
GL1
        call    #Load_FirstIntegralNew
      call    #FieldDec2Hex
      mov     &Value_Hex,&IntegralA_LimitNew

        call    #Load_FirstIntegralCritical
      call    #FieldDec2Hex
      mov     &Value_Hex,&IntegralA_LimitCritical

        call    #Load_FirstIntegralBad
      call    #FieldDec2Hex
      mov     &Value_Hex,&IntegralA_LimitBad

        call    #Load_FirstWindowWidth
      call    #FieldDec2Hex
      mov.b   &Value_Hex,&WindowA_Level

        call    #Load_FirstProcessingMethode
      call    #FieldDec2Hex
      mov.b   &Value_Hex,&ChamberA_Procedure
GLU1
;-------------------- Second Chamber -------------------------------
        bic     #SecondUsedFL
        call    #Load_SecondChamberUsed
        cmp.b   #0x01,&Temp_Buffer
        jne     GLU2
        bis     #SecondUsedFL

        call    #Load_SecondLowLimit
      call    #FieldDec2Hex
```

```
        mov     &Value_Hex,&TemperatureB_LowLimit

        call    #Load_SecondHighLimit
        call    #FieldDec2Hex
        mov     &Value_Hex,&TemperatureB_HighLimit

          cmp   &TemperatureB_HighLimit,&TemperatureB_LowLimit
          jl    GL2
          mov   &TemperatureB_LowLimit,&temp
          mov   &TemperatureB_HighLimit,&TemperatureB_LowLimit
          mov   &Temp,&TemperatureB_HighLimit
GL2
        call    #Load_SecondIntegralNew
        call    #FieldDec2Hex
        mov     &Value_Hex,&IntegralB_LimitNew

          call   #Load_SecondIntegralCritical
        call    #FieldDec2Hex
        mov     &Value_Hex,&IntegralB_LimitCritical

          call   #Load_SecondIntegralBad
        call    #FieldDec2Hex
        mov     &Value_Hex,&IntegralB_LimitBad

        call    #Load_SecondWindowWidth
        call    #FieldDec2Hex
        mov.b   &Value_Hex,&WindowB_Level

        call    #Load_SecondProcessingMethode
        call    #FieldDec2Hex
        mov.b   &Value_Hex,&ChamberB_Procedure
          bic   #HumidityUsedFL
          cmp.b #HumidityIntegral,&ChamberB_Procedure
          jne   GLU2
          bis   #HumidityUsedFL
          cmp.b #HumidityWindow,&ChamberB_Procedure
          jne   GLU2
          bis   #HumidityUsedFL
GLU2
;-------------  -        -  --------------------
          call   #Load_FirstChamberCode
          mov.b &Temp_Buffer,&ChamberCode
          cmp.b #0x01,&ChamberCode
          jeq   GTEx
          cmp.b #0x02,&ChamberCode
          jne   GTEx

GTEx    ret
;-----------------------------------------------------------------------------------------------
```

```
Value_Hex2Dec
        mov     #ValueBuffer,r9
LkLL    clr.b   0(r9)
        inc     r9
        cmp     #ValueBuffer+9,r9
        jlo     LkLL

        mov.b   #'*',0(r9)
        dec     r9
        dec     r9
        mov.b   #'.',0(r9)
        dec     r9
        push.b  counter
        push    &Temp
        rra     &temp
        tst     &temp
        jn      Negative_temp
        jmp     Positive_temp
Negative_temp
        inv     &temp
Positive_temp
        call    #div_temp_10
        mov.b   Left_Remainter,0(r9)
        bis.b   #0x30,0(r9)
        dec     r9
        mov     Right_Remainter,&Temp
        cmp     #0x0a,Right_Remainter
        jhs     Positive_Temp
        mov.b   Right_Remainter,0(r9)
        bis.b   #0x30,0(r9)
        dec     r9
        pop     &Temp
        tst     &temp
        jn      Negative_temperature
        jmp     Positive_temperature
Negative_Temperature
        mov.b   #'-',0(r9)
        dec     r9
        mov.b   #'@',0(r9)
        jmp     HDCont1
Positive_Temperature
HDCont          mov.b   #'@',0(r9)
HDCont1         rrc.b   &Temp
        jnc     HDC1
        mov.b   #'5',&ValueBuffer+8
        jmp     HDC5
HDC1    mov.b           #'0',&ValueBuffer+8
HDC5
        pop.b   counter
```

```
        ret
;-------------------------------------------------------------------------------
div_temp_10
        mov    &temp,right_remainter
        mov    #10,divisor
        clr    left_remainter
        mov.b  #16,Counter
        clrc
        rlc    right_remainter
        rlc    left_remainter

again   sub    divisor,left_remainter

        tst    left_remainter
        jl     restore
        clrc
        rlc    right_remainter
        rlc    left_remainter
        bis    #bit0,right_remainter
        jmp    no_rest
restoreadd     divisor,left_remainter
        clrc
        rlc    right_remainter
        rlc    left_remainter
        bic    #bit0,right_remainter
no_rest        dec.b  Counter
        jne    again
        clrc
        rrc    left_remainter
        ret
;*******************************************************************************
Load_UnitID             mov.b    #UnitID,&FieldCounter
                jmp     LoadInfoField
Load_passwordOld        mov.b    #PasswordOld,&fieldCounter
                jmp     loadinfofield
Load_CurrentPassword    mov.b    #CurrentPassword,&fieldCounter
                jmp     loadinfofield
Load_TerminalUsed       mov.b    #TerminalUsed,&fieldCounter
                jmp     loadinfofield
Load_TerminalPrefix     mov.b    #TerminalPrefix,&fieldCounter
                jmp     loadinfofield
Load_FirstCallingNumber     mov.b    #FirstCallingNumber,&fieldCounter
                jmp     loadinfofield
Load_SecondCallingNumber    mov.b    #SecondCallingNumber,&fieldCounter
                jmp     loadinfofield
Load_ThirdCallingNumber     mov.b    #ThirdCallingNumber,&fieldCounter
                jmp     loadinfofield
Load_RingCounts         mov.b    #RingCounts,&fieldCounter
                jmp     loadinfofield
```

```
Load_InformPeriod               mov.b    #InformPeriod,&fieldCounter
                    jmp      loadinfofield
Load_NumberOfChambers        mov.b    #NumberOfChambers,&fieldCounter
                    jmp      loadinfofield

Load_FirstChamberCode        mov.b    #FirstChamberCode,&fieldCounter
                    jmp      loadinfofield
Load_FirstChamberUsed        mov.b    #FirstChamberUsed,&fieldCounter
                    jmp      loadinfofield
Load_FirstLowLimit           mov.b    #FirstLowLimit,&fieldCounter
                    jmp      loadinfofield
Load_FirstHighLimit          mov.b    #FirstHighLimit,&fieldCounter
                    jmp      loadinfofield
Load_FirstIntegralNew           mov.b    #FirstIntegralNew,&fieldCounter
                    jmp      loadinfofield
Load_FirstIntegralCritical      mov.b    #FirstIntegralCritical,&fieldCounter
                    jmp      loadinfofield
Load_FirstIntegralBad           mov.b    #FirstIntegralBad,&fieldCounter
                    jmp      loadinfofield
Load_FirstProcessingMethode          mov.b    #FirstProcessingMethode,&fieldCounter
                    jmp      loadinfofield
Load_FirstWindowWidth        mov.b    #FirstWindowWidth,&fieldCounter
                    jmp      loadinfofield

Load_SecondChamberCode       mov.b    #SecondChamberCode,&fieldCounter
                    jmp      loadinfofield
Load_SecondChamberUsed       mov.b    #SecondChamberUsed,&fieldCounter
                    jmp      loadinfofield
Load_SecondLowLimit          mov.b    #SecondLowLimit,&fieldCounter
                    jmp      loadinfofield
Load_SecondHighLimit         mov.b    #SecondHighLimit,&fieldCounter
                    jmp      loadinfofield
Load_SecondIntegralNew       mov.b    #SecondIntegralNew,&fieldCounter
                    jmp      loadinfofield
Load_SecondIntegralCritical  mov.b    #SecondIntegralCritical,&fieldCounter
                    jmp      loadinfofield
Load_SecondIntegralBad       mov.b    #SecondIntegralBad,&fieldCounter
                    jmp      loadinfofield
Load_SecondWindowWidth       mov.b    #SecondWindowWidth,&fieldCounter
                    jmp      loadinfofield
Load_SecondProcessingMethode      mov.b    #SecondProcessingMethode,&fieldCounter
                    jmp      loadinfofield

LoadInfoField
       push     sr
       dint
       push     r14
         push    r15
         mov     #temp_buffer,r15
```

```
       mov      #0x1000,r14
       cmp.b    #0x00,&FieldCounter
       jne      Load_B
       jmp      Load_E
Load_B   mov.b        @r14,&tmp
         rrc.b    &tmp
       rrc.b    &tmp
       rrc.b    &tmp
       rrc.b    &tmp
       and.b    #0x0f,&tmp
       cmp.b    #0x0b,&tmp
         jne      Load_C
         dec.b   fieldCounter
         jz       Load_D
Load_Cmov.b  @r14,&tmp
         and.b   #0x0f,&tmp
         cmp.b   #0x0b,&tmp
         jeq      Load_A
         inc      r14
         jmp      load_B
Load_A          inc      r14
         dec.b   &fieldcounter
         jnz      load_B

Load_Emov.b  @r14,&tmp
         rrc.b    &tmp
       rrc.b    &tmp
       rrc.b    &tmp
       rrc.b    &tmp
         and.b   #0x0f,&tmp
         mov.b   &tmp,0(r15)
         cmp.b   #0x0b,0(r15)
         jeq      Load_EX
         inc      r15
         mov.b   @r14,&tmp
         and.b   #0x0f,&tmp
         mov.b   &tmp,0(r15)
         cmp.b   #0x0b,0(r15)
         jeq      Load_EX
         inc      r15
         inc      r14
         jmp      Load_E

Load_D   mov.b        @r14,&tmp
         and.b   #0x0f,&tmp
         mov.b   &tmp,0(r15)
         cmp.b   #0x0b,0(r15)
         jeq      Load_EX
         inc      r15
```

```
        inc     r14
        mov.b  @r14,&tmp
        rrc.b   &tmp
      rrc.b   &tmp
      rrc.b   &tmp
      rrc.b   &tmp
        and.b  #0x0f,&tmp
        mov.b  &tmp,0(r15)
        cmp.b  #0x0b,0(r15)
        jeq     Load_EX
        inc     r15
        jmp     Load_D
Load_EX       mov.b  #'*',0(r15)
        pop     r15
      pop     r14
      pop     sr
      ret
;---------  Table For temperature Convertion  With RTI Thermistor 3K & 10K Ref Resistor ----------
--------------
Temp_table:

        DB      127,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127;00-15
        DB      127,127,119,116,113,110,107,105,102,100,097,095,093,091,089,087;16-31
        DB      085,083,082,080,079,078,075,074,073,071,070,068,067,066,065,064;32-47
        DB      062,061,060,059,057,056,055,054,053,052,051,050,049,048,047,046;48-63
        DB      045,044,043,043,042,041,040,039,038,037,036,036,035,034,033,032;64-79
        DB      032,031,030,029,029,028,027,026,026,025,024,023,023,022,021,021;80-95
        DB      020,019,019,018,017,017,016,015,015,014,013,013,012,011,011,010;96-111
        DB      009,009,008,007,007,006,006,005,004,004,003,002,002,001,001,000;112-127
        DB      -01,-01,-02,-02,-03,-04,-04,-05,-05,-06,-07,-07,-08,-08,-09,-10;128-143
        DB      -10,-11,-12,-12,-13,-13,-14,-15,-15,-16,-16,-17,-18,-18,-19,-19;144-159
        DB      -20,-21,-21,-22,-22,-23,-24,-24,-25,-26,-26,-27,-27,-28,-29,-29;160-175
        DB      -30,-31,-31,-32,-33,-33,-34,-35,-35,-36,-37,-37,-38,-39,-39,-40;176-191
        DB      -41,-42,-42,-43,-44,-45,-45,-46,-47,-48,-48,-49,-50,-51,-51,-52;192-207
        DB      -53,-54,-55,-56,-57,-58,-59,-60,-60,-61,-62,-63,-64,-65,-67,-68;208-223
        DB      -69,-70,-71,-72,-73,-74,-76,-77,-79,-80,-82,-83,-85,-86,-88,-90;224-239
        DB      -92,-128,-128,-128,-128,-128,-128,-128,-128,-128,-128,-128,-128,-128,-128,-
128;240-255

PT100Table
        if      PT100=1
        db      -107
        db      -106
        db      -106
        db      -105
        db      -105
        db      -105
        db      -104
        db      -104
```

```
db      -103
db      -103
db      -103
db      -102
db      -102
db      -102
db      -101
db      -101
db      -100
db      -100
db      -100
db      -99
db      -99
db      -98
db      -98
db      -98
db      -97
db      -97
db      -97
db      -96
db      -96
db      -95
db      -95
db      -95
db      -94
db      -94
db      -93
db      -93
db      -93
db      -92
db      -92
db      -92
db      -91
db      -91
db      -90
db      -90
db      -90
db      -89
db      -89
db      -88
db      -88
db      -88
db      -87
db      -87
db      -87
db      -86
db      -86
db      -85
db      -85
```

```
db      -85
db      -84
db      -84
db      -83
db      -83
db      -83
db      -82
db      -82
db      -82
db      -81
db      -81
db      -80
db      -80
db      -80
db      -79
db      -79
db      -78
db      -78
db      -78
db      -77
db      -77
db      -77
db      -76
db      -76
db      -75
db      -75
db      -75
db      -74
db      -74
db      -73
db      -73
db      -73
db      -72
db      -72
db      -72
db      -71
db      -71
db      -70
db      -70
db      -70
db      -69
db      -69
db      -69
db      -68
db      -68
db      -67
db      -67
db      -67
db      -66
```

```
db      -66
db      -65
db      -65
db      -65
db      -64
db      -64
db      -64
db      -63
db      -63
db      -62
db      -62
db      -62
db      -61
db      -61
db      -60
db      -60
db      -60
db      -59
db      -59
db      -59
db      -58
db      -58
db      -57
db      -57
db      -57
db      -56
db      -56
db      -56
db      -55
db      -55
db      -54
db      -54
db      -54
db      -53
db      -53
db      -52
db      -52
db      -52
db      -51
db      -51
db      -51
db      -50
db      -50
db      -49
db      -49
db      -49
db      -48
db      -48
db      -48
```

```
db      -47
db      -47
db      -46
db      -46
db      -46
db      -45
db      -45
db      -45
db      -44
db      -44
db      -43
db      -43
db      -43
db      -42
db      -42
db      -41
db      -41
db      -41
db      -40
db      -40
db      -40
db      -39
db      -39
db      -38
db      -38
db      -38
db      -37
db      -37
db      -37
db      -36
db      -36
db      -35
db      -35
db      -35
db      -34
db      -34
db      -34
db      -33
db      -33
db      -32
db      -32
db      -32
db      -31
db      -31
db      -30
db      -30
db      -30
db      -29
db      -29
```

```
db      -29
db      -28
db      -28
db      -27
db      -27
db      -27
db      -26
db      -26
db      -26
db      -25
db      -25
db      -24
db      -24
db      -24
db      -23
db      -23
db      -23
db      -22
db      -22
db      -21
db      -21
db      -21
db      -20
db      -20
db      -20
db      -19
db      -19
db      -18
db      -18
db      -18
db      -17
db      -17
db      -17
db      -16
db      -16
db      -15
db      -15
db      -15
db      -14
db      -14
db      -14
db      -13
db      -13
db      -12
db      -12
db      -12
db      -11
db      -11
db      -11
```

```
db      -10
db      -10
db      -9
db      -9
db      -9
db      -8
db      -8
db      -8
db      -7
db      -7
db      -6
db      -6
db      -6
db      -5
db      -5
db      -5
db      -4
db      -4
db      -3
db      -3
db      -3
db      -2
db      -2
db      -2
db      -1
db      -1
db      0
db      0
db      0
db      1
db      1
db      1
db      2
db      2
db      3
db      3
db      3
db      4
db      4
db      4
db      5
db      5
db      6
db      6
db      6
db      7
db      7
db      7
db      8
```

```
db      8
db      9
db      9
db      9
db      10
db      10
db      10
db      11
db      11
db      12
db      12
db      12
db      13
db      13
db      13
db      14
db      14
db      15
db      15
db      15
db      16
db      16
db      16
db      17
db      17
db      18
db      18
db      18
db      19
db      19
db      19
db      20
db      20
db      21
db      21
db      21
db      22
db      22
db      22
db      23
db      23
db      24
db      24
db      24
db      25
db      25
db      25
db      26
db      26
```

```
db      26
db      27
db      27
db      28
db      28
db      28
db      29
db      29
db      29
db      30
db      30
db      31
db      31
db      31
db      32
db      32
db      32
db      33
db      33
db      34
db      34
db      34
db      35
db      35
db      35
db      36
db      36
db      37
db      37
db      37
db      38
db      38
db      38
db      39
db      39
db      39
db      40
db      40
db      41
db      41
db      41
db      42
db      42
db      42
db      43
db      43
db      44
db      44
db      44
```

```
db    45
db    45
db    45
db    46
db    46
db    47
db    47
db    47
db    48
db    48
db    48
db    49
db    49
db    49
db    50
db    50
db    51
db    51
db    51
db    52
db    52
db    52
db    53
db    53
db    54
db    54
db    54
db    55
db    55
db    55
db    56
db    56
db    56
db    57
db    57
db    58
db    58
db    58
db    59
db    59
db    59
db    60
db    60
db    61
db    61
db    61
db    62
db    62
db    62
```

```
db      63
db      63
db      63
db      64
db      64
db      65
db      65
db      65
db      66
db      66
db      66
db      67
db      67
db      68
db      68
db      68
db      69
db      69
db      69
db      70
db      70
db      70
db      71
db      71
db      72
db      72
db      72
db      73
db      73
db      73
db      74
db      74
db      74
db      75
db      75
db      76
db      76
db      76
db      77
db      77
db      77
db      78
db      78
db      79
db      79
db      79
db      80
db      80
db      80
```

```
db    81
db    81
db    81
db    82
db    82
db    83
db    83
db    83
db    84
db    84
db    84
db    85
db    85
db    85
db    86
db    86
db    87
db    87
db    87
db    88
db    88
db    88
db    89
db    89
db    89
db    90
db    90
db    91
db    91
db    91
db    92
db    92
db    92
db    93
db    93
db    93
db    94
db    94
db    95
db    95
db    95
db    96
db    96
db    96
db    97
db    97
db    97
db    98
db    98
```

```
        db      99
        db      99
        db      99
        db      100
        db      100
        db      100
        db      101
        endif
;-------------------------------------------------------------------------------------------------------
        ; MSP430x1xx Interrupt vectors

        ORG     0FFE4H          ; Port 1 Vector
        DW      PORT_1_ISR
        ORG     0FFEAH          ; ADC10 Vector
        DW      ADC10_ISR
        ORG     0FFECH          ; ADC10 Vector
        DW      SCITX_ISR
        ORG     0FFEEH          ; ADC10 Vector
        DW      SCIRX_ISR
        ORG     0FFF0h; Timer_AX Vector
        DW      TAX_ISR         ;
        ORG     0FFF2h; Timer_A0 Vector
        DW      TA0_ISR         ;
        ORG     0FFF4H          ; WDT Vector
        DW      RESET
        ORG     0FFFEH; Reset Vector
        DW      RESET
        END     main

;_____________________________________________________________________________________________
_____________________________________
```

10. Βιβλιογραφία

[1] “MSP430 microcontrollers”.
Available: https://www.ti.com/microcontrollers-mcus-processors/msp430-microcontrollers/overview.html

[2] “RS-485 Serial Interface Explained”.
Available: https://www.cuidevices.com/blog/rs-485-serial-interface-explained#:~:text=RS%2D485%20is%20an%20industrial,devices%20on%20the%20same%20bus.

[3] “RS-485”.
Available: https://en.wikipedia.org/wiki/RS-485

[4] “Temperature control systems”.
Available: https://www.who.int/