Πτυχιακή Εργασία *HOU-CS-UGP-2006-17*

Επικοινωνία με Δίκτυο Ψηφιακών Αισθητήρων *1-Wire*

ΓΙΩΡΓΟΣ ΒΙΟΛΕΤΤΑΣ

Η εργασία αυτή είναι αφιερωμένη στον Πατέρα μου, ως ελάχιστη ανταμοιβή για όλα αυτά που έχει κάνει για μένα. Περισσότερο από όλα όμως γιατί επιτέλους ικανοποιώ το «σαράκι» του να πάρω ένα πτυχίο……..

Ευχαριστίες:

Υπάρχουν αρκετοί άνθρωποι που με βοήθησαν πολύ αυτά τα πέντε χρόνια που κάθισα ξανά στα θρανία και θέλω να τους ευχαριστήσω όλους, ο καθένας τους ξέρει τι έχει κάνει για μένα……

Ξεχωριστή θέση στην καρδιά μου έχουν οι συνάδελφοι και πτυχιούχοι πλέον του Ε.Α.Π. Δημήτρης Θεοδωρίδης, Βάιος Ζιώγας και φυσικά ο Τρύφων Θεοδώρου, χωρίς την ανεκτίμητη βοήθεια του οποίου, αυτή η εργασία δεν θα είχε τελειώσει.

Επίσης θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή μου Σταύρο, γιατί εκτός από ένας άρτια καταρτισμένος επιστήμονας είναι και ένας υπέροχος –πάντα χαμογελαστός- άνθρωπος.

Θεσσαλονίκη, Μάιος 2006

Επικοινωνία με Δίκτυο Ψηφιακών Αισθητήρων *1-Wire*

Γιώργος Βιολέττας

Ονοματεπώνυμο Επιβλέποντα	**Ονοματεπώνυμο Μέλους 1**	**Ονοματεπώνυμο Μέλους 2**
Σταύρος Βουγιούκας	Αθανάσιος Σκόδρας	Οδυσσέας Κουφοπαύλου

Περίληψη: Σε πολλές εφαρμογές απαιτείται συνεχής μέτρηση μεγεθών τα οποία είναι κατανεμημένα στο χώρο, όπως π.χ. οι μετρήσεις θερμοκρασίας, πίεσης, υγρασίας σε θερμοκήπια, μετρήσεις στάθμης άλατος, αμμωνίας και άλλων οξειδωτικών παραγόντων σε ιχθυοτροφεία κτλ. Οι ταυτόχρονες μετρήσεις απαιτούν μεγάλο αριθμό αισθητήρων χαμηλού κόστους και απλή υποδομής διασύνδεσής τους. Για αυτό τον λόγο έχουν αναπτυχθεί από διάφορες εταιρείες, σειριακά πρωτόκολλα επικοινωνίας. Ένα εξ αυτών , είναι το ενσύρματο πρωτόκολλο επικοινωνίας 1-wire στο οποίο ένα μεγάλο πλήθος ψηφιακών αισθητήρων συνδέονται σε σειρά σε ένα «δίαυλο» (Bus), στον οποίο ισχύς και πληροφορία μεταδίδονται μαζί.

Η επικοινωνία ενός Η/Υ με τέτοιους αισθητήρες απαιτεί την ύπαρξη κατάλληλων μετατροπέων για τη σύνδεση στο φυσικό επίπεδο διαύλου 1-wire σε θύρα serial, USB ή Ethernet, και αντίστοιχα τη χρήση σειριακού ή τύπου TCP/IP πρωτοκόλλου. Τέτοιοι μετατροπείς παρέχονται από διαφορετικές εταιρίες, με αποτέλεσμα η υλοποίηση λογισμικού επικοινωνίας εφαρμογών με τους αισθητήρες να εξαρτάται από το μετατροπέα που χρησιμοποιείται.

Η πτυχιακή εργασία αφορά στην υλοποίηση ενός «Ενιαίου Αντικειμενοστραφούς Ενδιάμεσου» (Application Programming Interface) υψηλού επιπέδου, το οποίο θα απλοποιεί και θα διευκολύνει την επικοινωνία λογισμικού εφαρμογών με ενσύρματα δίκτυα ψηφιακών αισθητήρων 1-wire ανεξαρτήτως μετατροπέα σύνδεσης και ενδιάμεσου πρωτοκόλλου .

Λέξεις-κλειδιά: 1-wire, δίκτυο αισθητήρων, ψηφιακός αισθητήρας, ενσύρματο πρωτόκολλο επικοινωνίας, δίαυλος, bus .

Περιεχόμενο: Κείμενο, πρόγραμμα σε γλώσσα C, πρωτότυπο εφαρμογής

Abstract: In biosystems monitoring applications it is necessary to continuously measure spatially distributed variables, like for example temperature, pressure and humidity in greenhouses, salinity level, ammonia and other oxidizing agents in aquaculture facilities. Such measurements require a large number of low-cost sensors, which can be interconnected in a simple manner. Various companies have developed networking solutions to cover such demands. The 1-wire networking technology is a very successful device communications bus system, which allows connection of a large number of digital sensors on a single wire (a ground wire is also needed). Data, signaling and power are transmitted through the same wire enabling low-cost networking. Communication of such a network with a PC or embedded controller requires appropriate interfaces, such as serial or USB with a serial protocol, or Ethernet with a TCP/IP based protocol. Such interfaces are available from various vendors and hence the communication software with 1-wire digital networks is vendor-dependent. The goal of this work is to develop a high level object-oriented Application Programming Interface (API), which can be used by user application level software to communicate easily with 1-wire networks, independently of the hardware interface and its corresponding low-level communicating protocol.

Communications with 1-*Wire* Digital Sensor Networks

George Violettas

Supervisor	**Member 1**	**Member 2**
Stavros Vougioukas	Athanasios Skodras	Odysseas Koufopavlou

Introduction

In biosystems monitoring applications it is necessary to continuously measure spatially distributed variables, like for example temperature, pressure and humidity in greenhouses, salinity level, ammonia and other oxidizing agents in aquaculture facilities. Such measurements require a large number of low-cost sensors, which can be interconnected in a simple manner. Various companies have developed networking solutions to cover such demands.

The 1-wire networking technology is a very successful device communications bus system, which allows connection of a large number of digital sensors on a single wire (a ground wire is also

needed). Data, signaling and power are transmitted through the same wire enabling low-cost networking.

The advantages of the 1-Wire are:

- It is cost effective
- Users can add/remove devices in real time
- It offers enough data speed for applications demanding low-to-moderate speed
- It only needs "One Wire" to transfer data and power
- A large variety of digital 1-wire sensors and appliances has been developed

The disadvantages are:

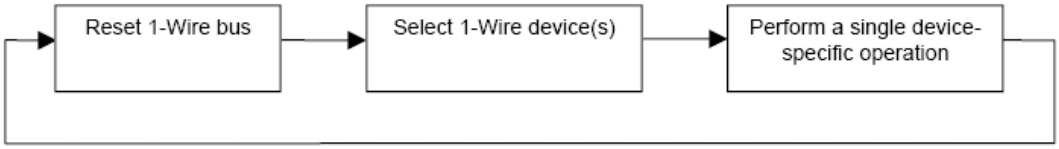
- It is not standardized (could be through IEEE), but is a product of a single company (Dallas Semiconductors)
- Relatively low data speeds for complicated future application
- Low quality of free code issued by the manufacturer for the individual devices.

Communication of such a network with a PC or embedded controller requires appropriate adaptor interfaces, such as serial or USB with a serial protocol, or Ethernet with a TCP/IP based protocol. Such interfaces are available from various vendors and hence the communication software with 1-wire digital networks is vendor-dependent. The goal of this work is to develop a high level object-oriented Application Programming Interface (API), which can be used by user application level software to communicate easily with 1-wire networks, independently of the hardware adaptor and its corresponding low-level communicating protocol.

Basic Principles of 1-Wire Networks

There are two basic devices to the network: the master device (adapter) and the slave devices (various sensors, hubs, LCDs etc). The master devices are the intermediaries between the sensors and the computer. These master devices connect to the computer thorough RS232, USB ports, or TCP/IP protocol. Depending on the device, the slave devices can get the power from the network or power up independently. There are 3 major operations between the master and the slave devices. Sending LOGICAL HIGH as logical digit one (1), sending LOGICAL LOW as logical digit zero (0), and READ BIT.

The full cycle of the basic operations is shown below:



Picture 1: full cycle of the 1-Wire basic operations

Every 1-Wire device has a worldwide unique digital address. All the communication with the device is sent to this address. Before the start of any communication a Reset signal must be sent. Then the master device selects the specific device (if desired) and then the specific command follows. The commands to the 1-wire devices may differ significantly, depending on the connection protocol of the specific master device. Some of them are native, supported by the Operating Systems (COM, serial devices protocol) and some need extra Operating System specific drivers (USB).

Existing Software

By the time this work started, there were 5 different API's provided free by the manufacturer, along with the source code and documentation. They are listed below:

API	Description
1-Wire Public Domain	C Language for all platforms
1-Wire API για Java	Java language for many platforms and some adaptors
1-Wire API για .NET	Created with #J. (.NET). Supports Windows CE
1-Wire COM	Windows Component Object Model (COM). Supports Windows with Java and Visual Basic Scripts; needs java virtual machine
TMEX API	"Closed" API. No source code

Software Implementation

The major goal was to give to every application the ability of using the same code no matter which protocol or adapter is used to communicate with the 1-Wire network.
In order to achieve this, a high-level "universal" API was developed, which did the following:

- Every element of the network is treated as an object
- One class is constructed for each device
- The fields of each class are the programmable functions of the device
- The methods of those classes are the functions of the device

In order to be able to call the same function for different adapter (e.g. USB, Ethernet), all the existing functions were modified, so that one function can call each and every adaptor specific function, one by one (case_adaptor1, case_adaptor2 etc.). The http code does not have the ability to implement all the functions of the other protocols and adaptors. So in certain cases there is no http case implementation. The following are all the libraries (modified and original) required for the API to run:

Acquire and release a Session	**Link Layer functions**
mowSesU.C	mowLLU.C
mpsesw32.C	mlpWin32.C
musbwses.c	musbwlnk.c
Network functions	**Transport functions**
mownetu.C	mowTrnU.C
mownet.C	mowTran.C
musbwnet.C	musbwtrn.C
Multi-build Libraries	**Existing Libraries (Used "as is")**
Mownet.h	Serial port adaptor DS2480 libraries
multises.c	mWn32Lnk.C
multilnk.c	m2480ut.c
multinet.C	mDS2480.h
multitran.c	
USB adaptor DS2490 libraries	**http Server HA7Net library**
m2490ut.c	HA7Net.c

d2490u.h	All the libraries created for the server are in the same library

List of files needed for the API

HA7Net.c		
ds2490.h	mownetu.c	musbwnet.c
m2480ut.c	mowsesu.c	musbwses.c
m2490ut.c	mowtran.c	musbwtrn.c
mcrcutil.c	mowtrnu.c	mwn32lnk.c
mds2480.h	mpsesw32.c	mowllu.c
mlpwin32.c	multilnk.c	mownet.c
mowerr.c	multinet.c	mownet.h
multitran.c	multises.c	musbwlnk.c

Demo Application

Based on the above files, a demo application was created. As master device, the USB DS9490R was used. As slave devices we used the DS1920, "Temperature logger in real time" and DS1921, "Temperature and Humidity Sampling device". We also used the DOS implemented, libraries for those two devices.

The demo application offers 3 major choices (menus)

- First choice: Presenting real time data from a DS1920 temperature measurement device. There are two matrices presenting the temperature in both Celsius and Fahrenheit degrees and a graph showing the variance of the temperature in time.
- Second choice: Detailed representation of the DS1921 stored data. After setting up the sampling rate of the device, it is left on the field to acquire the data in question. After the given period of time, the device is collected and connected back to the computer to "unload" the stored data.
- Third Choice: Presenting in real time graphs the data from two devices, the DS1920 and the DS2438 on the same time. This way we can have one of the devices inside a building and the

other one outside, under the sun. This way we can absorb the temperature difference between the two devices. The DS2438 is capable of measuring temperature and humidity, so there are two graphs for this device.

All the developed software was written in C++ Language. Every effort was made to make the code operating system and machine independent. All the code for both the API and the DEMO application is included in Appendix A and Appendix B, commented in English Language.
With this work, the basic body of the API is ready. Theoretically a programmer can start writing applications based on it. All the low level functions have been exhaustively tested through the continuous compilation of the code of all the libraries and functions created. Even the functions for the http server were all tested. Finally, two different experimental interfaces were produced which can be used to test and try various applications constructed.

Conclusions

The 1-Wire technology (bus network and sensors) is a fairly cheap and easy to implement mechanism for the measurement of variables in a variety of professional and scientific fields. The communication speed is relatively low, so it is ideal for continuously measuring distributed slow-varying physical variables like temperature and/or humidity. The reliability and relative simplicity of the 1-wire devices gives the ability to an inexperienced programmer to create custom applications. The API developed in this work makes it even easier to develop custom applications without going into the details of the protocol, or the specific master device used for the network. The API's functionality was tested extensively and demo applications were written, which can be used as "guides" for further experimentation and software development.

Περιεχόμενα

1. Εισαγωγή

1.1. Αντικείμενο

Σε πολλές εφαρμογές απαιτείται η συνεχής μέτρηση μεγεθών τα οποία είναι κατανεμημένα στο χώρο και συνεπώς απαιτούν μεγάλο αριθμό αισθητήρων. Τέτοιες εφαρμογές είναι οι βιομηχανικοί αυτοματισμοί, οι αυτοματισμοί θερμοκηπίων, και τα έξυπνα κτίρια. Σε αυτήν την κατεύθυνση έχουν αναπτυχθεί διάφορα πρωτόκολλα σειριακής επικοινωνίας από διάφορες εταιρείες όπως τα I^2C^*, SMBUS™, SPI™, MicroWirePLUS™, MBus(EN1434), CAN(ISO11838).

Ένα εξ αυτών - και αρκετά διαδεδομένο - είναι το *ενσύρματο πρωτόκολλο επικοινωνίας 1-wire* το οποίο αποτελεί μία φτηνή και πρακτική λύση για τέτοιου είδους εφαρμογές. Ένα μεγάλο πλήθος ψηφιακών αισθητήρων (π.χ., αισθητήρες μέτρησης θερμοκρασίας, υγρασίας, pH, συγκέντρωσης CO_2 ή A/D converters) συνδέονται σε σειρά σε έναν δίαυλο (Bus) στο οποίο ισχύς και πληροφορία μεταδίδονται μαζί. Κάθε αισθητήρας έχει μία μοναδική διεύθυνση (αντίστοιχη ενός MAC address) και υλοποιεί το 1-wire πρωτόκολλο. Η σύνδεση ενός Η/Υ με τέτοιους αισθητήρες απαιτεί την ύπαρξη ενός μετατροπέα (hardware) 1-wire σε serial, USB ή TCP/IP, HTTP πρωτόκολλο. Τέτοιοι μετατροπείς παρέχονται από διαφορετικές εταιρίες, η επικοινωνία όμως μαζί τους ποικίλλει.

Στόχος της πτυχιακής εργασίας είναι η ενοποίηση /ομαδοποίηση όλης αυτής της σκόρπιας πληροφορίας και η υλοποίηση ενός ενιαίου αντικειμενοστραφούς ενδιάμεσου (Application Programming Interface) υψηλού επιπέδου το οποίο θα απλοποιεί και θα διευκολύνει την επικοινωνία λογισμικού εφαρμογών με ενσύρματα δίκτυα ψηφιακών αισθητήρων 1-wire ανεξαρτήτως τρόπου σύνδεσης (σειριακής θύρας, USB, LPT, RJ-45) και πρωτοκόλλου επικοινωνίας (RS232, HTTP).

1.2. Περιγραφή του 1-Wire

Το *1-Wire*® είναι κατατεθειμένο trademark της Dallas Semiconductor. Αφορά στην διασύνδεση συσκευών που πληρούν συγκεκριμένα χαρακτηριστικά, μέσω ενός καλωδίου (wire), πάνω από το οποίο περνούν μαζί τροφοδοσία και δεδομένα. Ουσιαστικά βέβαια χρειάζεται και άλλο ένα καλώδιο που λειτουργεί ως ουδέτερος (γη). Αυτό πρακτικά σημαίνει ότι σε μεταλλικές κατασκευές

ή άλλου τύπου αγώγιμα υλικά, μπορούμε να συνδέσουμε τις συσκευές αυτές με ένα μοναδικό καλώδιο, διαφορετικά χρειαζόμαστε ένα ζεύγος καλωδίων, όπως για παράδειγμα το τηλεφωνικό καλώδιο. Όλες οι συσκευές 1-Wire έχουν ένα 64μπιτο αριθμό γραμμένο σε «μνήμη μόνο ανάγνωσης» ROM ώστε να αναγνωρίζονται μοναδικά σε ένα τέτοιο δίκτυο. Η μορφή του αριθμού αυτού, φαίνεται στο Σχήμα 1. *(3)*

MSB	64 bit Registration number			LSB
8 bit CRC		48 bit Serial Number		8 bit Family Code
MSB	LSB	MSB	LSB	MSB LSB

Σχήμα 1: μοναδικός αριθμός συσκευής 1-Wire

Υπόμνημα
MSB: Most Significant Bit, το πλέον σημαντικό δυαδικό ψηφίο
LSB: Least Significant Bit, το λιγότερο σημαντικό δυαδικό ψηφίο
CRC: Cyclic Redundancy Code: κώδικας επισήμανσης/διόρθωσης λαθών
Family Code: ο αριθμός που συνδέει την συγκεκριμένη συσκευή με μία οικογένεια με συγκεκριμένα χαρακτηριστικά

Υπάρχουν δύο ειδών βασικές συσκευές στο κύκλωμα. Η «κύρια συσκευή» (master device) που είναι ουσιαστικά και ο ελεγκτής του κυκλώματος και οι «εξαρτώμενες συσκευές» (slave devices) που μπορούν να είναι διαφορετικών τύπων και δυνατοτήτων. Υπάρχουν και άλλες δευτερεύουσες συσκευές 1-Wire όπως για παράδειγμα Hubs, ή συσκευές πολλαπλής εισόδου/ εξόδου (multiple I/O board) ή οθόνες LCD για εμφάνιση μηνυμάτων ή/και αλληλεπίδρασης με το δίκτυο. Οι συσκευές στο 1-Wire συνδέονται εν σειρά. Μπορούμε όμως με την κατάλληλη προσθήκη ειδικών Hubs να δημιουργήσουμε τοπολογία αστέρα (star). Οι κύριες συσκευές (Master Devices), δηλαδή οι οδηγοί του κυκλώματος εκτελούν τις εργασίες ανάγνωσης/ εγγραφής των συνδεδεμένων στο κύκλωμα slave συσκευών, και αποτελούν τον ενδιάμεσο του bus με τον οποιαδήποτε Η/Υ (PC, Smart phone κτλ) που θα θέλαμε να επικοινωνεί με το δίκτυο. Οι συσκευές αυτές μπορούν να συνδέονται είτε μέσω πρωτοκόλλου RS232 (σειριακή θύρα) είτε μέσω USB, είτε μέσω Ethernet

interface, υλοποιώντας το πρωτόκολλο HTTP. Οι συσκευές slave μπορούν να είναι διαφορετικών τύπων και σκοπού (A/D μετατροπείς, μνήμης ανάγνωσης (ROM), μνήμης εγγραφής/ανάγνωσης (EPROM), αισθητήρες διαφόρων μεγεθών κτλ). Αναλόγως των απαιτήσεων και δυνατοτήτων των slave συσκευών, αυτές μπορεί να παίρνουν τροφοδοσία μόνο από το κύκλωμα 1-Wire είτε να έχουν δική τους τροφοδοσία (μπαταρία λιθίου). Η ταχύτητα που μπορεί να αναπτυχθεί σε ένα κύκλωμα 1-wire ξεκινά από 16,3Kbps και μπορεί να φτάσει μέχρι 142 Kbps. Το μήκος ενός δικτύου 1-wire μπορεί να φτάσει τα 300 μέτρα ενώ υποστηρίζεται πλήρως η δυναμική αλλαγή της τοπολογίας του δικτύου (προσθαφαίρεση συσκευών σε πραγματικό χρόνο). Υποστηρίζεται επίσης προστασία και επιδιόρθωση λαθών μέσω «Κώδικα Κυκλικού Πλεονασμού» (Cyclic Redundancy Code). Ένα κλάδος δικτύου 1-Wire μπορεί να περιέχει 256 συσκευές παντός τύπου , περιλαμβανομένης και της «κύριας» συσκευής (Master).

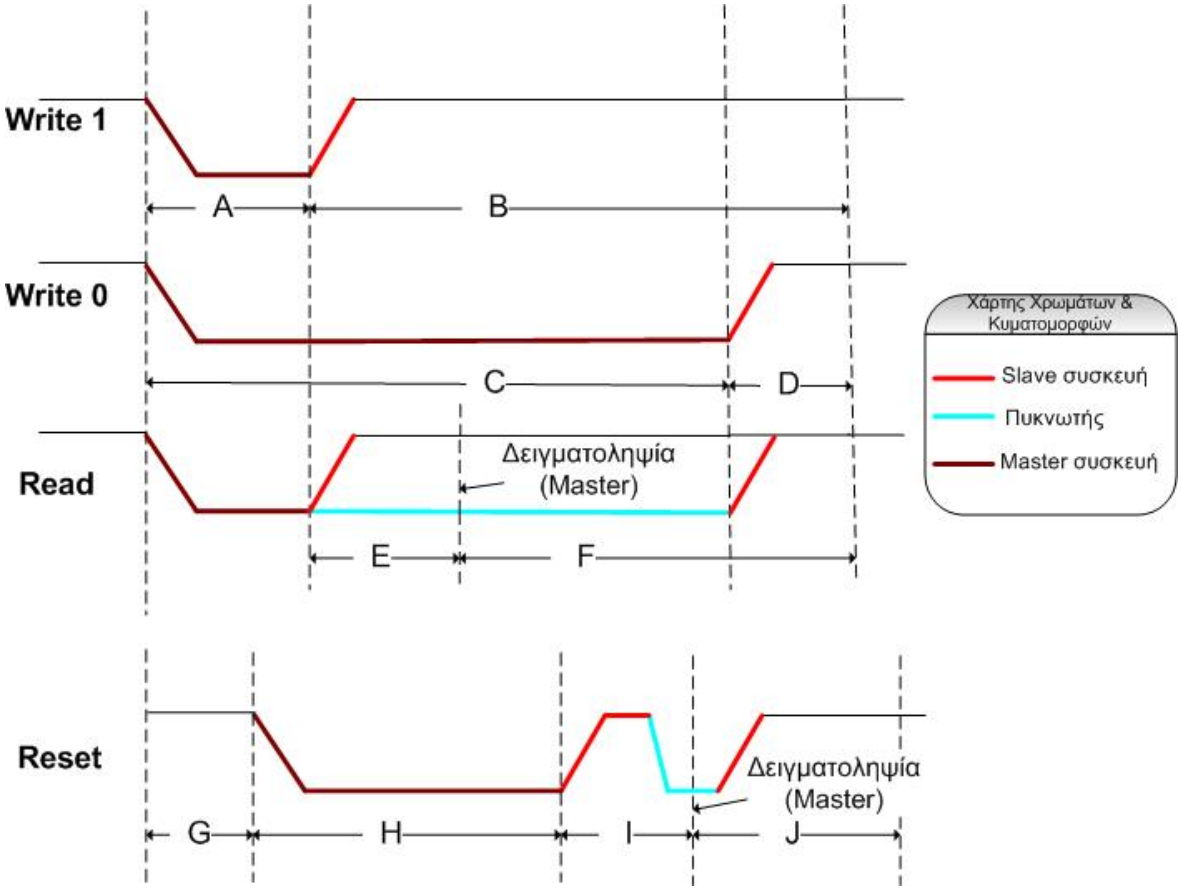
Η επικοινωνία του Master με τους 1-wire slaves γίνεται σειριακά μέσω αποστολής διαφορετικών τάσεων στον δίαυλο, για συγκεκριμένες χρονικές περιόδους. Η επικοινωνία είναι ημι-αμφίδρομη (half duplex). Η αποστολή του λογικού 1 (HIGH) γίνεται με αποστολή τάσης μεγαλύτερης των 2,2V ενώ η αποστολή του λογικού 0 (LOW) γίνεται με την αποστολή τάσης μικρότερης των 0,8V. Οι συσκευές slave που δεν έχουν δική τους τροφοδοσία, εμπεριέχουν έναν πυκνωτή μεγέθους 800pF ώστε να έχουν τάση όση ώρα η γραμμή του διαύλου δεν έχει ρεύμα. Το 1-Wire ανταλλάσσει δεδομένα (δηλαδή λογικά ψηφία 0 ή 1) μέσω χρονοθυρίδων (time slots). Οι χρονοθυρίδες, είναι ο απαραίτητος χρόνος αποστολής ενός συγκεκριμένου ηλεκτρικού παλμού στο δίκτυο, ώστε να θεωρηθεί ως αποδεκτή λογική εντολή (LOGICAL HIGH , LOGICAL LOW READ BIT). Οι χρονοθυρίδες αυτές, είναι 3 ειδών *(1)*:

- χρονοθυρίδα εγγραφής λογικού 1, *Write-1 time slot*
- χρονοθυρίδα εγγραφής λογικού 0, *Write-0 time slot*
- χρονοθυρίδα ανάγνωσης , *Read bit slot*

Το 1-Wire μπορεί να λειτουργήσει σε δύο ταχύτητες, αναλόγως των συσκευών που έχουν συνδεθεί στον δίαυλο. Οι δύο ταχύτητες αυτές είναι η ταχύτητα Standard (16,3 Kbps) και ταχύτητα Overdrive (142 Kbps). Αναλόγως της ταχύτητας λειτουργίας του δικτύου αλλάζουν και οι χρόνοι που απαιτούνται για την κάθε επιμέρους λειτουργία του. Πριν από κάθε λειτουργία στο δίκτυο, πρέπει να προηγηθεί η αποστολή από τον master της εντολής αρχικοποίησης των συσκευών του

δικτύου (ώστε να είναι ανιχνεύσιμες). Στο Σχήμα 2 *(2)* φαίνεται ο τρόπος που το 1-Wire υλοποιεί τις βασικές λειτουργίες του, δηλαδή:

- Αρχικοποίηση & Ανίχνευση Συσκευών (Reset & Presence Detect)
- Αποστολή λογικού 1, (Write 1 bit)
- Αποστολή λογικού 0 (Write 0 bit)
- Ανάγνωση από τις Slave συσκευές του δικτύου του λογικού ψηφίου (Read bit)



Σχήμα 2: Χρονικές σχισμές (Time Slots) 1-Wire.

Πίνακας 1: Χρονικά όρια Παλμών 1-Wire σε κανονική λειτουργία

ΕΠΕΞΗΓΗΣΗ ΣΥΜΒΟΛΩΝ ΕΙΚΟΝΑΣ 2: Χρονικές σχισμές 1-Wire (Time Slots) (Standard Mode)		
ΣΥΜΒΟΛΟ	ΧΡΟΝΟΣ (μs)	Μέγιστος χρόνος (μs)
A	6	15
B	64	Δεν υπάρχει (Δ/Υ)
C	60	120
D	10	(Δ/Υ)
E	9	12
F	55	(Δ/Υ)
G	0	(Δ/Υ)
H	480	640
I	70	78
J	410	(Δ/Υ)

Πίνακας 2: Χρονικά όρια Παλμών 1-Wire σε λειτουργία Overdrive

ΕΠΕΞΗΓΗΣΗ ΣΥΜΒΟΛΩΝ ΕΙΚΟΝΑΣ 2: Χρονικές σχισμές 1-Wire (Time Slots) (Overdrive Mode)		
ΣΥΜΒΟΛΟ	ΧΡΟΝΟΣ (μs)	Μέγιστος χρόνος (μs)
A	1,5	1,85
B	7,5	(Δ/Υ)
C	7,5	14
D	2,5	(Δ/Υ)
F	7	(Δ/Υ)

G	2,5	(Δ/Υ)
H	70	80
I	8,5	8,8
J	40	(Δ/Υ)

Σε ένα κύκλωμα 1-Wire, η επικοινωνία ξεκινά πάντα με την αποστολή από την master συσκευή του σήματος "reset and presence detect", δηλαδή ενός σήματος που «ειδοποιεί» όλες τις συσκευές στο δίκτυο να αρχικοποιηθούν και να στείλουν ένα αναγνωριστικό παλμό/σήμα ώστε να ταυτοποιηθεί η ύπαρξη τους στο δίκτυο. Μία περιληπτική περιγραφή της λειτουργίας και υλοποίησης των τεσσάρων βασικών εντολών-λειτουργιών (σε standard mode) δίνεται παρακάτω: *(4), (5)*

Αρχικοποίηση & Ανίχνευση (Reset & Presence Detect)

Η εντολή αυτή αρχικοποιεί και ανιχνεύει όσες συσκευές τυγχάνει να υπάρχουν στο δίκτυο. Μετά από χρόνο G=0 μs ο δίαυλος παραμένει σε χαμηλή τάση LOW (τάση < 0,8 V) για χρόνο H=480 μs. Μετά από χρόνο I=70 μs γίνεται δειγματοληψία του διαύλου. Εάν υπάρχει (ουν) συσκευή (ές), τότε ο δίαυλος επιστρέφει λογικό 0 (δεν υπάρχει τάση στο δίαυλο) , αλλιώς επιστρέφει λογικό 1 (αποστολή παλμού αναγνώρισης).

Εγγραφή λογικού 1 (Write 1 bit)

Η εντολή αυτή, «γράφει» το λογικό ψηφίο 1 (binary digit 1), σε μία συσκευή του δικτύου. Ο δίαυλος παραμένει σε χαμηλή τάση LOW, για χρόνο A=6 μs και μετά απελευθερώνει τον δίαυλο παραμένοντας αδρανής για χρόνο B=64 μs.

Εγγραφή λογικού 0 (Write 0 bit)

Η εντολή αυτή, «γράφει» το λογικό ψηφίο 0 (binary digit 0), σε μία συσκευή του δικτύου. Ο δίαυλος παραμένει σε χαμηλή τάση LOW για χρόνο C=60 μs και μετά απελευθερώνεται παραμένοντας αδρανές για χρόνο D=10 μs.

Ανάγνωση λογικού ψηφίου (Read bit)

Η εντολή αυτή, «διαβάζει» το λογικό ψηφίο (0 ή 1) (binary digit 0 or 1), που στέλνει μία συσκευή του δικτύου. Ο δίαυλος παραμένει σε χαμηλή τάση LOW για χρόνο A=6 μs και μετά

απελευθερώνεται παραμένοντας αδρανής για χρόνο E=9 μs. Μετά γίνεται δειγματοληψία του διαύλου, ώστε να διαβάσει το ψηφίο που έστειλε η slave συσκευή. Μετά παραμένει αδρανές για χρόνο F=55 μs.

1.3. Σύνοψη και Αξιολόγηση

Το 1-Wire® αφορά στη διασύνδεση συσκευών, μέσω ενός καλωδίου (wire), πάνω από το οποίο περνούν μαζί τροφοδοσία και δεδομένα. Όλες οι συσκευές 1-Wire έχουν έναν μοναδικό 64μπιτο αριθμό γραμμένο σε μνήμη ROM, ενώ υπάρχουν δύο ειδών βασικές συσκευές στο κύκλωμα: Η «κύρια συσκευή» (master device) και οι «εξαρτώμενες συσκευές» (slave devices). Οι συσκευές στο 1-Wire συνδέονται εν σειρά, και μέσω ενός master – μετατροπέα, μπορούν να συνδέονται μέσω RS232, (σειριακής θύρας), μέσω παράλληλης θύρας (LPT), μέσω USB, είτε υλοποιώντας το πρωτόκολλο http, δηλαδή μέσω ενός web interface. Το μήκος ενός δικτύου 1-wire μπορεί να φτάσει τα 300 μέτρα και υπάρχει προστασία και επιδιόρθωση λαθών (CRC). Ένα κλάδος δικτύου 1-Wire μπορεί να περιέχει 256 συσκευές παντός τύπου και η τοπολογία του να αλλάζει δυναμικά, δηλαδή μπορούν να προσθαφαιρούνται συσκευές σε πραγματικό χρόνο. Η επικοινωνία του Master με τους slaves γίνεται μέσω αποστολής διαφορετικών τάσεων στον δίαυλο και είναι ημι-αμφίδρομη (half duplex).. Το 1-Wire ανταλλάσσει δεδομένα μέσω χρονοθυρίδων (time slots) και μπορεί να λειτουργήσει σε δύο ταχύτητες, αναλόγως των συσκευών που έχουν συνδεθεί στον δίαυλο με ταχύτητες που ξεκινούν από 16,3Kbps και μπορεί να φτάσουν μέχρι 142 Kbps. Σε ένα κύκλωμα 1-Wire, η επικοινωνία ξεκινά πάντα με την αποστολή από την master συσκευή ενός σήματος που «ειδοποιεί» όλες τις συσκευές στο δίκτυο να αρχικοποιηθούν και να στείλουν ένα αναγνωριστικό παλμό/σήμα ώστε να ταυτοποιηθεί η ύπαρξη τους στο δίκτυο. Οι υπόλοιπες βασικές λειτουργίες του δικτύου είναι «Αποστολή λογικού 1», «Αποστολή λογικού 0» , και «Ανάγνωση από τις συσκευές – πελάτες του δικτύου του λογικού ψηφίου».

Ως πλεονεκτήματα του 1-Wire θεωρούνται η δυναμική αλλαγή τοπολογίας στο δίκτυο, δηλαδή η προσθαφαίρεση συσκευών σε πραγματικό χρόνο, η ταχύτητα μεταγωγής των δεδομένων, που για τις εφαρμογές που υλοποιεί μέχρι τώρα θεωρείται αρκετά ικανοποιητική και φυσικά η δυνατότητα του δικτύου να υλοποιηθεί πάνω σε ένα καλώδιο (1 wire) οποιουδήποτε είδους, κάτι που συμπιέζει το κόστος υλοποιήσης προς τα κάτω.

Μειονεκτήματα του 1-Wire θα μπορούσαν να θεωρηθούν η μη ύπαρξη αναγνωρισμένου προτύπου αφού είναι προϊόν μίας εταιρείας (Dallas Semiconductors). Επίσης η σχετικά χαμηλή ταχύτητα που προσφέρει ο δίαυλος (ταχύτητα standard) μπορεί να αποτελέσει πρόβλημα για όποιες σύνθετες και απαιτητικές μελλοντικές εφαρμογές και υπηρεσίες. Τέλος, η κακή ποιότητα του κώδικα για τις επιμέρους συσκευές, που διανέμεται δωρεάν από την εταιρεία, καθιστά τη χρήση του 1-wire αρκετά φιλόδοξο εγχείρημα. Αυτό όμως είναι κάτι που φιλοδοξεί να ανατρέψει η παρούσα εργασία.

Τα κυριότερα ανταγωνιστικά προϊόντα με το 1-Wire είναι το I^2C^* και το MicroWirePLUS^{tm} *(9)*. Επίσης στην ίδια κατηγορία βρίσκονται και τα πρωτόκολλα MBus(EN1434), CAN(ISO11838), LIN Bus, SMBUS^{TM} , SPI^{TM} **.** Στον πίνακα που ακολουθεί αναλύονται οι κυριότερες παράμετροι των ανταγωνιστικών πρωτοκόλλων και συγκρίνονται με αυτές του 1-Wire *(9)*.

1.4. Συγκριτικός Πίνακας Σειριακών Πρωτοκόλλων

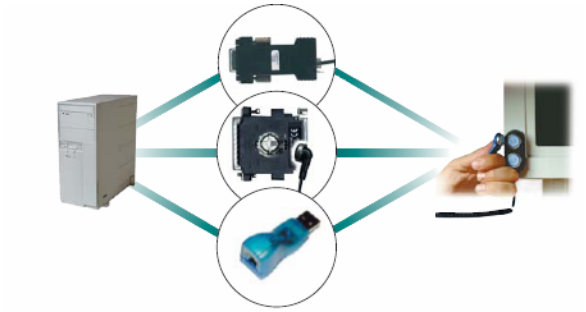
	1-Wire	I^2C^*	SMBUS™	SPI™	MicroWirePLUS™	MBus(EN1434)	CAN(ISO11838)	LIN Bus
Αριθμός καλωδίων που απαιτούνται	1 E/E	2	2	4	4	2	2	1
Μέγεθος δικτύου	300μ	Περιορίζεται από την απαίτηση 400pF φορτίο του BUS	Περιορίζεται από την απαίτηση 400pF φορτίο του BUS	Δ/Υ	Δ/Υ	350μ και 250 slave συσκευές	40μ 1Mbps 1000μ 50 kbps (Παράδειγμα)	40μ 10nF Μέγιστο
Διευθυνσιοδότηση συσκευών	56bits	7 bits	7 bits	Δ/Υ	Δ/Υ	8 bits	Ταυτοποίηση μηνύματος 11 bits & 29 bits	Ταυτοποίηση μηνύματος 8 bits + 2 parity bits
Τοπολογία δικτύου	Δυναμική αλλαγή	Δ/Υ	Πρωτόκολλο εύρεσης διευθύνσεων	Δ/Υ	Δ/Υ	Αυτόματη	Δ/Υ	Δ/Υ
Ταχύτητα μεταφοράς δεδομένων	Standard -16,3 kbps Overdrive 142 kbps	Standard 100kbps Fast 400kbps High 3,4Mbps	100 kbps	Αναλόγως συσκευής (μέχρι 100 kbps)	Αναλόγως συσκευής (μέχρι 100 kbps)	9600 bps	1 Mbps	20 kbps
Προστασία δεδομένων	8 & 16 bit CRC	Ναι	Ναι	Δ/Υ	Δ/Υ	Ναι	Ναι (CSMA/CD)	Ναι
Παροχή ενέργειας	Παρασιτική & αφοσιωμένη	αφοσιωμένη	αφοσιωμένη	αφοσιωμένη	αφοσιωμένη	Παρασιτική και/ή αφοσιωμένη	Παρασιτική και/ή αφοσιωμένη	Παρασιτική

1.5. Χρήσεις του 1-Wire

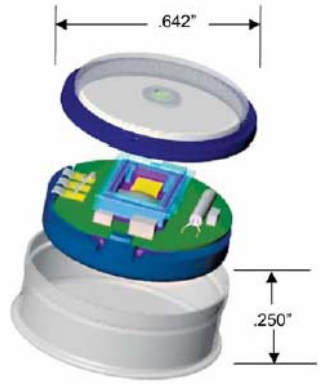
Η Dallas Semiconductors έχει στην γραμμή παραγωγής πάνω από 300 συσκευές-προϊόντα 1-Wire *(Dallas Semiconductors/Maxim Dallas Product Line Card, 22.05.06),* καλύπτοντας μία ευρύτατη γκάμα. Υπάρχουν αισθητήρες διαφόρων ειδών (πίεσης , υγρασίας, θερμοκρασίας αλλά και οξειδίων, αερίων κτλ) , ενισχυτές ήχου, μετατροπείς τάσεως , φίλτρα (αναλογικά), μικρο-ελεγκτές, αναλογικοί πολυπλέκτες και διακόπτες και πολλά άλλα προϊόντα. Όλα αυτά μπορούν να συνδυαστούν μεταξύ τους σε ένα δίκτυο 1-Wire.

1.5.1. Υλοποιήσεις iButton

Μία από τις κυριότερες και πιο επιτυχημένες εμπορικά εφαρμογές του 1-Wire της Dallas Semiconductors είναι η υλοποίηση 1-wire συσκευών σε iButton.



Εικόνα 2: Διασύνδεση iButton



Εικόνα 3: Ανατομία iButton

Το iButton είναι ένα ηλεκτρονικό chip το οποίο είναι «σφραγισμένο» μέσα σε ένα ανοξείδωτο περίβλημα 16mm και μπορεί να επικοινωνήσει με πρωτόκολλο 1-wire. Έχει κατασκευαστεί έτσι ώστε να μπορεί να ταξιδέψει οπουδήποτε, με έναν άνθρωπο ή ένα προϊόν, είτε έχοντας ήδη αποθηκευμένη την πληροφορία, ή αποθηκεύοντας την πληροφορία κατά τακτά (παραμετροποιήσιμα) χρονικά διαστήματα κατά την διάρκεια του ταξιδιού *(Βλέπε ΑΠΟΣΦΑΛΜΑΤΩΣΗ & ΠΙΛΟΤΙΚΗ ΕΦΑΡΜΟΓΗ συσκευή DS1921)*. Μπορεί να λειτουργήσει σε εχθρικά περιβάλλοντα (σκόνη, υγρασία, βροχή κτλ) και λόγω μεγέθους μπορεί να μεταφερθεί εύκολα όπως π.χ. πάνω σε ένα μπρελόκ, και μπορεί να χρησιμοποιηθεί σαν «κλειδί» εισόδου σε κτίρια ή Η/Υ.

Κάθε iButton έχει μία παγκοσμίως μοναδική διεύθυνση (αντίστοιχη της Mac Address των καρτών δικτύου των Η/Υ) η οποία είναι χαραγμένη με laser στο εσωτερικό του τσιπ και δεν μπορεί να αλλοιωθεί. Άρα μπορεί να χρησιμοποιηθεί σαν το μοναδικό αναγνωριστικό όνομα για κάθε διαφορετικό iButton. Οι βασικές κατηγορίες που χωρίζονται τα iButtons είναι:

- Μόνο διεύθυνση
- Μνήμης
- Ρολόι πραγματικού χρόνου
- Ασφάλειας
- Θερμοκρασίας

Το ατσάλινο περίβλημα του iButton, είναι το μέσο επικοινωνίας του με την όποια μονάδα 1-wire master. Απλώς χρησιμοποιούμε ως «θετική επαφή» ("Lid" data contact) το πάνω μέρος του, και ως «γείωση» ("Base" ground contact) το κάτω μέρος του, ώστε να μπορέσουμε να έχουμε κύκλωμα. Με αυτόν τον τρόπο είναι πολύ εύκολο να το συνδέσουμε σε δύο οποιεσδήποτε επαφές και να επικοινωνήσουμε με το τσιπ. Εάν συνδέσουμε ένα οποιοδήποτε καλώδιο με τις δύο επαφές του iButton μπορούμε να επικοινωνήσουμε μαζί του με το πρωτόκολλο 1-wire.

Εικόνα 4: επικοινωνία iButton με συσκευή βάσης

Η πληροφορία από και προς τα iButton μεταφέρεται από και προς τον Η/Υ με στιγμιαία επαφή, με ταχύτητα μέχρι 142kbps. Ακουμπάμε το iButton στην ειδική επαφή (Blue Dot receptor) που είναι συνδεδεμένη σε έναν Η/Υ. Ο Blue Dot receptor είναι συνδεδεμένος με το κύκλωμα 1-Wire μέσω του ειδικού 1-Wire μετατροπέα (adaptor) που είναι συνδεδεμένος στην σειριακή, παράλληλη ή USB πόρτα ενός Η/Υ. Το τσιπ στο εσωτερικό του iButton είναι

προστατευμένο από το ατσαλένιο περίβλημα. Μπορούμε να το πετάξουμε, να το πατήσουμε, να το χαράξουμε χωρίς να αλλοιωθεί η πληροφορία που περικλείει.

Εικόνα 5: Διάφορες υλοποιήσεις iButton

Εικόνα 6: Διάφορες υλοποιήσεις iButton

Το iButton είναι ιδανικό εκεί που χρειάζεται η πληροφορία να ταξιδέψει με ένα αντικείμενο ή με ένα πρόσωπο. Μπορεί να κρεμαστεί σε ένα μπρελόκ κλειδιών, να ενσωματωθεί σε ένα δαχτυλίδι, ή ρολόι και να χρησιμοποιηθεί σαν κλειδί εισόδου σε κτίρια, Η/Υ, οχήματα ή και χρήσης εργαλείων και μηχανημάτων. Μπορεί επίσης να μετρήσει διάφορες παραμέτρους σε γραμμές παραγωγής ή μεταφοράς. Κάποιες εκδόσεις του μπορούν να χρησιμοποιηθούν για μικρού κόστους συναλλαγές όπως μέσα μεταφοράς, παρκόμετρα ή αυτόματους πωλητές. Μπορεί επίσης να χρησιμοποιηθεί για να αναβρίσκονται - παρακολουθούνται πολύτιμα αντικείμενα μέσα στον χώρο, όπως π.χ. ιατρικά εργαλεία μέσα σε νοσοκομεία ή αντικείμενα αξίας ή μικρού μεγέθους σε εργαστήρια κτλ.

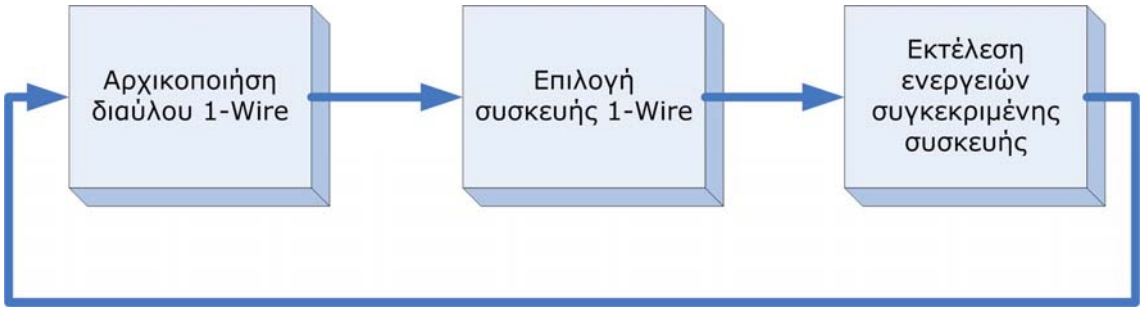
Εάν θέλουμε να συγκρίνουμε το iButton έναντι των ετικετών Barcodes ή των μαγνητικών καρτών, το πλεονέκτημα τους είναι ότι μπορούμε όχι μόνο να διαβάσουμε αλλά και να γράψουμε/ αποθηκεύσουμε πληροφορία στο iButton. Έχει μεγαλύτερη ταχύτητα μετάδοσης και φυσικά μεγάλη αντοχή αφού πρακτικά δεν καταστρέφεται κάτω από σκληρές συνθήκες εργασίας. Είναι απόλυτα ασφαλές όταν υπάρχουν αξιώσεις ταυτοποίησης. Επίσης είναι πολύ φτηνό αν σκεφτούμε ότι το κόστος κάθε iButton ξεκινάει από 2€ περίπου και το κόστος της συσκευής I/O (Master Device) δεν ξεπερνάει τα 35€. *(7),(8)*

2. Υφιστάμενη κατάσταση

2.1. Περιγραφή Αρχών Λειτουργίας

Γνωρίζοντας ότι η κάθε συσκευή 1-wire εμπεριέχει (χαραγμένη με laser) μία μοναδική διεύθυνση, αντιλαμβανόμαστε πως η όποια επικοινωνία και ανταλλαγή δεδομένων με τις συσκευές γίνεται βάσει αυτού του αριθμού. Για να γίνει αυτό , πριν από κάθε επικοινωνία με τις συσκευές, ο master στέλνει ένα σήμα αρχικοποίησης. Ακολουθεί η επιλογή της συγκεκριμένης συσκευής (ή όλων των συσκευών στο δίαυλο αν αυτό είναι επιθυμητό) και ακολουθεί η συγκεκριμένη εντολή-ενέργεια προς την επιλεγμένη συσκευή.

Σε όλα τα διαφορετικά API που έχει δημιουργήσει η εταιρεία για διαφορετικές master συσκευές και διαφορετικά Λειτουργικά Συστήματα, υπάρχουν πολλά κοινά σημεία. Αυτό συμβαίνει γιατί οι λειτουργίες που χρειάζονται να γίνουν στον δίαυλο, αλλά και στις όποιες συσκευές, είναι αυτές που περιγράφονται στο Σχήμα 3.



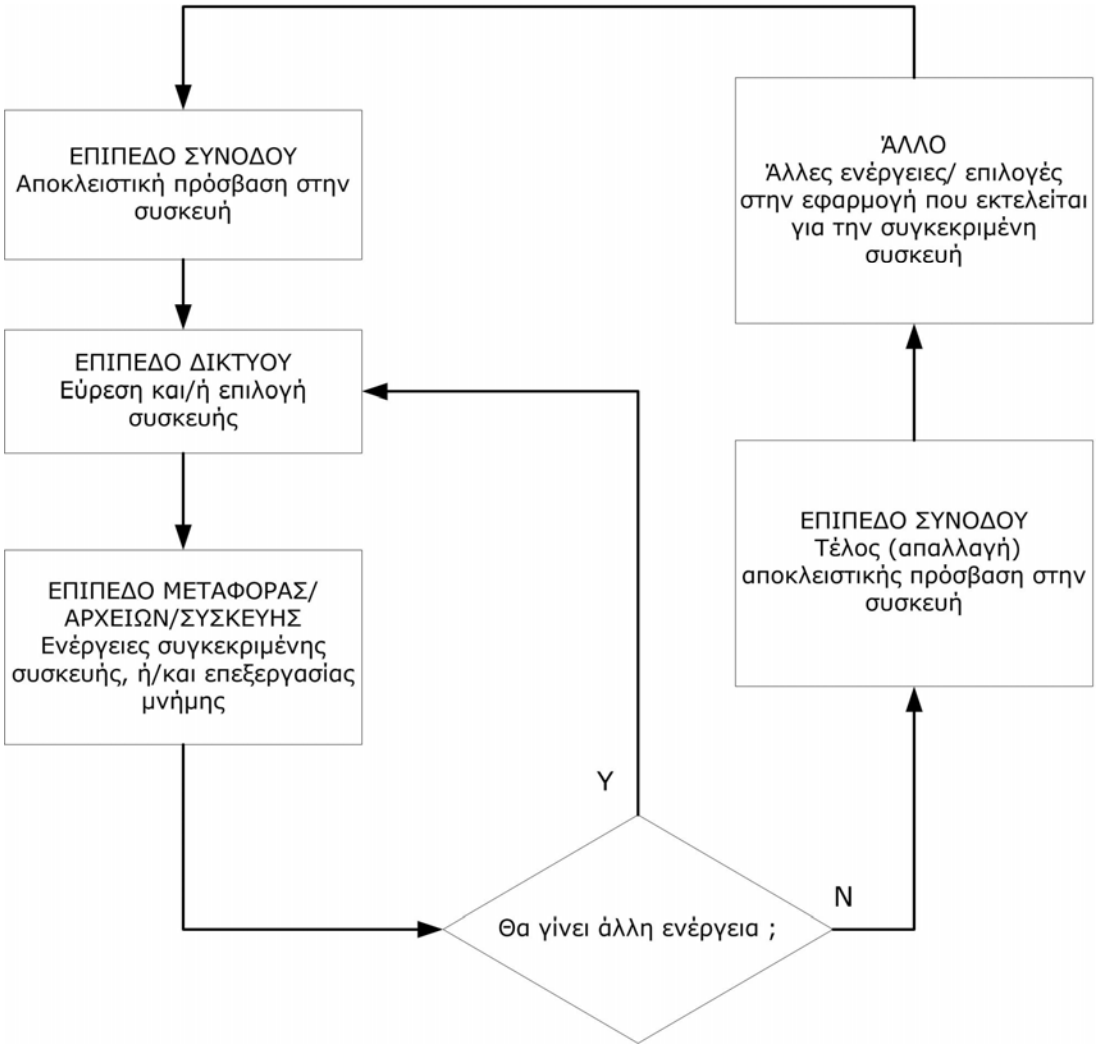
Σχήμα 3: Τυπική Ροή εργασιών στο 1-Wire *(10)*

Αυτές είναι: αρχικοποίηση συσκευών, επιλογή συγκεκριμένης συσκευής και αποστολή εντολών στην συγκεκριμένη συσκευή,. Οι λειτουργίες υλοποιούνται χρησιμοποιώντας τις 4 βασικές εντολές που περιγράφονται στην 1.2, δηλαδή την αρχικοποίηση και ανίχνευση συσκευής, τις εντολές εγγραφής λογικού 1, εγγραφής λογικού 0, και την εντολή ανάγνωσης λογικού ψηφίου. Όλες οι συναρτήσεις που έχουν δημιουργηθεί βασίζονται πάνω σε αυτές τις πρωταρχικές λειτουργίες, και μπορούν να αντιστοιχηθούν/ κατηγοριοποιηθούν σε 6 διαφορετικά επίπεδα. Υπάρχει δηλαδή μερική αντιστοίχηση με τα 7 επίπεδα του OSI *(Walrand Jean (1997) «Δίκτυα Επικοινωνιών», σελ 156-163)* θεωρώντας ότι το επίπεδο διασύνδεσης (link) έχει συγχωνευθεί με το φυσικό επίπεδο, το επίπεδο συσκευής αντιστοιχεί στο επίπεδο εφαρμογής και το επίπεδο αρχείου στο επίπεδο παρουσίασης. Αυτά τα έξι επίπεδα περιγράφονται περιληπτικά στον Πίνακας 3 που ακολουθεί:

Πίνακας 3: Τα 6 επίπεδα συναρτήσεων API *(10)*

(ΕΠΙΠΕΔΟ ΣΥΝΟΔΟΥ-SESSION
Αναλαμβάνει να ορίσει την αποκλειστική χρήση του διαύλου. Συνήθως είναι απαραίτητη ως διαδικασία όταν η συσκευή/ές που χρησιμοποιούνται εκείνη την στιγμή στον δίαυλο, χρειάζεται να έχουν την αποκλειστική χρήση του
ΕΠΙΠΕΔΟ ΔΙΑΣΥΝΔΕΣΗΣ- LINK
Περιλαμβάνει όλες τις πρωτογενείς λειτουργίες που συμβαίνουν στον δίαυλο. Όλες οι λειτουργίες που εκτελούν οι συσκευές μπορούν να αναλυθούν σε αρχικοποίηση και εγγραφή/ ανάγνωση λογικού ψηφίου
ΕΠΙΠΕΔΟ ΔΙΚΤΥΟΥ - NETWORK
Λειτουργίες που αφορούν την ανεύρεση και επιλογή των συσκευών του διαύλου. Η μοναδική «διεύθυνση» της κάθε συσκευής χρησιμοποιείται ως διεύθυνση δικτύου. Οι συναρτήσεις που χρησιμοποιούνται εδώ μπορούν να δημιουργηθούν από τις συναρτήσεις του επιπέδου διασύνδεσης. Μερικές συσκευές Master εμπεριέχουν κάποιες από αυτές τις εντολές, ως πιο αποτελεσματικές από τις συναρτήσεις του επιπέδου διασύνδεσης.
ΕΠΙΠΕΔΟ ΜΕΤΑΦΟΡΑΣ - TRANSPORT
Συναρτήσεις αποκλεισμού επικοινωνίας (όταν έχει επιλεγεί μία συσκευή, οι υπόλοιπες πρέπει να σταματήσουν-αποκλειστούν από την master συσκευή μέχρι να τελειώσει η επικοινωνία με την επιλεγμένη συσκευή) και συναρτήσεις πρωτογενών λειτουργιών εγγραφής/ ανάγνωσης μνήμης (EPROM). Οι συναρτήσεις αυτές κατασκευάζονται από τις συναρτήσεις επιπέδου δικτύου και επιπέδου διασύνδεσης.
ΕΠΙΠΕΔΟ ΑΡΧΕΙΩΝ - FILE
Αυτές οι συναρτήσεις χρησιμοποιούνται μόνο όταν η συσκευή έχει μνήμη μεγαλύτερη της μίας σελίδας και κατασκευάζονται από τις συναρτήσεις επιπέδου δικτύου και επιπέδου μεταφοράς.
ΕΠΙΠΕΔΟ ΣΥΣΚΕΥΗΣ - DEVICE
Οι συναρτήσεις αυτές (λεγόμενες και υψηλού επιπέδου) έχουν δημιουργηθεί για συγκεκριμένη συσκευή και φυσικά εμπλέκουν όλες τις προηγούμενες κατηγορίες στην δημιουργία τους. Μπορούν για παράδειγμα να διαβάσουν την θερμοκρασία από έναν αντίστοιχο αισθητήρα 1-Wire

Η τυπική αλληλουχία χρήσης των παραπάνω κλάσεων συναρτήσεων περιγράφεται στο επόμενο διάγραμμα ροής (Σχήμα 4):



Σχήμα 4: Διάγραμμα ροής βασικών συναρτήσεων *(10)*

Εδώ πρέπει να αναφερθεί ότι οι master συσκευές αναλόγως του πρωτοκόλλου με το οποίο υλοποιούν την διασύνδεση τους με τον Η/Υ (ή την συσκευή επεξεργασίας εν γένει) διαφέρουν ριζικά ως προς τις δυνατότητες τους, όπως επίσης ως προς τον τρόπο με τον οποίο επικοινωνούν και με τον δίαυλο 1-Wire, και με την συσκευή διασύνδεσης (Η/Υ). Έτσι π.χ. οι συσκευές που συνδέονται με τον Η/Υ μέσω της RS232 (Σειριακή θύρα, COM1 & COM2) έχουν σημαντικότατους περιορισμούς μεταγωγής δεδομένων και εντολών αφού λόγω των FIFO Buffers πρέπει να περάσει κάποιος χρόνος ανάμεσα στην αποστολή συνεχόμενων

δεδομένων στην COM εάν γεμίσει η buffer (*13)*. Η υποστήριξη από την άλλη μεριά τέτοιων συσκευών είναι εγγενής από όλα τα λειτουργικά συστήματα (λόγω διασύνδεσης σε RS232, υπάρχουν πρωτόκολλα μεταφοράς δεδομένων , διόρθωσης λαθών κτλ) και έτσι δεν χρειάζεται η εγκατάσταση «οδηγών» (drivers) για τέτοιου είδους συσκευές. Από την άλλη μεριά , οι master συσκευές υλοποιούν το πρωτόκολλο USB σύνδεσης, είναι ταχύτερες (και απολύτως αξιόπιστες όπως αποδείχθηκε στην πράξη), αλλά χρειάζεται να εγκατασταθούν οι «οδηγοί» της συσκευής για το συγκεκριμένο Λειτουργικό Σύστημα, κάτι που προφανώς μειώνει την χρηστικότητα τους.

2.2. Κατηγοριοποιήση Υπάρχοντος Κώδικα

Την στιγμή που ξεκίνησε η εκπόνηση της πτυχιακής εργασίας υπήρχαν 5 διαφορετικά API για το 1-Wire κύκλωμα εκ των οποίων για τα 4 από αυτά διατίθετο δωρεάν ο πηγαίος κώδικας από την εταιρεία Dallas Semiconductors και για τα οποία υπήρχε τεκμηρίωση. Το καθένα λειτουργεί/ δημιουργήθηκε για διαφορετικές πλατφόρμες και έχει υλοποιηθεί με διαφορετικές γλώσσες προγραμματισμού *(10)*. Στον πίνακα που ακολουθεί φαίνονται αυτά τα API , με σύντομη περιγραφή:

Πίνακας 4: Τα 5 βασικά API με περιληπτική περιγραφή

API	ΠΕΡΙΓΡΑΦΗ
1-Wire Public Domain	Γλώσσα C για όλες τις πλατφόρμες PC Περιέχει ένα low level API γραμμένο εξ ολοκλήρου σε γλώσσα C. Επειδή είναι γραμμένο σε C , έχει την δυνατότητα της μεταφερσιμότητας σε πολλές διαφορετικές πλατφόρμες και λειτουργικά συστήματα. Εμπεριέχει κώδικα και διάφορα παραδείγματα εφαρμογών. Επίσης περιέχει τους αλγόριθμους για τις βασικές λειτουργίες (reset/presence detect, byte I/O, and bit I/O). Υποστηρίζει συγκεκριμένες master συσκευές, την DS9097U σειριακή και την DS9490 USB. Αυτό το SDK απευθύνεται προφανώς σε μηχανές που τρέχουν κάποια έκδοση των Windows. υπάρχουν πολλές πιλοτικές εφαρμογές , γραμμένες σε διάφορες γλώσσες όπως VB, C, Delphi, Jscript , κτλ. Υπάρχει HTML τεκμηρίωση για την χρήση του TMEX API και του -Wire COM. Προαιρετικά υποστηρίζει

	και την πλατφόρμα .NET
1-Wire API για Java	Υποστηρίζει σχεδόν όλες τις συσκευές master, και κάποια USB με βιβλιοθήκες ανεξάρτητες πλατφόρμας. Εμπεριέχει πολλές java κλάσεις και interfaces. Εμπεριέχει την υλοποίηση ενός virtual 1-Wire adapter για χρήση σε TCP/IP δίκτυο. Υποστηρίζει τα iButtons και τις περισσότερες 1-Wire συσκευές της Dallas Semiconductorς. Υποστηρίζει συγκεκριμένες master συσκευές τις DS9097U σειριακή και την DS9490 USB
1-Wire API για .NET	Βιβλιοθήκες δημιουργημένες σε #J. (.NET). Υποστηρίζει Windows CE
1-Wire COM	Windows Component Object Model (COM). Υποστηρίζει Windows με Java scripts και Visual Basic Scripts και φυσικά χρειάζεται java virtual machine για να «τρέξει»
TMEX API	«κλειστό» API που χρειάζεται άλλο API για να έχει πλήρη έλεγχο των συσκευών του δικτύου (δεν υπάρχει πηγαίος κώδικας)

΄Οπως είναι φανερό, υπάρχουν master συσκευές για κάθε είδους πρωτόκολλο και για την κάθε μία από αυτές υπάρχουν εκτός από τους «οδηγούς» με τους οποίους επικοινωνεί μαζί τους ο Η/Υ, και συγκεκριμένες συναρτήσεις για να μπορέσουν να συνδεθούν και να διαχειριστούν τον δίαυλο 1-wire αφού η κάθε master συσκευή αναλόγως του πρωτοκόλλου με το οποίο επικοινωνεί με τον Η/Υ έχει και πολλούς περιορισμούς και ιδιαιτερότητες στην σύνδεση και προς τις δύο μεριές. (και προς τον Η/Υ και προς τον δίαυλο 1-Wire).

Ακολουθεί συνοπτικός πίνακας με τους προσαρμογείς-master συσκευές που κυκλοφορούν αυτή την στιγμή στην αγορά και με την θύρα στην οποία συνδέεται. Ο κωδικός αριθμός τους είναι και το όνομα τους (*11)*.

Πίνακας 5: Master συσκευές- μετατροπείς 1-Wire

Κωδικός	Περιγραφή	Δυνατότητες
DS1402D-DB8	Blue Dot Receptor, Parallel Port Παράλληλη θύρα	Parallel Port Adapter Cables, Blue Dot, F5 Package Compatible

DS1410E-001	Parallel Port Adapter Παράλληλη θύρα	With ID, F5 Package Compatible
DS1411-009	Serial Port Button Holder Σειριακή θύρα	With ID, F5 Package Compatible
DS1411-S09	Serial Port Button Holder Σειριακή θύρα	F5 Package Compatible
DS1413	Passive Serial Port Button Holder Σειριακή θύρα	F5 Package Compatible
DS9097	Serial Port Adapter Σειριακή θύρα	-
DS9097E	Serial Port Adapter Σειριακή θύρα	EPROM Compatible Adapter
DS9097U-009	Universal Serial Port Adapter Σειριακή θύρα	With ID
DS9097U-E25	Universal Serial Port Adapter Σειριακή θύρα	EPROM Compatible Adapter
DS9097U-S09	Universal Serial Port Adapter Σειριακή θύρα	-
DS9490B	USB Button holder Θύρα USB	USB Port, With ID, F5 Package Compatible
DS9490R	USB to 1-Wire RJ11 Θύρα USB	USB Port, With ID
DS1410E	LPT port Παράλληλη θύρα	LPT port

Το πληρέστερο και εκτενέστερο από τα API που περιγράφονται στον Πίνακα 4, είναι το 1-Wire Public Domain. Αυτό περιέχει κατηγορίες συναρτήσεων και βιβλιοθηκών που περιγράφονται στον πίνακα *(10)*.

Πίνακας 6: αναλυτικά περιεχόμενα του 1-Wire Public Domain

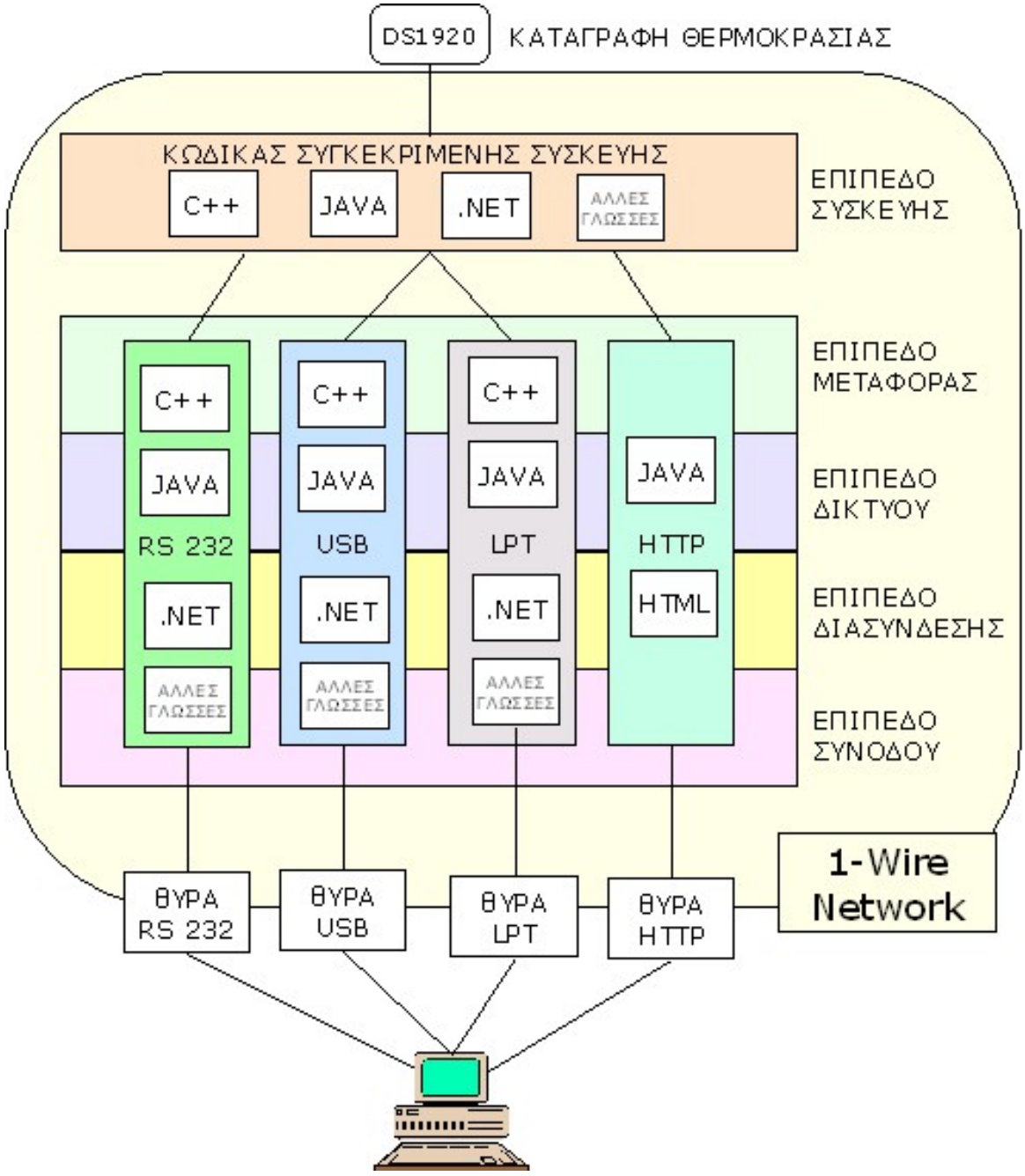
Κατηγορία πηγαίου κώδικα (builds)	**Πλατφόρμα**	**Θύρα-Πρωτόκολλο**	**Περιγραφή**
Userial	Win32, Win16, DOS, WinCE, Pocket PC Linux,	COM	Υποστηρίζει την master συσκευή DS9097U και όλες όσες βασίζονται στο chip

	UNIX, DS550		DS2480B
General	Win32	LPT	Υποστηρίζει την master συσκευή DS1410E (παράλληλη θύρα)
Other "libusb"	Win32, Linux, Macintosh, Unix	USB	Υποστηρίζει την master συσκευή DS9490 USB και όλες όσες βασίζονται στο chip DS2490 USB συσκευές εφόσον έχουν εγκατασταθεί οι αντίστοιχοι «οδηγοί»
Other "usb"	Win32	USB	Υποστηρίζει την master συσκευή DS9490 USB και όλες όσες βασίζονται στο chip DS2490 USB συσκευές εφόσον έχει εγκατασταθεί ο αντίστοιχος «οδηγός» DS2490.sys
Other "wrapper" TMEX	Win32	USB, LPT, COM	Εμπερικλείει το TMEX API το οποίο μπορεί να δουλέψει με πολλές master συσκευές (δεν υπάρχει πηγαίος κώδικας για αυτό)
Other "multi port"	Win32	USB, LPT, COM	Δίνει πρόσβαση σε όλες τις θύρες , χρησιμοποιώντας τους οδηγούς χαμηλού επιπέδου των Windows 32

Για τις http συσκευές (εν προκειμένω για την HA7net της Embedded Data Systems η οποία βρισκόταν στην κατοχή μας την εποχή που ξεκίνησε η πτυχιακή εργασία- Οκτώβριος 2005) , υπάρχουν από την μεριά των εταιρειών που τις κατασκευάζουν, μόνον οι βασικές εντολές low level προγραμματισμού *(Embedded Data Systems, Low Level 1-Wire support, 29.10.05)*. Π.χ. εντολές ανάγνωσης/ εγγραφής bit και byte κτλ *(Παράρτημα Γ Εντολές που υποστηρίζονται από τον HA7net)*. Απουσιάζουν παντελώς εντολές υψηλού επιπέδου, και φυσικά οποιαδήποτε δυνατότητα παραμετροποίησης 1-Wire αισθητήρων.

Η υπάρχουσα βιβλιοθήκη της Dallas Semiconductor σχετικά με τον http server HA7Net *(Παράρτημα Γ, Υπάρχουσα βιβλιοθήκη για τον HA7net)* περιείχε μόνο τις γραμμές εντολών προς τον http server για κάθε συνάρτηση και τίποτα άλλο. Επίσης περιείχε λάθη όπως π.χ. εντολή While την οποία ακολουθούσε else, η οποία ως γνωστό είναι λανθασμένη δήλωση στη C++. Επίσης ο έλεγχος της ορθότητας της IP διευθύνσεως θεωρήθηκε αναποτελεσματικός και λανθασμένος εν γένει. *(1), (21)*. Προσθέτοντας και τα πολλά συντακτικά λάθη και τις λάθος δηλώσεις στις συναρτήσεις η βιβλιοθήκη (HA7Net.c) έπρεπε να γραφτεί εξολοκλήρου από την αρχή.

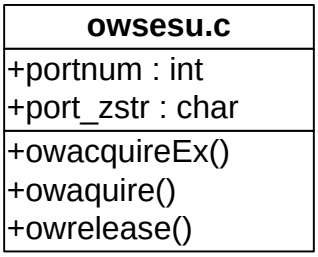
Στο Σχήμα 5 φαίνεται αναλυτικά ο χάρτης της υπάρχουσας κατάστασης όσον αφορά στον κατακερματισμό του υπάρχοντος λογισμικού. Έχουμε διαφορετικά API τα οποία απευθύνονται-ενεργοποιούν συγκεκριμένες master συσκευές, και για τα οποία υπάρχει γραμμένος κώδικας για κάποιες από τις συσκευές-πελάτες. Έτσι παρατηρείται το φαινόμενο, να κατέχουμε δύο συσκευές (π.χ. έναν καταγραφέα θερμοκρασίας πραγματικού χρόνου DS1920 και έναν καταγραφέα υγρασίας DS1923) και να μην μπορούμε να «δούμε» τις δύο συσκευές μέσα από ένα πρόγραμμα γιατί η συσκευή DS1923 υποστηρίζεται μόνο από το πρόγραμμα σε JAVA που έχει κατασκευάσει η εταιρεία, και επίσης υπάρχουν οδηγοί και κώδικας καταγραφής των δεδομένων της συγκεκριμένης συσκευής (επίπεδο συσκευής) μόνο στο "Userial" κομμάτι του API "1-wire public domain".



Σχήμα 5 : Αναλυτική καταγραφή υπάρχουσας κατάστασης

2.3. Αναλυτική περιγραφή συναρτήσεων 4 επιπέδων

Αναλυτικά οι περιεχόμενες συναρτήσεις της κάθε βιβλιοθήκης είναι *(12)* :



Σχήμα 6: UML διάγραμμα ownet.c

2.3.1. Συναρτήσεις Επιπέδου Συνόδου

Οι συναρτήσεις του επιπέδου περιέχονται στο αρχείο-βιβλιοθήκη **owSesU.C** *(ΠΑΡΑΡΤΗΜΑ Α)*

Η βιβλιοθήκη αυτή εμπεριέχει 2 συναρτήσεις:

owAcquire

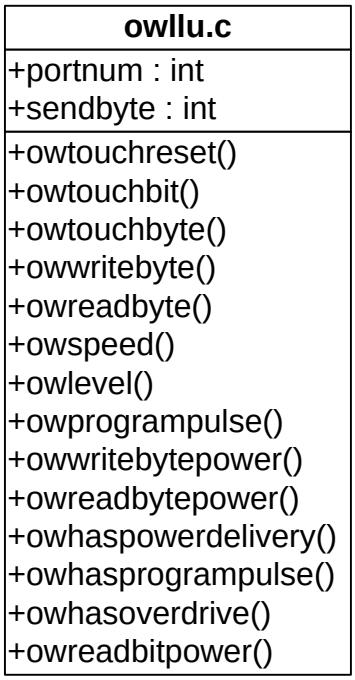
SMALLINT owAcquire(int portnum, char *port_zstr)

Η συνάρτηση owAcquire, «βρίσκει» και «κλειδώνει» υπό την κατοχή της, ένα δίκτυο 1-wire.

owRelease

void owRelease(int portnum)

Η συνάρτηση owRelease, ελευθερώνει ένα δίκτυο που πριν είχε «ανακαλυφθεί» από την owAcquire.



Σχήμα 7: UML διάγραμμα ownet.c

2.3.2. Συναρτήσεις Επιπέδου Διασύνδεσης

Οι συναρτήσεις του επιπέδου περιέχονται στο αρχείο-βιβλιοθήκη **owLLU.C** *(ΠΑΡΑΡΤΗΜΑ Α)*.

Η βιβλιοθήκη αυτή περιέχει 13 συναρτήσεις:

owTouchReset

SMALLINT owTouchReset(int portnum)

Η συνάρτηση owTouchReset επαναφέρει το 1-wire δίκτυο στην αρχική κατάσταση (Reset function) ώστε να είναι έτοιμο να πάρει εντολές:

owTouchBit

SMALLINT owTouchBit(int portnum, SMALLINT sendbit)

Η συνάρτηση owTouchBit στέλνει ένα μπιτ στο δίκτυο και επιστρέφει το αποτέλεσμα αναλόγως από την απάντηση του δικτύου (ποια ή ποιες συσκευές απάντησαν.

owWriteByte

SMALLINT owWriteByte(int portnum, SMALLINT sendbyte)

Η συνάρτηση owWriteByte εξετάζει την δυνατότητα της/των συσκευών του δικτύου να «γράφει» σωστά ένα byte.

owTouchByte

SMALLINT owTouchByte(int portnum, SMALLINT sendbyte)

Η συνάρτηση owTouchByte εξετάζει την δυνατότητα της/των συσκευών του δικτύου να «διαβάζει» σωστά ένα byte.

owSpeed

SMALLINT owSpeed(int portnum, SMALLINT new_speed)

Η συνάρτηση owSpeed εξετάζει και επιστρέφει την ταχύτητα της/των συσκευών του δικτύου.

owLevel

SMALLINT owLevel(int portnum, SMALLINT new_level)

Η συνάρτηση owLevel εξετάζει και επιστρέφει το είδος *(άρα και το είδος των εντολών που μπορούν να δεχτούν*) των συσκευών του δικτύου.

owProgramPulse

SMALLINT owProgramPulse(int portnum)

Η συνάρτηση owProgramPulse δημιουργεί ένα παλμό 12V και εξετάζει και επιστρέφει το αν υπάρχουν συσκευές στο δίκτυο που μπορούν να τον δεχτούν άρα και να προγραμματιστούν με αυτόν.

owWriteBytePower

SMALLINT owWriteBytePower(int portnum, SMALLINT sendbyte)

Η συνάρτηση owWriteBytePower εξετάζει την δυνατότητα της/των συσκευών του δικτύου να «γράφει» και να «διαβάζει» σωστά ένα byte.

owReadBytePower

SMALLINT owReadBytePower(int portnum)

Η συνάρτηση owReadBytePower εξετάζει την δυνατότητα της/των συσκευών του δικτύου να «γράφει» και να «διαβάζει» σωστά το Byte μετά από το οποίο θα ξεκινήσει να στέλνει ισχύ (power) στο δίκτυο.

owReadBitPower

SMALLINT owReadBitPower(int portnum, SMALLINT applyPowerResponse)

Η συνάρτηση owReadBitPower εξετάζει την δυνατότητα της/των συσκευών του δικτύου να «γράφει» και να «διαβάζει» σωστά το bit μετά από το οποίο θα ξεκινήσει να στέλνει ισχύ (power) στο δίκτυο.

owHasPowerDelivery

SMALLINT owHasPowerDelivery(int portnum)

Η συνάρτηση owHasPowerDelivery εξετάζει την δυνατότητα του μετατροπέα (adaptor) του δικτύου να «μεταφέρει» ισχύ (power) στο δίκτυο.

owHasOverDrive

SMALLINT owHasOverDrive(int portnum)

Η συνάρτηση owHasOverDrive εξετάζει την δυνατότητα του μετατροπέα (adaptor) του δικτύου να «μεταφέρει» ισχύ (power) στο δίκτυο σε κατάσταση overdrive.

owHasProgramPulse

SMALLINT owHasProgramPulse(int portnum)

Η συνάρτηση owHasProgramPulse δημιουργεί ένα παλμό 12V και εξετάζει και επιστρέφει το αν υπάρχουν συσκευές στο δίκτυο που μπορούν να τον δεχτούν άρα και να προγραμματιστούν με αυτόν.

ownetu.c
+portnum : int +do_reset : int +alarm_only : int +do_read : int
+owFirst() +ownext() +owserialnum() +owfamilysearchsetup() +owskipfamily() +owaccess() +owverify() +owoverdriveaccess()

Σχήμα 8: UML διάγραμμα ownet.c

2.3.3. Συναρτήσεις Επιπέδου δικτύου

Οι συναρτήσεις του επιπέδου περιέχονται στο αρχείο-βιβλιοθήκη **ownet.c** *(ΠΑΡΑΡΤΗΜΑ Α-ΤΡΟΠΟΠΟΙΗΜΕΝΕΣ ΣΥΝΑΡΤΗΣΕΙΣ).*

Η βιβλιοθήκη αυτή περιέχει 9 συναρτήσεις:

owFirst

SMALLINT owFirst(int portnum, SMALLINT do_reset, SMALLINT alarm_only).

Η συνάρτηση owFirst βρίσκει την $1^{η}$ διαθέσιμη συσκευή στο δίκτυο, αναλόγως την παράμετρο (alarm_only) επιστρέφει την $1^{η}$ από τις -ενεργές εκείνη την στιγμή- συσκευές του δικτύου

owNext

SMALLINT owNext(int portnum, SMALLINT do_reset, SMALLINT alarm_only)

Η συνάρτηση owNext βρίσκει την επόμενη διαθέσιμη συσκευή στο δίκτυο, αναλόγως την παράμετρο (alarm_only) επιστρέφει την $1^{η}$ επόμενη από τις -ενεργές εκείνη την στιγμή- συσκευές του δικτύου. Φυσικά με επανειλημμένες κλήσεις επιστρέφει όλες τις συσκευές του δικτύου.

owSerialNum

void owSerialNum(int portnum, uchar *serialnum_buf, SMALLINT do_read)

Η συνάρτηση owSerialNum διαβάζει ή γράφει στην μνήμη προσωρινής αποθήκευσης (buffer) το σειριακό αριθμό των συσκευών (device serial number) την οποία χρησιμοποιούν οι δύο προηγούμενες συναρτήσεις για να βρουν τις συσκευές του δικτύου.

owFamilySearchSetup
void owFamilySearchSetup(int portnum, SMALLINT search_family)
Η συνάρτηση owFamilySearchSetup ενεργοποιεί τον αλγόριθμο που ανιχνεύει/καταγράφει την οικογένεια των συσκευών που ανεβρέθησαν από τις προηγούμενες συναρτήσεις.

owSkipFamily
void owSkipFamily(int portnum)
Η συνάρτηση owSkipFamily απενεργοποιεί τον αλγόριθμο που ανιχνεύει/καταγράφει την οικογένεια των συσκευών που ανεβρέθησαν από τις προηγούμενες συναρτήσεις.

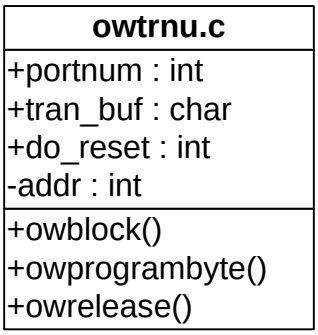
owAccess
SMALLINT owAccess(int portnum)
Η συνάρτηση owAccess κάνει αρχικοποίηση (reset) των συσκευών του δικτύου και στέλνοντας την αντίστοιχη εντολή, ετοιμάζει τις συσκευές του δικτύου να δεχτούν τις όποιες εξειδικευμένες εντολές μπορούν να υποστηρίξουν (οι συγκεκριμένες συσκευές).

owVerify
SMALLINT owVerify(int portnum, SMALLINT alarm_only)
Η συνάρτηση owVerify ελέγχει και πιστοποιεί ότι η συγκεκριμένη συσκευή είναι συνδεδεμένη και με συγκεκριμένη εντολή/διακόπτη ότι η συσκευή αυτή βρίσκεται σε κατάσταση εγρήγορσης (alarm state).

owOverdriveAccess
SMALLINT owOverdriveAccess(int portnum)
Η συνάρτηση owOverdriveAccess ελέγχει και πιστοποιεί ότι η συγκεκριμένη συσκευή είναι συνδεδεμένη και με συγκεκριμένη εντολή/διακόπτη η συσκευή αυτή μπορεί να λειτουργήσει σε κατάσταση overdrive.

bitacc
SMALLINT bitacc(SMALLINT op, SMALLINT state, SMALLINT loc, uchar *buf)

Η συνάρτηση bitacc «γράφει» και «διαβάζει» ένα μπιτ από την μνήμη προσωρινής αποθήκευσης (buffer).



Σχήμα 9: UML διάγραμμα owtrnu.c

2.3.4. Συναρτήσεις Επιπέδου Μεταφοράς

Transport Functions

Οι συναρτήσεις του επιπέδου περιέχονται στο αρχείο-βιβλιοθήκη **owTran.c** *(ΠΑΡΑΡΤΗΜΑ Α)*.

Η βιβλιοθήκη αυτή περιέχει 2 συναρτήσεις:

owBlock

SMALLINT owBlock(int portnum, SMALLINT do_reset, uchar *tran_buf, SMALLINT tran_len)

Η συνάρτηση owBlock είναι η βασική συνάρτηση μεταφοράς από και προς την συσκευή, ενός μπλοκ δεδομένων. Προαιρετικά χρησιμοποιεί την reset (Αρχικοποίηση) αν κριθεί απαραίτητη. Το αποτέλεσμα επιστρέφει στην ίδια μνήμη buffer από την οποία ξεκίνησε η μεταφορά.

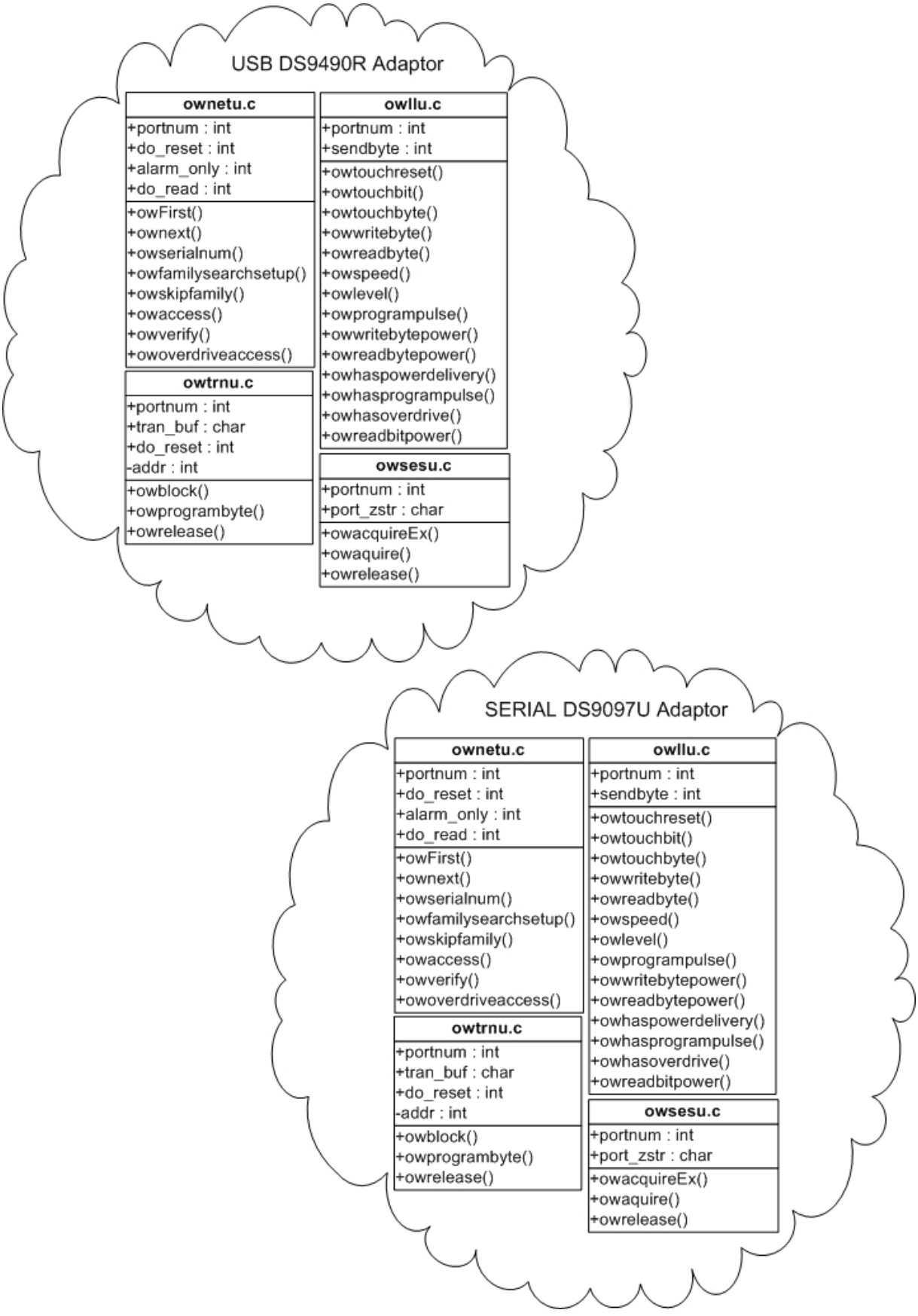
owProgramByte

SMALLINT owProgramByte(int portnum, SMALLINT write_byte, int addr, SMALLINT write_cmd, SMALLINT crc_type, SMALLINT do_access)

Η συνάρτηση owProgramByte γράφει ένα byte σε όλες τις 1-Wire συσκευές που είναι εφοδιασμένες με μνήμη EPROM.

Σχήμα 10: UML διάγραμμα κλάσεων 4 αρχικών βιβλιοθηκών

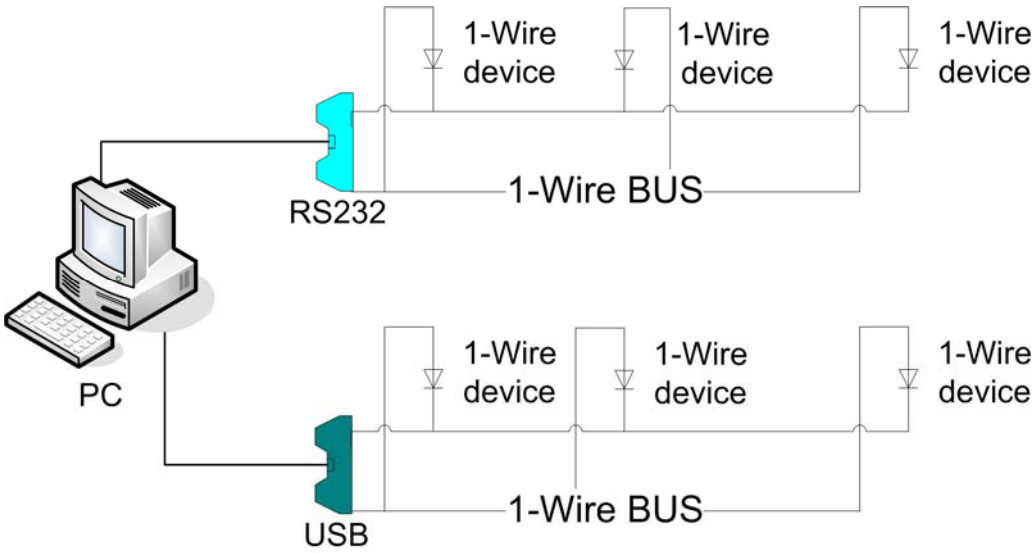
Αυτές είναι οι αρχικές υπάρχουσες βιβλιοθήκες και περιέχουν τις βασικές συναρτήσεις στην γενική τους μορφή. Αυτές οι βιβλιοθήκες υπήρχαν για κάθε μία master συσκευή, με την ίδια ονομασία και βιβλιοθηκών και επιμέρους συναρτήσεων. Όπως θα εξηγηθεί λεπτομερώς αργότερα, οι βιβλιοθήκες και οι συναρτήσεις τους έχουν τροποποιηθεί ώστε να μπορούν να κληθούν για τους διαφορετικούς μετατροπείς και πρωτόκολλα που υπάρχουν από μία κεντρική εφαρμογή. Στην σελίδα 44, σχήμα 11, βλέπουμε μία σχηματική αναπαράσταση της υπάρχουσας κατάστασης. Υπάρχουν βιβλιοθήκες για δύο διαφορετικές master συσκευές των οποίων οι συναρτήσεις έχουν τα ίδια ονόματα. Φυσικά η κάθε βιβλιοθήκη υλοποιεί διαφορετικά τις συναρτήσεις που εμπεριέχει , αφού ο κάθε προσαρμογέας έχει διαφορετικές ανάγκες και προτεραιότητες λόγω πρωτοκόλλου σύνδεσης.


USB DS9490R Adaptor
ownetu.c
+portnum : int
+do_reset : int
+alarm_only : int
+do_read : int
+owFirst()
+ownext()
+owserialnum()
+owfamilysearchsetup()
+owskipfamily()
+owaccess()
+owverify()
+owoverdriveaccess()
owtrnu.c
+portnum : int
+tran_buf : char
+do_reset : int
-addr : int
+owblock()
+owprogrambyte()
+owrelease()
owllu.c
+portnum : int
+sendbyte : int
+owtouchreset()
+owtouchbit()
+owtouchbyte()
+owwritebyte()
+owreadbyte()
+owspeed()
+owlevel()
+owprogrampulse()
+owwritebytepower()
+owreadbytepower()
+owhaspowerdelivery()
+owhasprogrampulse()
+owhasoverdrive()
+owreadbitpower()
owsesu.c
+portnum : int
+port_zstr : char
+owacquireEx()
+owaquire()
+owrelease()
SERIAL DS9097U Adaptor
ownetu.c
+portnum : int
+do_reset : int
+alarm_only : int
+do_read : int
+owFirst()
+ownext()
+owserialnum()
+owfamilysearchsetup()
+owskipfamily()
+owaccess()
+owverify()
+owoverdriveaccess()
owtrnu.c
+portnum : int
+tran_buf : char
+do_reset : int
-addr : int
+owblock()
+owprogrambyte()
+owrelease()
owllu.c
+portnum : int
+sendbyte : int
+owtouchreset()
+owtouchbit()
+owtouchbyte()
+owwritebyte()
+owreadbyte()
+owspeed()
+owlevel()
+owprogrampulse()
+owwritebytepower()
+owreadbytepower()
+owhaspowerdelivery()
+owhasprogrampulse()
+owhasoverdrive()
+owreadbitpower()
owsesu.c
+portnum : int
+port_zstr : char
+owacquireEx()
+owaquire()
+owrelease()

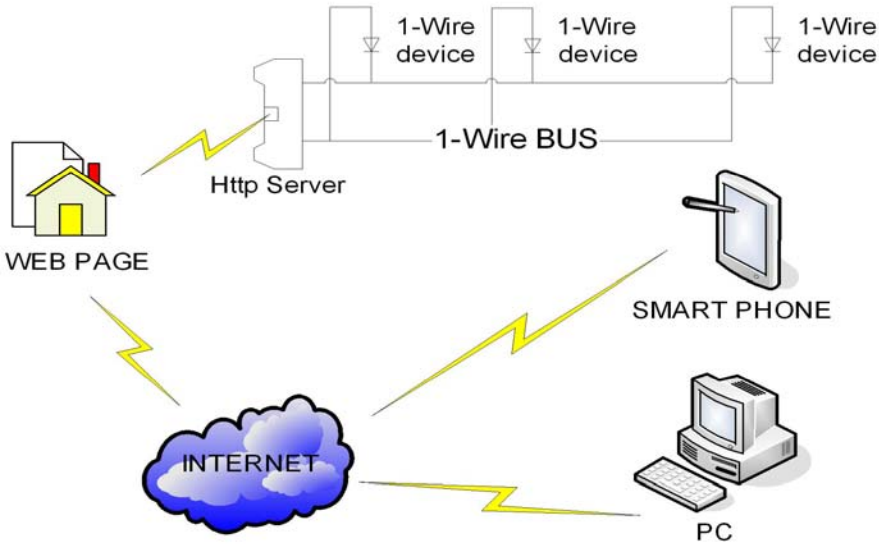
Σχήμα 11: UML απεικόνιση 2 ίδιων υπαρχουσών βιβλιοθηκών

3. Ανάπτυξη Λογισμικού

3.1. Ανάλυση απαιτήσεων



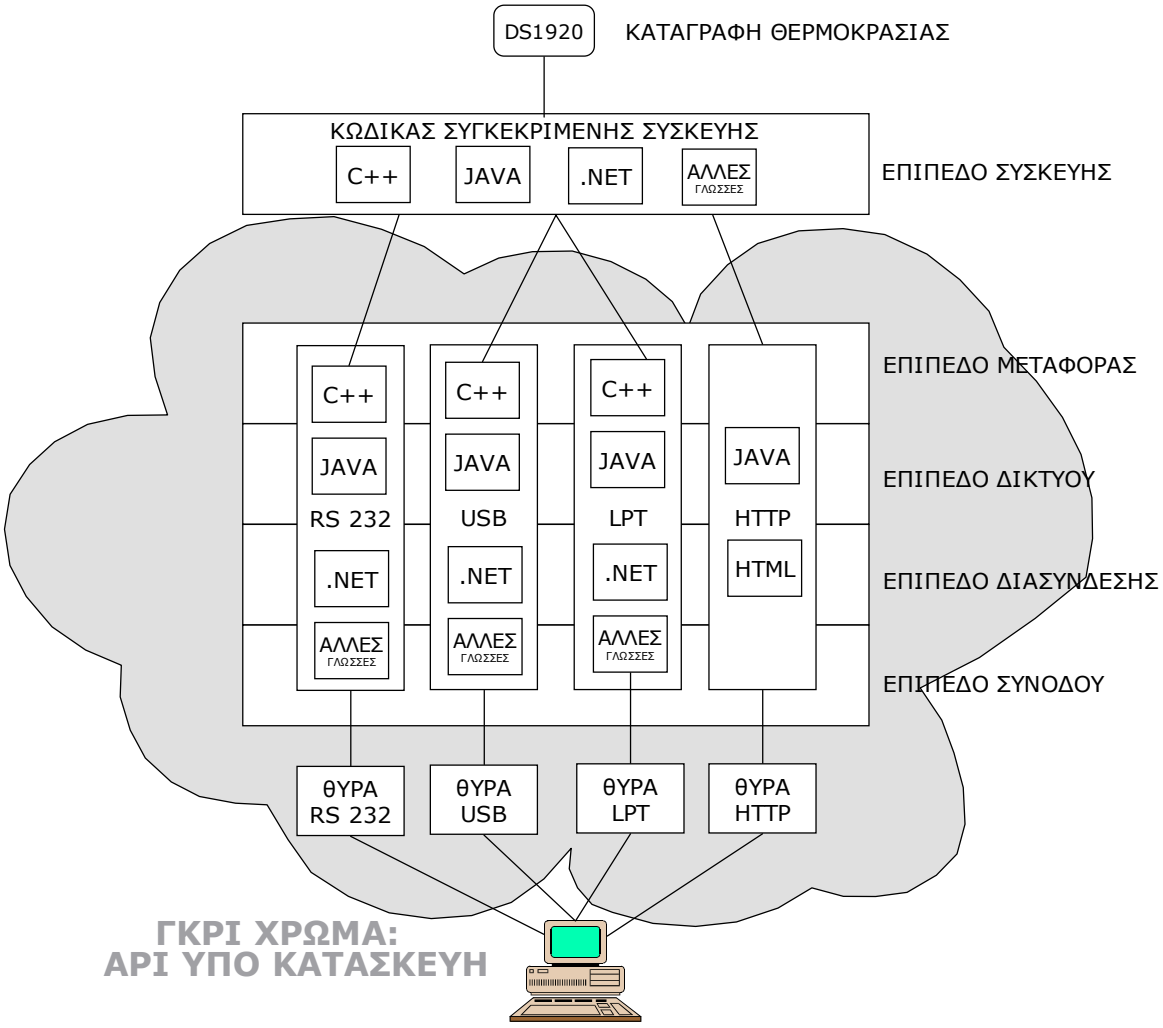
Σχήμα 12: Σύνδεση μέσω USB & RS232



Σχήμα 13: Απομακρυσμένη διαχείριση μέσω HTTP

Στα Σχήμα 12 & Σχήμα 13 , φαίνονται δύο τυπικές συνδέσεις ενός δικτύου 1-Wire με τον Η/Υ. Στην πρώτη περίπτωση ένα δίκτυο 1-Wire συνδέεται μέσω της σειριακής πόρτας RS232 ή/και ένα άλλο δίκτυο 1-Wire μέσω μίας θύρας USB. Στην δεύτερη περίπτωση το δίκτυο 1-Wire είναι συνδεδεμένο πάνω σε έναν http Server, δίνοντας μας έτσι την δυνατότητα να συνδεθούμε σε αυτό , ακόμη και μέσω του διαδικτύου.

Στα πλαίσια της παρούσας πτυχιακής εργασίας θα αντιμετωπισθεί το πρόβλημα του κατακερματισμού του κώδικα που παρατηρείται αυτήν την στιγμή όσον αφορά τις συσκευές master και τα πρωτόκολλα επικοινωνίας. Η εργασία φιλοδοξεί να συνενώσει τον «κατακερματισμένο» κώδικα που υπάρχει και να δημιουργήσει ένα Interface το οποίο θα μπορεί να χρησιμοποιήσει τον κώδικα που έχει γραφτεί για συγκεκριμένη συσκευή-πελάτη από οποιοδήποτε πρωτόκολλο και θύρα. Μία πρωταρχική αναπαράσταση των πεδίων που καλύπτει το API που δημιουργήθηκε αποτυπώνεται στο Σχήμα 14. Με την ολοκλήρωση του API δίνεται πλέον στον προγραμματιστή το εργαλείο που «παρακάμπτει» τα προβλήματα που δημιουργούνται από την χρήση διαφορετικών master-συσκευών και πρωτοκόλλων επικοινωνίας. Έτσι ο κώδικας και η εφαρμογή που θα δημιουργήσει για μία συγκεκριμένη συσκευή-πελάτη, θα γραφτεί μία φορά μόνο, και θα «τρέχει» σε οποιαδήποτε υλοποίηση του 1-Wire. Επίσης το API γράφτηκε ως ανεξάρτητο λειτουργικού συστήματος και συσκευής. Έτσι μπορεί να εκτελέσει τον κώδικα σε οποιοδήποτε λειτουργικό σύστημα και μηχανή (μερική εξαίρεση αποτελούν οι master-συσκευές USB, που χρειάζονται τους «οδηγούς»-drivers, για το λειτουργικό σύστημα στο οποίο εγκαθίστανται).



Σχήμα 14: πρωταρχική αναπαράσταση του υπό κατασκευή API

Όπως προαναφέρθηκε (Πίνακας 3: Τα 6 επίπεδα συναρτήσεων API), υπάρχουν 6 βασικά επίπεδα διαστρωμάτωσης στο 1-wire δίκτυο:

1. ΕΠΙΠΕΔΟ ΣΥΝΟΔΟΥ-SESSION
2. ΕΠΙΠΕΔΟ ΔΙΑΣΥΝΔΕΣΗΣ- LINK
3. ΕΠΙΠΕΔΟ ΔΙΚΤΥΟΥ - NETWORK
4. ΕΠΙΠΕΔΟ ΜΕΤΑΦΟΡΑΣ - TRANSPORT

5. ΕΠΙΠΕΔΟ ΑΡΧΕΙΩΝ - FILE
6. ΕΠΙΠΕΔΟ ΣΥΣΚΕΥΗΣ - DEVICE

Από αυτά, τα πρώτα τέσσερα είναι αυτά τα οποία ασχολούνται με το δίκτυο και τις master συσκευές σε αυτό (το επίπεδο αρχείων υλοποιείται μόνο όταν η συγκεκριμένη συσκευή έχει μνήμη μεγαλύτερη της μίας σελίδας). Σε αυτά τα τέσσερα επίπεδα, υπάρχουν υλοποιημένες ξεχωριστές βιβλιοθήκες-συναρτήσεις για κάθε μία master συσκευή *(Dallas Semiconductors/ Maxim, 1-Wire Public Domain Kit, 22.10.06)*. Όλες οι βιβλιοθήκες-συναρτήσεις αυτές, έχουν ακριβώς την ίδια δομή και χρησιμοποιούν τα ίδια ονόματα για τις επιμέρους συναρτήσεις τους. Οι συναρτήσεις αυτές είναι χωρισμένες σε 4 ομάδες κατηγορίες αναλόγως το επίπεδο στο οποίο ανήκουν και εμπεριέχουν όλες τις βασικές συναρτήσεις που είναι απαραίτητες για να λειτουργήσει ένα 1-Wire δίκτυο. Ακολουθούν τα ονόματα των 4 αυτών βιβλιοθηκών αντιστοιχισμένα σε κάθε ένα από τα 4 επίπεδα:

ΣΥΝΑΡΤΗΣΕΙΣ ΕΠΙΠΕΔΟΥ ΣΥΝΟΔΟΥ-SESSION

Acquire & release a Session

Αρχείο, owSesU.C

Εμπεριέχει όλες τις συναρτήσεις του επιπέδου συνόδου

ΣΥΝΑΡΤΗΣΕΙΣ ΕΠΙΠΕΔΟΥ ΔΙΑΣΥΝΔΕΣΗΣ- LINK

Link Layer functions

Αρχείο owLLU.C

Εμπεριέχει όλες τις συναρτήσεις του επιπέδου διασύνδεσης

ΣΥΝΑΡΤΗΣΕΙΣ ΕΠΙΠΕΔΟΥ ΔΙΚΤΥΟΥ - NETWORK

Network functions

Αρχείο ownet.c

Εμπεριέχει όλες τις συναρτήσεις του επιπέδου δικτύου

ΣΥΝΑΡΤΗΣΕΙΣ ΕΠΙΠΕΔΟΥ ΜΕΤΑΦΟΡΑΣ - TRANSPORT

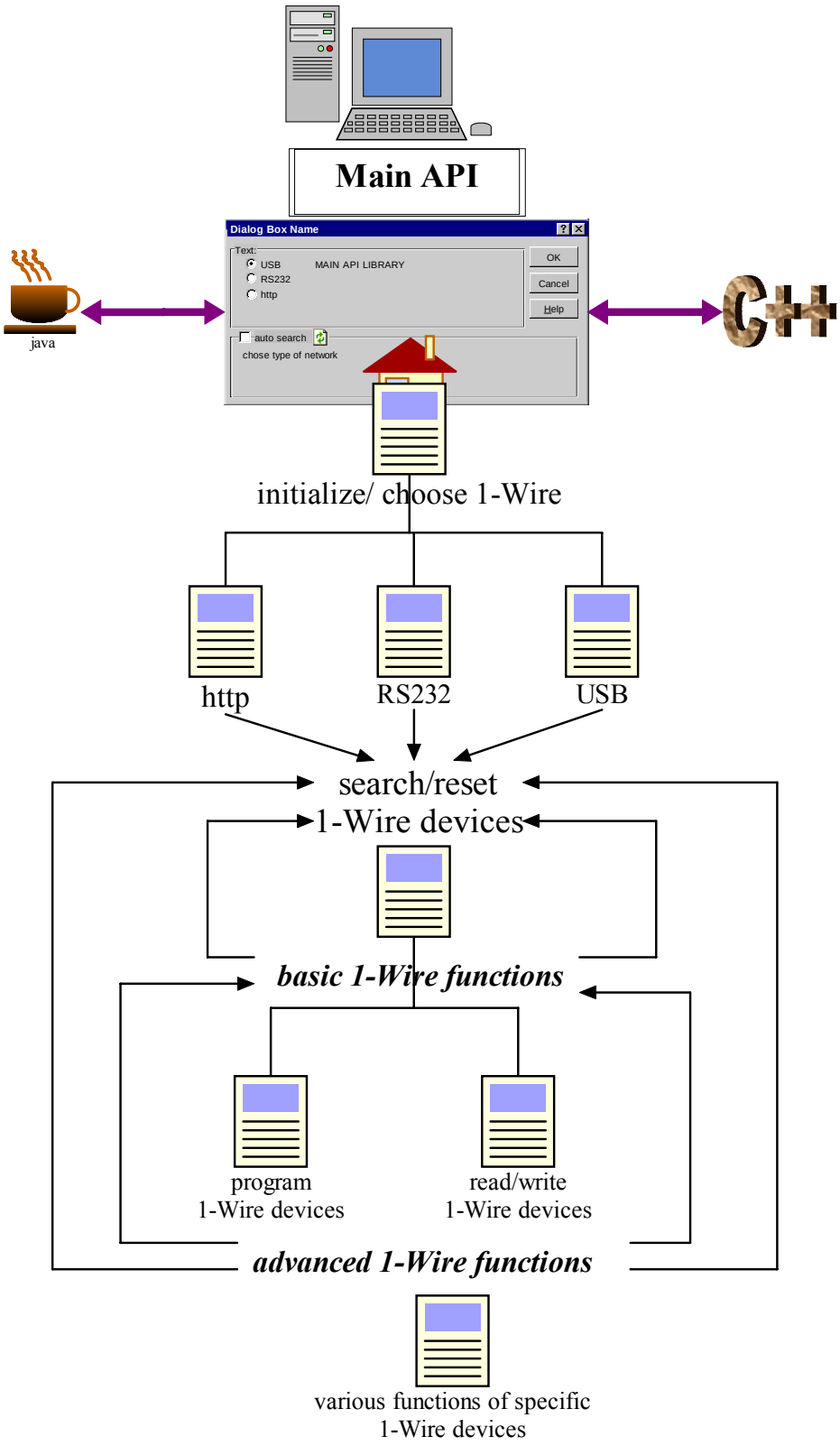
Transport functions

Αρχείο owTran.C

Εμπεριέχει όλες τις συναρτήσεις του επιπέδου μεταφοράς

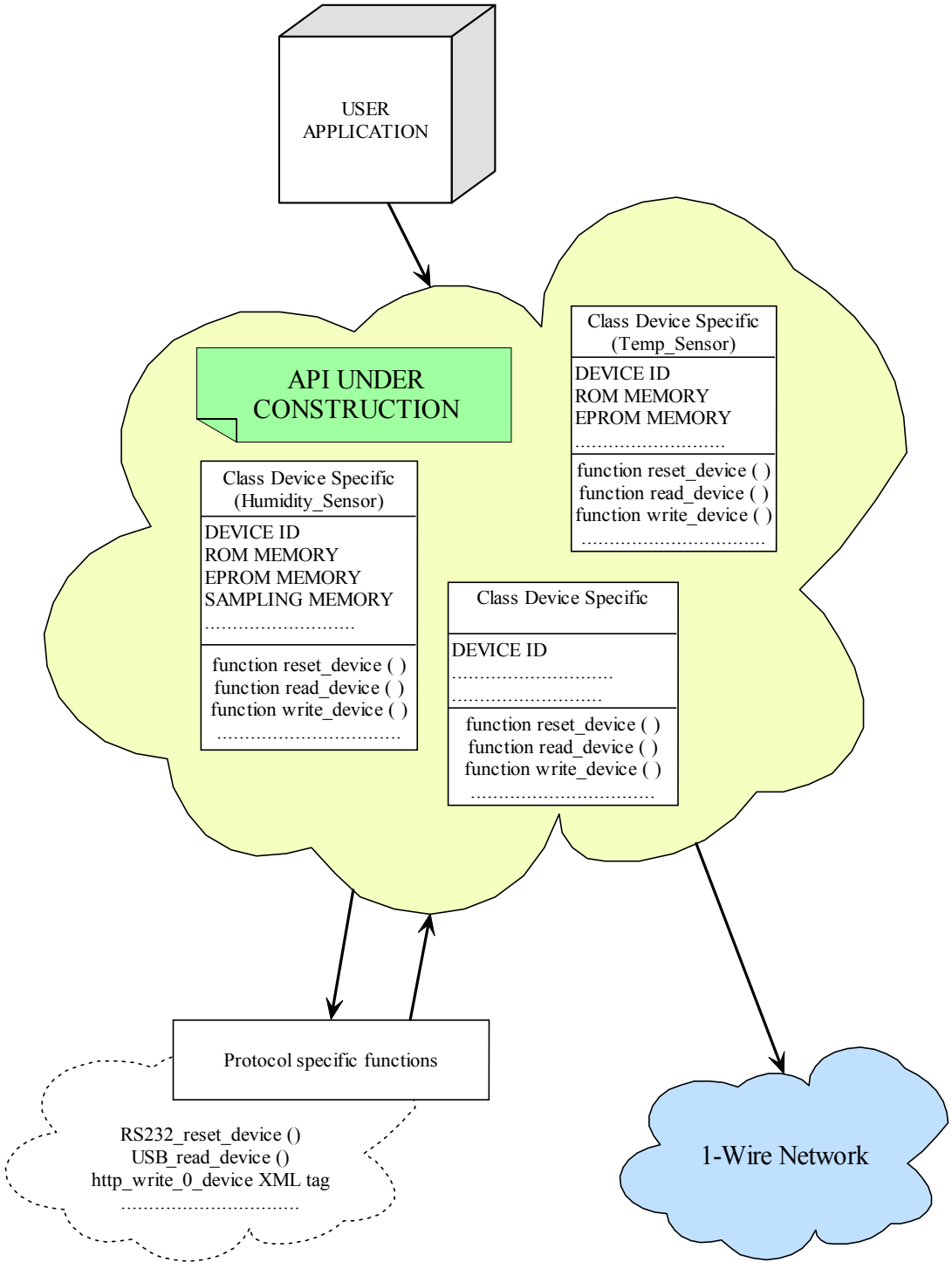
3.2. Σχεδίαση – αρχιτεκτονική

Στο Σχήμα 15, αναπαρίσταται η λογική του API που δημιουργήθηκε. Δεν έχει σημασία από ποια θύρα ή πρωτόκολλο επικοινωνεί ο Η/Υ με το δίκτυο 1-wire. Εφόσον υπάρχουν οι συναρτήσεις και οι εντολές της συγκεκριμένης συσκευής (σε όποια γλώσσα και αν είναι γραμμένες) το API αναλαμβάνει να κάνει τον ενδιάμεσο.



Σχήμα 15: Πρωταρχική Αναπαράσταση του API.

3.3. Χάρτης software επικοινωνίας



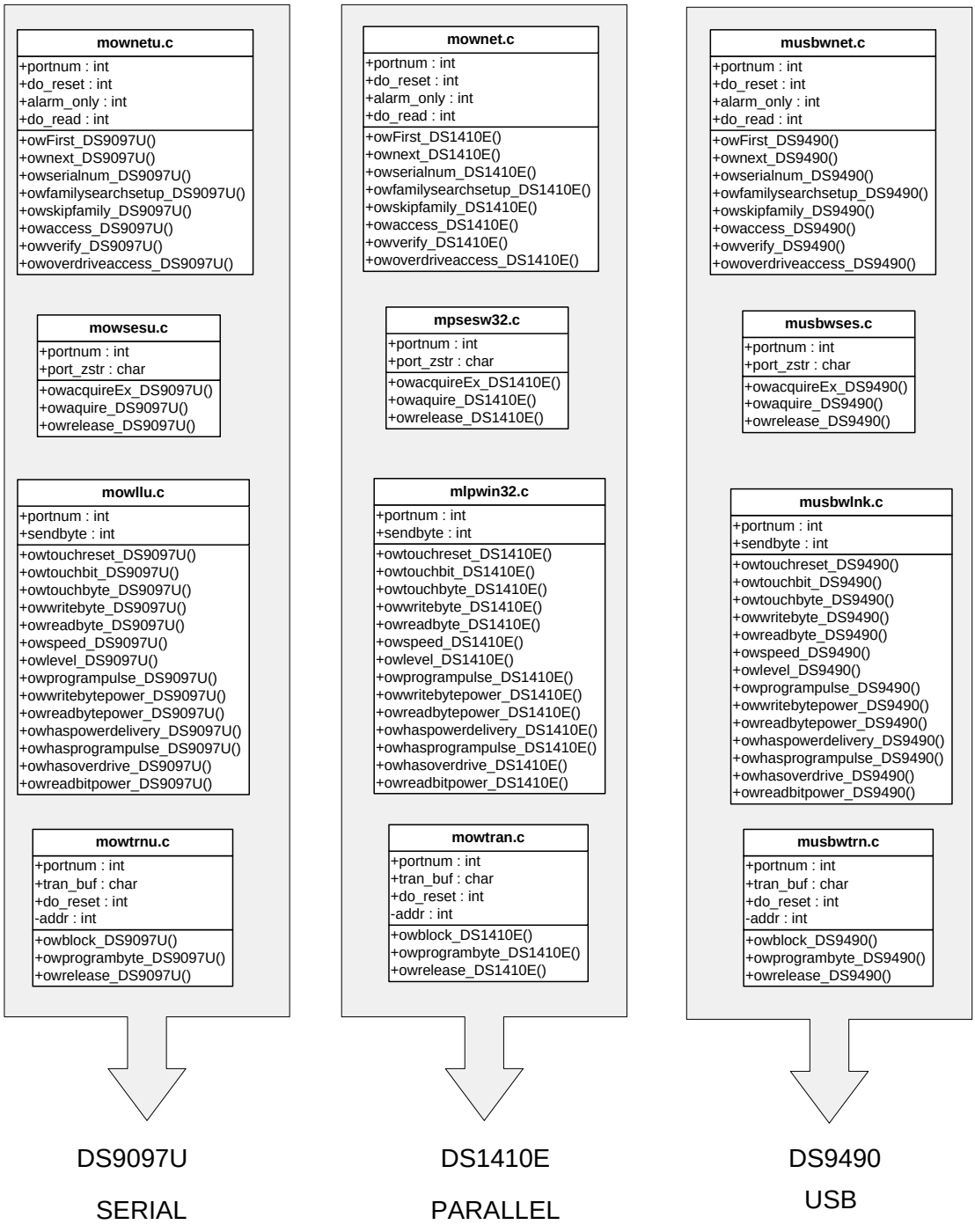
Σχήμα 16: Αναπαράσταση κεντρικού σχεδιασμού του API

Το ζητούμενο στην κατασκευή του API, είναι να δώσουμε στην όποια εφαρμογή, την δυνατότητα να χρησιμοποιεί τις ίδιες ρουτίνες ανεξαρτήτως του τι υπάρχει κάτω από αυτές,

και ποιο πρωτόκολλο χρησιμοποιείται ώστε να επικοινωνήσει η εφαρμογή με το 1-Wire. Για να καταστεί αυτό εφικτό, το API που κατασκευάστηκε θα γίνει ο ενδιάμεσος ανάμεσα στις συγκεκριμένες συναρτήσεις του κατασκευαστή και στην όποια εφαρμογή ανώτερου επιπέδου του χρήστη (Σχήμα 16). Έτσι ο χρήστης δεν χρειάζεται να γνωρίζει, ούτε να ασχοληθεί με το ποια συνάρτηση θα καλέσει κάθε φορά. Για να το επιτύχουμε αυτό εργαστήκαμε ως εξής:

Κάθε στοιχείο (συσκευή) του δικτύου γίνεται αντιληπτό ως αντικείμενο (object), και κατασκευάστηκε μία κλάση για κάθε τύπου συσκευές. (ίσως χρειαστεί αργότερα να δημιουργηθεί μία master class που να περιέχει τις γενικές παραμέτρους όλων των συσκευών, (π.χ. Device_id κτλ) και μετά να δημιουργηθούν υπο-κλάσεις για κάθε διαφορετική συσκευή που να κληρονομούν τα γνωρίσματα της αρχικής κλάσης. Οι κλάσεις σαν πεδία έχουν τις προγραμματιζόμενες λειτουργίες των συσκευών αυτών, και σαν μεθόδους θα έχουν τις λειτουργίες αυτών των συσκευών. Έτσι επιτυγχάνουμε να ομαδοποιήσουμε τις λειτουργίες των συσκευών στο δίκτυο, και μεταφράζοντας εσωτερικά τις εντολές στο κατάλληλο πρωτόκολλο κάθε φορά (RS232, USB, http), ο χρήστης θα χρησιμοποιεί τις συσκευές χωρίς να ενδιαφέρεται για το είδος της σύνδεσης τους στο δίκτυο.

Για να καταστεί δυνατή η κλήση των διαφορετικών συναρτήσεων και βιβλιοθηκών μέσα από το API, και η χρήση των διαφορετικών προσαρμογέων σε υψηλό επίπεδο, τροποποιήθηκαν αναλόγως οι συναρτήσεις που αφορούν στον κάθε προσαρμογέα. Η κάθε συνάρτηση πλέον είναι συγκεκριμενοποιημένη στον προσαρμογέα που αντιστοιχεί (adaptor specific) *(έχει στο όνομα της προσαρμοσμένο το όνομα του προσαρμογέα για τον οποίον καλείται, και έχει δηλωμένες τυχόν παραμέτρους-local variables- οι οποίες υλοποιούνται μόνο για τον συγκεκριμένο προσαρμογέα)* και έτσι δίδεται η δυνατότητα να κληθούν όλες μαζί μέσα από ένα API. Στο Σχήμα 17, φαίνονται οι τροποποιημένες σε όνομα συναρτήσεις των μητρικών συσκευών DS9097U & DS9490, σε σχέση με τα ίδια ονόματα που είχαν οι συναρτήσεις που υπήρχαν για τις δύο συσκευές όπως φαίνονται στο Σχήμα 10.



Σχήμα 17: UML διάγραμμα τροποποιημένων κλάσεων

Στο υψηλό επίπεδο του API για τις 4 βασικές ομάδες συναρτήσεων, δηλαδή αυτές που περιέχονται στις παρακάτω 4 κατηγορίες:

- Acquire & release a Session
- Link Layer functions

- Network functions
- Transport functions

Είναι πλέον δυνατόν να κληθεί μία συνάρτηση για την κάθε κλάση, και αυτή η συνάρτηση με την σειρά της, δοκιμάζει όλες τις περιπτώσεις , καλώντας όλες τις συναρτήσεις που γνωρίζει για τους προσαρμογείς που υπάρχουν μέχρι τώρα. Δηλαδή για παράδειγμα η συνάρτηση owTouchReset() καλεί όλες τις επιμέρους συναρτήσεις που έχουν στην ονομασία τους πλέον ως δεύτερο συνθετικό το όνομα του προσαρμογέα τους, δηλαδή τις:

owTouchReset _DS9490()
owTouchReset _DS1410E()
owTouchReset_ DS9097U()

Η κλήση τους γίνεται με case:

```
  switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
  {
    case DS9490:  return owTouchReset_DS9490(portnum & 0xFF);
    case DS1410E: return owTouchReset_DS1410E(portnum & 0xFF);
    default:
    case DS9097U: return owTouchReset_DS9097U(portnum & 0xFF);
  };
```

Αυτός είναι και ο πιο ανώδυνος και εύκολος τρόπος να κληθούν οι διαφορετικές συναρτήσεις. Και αυτό γιατί ο προγραμματιστής που θα χρησιμοποιήσει το υψηλού επιπέδου API για να γράψει εφαρμογές δεν χρειάζεται να γνωρίζει τίποτα για την επιμέρους υλοποίηση των συναρτήσεων. Επίσης έτσι ο κώδικας είναι συμπαγής και είναι πάρα πολύ εύκολο να προστεθούν οι συναρτήσεις για καινούριο προσαρμογέα ή πρωτόκολλο, όταν γραφτούν. Απλώς έχει προστεθεί ένα επιπλέον case σε όλες τις συναρτήσεις που υλοποιούν τα 4 αρχεία:

multilnk.c
multinet.C
multises.c
multitran.c

Με αυτόν τον τρόπο έχουν υλοποιηθεί και οι συναρτήσεις για το πρωτόκολλο http. Στο προηγούμενο παράδειγμα προστίθεται η 4η συνάρτηση:

owTouchReset _DS9490()

owTouchReset _DS1410E()

owTouchReset_ DS9097U()

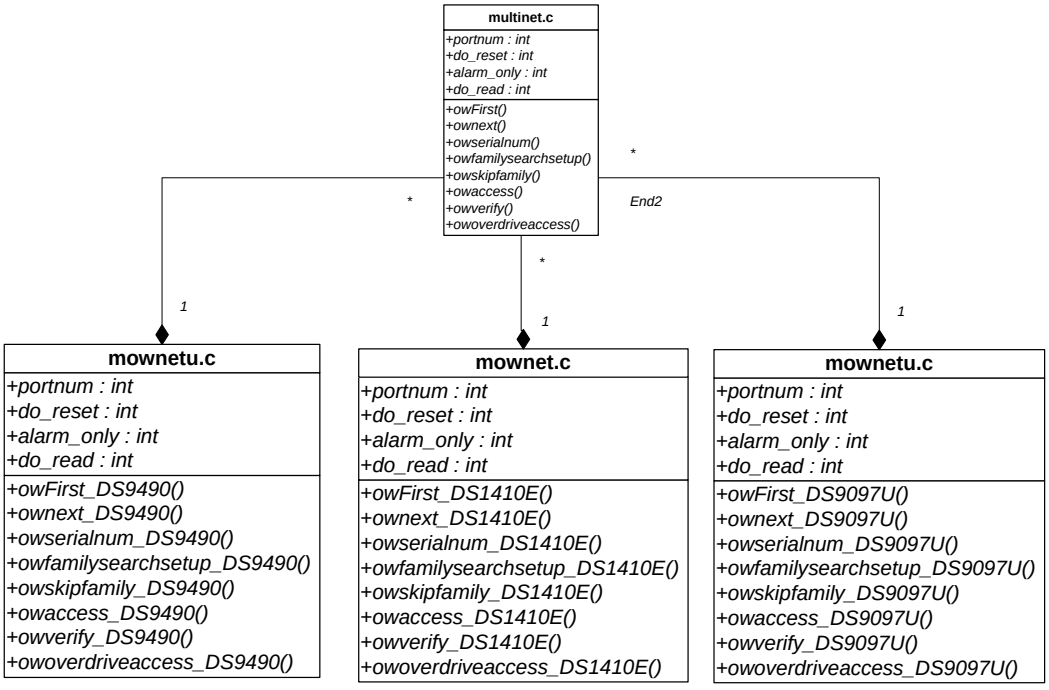
owTouchReset_ HA7Net()

Η κλήση τους γίνεται με case:

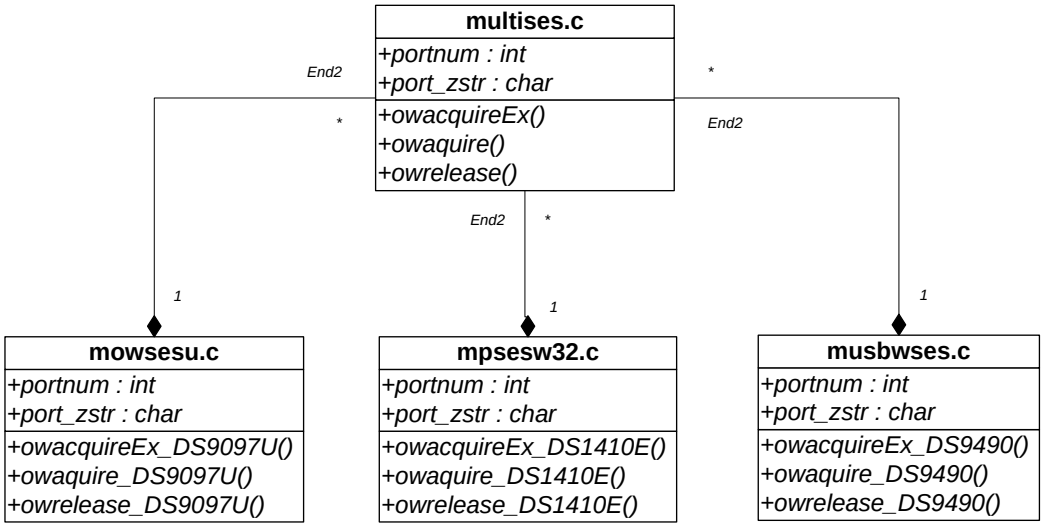
```
  switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
  {
    case DS9490:  return owTouchReset_DS9490(portnum & 0xFF);
    case DS1410E: return owTouchReset_DS1410E(portnum & 0xFF);
    default:
    case DS9097U: return owTouchReset_DS9097U(portnum & 0xFF);
         case HA7Net: return owTouchReset_ HA7Net(portnum & 0xFF);

  };
```

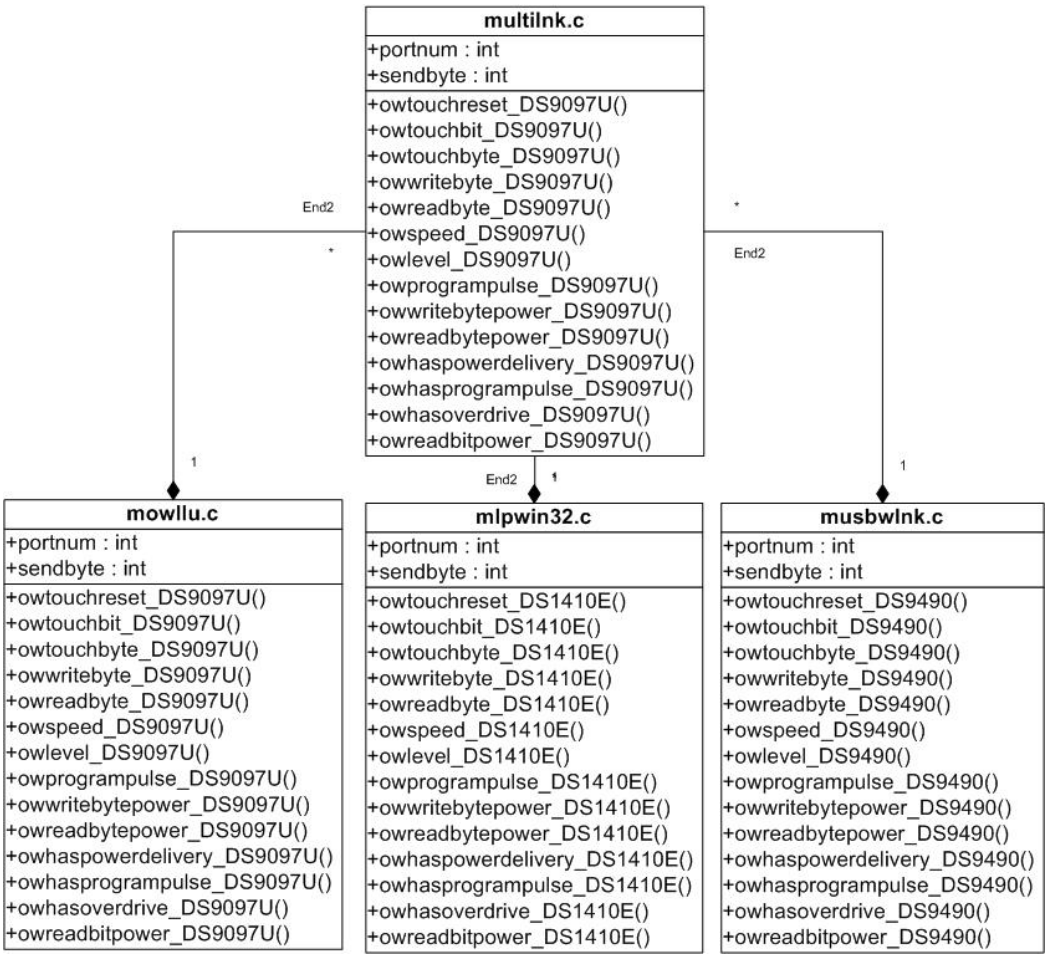
Στα διαγράμματα UML που ακολουθούν φαίνεται πως καλούνται πλέον οι επιμέρους (device specific) συναρτήσεις μέσα από τις ίδιες βιβλιοθήκες που καλούσαν και πριν όλα τα προγράμματα-συναρτήσεις που ανήκουν στο 5ο επίπεδο.



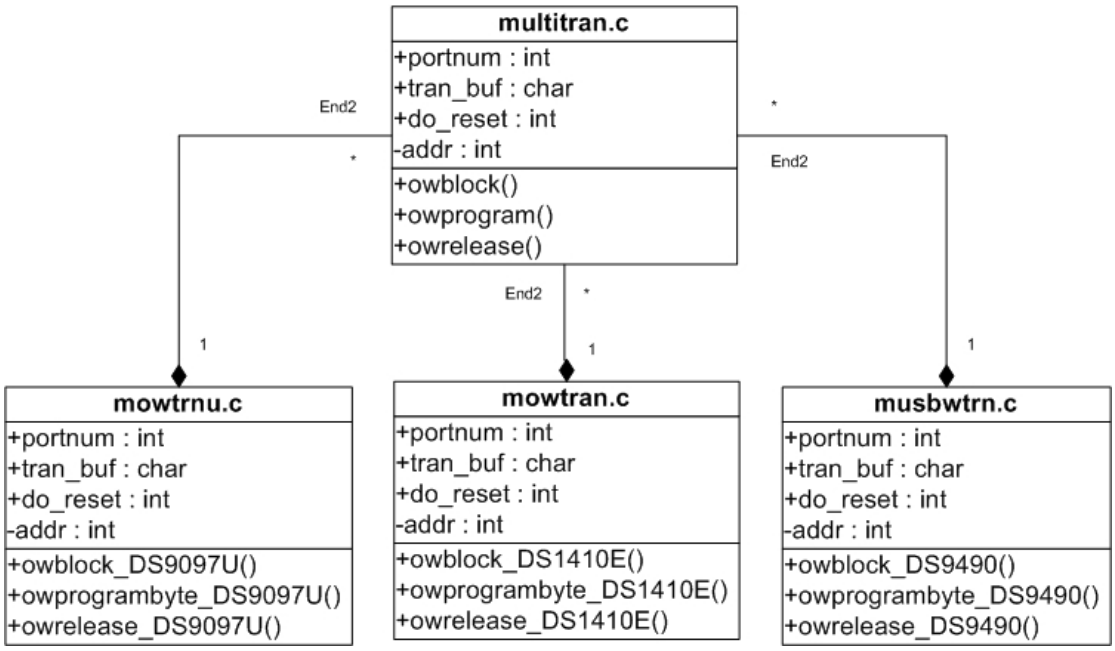
Σχήμα 18: UML διάγραμμα τροποποιημένης κλάσης multinet.c



Σχήμα 19: UML διάγραμμα τροποποιημένης κλάσης multises.c



Σχήμα 20: UML διάγραμμα τροποποιημένης κλάσης multilink.c



Σχήμα 21: UML διάγραμμα τροποποιημένης κλάσης multitran.c

Στο πρωτόκολλο http δεν υλοποιούνται όλες οι δυνατότητες και όλες οι συναρτήσεις που υλοποιούν τα υπόλοιπα 3. Αυτό όμως δεν είναι πρόβλημα αφού απλώς δεν θα υπάρχει το αντίστοιχο case στο πεδίο το οποίο δεν υλοποιείται σε http. Επίσης με αυτόν τον τρόπο οι όποιες προσθήκες μπορούν να γίνονται σταδιακά, αφού πρώτα αποσφαλματωθούν οι καινούριες εντολές ή πρωτόκολλα, τότε και μόνον τότε να προστίθενται οι αντίστοιχες επιλογές στον κώδικα.

Δημιουργήθηκαν εξαρχής οι συναρτήσεις της βιβλιοθήκης HA7net.c οι οποίες χρησιμοποιούν τον http Server HA7net της εταιρείας Embedded Data Systems. Η δυσκολία της δημιουργίας των συγκεκριμένων συναρτήσεων (ή/και βιβλιοθηκών) ήταν ότι έπρεπε να είναι ανεξάρτητες πλατφόρμας. Η δεύτερη δυσκολία έγκειται στο γεγονός ότι το υπό κατασκευή API γράφτηκε σε γλώσσα C++ ακριβώς για να έχει την δυνατότητα της εφαρμογής του ανεξάρτητα από πλατφόρμα (interpretability). Ως γνωστόν όμως η γλώσσα C++ δεν υποστηρίζει εγγενώς την http αφού προηγείται αυτής *(Jesse Liberty 2000)*. Η όποια διαστρωμάτωση έχει γίνει ανάμεσα στην C++ και στην http, έγινε εκ των υστέρων και σε υψηλό επίπεδο για λόγους συμβατότητας. Έτσι η γλώσσα C++ δεν προσφέρει τα εργαλεία εκείνα που προσφέρουν άλλες γλώσσες (όπως π.χ. η JAVA) *(16), (17)* για την επικοινωνία

και ανταλλαγή δεδομένων με το πρωτόκολλο http. Όλα τα έτοιμα εργαλεία της C++ εμπίπτουν σε μία από τις παρακάτω περιπτώσεις:

- Προϋπέθεταν την χρήση/ εγκατάσταση συγκεκριμένων βιβλιοθηκών στην μεριά του πελάτη (client) όσο και στην μεριά του εξυπηρετητή (server)
    - Κατηγορία Internet TServerSocket- TClientSocket
- Χρησιμοποιούν java scripts & XML
    - Κατηγορία MIDAS- WebConnection , Internet, Internet Express.

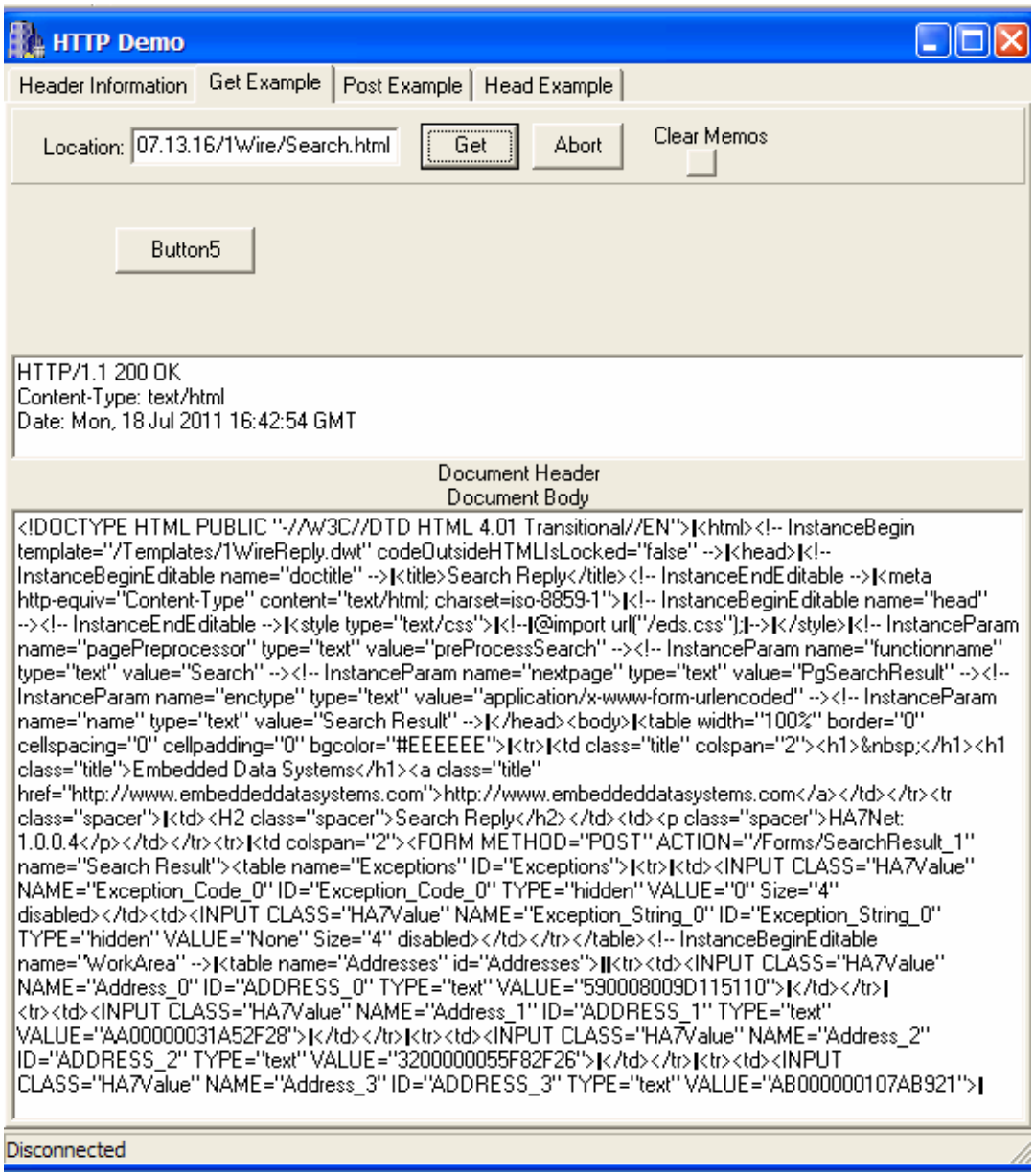
Μετά από έρευνα στο διαδίκτυο, παρατηρήθηκε επίσης ότι υπάρχει πολύ μεγάλη ποικιλία δυνατοτήτων πρόσβασης σε http πληροφορία από την πλατφόρμα windows, δεν υπάρχει αρκετή πληροφορία που να δίνει την δυνατότητα πρόσβασης στην http πληροφορία χωρίς να χρειάζονται κάποιες βιβλιοθήκες (dll's) που να δεσμεύουν-κατευθύνουν αυτήν την πρόσβαση , μέσα από συγκεκριμένο λειτουργικό σύστημα. Ακόμη και οι όποιες ερασιτεχνικές λύσεις που ανευρέθηκαν στο διαδίκτυο, εμπεριείχαν συγκεκριμένα dll's που περιορίζουν τρομερά την χρήση τους σε συγκεκριμένες πλατφόρμες. Έτσι επελέγη η αναγκαστικά η λύση του εξαρχής σχεδιασμού σε χαμηλό επίπεδο της επικοινωνίας με τον http server.

Χρησιμοποιήθηκε η κατηγορία FastNet της C++, και συγκεκριμένα το στοιχείο (component) NMHTTP. Το στοιχείο αυτό δεν είναι φιλικό στην χρήση του, και στερείται δυνατοτήτων , αλλά έχει κάποια σοβαρά πλεονεκτήματα:

- Δεν προϋποθέτει κάτι ιδιαίτερο από την μεριά του εξυπηρετητή (server). Ούτε καν ενδιαφέρεται για το είδος των δεδομένων που μεταφέρει. Και αυτό γιατί ουσιαστικά μεταφέρει ολόκληρη την σελίδα (υπάρχει μόνον διαχωρισμός σε body & header) που παράγει ο εξυπηρετητής χωρίς καν να την εξετάζει για το τι περιέχει. Αυτό πρακτικά σημαίνει ότι μεταφέρει όλη την πληροφορία που αποστέλλεται σε αυτόν που απέστειλε στον εξυπηρετητή την συγκεκριμένη εντολή.
- Επιστρέφει την όποια πληροφορία οπουδήποτε του ζητηθεί. Μπορεί να αποθηκεύσει την σελίδα που επιστρέφεται, ακόμη και σε ένα απλό αρχείο κειμένου (txt) εάν ζητηθεί.
- Δεν εγκαθιστά και δεν απαιτεί καμία ιδιαίτερη βιβλιοθήκη συγκεκριμένης πλατφόρμας (platform oriented libraries- platform dll's).

Με αυτόν τον τρόπο μέχρι τώρα δεν υπάρχει καμία δέσμευση σχετικά με την υλοποίηση του. Το δύσκολο σε αυτήν την περίπτωση είναι ότι όπως προαναφέρθηκε, αυτό που έχουμε μέχρι τώρα είναι μία ακατέργαστη σελίδα html σε μορφή κειμένου. Αυτό σημαίνει ότι πρέπει να

βρεθεί κάθε φορά η ζητούμενη πληροφορία με συγκεκριμένο τρόπο. Στην εικόνα 1 φαίνεται ανάγλυφα το πρόβλημα που υπάρχει. Στην προκειμένη περίπτωση το ζητούμενο είναι να εξαχθούν οι πέντε (5) μοναδικοί 16αδικοί αριθμοί που χαρακτηρίζουν τις συσκευές 1-Wire που ανευρέθηκαν στο δίκτυο http- 1-Wire, και εμφανίστηκαν στο «κυρίως μέρος» (body) της συγκεκριμένης σελίδας μετά τις επικεφαλίδες VALUE="… , που επέστρεψε ο http server μετά την εντολή Search.html που του εστάλη.



Εικόνα 7: DEMO εφαρμογή υλοποίησης παραλαβής http δεδομένων

Επειδή δε ως γνωστό, κάθε σελίδα html περιέχει τους τίτλους και τις ονομασίες των πινάκων της, πλέον της μίας φοράς, και επειδή σύμφωνα με όλα τα παραπάνω ο μόνος τρόπος ανεύρεσης της πληροφορίας, είναι κυριολεκτικά το «σκανάρισμα» όλης της σελίδας λέξη προς λέξη, επιλέχθηκαν για κάθε συγκεκριμένη συνάρτηση διαφορετικά κριτήρια εύρεσης με βάση φυσικά την μοναδικότητα τους.

Έτσι για παράδειγμα στο απόσπασμα κώδικα που ακολουθεί,

```
int n=0,pos,start, end;
        //execute the specific command
        NMHTTP1-
>Get("http://"+IPAddress_Prop_formated+"/1Wire/"+HttpSpecCommand);
        //collect the body of the http page
        Command = NMHTTP1->Body;
        //narrow the search starting from the field "Addresses"
        start = Command.AnsiPos("Addresses");
        //dont search further than the field "Statistics"
        end = Command.AnsiPos("Statistics");
        Command = Command.SubString(start,end - start);
        //if it finds the string "VALUE" with capital letters
        while((pos = Command.AnsiPos("VALUE")) != 0) {
        //there is a hex number of two bytes (16 bits)
          pos=pos+7;
          Values[n] = Command.SubString(pos,16);
          Command = Command.SubString(pos+16,Command.Length());
```

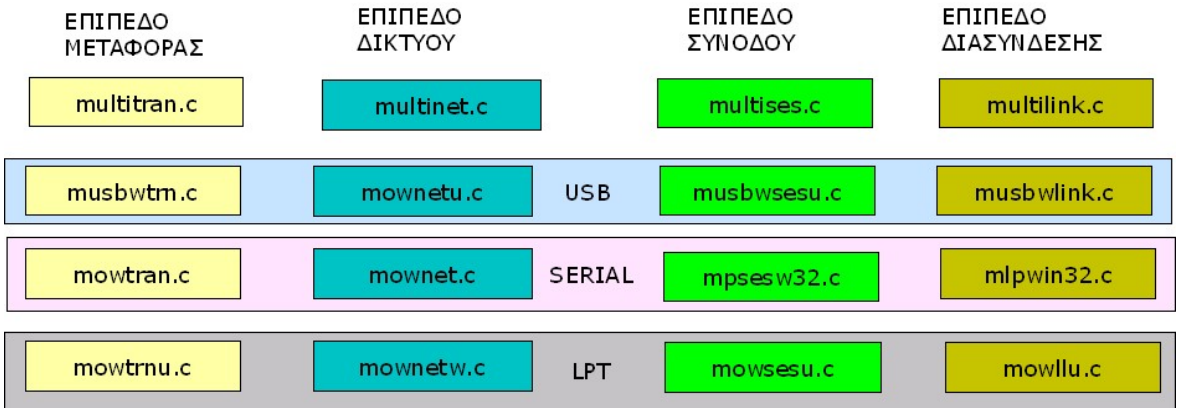
Μας ενδιαφέρει να βρούμε (ψάξουμε) ανάμεσα στα σημεία της σελίδας που ορίζονται από την λέξη "Addresses" και την λέξη "Statistics".
Αυτό δε που ψάχνουμε είναι οι 16δικοί αριθμοί (αγνώστου πλήθους) οι οποίοι έπονται πάντα της λέξης "VALUE" με κεφαλαία. Ο τρόπος αυτός αν και δύσκολος και κουραστικός στην υλοποίηση, πληρεί όλες τις προϋποθέσεις που τέθηκαν εξαρχής, δηλαδή δεν χρειάζεται καμία ιδιαίτερη βιβλιοθήκη, πέρα από τις κλασσικές ανεξάρτητες βιβλιοθήκες της C++, δεν χρειάζεται κανενός είδους ειδική εγκατάσταση, και δουλεύει σε οποιαδήποτε πλατφόρμα, αφού επεξεργάζεται κείμενο στην απλούστερη μορφή του (txt). Άρα ξεφεύγουμε αμέσως

από προβλήματα ασυμβατότητας μεταξύ λειτουργικών συστημάτων, μεταξύ φυλλομετρητών (browsers) και μεταξύ διαφορετικών εκδόσεων της C++.

Σαν τελευταία προσθήκη το component NHTTP αφαιρέθηκε τελείως από την http βιβλιοθήκη (HA7net.com). Μέσα στη βιβλιοθήκη παρέμειναν μόνο οι συναρτήσεις οι οποίες επεξεργάζονται τον κορμό (body) της «σελίδας» που επιστρέφει ο http server. Η αποστολή της συγκεκριμένης εντολής και η παραλαβή του κορμού (body) που επιστρέφει ο http server γίνεται με την κλήση εξωτερικής συνάρτησης (ExecHttp), η οποία δηλώνεται πλέον στην κεντρική (main) εφαρμογή. Πρέπει να τονιστεί ότι αυτός ο τρόπος δίνει τρομερά πλεονεκτήματα στον προγραμματιστή. Δεν μας ενδιαφέρει πλέον με ποιον τρόπο θα αποσταλεί η εντολή στον http server και πως θα ληφθεί η σελίδα που επιστρέφει. Ο προγραμματιστής μπορεί να χρησιμοποιήσει οποιαδήποτε της C++ ή οποιασδήποτε άλλης γλώσσας εξωτερικά components και dll's προσανατολισμένα στο λειτουργικό σύστημα του προγραμματιστή ή στη συγκεκριμένη εφαρμογή του. Το μόνο που μας ενδιαφέρει είναι να υπάρχει ένας τρόπος αποστολής http εντολών και παραλαβής του κορμού (body) της σελίδας που επιστρέφει ο server.

3.4. Χάρτης ενοποιημένων συναρτήσεων

Οι βιβλιοθήκες που παρατίθενται εδώ, είναι αυτές που συνολικά χρειάζονται για να υπάρξει ένα ολοκληρωμένο περιβάλλον εκτέλεσης εφαρμογών σε ένα δίκτυο 1-Wire το οποίο μπορεί να υλοποιείται μέσω σειριακής σύνδεσης, USB, παράλληλης σύνδεσης, ή μέσω πρωτοκόλλου http. Αρκετές από αυτές τις βιβλιοθήκες τροποποιήθηκαν εξ ολοκλήρου και άλλες λιγότερο, ενώ κάποιες από αυτές παρέμειναν αυτούσιες. Όπως φαίνεται και στο που ακολουθεί οι βιβλιοθήκες για κάθε master συσκευή έχουν αλλάξει όνομα , αλλά μπορούν να κληθούν μέσω των «κεντρικών» βιβλιοθηκών για κάθε επίπεδο, που διατήρησαν τα προηγούμενα ονόματα τους.



Εικόνα 8: Χρωματική απόδοση τροποποιημένων βιβλιοθηκών

3.4.1. Επίπεδο Συνόδου

Acquire & release a Session

mowSesU.C

βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις συνόδου για τον DS2480B (U) σειριακό μετατροπέα. Επίσης υποστηρίζονται οι μετατροπείς DS2480/DS2480B (U) (DS9097U).

mpsesw32.C

βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις συνόδου για τον DS1410E παράλληλο (LPT) μετατροπέα.

musbwses.c

βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις συνόδου για τον (DS9490) μετατροπέα θύρας USB .

Σημείωση: όπως προαναφέρθηκε η multises.c καλεί με την εντολή case τις αντίστοιχες συναρτήσεις από τις 3 adapter specific βιβλιοθήκες. Η λεπτομερής περιγραφή των γενικών συναρτήσεων της βιβλιοθήκης, υπάρχει στο Παράρτημα Α.

3.4.2. Επίπεδο Διασύνδεσης

Link Layer functions

mowLLU.C

βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις επιπέδου διασύνδεσης για τον DS2480B (U) σειριακό μετατροπέα. Επίσης υποστηρίζονται οι επίσης σειριακοί μετατροπείς DS2480/DS2480B (U) (DS9097U).

mlpWin32.C

βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις επιπέδου διασύνδεσης για τον DS1410E παράλληλο (LPT) μετατροπέα.

musbwlnk.c

βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις επιπέδου διασύνδεσης για τον (DS9490) μετατροπέα θύρας USB. (Χρειάζεται η εγκατάσταση του DS2490.SYS).

Σημείωση: όπως προαναφέρθηκε η multilink.c καλεί με την εντολή case τις αντίστοιχες συναρτήσεις από τις 3 εξειδικευμένες σε μετατροπέα βιβλιοθήκες (adapter specific) βιβλιοθήκες. Η λεπτομερής περιγραφή των γενικών συναρτήσεων της βιβλιοθήκης, υπάρχει στο Παράρτημα Α .

3.4.3. Επίπεδο δικτύου

Network functions

mownetu.C

βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις επιπέδου δικτύου για τον DS2480B (U) σειριακό μετατροπέα. Επίσης υποστηρίζονται οι σειριακοί μετατροπείς DS2480/DS2480B (U) (DS9097U).

mownet.C

βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις επιπέδου δικτύου για τον DS1410E παράλληλο (LPT) μετατροπέα.

musbwnet.C

βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις επιπέδου δικτύου για τον (DS9490) μετατροπέα θύρας USB.

Σημείωση: όπως προαναφέρθηκε η multinet.c καλεί με την εντολή case τις αντίστοιχες συναρτήσεις από τις 3 adapter specific βιβλιοθήκες. Η λεπτομερής περιγραφή των γενικών συναρτήσεων της βιβλιοθήκης, υπάρχει στο Παράρτημα Α..

3.4.4. Επίπεδο Μεταφοράς
Transport functions

mowTrnU.C
βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις επιπέδου μεταφοράς για τον DS2480B (U) σειριακό μετατροπέα.

mowTran.C
βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις επιπέδου δικτύου για τον DS1410E παράλληλο (LPT) μετατροπέα.

musbwtrn.C
βιβλιοθήκη η οποία υλοποιεί όλες τις συναρτήσεις επιπέδου δικτύου για τον (DS9490) μετατροπέα θύρας USB.

Σημείωση: όπως προαναφέρθηκε η multitran.c καλεί με την εντολή case τις αντίστοιχες συναρτήσεις από τις 3 adapter specific βιβλιοθήκες. Η λεπτομερής περιγραφή των γενικών συναρτήσεων της βιβλιοθήκης, υπάρχει στο Παράρτημα Α .

3.4.5. Βιβλιοθήκες Multi-build

Οι παρακάτω βιβλιοθήκες είναι αυτές που υλοποιήθηκαν για να υποστηρίξουν αντίστοιχα όλους τους γνωστούς μέχρι τώρα προσαρμογείς και θύρες *(ΠΑΡΑΡΤΗΜΑ Α)*
Mownet.h
Δηλώσεις των συναρτήσεων των επιμέρους βιβλιοθηκών *(ΠΑΡΑΡΤΗΜΑ Α).*

multises.c
Υποστηρίζει όλους τους προσαρμογείς για το επίπεδο συνόδου
(ΠΑΡΑΡΤΗΜΑ Α).

multilnk.c

Υποστηρίζει όλους τους προσαρμογείς για το επίπεδο διασύνδεσης *(ΠΑΡΑΡΤΗΜΑ Α).*

multinet.C

Υποστηρίζει όλους τους προσαρμογείς για το επίπεδο δικτύου *(ΠΑΡΑΡΤΗΜΑ Α).*

multitran.c

Υποστηρίζει όλους τους προσαρμογείς για το επίπεδο μεταφοράς (*ΠΑΡΑΡΤΗΜΑ Α).*

3.4.6. Υπάρχουσες κοινές βιβλιοθήκες

Υπήρξαν αρκετές βιβλιοθήκες που δεν υπάρχει λόγος να αλλαχθούν και χρησιμοποιήθηκαν αυτούσιες. Αυτές είναι βιβλιοθήκες επεξεργασίας και εμφάνισης λαθών, βιβλιοθήκες που χειρίζονται τις ιδιαιτερότητες της κάθε master συσκευής-προσαρμογέα κτλ. *(ΠΑΡΑΡΤΗΜΑ Α). Οι βιβλιοθήκες αυτές είναι οι εξής:*

Οι παρακάτω 2 βιβλιοθήκες είναι κοινές για όλους τους προσαρμογείς και υλοποιούν αντίστοιχα:

Η **mcrcutil.c** τον έλεγχο των λειτουργιών CRC (Cyclic Redudancy Code) 8 & 16 bit, όπου και όταν απαιτείται *(ΠΑΡΑΡΤΗΜΑ Α)* .

Η **mower.c** υλοποιεί αναλυτικά όλα τα πιθανά λάθη που μπορεί να εμφανιστούν. Να σημειωθεί ότι είναι γραμμένη έτσι ώστε να είναι ανεξάρτητη προσαρμογέα , αφού υλοποιεί τα λάθη σε επίπεδο δικτύου και όχι σε επίπεδο συσκευών ή προσαρμογέα. Δηλαδή υπάρχουν κώδικες συγκεκριμένων λαθών για δυσλειτουργία που προκύπτει από την κλήση μίας γενικής συνάρτησης (π.χ. την συνάρτηση owReset() την οποία υλοποιούν όλοι οι προσαρμογείς και όλες οι θύρες. Αυτό σημαίνει ότι δεν χρειάζεται να προστεθεί ή να αλλαχτεί κώδικας κάθε φορά που προστίθεται ένας προσαρμογέας ή πρωτόκολλο *(ΠΑΡΑΡΤΗΜΑ Α)*.

3.4.7. Υπάρχουσες βιβλιοθήκες μετατροπέων

Βιβλιοθήκες για τον σειριακό μετατροπέα DS2480

Η σειριακή πόρτα έχει αρκετές ιδιαιτερότητες, όπως ότι μπορεί να υπάρχουν πολλές σειριακές θύρες, κάποιες από αυτές να είναι κατειλημμένες από άλλες συσκευές που επίσης στέλνουν δεδομένα. Παράμετροι όπως το baud rate, οι FIFO buffers αναγκαστικά πρέπει να έχουν συγκεκριμένες συναρτήσεις που να υλοποιούν τις διαφορετικές παραμέτρους τους.

mWn32Lnk.C

Η βιβλιοθήκη χρησιμοποιείται και ως γενική βιβλιοθήκη για όλους τους σειριακούς μετατροπείς *(ΠΑΡΑΡΤΗΜΑ Α – Υπάρχουσες κοινές βιβλιοθήκες - Βιβλιοθήκες για τον σειριακό μετατροπέα DS2480- mWn32Lnk.C)*.

m2480ut.c

Η βιβλιοθήκη περιέχει όλες τις συναρτήσεις χρήσης του μετατροπέα (*ΠΑΡΑΡΤΗΜΑ Α – Υπάρχουσες κοινές βιβλιοθήκες - Βιβλιοθήκες για τον σειριακό μετατροπέα DS2480- m2480ut.c)*.

mDS2480.h

Η master βιβλιοθήκη περιέχει όλες τις δηλώσεις των συναρτήσεων του προσαρμογέα *(ΠΑΡΑΡΤΗΜΑ Α – Υπάρχουσες κοινές βιβλιοθήκες - Βιβλιοθήκες για τον σειριακό μετατροπέα DS2480- mDS2480.h)*.

Βιβλιοθήκες για τον USB μετατροπέα DS2490

m2490ut.c

Η βιβλιοθήκη περιέχει όλες τις επιμέρους συναρτήσεις που υλοποιεί ο προσαρμογέας *(ΠΑΡΑΡΤΗΜΑ Α)*. Για να δουλέψουν οι συναρτήσεις της βιβλιοθήκης χρειάζεται να έχουν εγκατασταθεί οι οδηγοί (drivers) για την συσκευή (DS2490.SYS).

d2490u.h

Η master βιβλιοθήκη περιέχει όλες τις δηλώσεις των συναρτήσεων του προσαρμογέα *(ΠΑΡΑΡΤΗΜΑ Α)*.

Βιβλιοθήκη για τον http Εξυπηρετητή HA7Net

HA7Net.c

Όλες οι συναρτήσεις που δημιουργήθηκαν για τον http server έχουν ενσωματωθεί σε ένα αρχείο/βιβλιοθήκη.

Οι προσθήκες / μετατροπές που έγιναν στα 4 βασικά αρχεία:
multilnk.c, multinet.C, multises.c, multitran.c , για την ενσωμάτωση του http server έχουν όλες τις ίδια φιλοσοφία. Εκεί που υπάρχει 1 προς 1 αντιστοίχηση με γνωστές συναρτήσεις, προστέθηκε κώδικας που αντιστοιχεί σε κλήση "case". Αυτές είναι:

multitran.c
στην συνάρτηση owBlock προστέθηκε η γραμμή:
"case HA7Net:....."

multises.c
στην συνάρτηση owAcquire προστέθηκε η γραμμή:
"case HA7Net:....."
στην συνάρτηση owRelease προστέθηκε η γραμμή:
"case HA7Net:....."

multinet.C
στην συνάρτηση owNext προστέθηκε η γραμμή:
"case HA7Net:....."
στην συνάρτηση owFamilySearchSetup προστέθηκε η γραμμή:
"case HA7Net:....."
στην συνάρτηση owVerify προστέθηκε η γραμμή:
"case HA7Net:....."

multilnk.C
στην συνάρτηση owTouchReset προστέθηκε η γραμμή:
"case HA7Net:....."
στην συνάρτηση owReadBitPower προστέθηκε η γραμμή: "case HA7Net:....."

Οι υπόλοιπες συναρτήσεις που υλοποιεί ο http server δεν αντιστοιχούν 100% στις υπάρχουσες συναρτήσεις των υπόλοιπων πρωτοκόλλων. Έτσι ναι μεν γράφτηκαν συναρτήσεις για αυτές και δοκιμάστηκαν στην πράξη, αλλά δεν μπόρεσαν να ενσωματωθούν

στα "case" των υπολοίπων συναρτήσεων των υπόλοιπων πρωτοκόλλων. Είναι δυνατόν όμως χρησιμοποιώντας την υπάρχουσα υλοποίηση, να αναπτύξει κάποιος τις υπόλοιπες συναρτήσεις (υλοποιώντας τες κάνοντας χρήση των υπαρχουσών) με μακρυπρόσθεσμο στόχο την 100% προσομοίωση του http πρωτοκόλλου με τις υπόλοιπες διασυνδέσεις.

3.4.8. Πίνακας περιεχόμενων αρχείων στο API

HA7Net.c

ds2490.h	mownetu.c	musbwnet.c
m2480ut.c	mowsesu.c	musbwses.c
m2490ut.c	mowtran.c	musbwtrn.c
mcrcutil.c	mowtrnu.c	mwn32lnk.c
mds2480.h	mpsesw32.c	mowllu.c
mlpwin32.c	multilnk.c	mownet.c
mowerr.c	multinet.c	mownet.h
multitran.c	multises.c	musbwlnk.c

3.4.9. Αρχές Σχεδίασης και Τεκμηρίωση

Ακολουθήθηκαν οι αρχές του αντικειμενοστραφούς οπτικού (visual) προγραμματισμού. Χρησιμοποιήθηκε ως αντικειμενοστραφής γλώσσα η γλώσσα C++. Η δυνατότητα της C++ να λειτουργεί κάτω από διαφορετικές πλατφόρμες και η ευρεία διάδοση της είναι τα βασικά στοιχεία που συνηγορούν στην χρησιμοποίηση της. Ενσωματώθηκε σε αρκετά μεγάλο βαθμό το Low-Level software που έχει αναπτυχθεί κατά βάση από την Dallas Semiconductor αλλά και κομμάτια άλλων εταιρειών (Embedded Data Systems) που μπόρεσαν να ενσωματωθούν στο υψηλού επιπέδου API που δημιουργήθηκε. Το API που δημιουργήθηκε είναι εύκολα επεκτάσιμο εφόσον στα πλαίσια της εργασίας εμπεριέχει μόνο τις βασικές λειτουργίες και τις βασικές συσκευές 1-Wire. Ο επόμενος προγραμματιστής μπορεί εύκολα να ενσωματώσει τις όποιες δυνητικές επεκτάσεις θεωρεί αναγκαίες ή/και ωφέλιμες. Υλοποιήθηκε σε περιβάλλον .NET της Microsoft για λόγους εύκολης αποσφαλμάτωσης και τεκμηρίωσης αλλά και στο περιβάλλον Borland C++ Builder 5 για λόγους συμβατότητας προς τα πίσω, αφού πολλά κομμάτια του κώδικα της Dallas Semiconductors ήταν αδύνατο να «ανοιχτούν» σε

περιβάλλον .NET αφού περιείχαν ασυμβατότητες και η .NET προσπαθούσε να «αναβαθμίσει» τον κώδικα , αλλάζοντας φυσικά την δομή του.

Η τεκμηρίωση του κώδικα έγινε αποκλειστικά στα αγγλικά, πρώτον για να είναι συμβατή με οποιοδήποτε Η/Υ και λειτουργικό σύστημα και δεύτερον γιατί πολλοί αποσφαλματωτές της γλώσσας C++ συναντούν σοβαρά προβλήματα όταν ανάμεσα στον κώδικα συναντούν σχόλια με ελληνικούς χαρακτήρες.

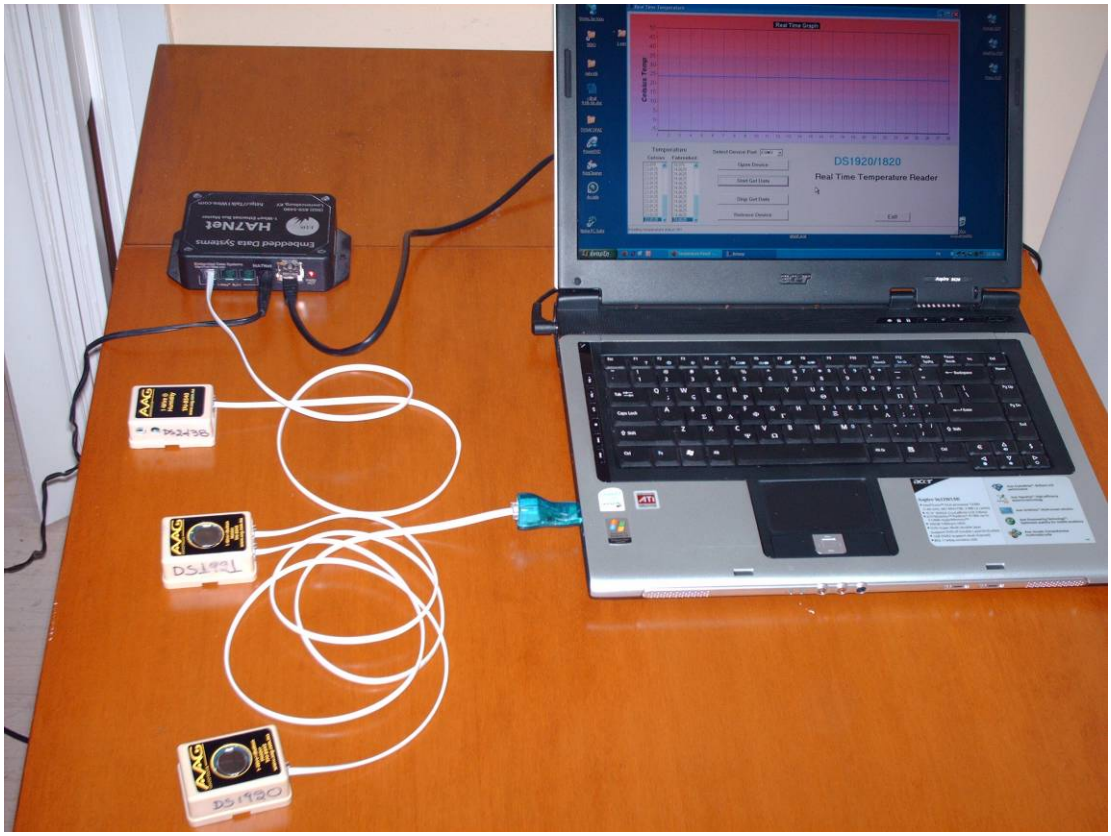
4. Αποσφαλμάτωση και Πιλοτική Εφαρμογή

4.1. Διαθέσιμες συσκευές

Όταν ξεκίνησε η εργασία (Οκτώβριος 2005) στην κατοχή μας βρισκόντουσαν οι εξής συσκευές:

- Σειριακή master συσκευή DS2480
- USB master συσκευή DS9490R
- 1 συσκευή καταγραφής θερμοκρασίας DS1920
- 1 συσκευή καταγραφής δειγμάτων θερμοκρασίας DS1921
- 1 συσκευή καταγραφής θερμοκρασίας και/ή υγρασίας DS2438
- 2 συσκευές καταγραφής θερμοκρασίας DS18B20
- 1 http Server HA7net

Στην επόμενη εικόνα , φαίνεται η διασύνδεση αυτών των συσκευών σε δύο δίκτυα 1-Wire. Ένα δίκτυο μέσω της USB master συσκευής και ένα άλλο δίκτυο μέσω του http server (Ο συγκεκριμένος φορητός Η/Υ δεν είχε σειριακή πόρτα (RS232).



Εικόνα 9: Υλοποίηση 2 δικτύων 1-Wire σε φορητό Η/Υ

4.2. Δημιουργία Demo Εφαρμογής

Για τις ανάγκες της εργασίας αποφασίστηκε να δημιουργηθεί μία εφαρμογή με τα εξής χαρακτηριστικά:

- Ως βιβλιοθήκες χρησιμοποιήθηκαν όλα τα αρχεία που δημιουργήθηκαν για την εργασία.
- Ως μετατροπέας (adaptor) αποφασίστηκε να χρησιμοποιηθεί ο μόνος αξιόπιστος [(1)] που υπήρχε στην διάθεση μας την δεδομένη χρονική στιγμή, ο **USB DS9490R**.
- Ως συσκευές ανάγνωσης-καταγραφής δεδομένων χρησιμοποιήθηκαν οι:
    - **DS1920** συσκευή καταγραφής θερμοκρασίας σε πραγματικό χρόνο
    - **DS1921** συσκευή παραμετροποίησης καταγραφής δειγμάτων θερμοκρασίας σε τακτά χρονικά διαστήματα (Temperature Sampling)
    - **DS2438** συσκευή καταγραφής θερμοκρασίας και υγρασίας σε πραγματικό χρόνο
- Ως σημεία αναφοράς χρησιμοποιήθηκαν οι βιβλιοθήκες που συνοδεύουν τις δύο παραπάνω συσκευές.

Οι βιβλιοθήκες αυτές είναι γραμμένες για περιβάλλον DOS και μόνο για χρήση με τις βιβλιοθήκες του USB μετατροπέα. Χρησιμοποιήθηκε το κομμάτι του κώδικα που αναφέρεται στις επιμέρους λειτουργίες που χρησιμοποιεί η κάθε συσκευή, και όλο το άλλο πρόγραμμα (παραθυρική εφαρμογή), «χτίστηκε» από την αρχή. Οι βιβλιοθήκες που χρησιμοποιήθηκαν περιγράφονται στο τέλος του κεφαλαίου και εμπεριέχονται στο παράρτημα. Φυσικά όλες οι βιβλιοθήκες ανήκουν στην Dallas Semiconductors και βρέθηκαν στο σχετικό δικτυακό τόπο της εταιρείας (www.1wire.com/downloads/ usbW32VC300.zip

Πρέπει να αναφέρουμε ότι στην διάθεση μας υπήρχε επίσης ο σειριακός μετατροπέας DS9097U, για τον οποίον μάλιστα ορισμένες υλοποιήσεις ήταν πάρα πολύ πιο εύκολές από τον USB μετατροπέα που χρησιμοποιήθηκε τελικώς, και φυσικά ο σειριακός μετατροπέας δεν χρειάζεται κανενός είδους «οδηγό» (driver) αφού υποστηρίζεται εγγενώς από όλα τα Λειτουργικά Συστήματα. Δυστυχώς ενώ από τον Οκτώβριο που ήταν διαθέσιμος μέχρι τα Χριστούγεννα περίπου συνεργαζόταν άψογα με δύο Η/Υ που είχαμε στην διάθεση μας –έναν φορητό Η/Υ και έναν επιτραπέζιο- κάποια στιγμή σταμάτησε να αναγνωρίζεται από οποιοδήποτε πρόγραμμα 1-Wire ακόμη και αυτά που έχουν δημιουργηθεί από την εταιρεία μόνο για τον συγκεκριμένο προσαρμογέα. Επίσης στην απευθείας σύνδεση με το Hyper Terminal δεν έδινε σωστά στοιχεία επικοινωνίας. Έτσι αποφασίστηκε τελικά η χρήση μόνο του USB προσαρμογέα για την υλοποίηση της όποιας πιλοτικής εφαρμογής.

4.3. Περιγραφή των χρησιμοποιηθέντων συσκευών

4.3.1. DS1920

Καταγραφή θερμοκρασίας σε πραγματικό χρόνο

Η συγκεκριμένη συσκευή , δεν έχει μνήμη (EPROM) και έτσι δεν αποθηκεύει δεδομένα. Για να λειτουργήσει χρειάζεται ισχύ (power) από το δίκτυο 1-Wire. Όταν είναι συνδεδεμένη στο δίκτυο , δείχνει την θερμοκρασία του περιβάλλοντος σε πραγματικό χρόνο. Μπορεί να καταγράψει θερμοκρασίες από -40^{o} C μέχρι 85^{o} C., με ακρίβεια 3 δεκαδικών ψηφίων.

4.3.2. DS2438

Καταγραφή υγρασίας και θερμοκρασίας σε πραγματικό χρόνο

Η συγκεκριμένη συσκευή , δεν έχει μνήμη (EPROM) και έτσι δεν αποθηκεύει δεδομένα. Για να λειτουργήσει χρειάζεται ισχύ (power) από το δίκτυο 1-Wire. Όταν είναι συνδεδεμένη στο δίκτυο , δείχνει την υγρασία και την θερμοκρασία του περιβάλλοντος σε πραγματικό χρόνο. Μπορεί να καταγράψει την υγρασία του περιβάλλοντος με ακρίβεια 6 δεκαδικών ψηφίων (αναπαράσταση double C++) και θερμοκρασίες από -40^{o} C μέχρι 85^{o} C., με ακρίβεια 4 δεκαδικών ψηφίων.

4.3.3. DS1921

Καταγραφή δειγμάτων θερμοκρασίας σε προκαθορισμένο χρόνο (temperature sampling)
Η συγκεκριμένη συσκευή έχει μνήμη EPROM. Αυτό σημαίνει ότι η συσκευή μπορεί να λειτουργήσει αυτόνομα. Μπορεί δηλαδή να προγραμματιστεί όντας συνδεδεμένη πάνω στο δίκτυο 1-wire, μια μετά να αποσυνδεθεί και να παίρνει δείγματα θερμοκρασίας από τον περιβάλλοντα χώρο της. Έχει δυνατότητα πλήρους παραμετροποίησης, δηλαδή ποιο θα είναι το χρονικό διάστημα που θα μεσολαβεί μεταξύ δύο δειγμάτων, αν θα ξανα-αρχίζει από την αρχή αν η μνήμη της γεμίσει και δεν έχει τελειώσει η αποστολή της, αν θα αποθηκεύει τις θερμοκρασίες σε βαθμούς Κελσίου ή Φαρενάιτ κτλ. επίσης έχει την δυνατότητα να μας δείξει αν κατά την διάρκεια της αποστολής της, οι θερμοκρασίες υπερέβησαν κάποιο ή κάποια ανώτατα ή κατώτατα όρια (High, Low Threshold) (χρήσιμο όταν π.χ. η συσκευή συνοδεύει

παγωτά που προφανώς μας ενδιαφέρει να μην έχει ανεβεί η θερμοκρασία τους πάνω από το μηδέν και έχουν αλλοιωθεί). Το βασικότερο όλων είναι ότι ο χρήστης πρέπει να διαλέξει τον ρυθμό καταγραφής θερμοκρασίας (sampling rate). Φυσικά δεδομένου του ότι η EPROM μνήμη της συσκευής είναι περιορισμένη και μάλλον μικρή σε μέγεθος, η επιλογή του ρυθμού καταγραφής έχει άμεση σχέση με την διάρκεια του χρόνου καταγραφής. Έτσι αν ο χρήστης επιλέξει χρόνο καταγραφής κάθε λεπτό, τότε η διάρκεια του συνολικού χρόνου καταγραφής θα είναι περίπου 1,4 ημέρες. Αντίθετα με έναν ρυθμό καταγραφής 30 λεπτών (1/2 ώρα) ο συνολικός χρόνος φτάνει στις 42 ημέρες, χρόνος που θεωρείται υπέρ-αρκετός για πολλών ειδών εφαρμογές (π.χ. υπερατλαντικά ταξίδια ευπαθών προϊόντων με πλοίο,).

Πρέπει να σημειωθεί εδώ, ότι η συσκευή χρειάζεται να συνδεθεί στο 1-Wire δίκτυο *μόνον* κατά την διάρκεια του προγραμματισμού-σχεδιασμού της λειτουργίας καταγραφής. Εφόσον έχει ολοκληρωθεί το πλάνο καταγραφής και η συσκευή είναι έτοιμη να καταγράψει, μπορεί να αφαιρεθεί από το δίκτυο και να τοποθετηθεί όπου χρειάζεται. Την επόμενη φορά που θα συνδεθεί σε ένα δίκτυο 1-Wire, μπορεί να παρουσιάσει τα δεδομένα που έχει αποθηκεύσει-καταγράψει.

4.4. Περιγραφή εφαρμογής

Η εφαρμογή που δημιουργήθηκε είχε ως πρωταρχικό σκοπό την επίδειξη της ικανότητας του δημιουργηθέντος API να «χτίζει» εφαρμογές πάνω σε οποιαδήποτε master συσκευή και φυσικά έπρεπε να δείξει την εμφανή ενσωμάτωση των δυνατοτήτων που παρέχει το API που δημιουργήθηκε. Έτσι δημιουργήθηκε μία εφαρμογή η οποία στην εισαγωγική οθόνη δίνει τρεις επιλογές:

Πρώτη επιλογή: Παρουσίαση σε πραγματικό χρόνο των δεδομένων από μία συσκευή Δεδομένα θερμοκρασίας από την συσκευή DS1920. Υπάρχουν συνολικά 2 πίνακες, που εμφανίζουν την θερμοκρασία της DS1920 σε δύο κλίμακες (Κελσίου και Φαρενάιτ) και ένα γράφημα που απεικονίζει την διακύμανση της θερμοκρασίας σε βαθμούς κελσίου. Φυσικά υπάρχουν τα απαραίτητα κουμπιά (buttons) για να καταδείξουν την ορθή χρήση των συναρτήσεων του API. Έτσι υπάρχει κουμπί (Open Devices) το οποίο εκτελεί τις συναρτήσεις FindDevices() & Aquire() (ΠΑΡΑΡΤΗΜΑ Α), κουμπί (Release Devices) που εκτελεί την owRelease() (ΠΑΡΑΡΤΗΜΑ Α), και φυσικά τα κουμπιά (Start Get Data) που

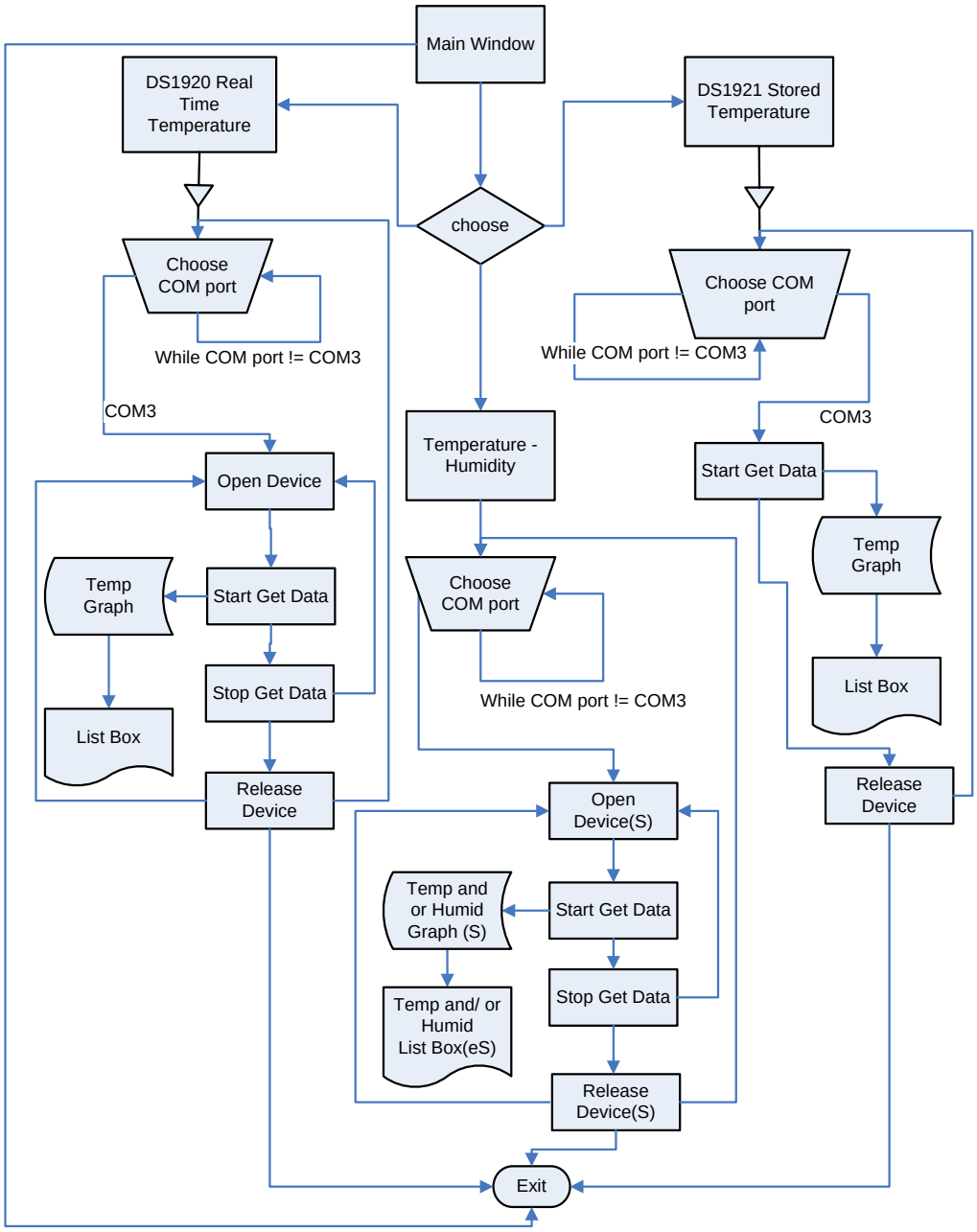
είναι απαραίτητα για την εμφάνιση των δεδομένων στους πίνακες , και το κουμπί (Stop Get Data) που σταματά την εμφάνιση των δεδομένων.

Δεύτερη επιλογή: Αναλυτική παρουσίαση των δεδομένων της συσκευής DS1921. Αφού έχει προηγηθεί η παραμετροποίηση και λήψη δειγμάτων από την συσκευή, αυτή συνδέεται στο δίκτυο για να «ξεφορτώσει» τα αποθηκευμένα δεδομένα της. Έτσι υπάρχει κουμπί (Open Devices) το οποίο εκτελεί τις συναρτήσεις FindDevices() & Aquire() (ΠΑΡΑΡΤΗΜΑ Α), κουμπί (Release Devices) που εκτελεί την owRelease() (ΠΑΡΑΡΤΗΜΑ Α), και φυσικά το κουμπί (Start Get Data) που είναι απαραίτητα για την εμφάνιση των δεδομένων στους πίνακες και στο γράφημα όπου φαίνεται αναλυτικά η διακύμανση των αποθηκευμένων στην συσκευή, δειγμάτων θερμοκρασίας.

Τρίτη επιλογή: Παρουσίαση σε πραγματικό χρόνο των δεδομένων από δύο συσκευές. Δεδομένα θερμοκρασίας στο ίδιο γράφημα από δύο συσκευές, την DS1920 και την DS2438. με αυτόν τον τρόπο μπορούμε να δούμε την διαφορά στις μετρήσεις των δύο συσκευών αν βρίσκονται σε διαφορετικά περιβάλλοντα. Στην ίδια οθόνη βλέπουμε ακόμη ένα επιπλέον γράφημα στο οποίο εμφανίζεται σε πραγματικό χρόνο η υγρασία που μετρά η DS2438. επίσης υπάρχουν συνολικά 4 πίνακες, εκ των οποίων οι δύο εμφανίζουν την θερμοκρασία της DS1920 σε δύο κλίμακες (Κελσίου και Φαρενάιτ) και δύο πίνακες για την DS2438 εκ των οποίων ο ένας εμφανίζει την θερμοκρασία και ο άλλος την υγρασία. Φυσικά υπάρχουν τα απαραίτητα κουμπιά (buttons) για να καταδείξουν την ορθή χρήση των συναρτήσεων του API. Έτσι υπάρχει κουμπί (Open Devices) το οποίο εκτελεί τις συναρτήσεις FindDevices() & Aquire() *(ΠΑΡΑΡΤΗΜΑ Α)*, κουμπί (Release Devices) που εκτελεί την owRelease() *(ΠΑΡΑΡΤΗΜΑ Α)*, και φυσικά τα κουμπιά (Start Get Data) που είναι απαραίτητα για την εμφάνιση των δεδομένων στους πίνακες , και το κουμπί (Stop Get Data) που σταματά την εμφάνιση των δεδομένων.

4.5. Διάγραμμα Ροής εφαρμογής

Στη συνέχεια παρουσιάζουμε το αναλυτικό διάγραμμα ροής της εφαρμογής.



Σχήμα 22: Διάγραμμα Ροής, Flow Chart

4.6. Περιπτώσεις Χρήσης Εφαρμογής

ΒΑΣΙΚΗ ΦΟΡΜΑ ΕΦΑΡΜΟΓΗΣ 1-WIRE DEMO

Περίπτωση Χρήσης 1.

1. Τίτλος περίπτωσης χρήσης

Real Time Temperature: Επιλογή φόρμας για την εμφάνιση/εκτύπωση δεδομένων θερμοκρασίας πραγματικού χρόνου από την συσκευή DS1820/1920

2. Σύντομη περιγραφή

Επιλέγοντας το κουμπί της εφαρμογής, ανοίγει η αντίστοιχη φόρμα DS1820/1920

3. Ροή γεγονότων.

1.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Real Time Temperature
2. κλείνει η ανοιχτή φόρμα και ανοίγει η φόρμα της αντίστοιχης περίπτωσης χρήσης

1.2 Εναλλακτικές ροές

Δεν υπάρχουν εναλλακτικές ροές

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα είναι στην αρχική κατάσταση

6. Κατάσταση εξόδου

Το σύστημα είναι στην αρχική κατάσταση

Περίπτωση Χρήσης 2.

1. Τίτλος περίπτωσης χρήσης

Stored Temperature: Επιλογή φόρμας για την εμφάνιση/εκτύπωση αποθηκεμένων δεδομένων θερμοκρασίας από την συσκευή DS1921

2. Σύντομη περιγραφή

Επιλέγοντας το κουμπί της εφαρμογής, ανοίγει η σχετική φόρμα

3. Ροή γεγονότων.

1.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Stored Temperature
2. κλείνει η ανοιχτή φόρμα και ανοίγει η φόρμα της αντίστοιχης περίπτωσης χρήσης

1.2 Εναλλακτικές ροές

Δεν υπάρχουν εναλλακτικές ροές

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα είναι στην αρχική κατάσταση

6. Κατάσταση εξόδου

Το σύστημα είναι στην αρχική κατάσταση

Περίπτωση Χρήσης 3.

1. Τίτλος περίπτωσης χρήσης

Exit: Κουμπί εξόδου από την εφαρμογή.

2. Σύντομη περιγραφή

Επιλέγοντας το κουμπί της εφαρμογής, κλείνει η αρχική φόρμα

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Exit

3.2 Εναλλακτικές ροές

Δεν υπάρχουν εναλλακτικές ροές

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα είναι στην αρχική κατάσταση

6. Κατάσταση εξόδου

Το σύστημα είναι στην αρχική κατάσταση

Περίπτωση Χρήσης 4.

1. Τίτλος περίπτωσης χρήσης

About: Κουμπί εμφάνισης πληροφοριών της εφαρμογής.

2. Σύντομη περιγραφή

Επιλέγοντας το κουμπί της εφαρμογής, εμφανίζονται η version της εφαρμογής και τα ονόματα προγραμματιστή και συμβούλου καθηγητή

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί About

3.2 Εναλλακτικές ροές

Δεν υπάρχουν εναλλακτικές ροές

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα είναι στην αρχική κατάσταση

6. Κατάσταση εξόδου

Το σύστημα είναι στην αρχική κατάσταση

ΦΟΡΜΑ ΑΝΑΓΝΩΣΗΣ ΘΕΡΜΟΚΡΑΣΙΑΣ ΣΥΣΚΕΥΩΝ DS1820/1920

Περίπτωση Χρήσης 5.

1. Τίτλος περίπτωσης χρήσης

Select Device Port: Επιλογή «πόρτας» (θύρας) που βρίσκεται η συσκευή 1-Wire

2. Σύντομη περιγραφή

Πρέπει να επιλεγεί η κατάλληλη θύρα του υπολογιστή, στην οποία βρίσκεται ο μετατροπέας- adaptor 1-Wire. Αν η θύρα είναι η σωστή, τότε εκτελείται η owAquire() και βγαίνει κατάλληλο μήνυμα στην Status Bar στο κάτω μέρος της οθόνης υποδοχής.

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί με τον αριθμό της πόρτας (drop down list).
2. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Port opened).
3. Εκτελείται η owAquire() με τις κατάλληλες παραμέτρους.
4. Το σύστημα αναμένει με την συγκεκριμένη πόρτα «ανοιχτή» (Port Aquired)

3.2 Εναλλακτικές ροές

Αν επιλεγεί οποιαδήποτε πόρτα που δεν έχει μετατροπέα συνδεδεμένο (<> COM3 στην περίπτωση μας) το σύστημα δεν αλλάζει κατάσταση και βγαίνει σχετικό μήνυμα στην status bar (device not found)

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα να είναι στην αρχική κατάσταση (Stop).

6. Κατάσταση εξόδου

Το σύστημα είναι σε κατάσταση αναμονής με την πόρτα-συσκευή έτοιμη.

Περίπτωση Χρήσης 6.

1. Τίτλος περίπτωσης χρήσης

Open Device: Επιλογή της συγκεκριμένης συσκευής 1-Wire (DS1920/1820)

2. Σύντομη περιγραφή

Πρέπει να υπάρχει η κατάλληλη συσκευή (DS1920/1820), στην οποία βρίσκεται ο μετατροπέας- adaptor 1-Wire. Αν η συσκευή είναι η σωστή, τότε εκτελείται η owAquire() και βγαίνει κατάλληλο μήνυμα στην Status Bar στο κάτω μέρος της οθόνης υποδοχής.

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Open Device.
2. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Device opened).
3. Εκτελείται η FindDevices() η οποία με την σειρά της καλεί τις owFamilySearchSetup(), owNext, owSerialNum()
4. Το σύστημα αναμένει με την συγκεκριμένη συσκευή «ανοιχτή» (Device Aquired)

3.2 Εναλλακτικές ροές

Αν δεν υπάρχει τέτοια συσκευή συνδεδεμένη στο δίκτυο το σύστημα δεν αλλάζει κατάσταση και βγαίνει σχετικό μήνυμα στην status bar (device not found)

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα να είναι σε κατάσταση αναμονής με την πόρτα-συσκευή έτοιμη.

6. Κατάσταση εξόδου

Το σύστημα είναι σε κατάσταση αναμονής με την συγκεκριμένη συσκευή (DS1920/1820) έτοιμη να αποστείλει δεδομένα..

Περίπτωση Χρήσης 7.

1. Τίτλος περίπτωσης χρήσης

Start Get Data: Υποδοχή δεδομένων (DS1920/1820)

2. Σύντομη περιγραφή

Πρέπει να είναι ήδη «ανοιχτή» η κατάλληλη συσκευή (DS1920/1820), στην οποία βρίσκεται ο μετατροπέας- adaptor 1-Wire. Αν η συσκευή είναι η σωστή, τότε αρχίζει το «κατέβασμα» (download) δεδομένων και βγαίνει κατάλληλο μήνυμα στην Status Bar στο κάτω μέρος της οθόνης υποδοχής.

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Start Get Data.
2. Το σύστημα αρχίζει να κατεβάζει δεδομένα από την συσκευή και να τα εμφανίζει στα δύο List-boxes (Celsius & Fahrenheit) και επίσης στέλνει τους βαθμούς Κελσίου στο γράφημα.
3. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Receiving Data).
4. Το σύστημα αναμένει με την συγκεκριμένη συσκευή «ανοιχτή» (Device Acquired)

3.2 Εναλλακτικές ροές

Αν πατηθεί για δεύτερη φορά το «κουμπί» τότε δεν συμβαίνει τίποτα. (device already on)

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα είναι σε κατάσταση αναμονής με την συγκεκριμένη συσκευή (DS1920/1820) έτοιμη να αποστείλει δεδομένα.

6. Κατάσταση εξόδου

Το σύστημα στέλνει δεδομένα σε πραγματικό χρόνο σε γράφημα και σε δύο list boxes ένα με βαθμούς Κελσίου και ένα με βαθμούς Φαρενάιτ.

Περίπτωση Χρήσης 8.

1. Τίτλος περίπτωσης χρήσης

Stop Get Data: Σταμάτημα Υποδοχής δεδομένων (DS1920/1820)

2. Σύντομη περιγραφή

Εάν ήδη έχει αρχίσει η υποδοχή δεδομένων από την συσκευή, τότε αυτή σταματάει και βγαίνει κατάλληλο μήνυμα στην status bar (Reading Temperature aborted by user).

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Stop Get Data.

2. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Reading Temperature aborted by user).

3. Το σύστημα αναμένει με την συγκεκριμένη συσκευή «ανοιχτή» (Device Acquired), αλλά δεν υποδέχεται δεδομένα.

3.2 Εναλλακτικές ροές

Αν δεν υπάρχει τέτοια συσκευή συνδεδεμένη στο δίκτυο, ή εάν έχει επιλεγεί λάθος θύρα ή δεν έχει γίνει προηγουμένως άνοιγμα της θύρας, το σύστημα δεν αλλάζει κατάσταση.

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα στέλνει δεδομένα σε πραγματικό χρόνο σε γράφημα και σε δύο list boxes ένα με βαθμούς Κελσίου και ένα με βαθμούς Φαρενάιτ. .

6. Κατάσταση εξόδου

Το σύστημα σταματάει να στέλνει δεδομένα σε πραγματικό χρόνο σε γράφημα και σε δύο list boxes.

Περίπτωση Χρήσης 9.

1. Τίτλος περίπτωσης χρήσης

Release Device: απελευθέρωση συσκευής (DS1920/1820)

2. Σύντομη περιγραφή

Εάν ήδη έχει αρχίσει η υποδοχή δεδομένων από την συσκευή, τότε αυτή σταματάει, απελευθερώνεται το δίκτυο και βγαίνει κατάλληλο μήνυμα στην status bar (Device Closed by user).

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Release Device.

2. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Device Closed by user).

3. Το σύστημα εκτελεί την owRelease() που απελευθερώνει την-τις συσκευές του δικτύου.

3.2 Εναλλακτικές ροές

1. Αν δεν υπάρχει τέτοια συσκευή συνδεδεμένη στο δίκτυο, ή εάν έχει επιλεγεί λάθος θύρα ή δεν έχει γίνει προηγουμένως άνοιγμα της θύρας, το σύστημα δεν αλλάζει κατάσταση

2. Πριν εκτελέσει την owRelease το σύστημα ελέγχει αν προηγουμένως έχει εκτελεστεί η owAquire() (γιατί αλλιώς δημιουργείται exception).

2. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

3. Κατάσταση εισόδου

1. Το σύστημα στέλνει δεδομένα σε πραγματικό χρόνο σε γράφημα και σε δύο list boxes ένα με βαθμούς Κελσίου και ένα με βαθμούς Φαρενάιτ.

2. Το σύστημα είναι σε κατάσταση αναμονής με την συγκεκριμένη συσκευή (DS1920/1820) έτοιμη να αποστείλει δεδομένα..

4. Κατάσταση εξόδου

Το σύστημα σταματάει να στέλνει δεδομένα και απελευθερώνει την-τις συσκευές του δικτύου.

Περίπτωση Χρήσης 10.

1. Τίτλος περίπτωσης χρήσης

Exit: Κλείσιμο φόρμας

2. Σύντομη περιγραφή

Καθαρίζουν τα δεδομένα από το γράφημα και τα 2 list boxes και κλείνει η φόρμα

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Exit.
2. Το σύστημα καθαρίζει το γράφημα και τα 2 list boxes .
3. Κλείνει η φόρμα.

3.2 Εναλλακτικές ροές

Δεν υπάρχουν εναλλακτικές ροές.

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα σταματάει να στέλνει δεδομένα και απελευθερώνει την-τις συσκευές του δικτύου.

6. Κατάσταση εξόδου

Το σύστημα επιστρέφει στην βασική φόρμα της εφαρμογής.

ΦΟΡΜΑ ΑΝΑΓΝΩΣΗΣ ΑΠΟΘΗΚΕΜΕΝΩΝ ΔΕΔΟΜΕΝΩΝ ΘΕΡΜΟΚΡΑΣΙΑΣ ΣΥΣΚΕΥΗΣ DS1921

Περίπτωση Χρήσης 11.

1. Τίτλος περίπτωσης χρήσης

Select Device Port: Επιλογή «πόρτας» (θύρας) που βρίσκεται η συσκευή 1-Wire

2. Σύντομη περιγραφή

Πρέπει να επιλεγεί η κατάλληλη θύρα του υπολογιστή, στην οποία βρίσκεται ο μετατροπέας- adaptor 1-Wire. Αν η θύρα είναι η σωστή, τότε εκτελείται η owAquire() και βγαίνει κατάλληλο μήνυμα στην Status Bar στο κάτω μέρος της οθόνης υποδοχής.

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί με τον αριθμό της πόρτας (drop down list).
2. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Port opened).
3. Εκτελείται η owAquire() με τις κατάλληλες παραμέτρους.
4. Το σύστημα αναμένει με την συγκεκριμένη πόρτα «ανοιχτή» (Port Aquired)

3.2 Εναλλακτικές ροές

Αν επιλεγεί οποιαδήποτε πόρτα που δεν έχει μετατροπέα συνδεδεμένο (<> COM3 στην περίπτωση μας) το σύστημα δεν αλλάζει κατάσταση και βγαίνει σχετικό μήνυμα στην status bar (device not found)

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα να είναι στην αρχική κατάσταση (Stop).

6. Κατάσταση εξόδου

Το σύστημα είναι σε κατάσταση αναμονής με την πόρτα-συσκευή έτοιμη.

Περίπτωση Χρήσης 12.

1. Τίτλος περίπτωσης χρήσης

Start Get Data: Υποδοχή δεδομένων (DS1921)

2. Σύντομη περιγραφή

Πρέπει να είναι ήδη «ανοιχτή» η κατάλληλη συσκευή (DS1921), στην οποία βρίσκεται ο μετατροπέας- adaptor 1-Wire. Αν η συσκευή είναι η σωστή, τότε αρχίζει το «κατέβασμα»

(download) δεδομένων και βγαίνει κατάλληλο μήνυμα στην Status Bar στο κάτω μέρος της οθόνης υποδοχής.

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Start Get Data.
2. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Reading Device, Please Wait).
3. Κατεβαίνουν τα δεδομένα από την συσκευή και τροποποιούνται ώστε να είναι δυνατόν να εμφανιστούν ξεχωριστά και ομαδοποιημένα
4. Δεδομένα εμφανίζονται ταξινομημένα στα δύο γραφήματα, στα δύο memoboxes και στο ένα listbox
5. ανοίγει ένα αρχείο με το όνομα output.txt στο οποίο αποθηκεύονται όλα τα παραπάνω δεδομένα

3.2 Εναλλακτικές ροές

Αν πατηθεί για δεύτερη φορά το «κουμπί» τότε δεν συμβαίνει τίποτα. (device already on)

4. Μη λειτουργικές απαιτήσεις

Αν δεν εκτελεστούν όλες οι λειτουργίες, το σύστημα εμφανίζει αντίστοιχο μήνυμα λάθους στην status bar

5. Κατάσταση εισόδου

Το σύστημα είναι σε κατάσταση αναμονής με την συγκεκριμένη συσκευή (DS1920/1820) έτοιμη να αποστείλει δεδομένα.

6. Κατάσταση εξόδου

Το σύστημα κατεβάζει δεδομένα από την συσκευή στην οθόνη και σε αρχείο.

Περίπτωση Χρήσης 13.

1. Τίτλος περίπτωσης χρήσης

Release Device: απελευθέρωση συσκευής (DS1920/1820)

2. Σύντομη περιγραφή

Εάν ήδη έχει αρχίσει η υποδοχή δεδομένων από την συσκευή, τότε αυτή σταματάει, απελευθερώνεται το δίκτυο και βγαίνει κατάλληλο μήνυμα στην status bar (Device Closed by user).

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Release Device.

2. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Device Closed by user).

3. Το σύστημα εκτελεί την owRelease() που απελευθερώνει την-τις συσκευές του δικτύου.

3.2 Εναλλακτικές ροές

1. Αν δεν υπάρχει τέτοια συσκευή συνδεδεμένη στο δίκτυο, ή εάν έχει επιλεγεί λάθος θύρα ή δεν έχει γίνει προηγουμένως άνοιγμα της θύρας, το σύστημα δεν αλλάζει κατάσταση

2. Πριν εκτελέσει την owRelease το σύστημα ελέγχει αν προηγουμένως έχει εκτελεστεί η owAquire() (γιατί αλλιώς δημιουργείται exception).

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα είναι σε κατάσταση αναμονής με την συγκεκριμένη συσκευή (DS1921) έτοιμη να αποστείλει δεδομένα..

6. Κατάσταση εξόδου

Το σύστημα απελευθερώνει την-τις συσκευές του δικτύου.

Περίπτωση Χρήσης 14.

1. Τίτλος περίπτωσης χρήσης

Exit: Κλείσιμο φόρμας

2. Σύντομη περιγραφή

Καθαρίζουν τα δεδομένα από το γράφημα και τα list boxes και κλείνει η φόρμα

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Exit.

2. Το σύστημα καθαρίζει το γράφημα και τα list boxes .

3. Κλείνει η φόρμα.

3.2 Εναλλακτικές ροές

Δεν υπάρχουν εναλλακτικές ροές.

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα βρίσκεται είτε στην αρχική κατάσταση.

6. Κατάσταση εξόδου

Το σύστημα επιστρέφει στην βασική φόρμα της εφαρμογής

ΦΟΡΜΑ ΑΝΑΓΝΩΣΗΣ ΘΕΡΜΟΚΡΑΣΙΑΣ & ΥΓΡΑΣΙΑΣ ΣΥΣΚΕΥΩΝ DS1920 & DS2438

Περίπτωση Χρήσης 15.

1. Τίτλος περίπτωσης χρήσης

Select Device Port: Επιλογή «πόρτας» (θύρας) που βρίσκονται οι συσκευές 1-Wire

2. Σύντομη περιγραφή

Πρέπει να επιλεγεί η κατάλληλη θύρα του υπολογιστή, στην οποία βρίσκεται ο μετατροπέας- adaptor 1-Wire. Αν η θύρα είναι η σωστή, τότε εκτελείται η owAquire() και βγαίνει κατάλληλο μήνυμα στην Status Bar στο κάτω μέρος της οθόνης υποδοχής.

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί με τον αριθμό της πόρτας (drop down list).
2. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Port opened).
3. Εκτελείται η owAquire() με τις κατάλληλες παραμέτρους.
4. Το σύστημα αναμένει με την συγκεκριμένη πόρτα «ανοιχτή» (Port Aquired)

3.2 Εναλλακτικές ροές

Αν επιλεγεί οποιαδήποτε πόρτα που δεν έχει μετατροπέα συνδεδεμένο (<> COM3 στην περίπτωση μας) το σύστημα δεν αλλάζει κατάσταση και βγαίνει σχετικό μήνυμα στην status bar (device not found)

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα να είναι στην αρχική κατάσταση (Stop).

6. Κατάσταση εξόδου

Το σύστημα είναι σε κατάσταση αναμονής με την πόρτα-συσκευή έτοιμη.

Περίπτωση Χρήσης 16.

1. Τίτλος περίπτωσης χρήσης

Open Device: Επιλογή της συγκεκριμένης συσκευής 1-Wire (DS1920/1820)

2. Σύντομη περιγραφή

Πρέπει να υπάρχει η κατάλληλη συσκευή (DS1920 ή/και η DS2438), με τις οποίες είναι συνδεδεμένος ο μετατροπέας- adaptor 1-Wire. Αν η συσκευή είναι η σωστή, τότε εκτελείται η owAquire() και βγαίνει κατάλληλο μήνυμα στην Status Bar στο κάτω μέρος της οθόνης υποδοχής. (Device(s) found)

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Open Device.
2. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Device opened).
3. Εκτελείται η FindDevices() η οποία με την σειρά της καλεί τις owFamilySearchSetup(), owNext, owSerialNum()
4. Το σύστημα αναμένει με την συγκεκριμένη συσκευή «ανοιχτή» (Device Aquired)

3.2 Εναλλακτικές ροές

Αν δεν υπάρχει τέτοια συσκευή συνδεδεμένη στο δίκτυο το σύστημα δεν αλλάζει κατάσταση και βγαίνει σχετικό μήνυμα στην status bar (device not found)

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα να είναι σε κατάσταση αναμονής με την πόρτα-συσκευή έτοιμη.

6. Κατάσταση εξόδου

Το σύστημα είναι σε κατάσταση αναμονής με την συγκεκριμένη συσκευή (DS1920 και/ή DS2438) έτοιμη να αποστείλει δεδομένα..

Περίπτωση Χρήσης 17.

1. Τίτλος περίπτωσης χρήσης

Start Get Data: Υποδοχή δεδομένων *(DS1920 και/ή DS2438)*

2. Σύντομη περιγραφή

Πρέπει να είναι ήδη «ανοιχτή» η κατάλληλη συσκευή *(DS1920 και/ή DS2438)* στην οποία βρίσκεται ο μετατροπέας- adaptor 1-Wire. Αν η συσκευή/συσκευές είναι η σωστή/σωστές, τότε αρχίζει η εμφάνιση δεδομένων στο γράφημα και στους πίνακες και βγαίνει κατάλληλο μήνυμα στην Status Bar στο κάτω μέρος της οθόνης υποδοχής.

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Start Get Data.

2. Το σύστημα αρχίζει να κατεβάζει δεδομένα από τις συσκευές και να τα εμφανίζει στα δύο List-boxes (Celsius & Fahrenheit) και επίσης στέλνει τους βαθμούς Κελσίου στο γράφημα.

3. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Receiving Data).

4. Το σύστημα αναμένει με τις συγκεκριμένες συσκευές «ανοιχτές» (Device(S) Acquired)

3.2 Εναλλακτικές ροές

Αν πατηθεί για δεύτερη φορά το «κουμπί» τότε δεν συμβαίνει τίποτα. (device already on)

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα είναι σε κατάσταση αναμονής με την συγκεκριμένη συσκευή (DS1920 και/ή DS2438) έτοιμη να αποστείλει δεδομένα.

6. Κατάσταση εξόδου

Το σύστημα στέλνει δεδομένα σε πραγματικό χρόνο σε δύο γραφήματα και σε δύο list boxes ένα με βαθμούς Κελσίου και ένα με βαθμούς Φαρενάιτ.

Περίπτωση Χρήσης 18.

1. Τίτλος περίπτωσης χρήσης

Stop Get Data: Σταμάτημα Υποδοχής δεδομένων (DS1920 και/ή DS2438)

2. Σύντομη περιγραφή

Εάν ήδη έχει αρχίσει η υποδοχή δεδομένων από τις συσκευές, τότε αυτή σταματάει και βγαίνει κατάλληλο μήνυμα στην status bar (Reading Data aborted by user).

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Stop Get Data.

2. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Reading Temperature aborted by user).

3. Το σύστημα αναμένει με την συγκεκριμένη συσκευή (DS1920 και/ή DS2438) «ανοιχτή» (Device Acquired), αλλά δεν υποδέχεται δεδομένα.

3.2 Εναλλακτικές ροές

Αν δεν υπάρχει καμία τέτοια συσκευή (DS1920 και/ή DS2438) συνδεδεμένη στο δίκτυο, ή εάν έχει επιλεγεί λάθος θύρα ή δεν έχει γίνει προηγουμένως άνοιγμα της θύρας, το σύστημα δεν αλλάζει κατάσταση.

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα στέλνει δεδομένα σε πραγματικό χρόνο σε δύο γραφήματα και σε δύο list boxes ένα με βαθμούς Κελσίου και ένα με βαθμούς Φαρενάιτ. .

6. Κατάσταση εξόδου

Το σύστημα σταματάει να στέλνει δεδομένα σε πραγματικό χρόνο σε γράφημα και σε δύο list boxes.

Περίπτωση Χρήσης 19.

1. Τίτλος περίπτωσης χρήσης

Release Device: απελευθέρωση συσκευών (DS1920 και/ή DS2438)

2. Σύντομη περιγραφή

Εάν ήδη έχει αρχίσει η υποδοχή δεδομένων από την συσκευή, τότε αυτή σταματάει, απελευθερώνεται το δίκτυο και βγαίνει κατάλληλο μήνυμα στην status bar (Device(S) Closed by user).

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Release Device.
2. Εμφανίζεται ένα κατάλληλο μήνυμα στην Status Bar (Device(S) Closed by user).
3. Το σύστημα εκτελεί την owRelease() που απελευθερώνει την-τις συσκευές του δικτύου.

3.2 Εναλλακτικές ροές

1. Αν δεν υπάρχει τέτοια συσκευή (DS1920 και/ή DS2438) συνδεδεμένη στο δίκτυο, ή εάν έχει επιλεγεί λάθος θύρα ή δεν έχει γίνει προηγουμένως άνοιγμα της θύρας, το σύστημα δεν αλλάζει κατάσταση
2. Πριν εκτελέσει την owRelease το σύστημα ελέγχει αν προηγουμένως έχει εκτελεστεί η owAquire() (γιατί αλλιώς δημιουργείται exception).

2. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

3. Κατάσταση εισόδου

1. Το σύστημα στέλνει δεδομένα σε πραγματικό χρόνο σε δύο γραφήματα και σε δύο list boxes ένα με βαθμούς Κελσίου και ένα με βαθμούς Φαρενάιτ.
2. Το σύστημα είναι σε κατάσταση αναμονής με τις συγκεκριμένες συσκευές (DS1920 και/ή DS2438) έτοιμη να αποστείλει δεδομένα..

4. Κατάσταση εξόδου

Το σύστημα σταματάει να στέλνει δεδομένα και απελευθερώνει την-τις συσκευές του δικτύου.

Περίπτωση Χρήσης 20.

1. Τίτλος περίπτωσης χρήσης

Exit: Κλείσιμο φόρμας

2. Σύντομη περιγραφή

Καθαρίζουν τα δεδομένα από τα γραφήματα και τα 2 list boxes και κλείνει η φόρμα

3. Ροή γεγονότων.

3.1 Βασική ροή

1. Η περίπτωση χρήσης ξεκινά όταν ο χρήστης πατάει το κουμπί Exit.
2. Το σύστημα καθαρίζει τα γραφήματα και τα 2 list boxes .
3. Κλείνει η φόρμα.

3.2 Εναλλακτικές ροές

Δεν υπάρχουν εναλλακτικές ροές.

4. Μη λειτουργικές απαιτήσεις

Δεν υπάρχουν ιδιαίτερες απαιτήσεις.

5. Κατάσταση εισόδου

Το σύστημα σταματάει να στέλνει δεδομένα και απελευθερώνει την-τις συσκευές του δικτύου.

6. Κατάσταση εξόδου

Το σύστημα επιστρέφει στην βασική φόρμα της εφαρμογής

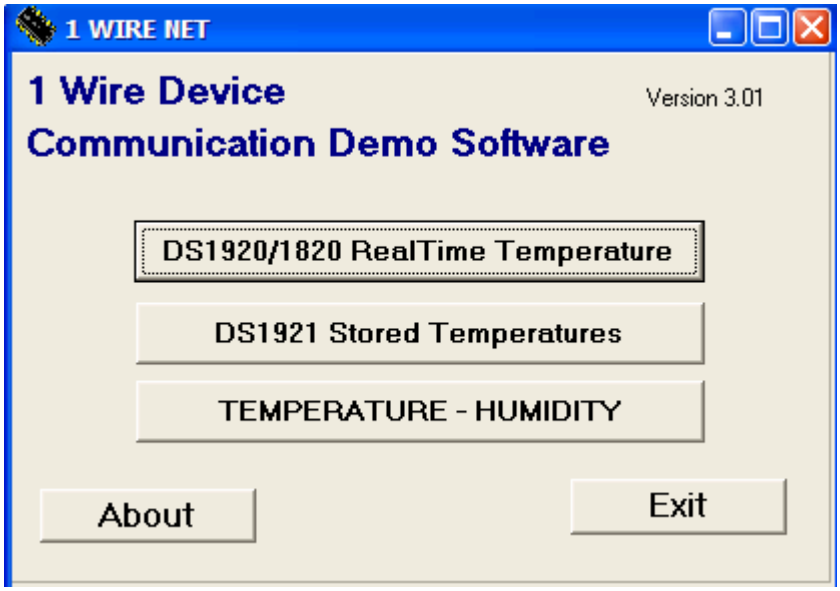
4.7. Αναλυτική Περιγραφή της Εφαρμογής

Η εφαρμογή γράφτηκε εξ' ολοκλήρου στον C++ Builder 5 της Borland. Μετά από εκτεταμένες συζητήσεις και αναλύσεις με συναδέλφους προγραμματιστές προτιμήθηκε η λύση αυτή, από την -σε πολλούς τομείς ανώτερη- πλατφόρμα .NET της Microsoft, καθαρά για λόγους συμβατότητας προς τα πίσω. Δυστυχώς ο κώδικας που παρέχεται από την

κατασκευάστρια εταιρεία Dallas Semiconductors ως συνοδευτικός των συσκευών, δεν ανταποκρίνεται στις καινούριες δομές της .NET και δεν μπορούσε να «περάσει» επιτυχώς από τους Compilers. Επειδή ο σκοπός αυτής της εργασίας είναι η δημιουργία ενός υψηλού επιπέδου API και όχι η συγγραφή κώδικα για τις συσκευές, προτιμήθηκε τελικά η παραπάνω λύση.

Εδώ πρέπει να τονιστεί ότι η εφαρμογή που δημιουργήθηκε αποσκοπεί στο να δείξει ότι είναι δυνατή η συγγραφή και παραγωγή προγραμμάτων με χρήση μόνο των βιβλιοθηκών που δημιουργήθηκαν στο πλαίσιο αυτής της διπλωματικής εργασίας. Μέσα σε αυτά τα πλαίσια η εφαρμογή «διαχειρίζεται» τρεις συσκευές 1-Wire με αρκετή επιτυχία. Παρόλο που δοκιμάστηκε αρκετά (debugging) είναι φυσικό να απέχει αρκετά από τα υψηλά στάνταρτς που θα έθετε μία εμπορική εφαρμογή και σε καμία περίπτωση δεν θα πρέπει να αντιμετωπίζεται ως τέτοια. Επίσης για να επιδειχθεί η ορθή χρήση των συναρτήσεων που έχουν δημιουργηθεί, δεν αυτοματοποιήθηκαν αρκετά οι λειτουργίες των συσκευών. Δηλαδή, θα μπορούσε στην φόρμα της DS1920/1820 να έχουμε λιγότερα κουμπιά, συγχωνεύοντας το κουμπί "Open Device" με το κουμπί "Start Get Data". Ο διαχωρισμός υφίσταται στην εφαρμογή μόνο για να καταδειχθεί και στον επίδοξο προγραμματιστή-συνεχιστή η χρήση των διαφορετικών συναρτήσεων. Κάτω από προϋποθέσεις θα μπορούσε επίσης το κουμπί "Release Device" να ενσωματωθεί στο "Stop Get Data" και/ή στο "Exit". Αντίστοιχα στην φόρμα της συσκευής DS1921, θα μπορούσε να υπάρχει μόνο ένα κουμπί που να ενσωματώνει τις λειτουργίες του "Open Device" και του κουμπιού "Start Get Data". Όταν τελειώσει η επιτυχής ανάγνωση των δεδομένων, εάν ο χρήστης ξανά-πατήσει το ίδιο κουμπί, η εφαρμογή μπορεί πολύ εύκολα να «καθαρίσει» τα προηγούμενα δεδομένα και να φέρει τα καινούρια. (από την ίδια ή άλλη συσκευή). Με την ίδια λογική θα μπορούσε το "Release Device" να ενσωματωθεί στο "Exit".

4.7.1. Σχολιασμός των επιμέρους οθονών



Εικόνα 10: Αρχική οθόνη Demo Εφαρμογής

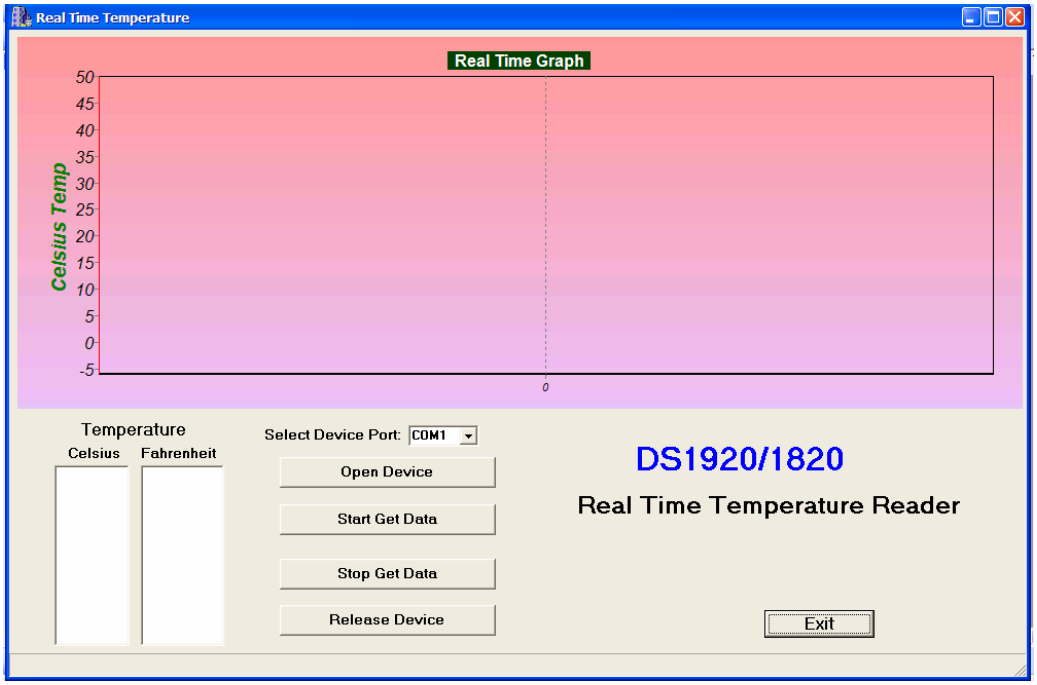
Η εφαρμογή εκκινεί από την αρχική οθόνη όπου έχουμε τρεις επιλογές:

- Παρακολούθηση θερμοκρασίας σε πραγματικό χρόνο μέσω της συσκευής 1-Wire DS1820/1920 (RealTime Temperature).
- Προβολή- επεξεργασία υποθηκευμένων δεδομένων θερμοκρασίας (sampling) μέσω της συσκευής 1-Wire DS1921 (Stored Temperarature).
- Παρακολούθηση θερμοκρασίας σε πραγματικό χρόνο μέσω της συσκευής 1-Wire DS1920 και παρακολούθηση θερμοκρασίας και υγρασίας μέσω της συσκευής DS2438.

Επιπρόσθετα υπάρχουν οι επιλογές του κουμπιού About, όπου απλώς παρέχονται στοιχεία για τον προγραμματιστή και την έκδοση (Version) της εφαρμογής και φυσικά το κουμπί Exit που απλώς κλείνει την εφαρμογή.

Εικόνα 11: : DS1920 Demo Εφαρμογή- About

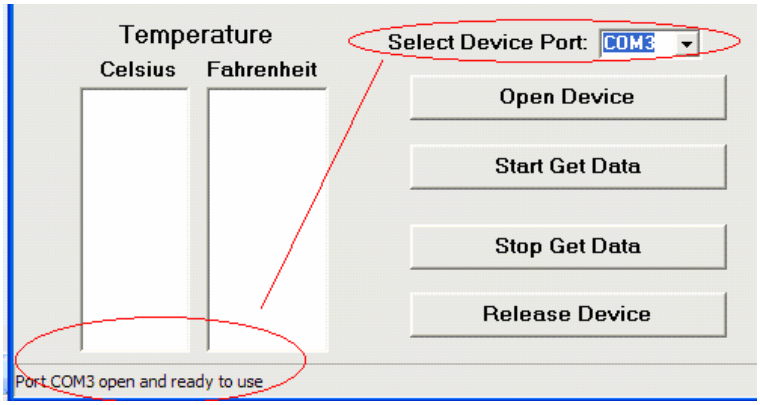
4.7.2. Θερμοκρασία πραγματικού χρόνου



Εικόνα 12: DS1920 Demo Εφαρμογή- Αρχική οθόνη

Στην αρχική οθόνη της φόρμας φαίνονται όλες οι λειτουργίες της. Προφανώς το κουμπί "Exit" κλείνει την εφαρμογή. Να σημειωθεί εδώ ότι μέσω μίας Boolean μεταβλητής όταν πατιέται το "Exit" η εφαρμογή ελέγχει αν έχει γίνει «απελευθέρωση» (Release) της

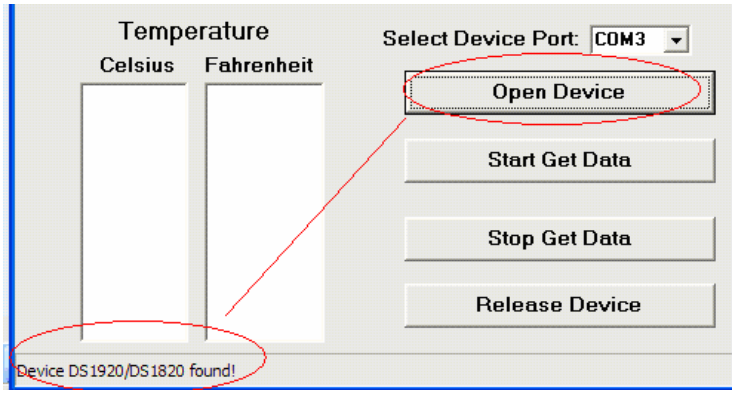
συσκευής (Closed=true) και εάν βρει την μεταβλητή false τότε εκτελεί την συνάρτηση owRelease() πριν κλείσει την φόρμα.



Εικόνα 13: DS1920 Demo Εφαρμογή- Επιλογή θύρας (COM port)

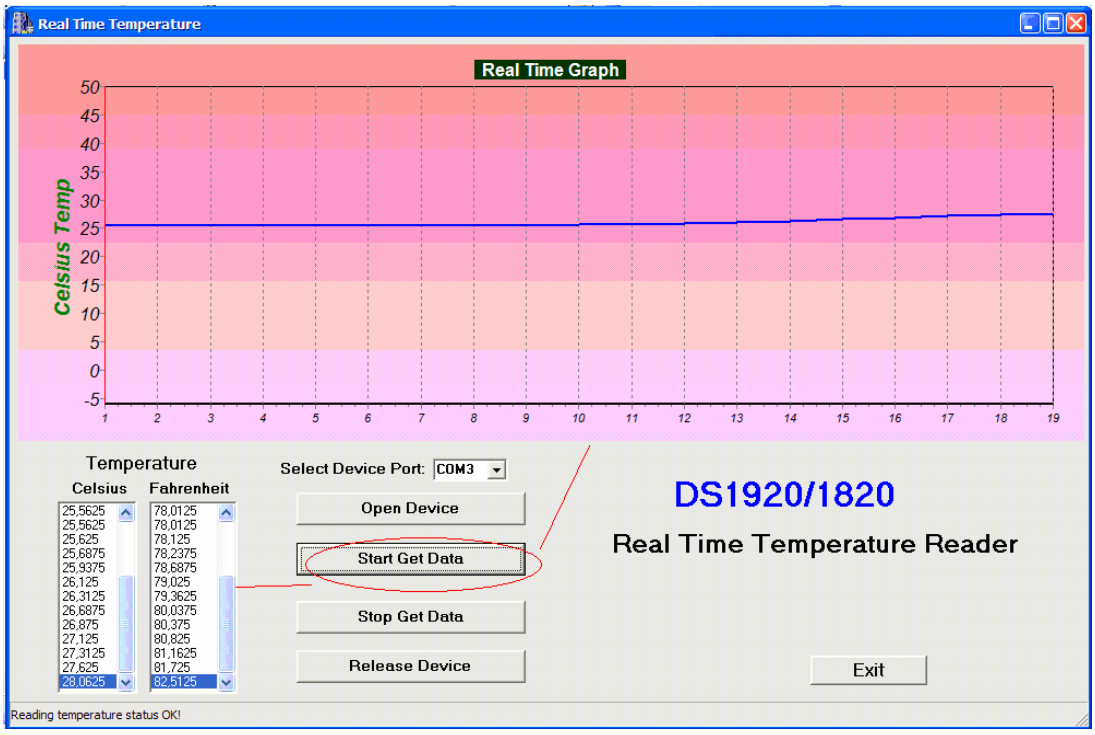
Ως πρώτη ενέργεια στην εφαρμογή, πρέπει να επιλεγεί η θύρα που είναι εγκατεστημένος ο προσαρμογέας του δικτύου 1-Wire. Εδώ πρέπει να σημειωθεί ότι η αυτόματη εύρεση του μετατροπέα (1-Wire adaptor) είναι εφικτή με κάποιες αλλαγές/προσθήκες στον κώδικα, αλλά θα επιφέρει μεγαλύτερα προβλήματα από αυτά που θα λύσει. Τι θα γίνει στην ταυτόχρονη ύπαρξη δύο μετατροπέων στον ίδιο υπολογιστή; Πώς θα ξέρουμε ποια/ποιες συσκευές ενεργοποιούμε σε ποιο δίκτυο κάθε φορά αν έχουμε δύο ή περισσότερα δίκτυα 1-Wire συνδεδεμένα στον Η/Υ; Μόνο μέσω του μοναδικού τους σειριακού αριθμού , αλλά ο ανθρώπινος νους πολύ δύσκολα θυμάται μεγάλους αριθμούς οπότε η πιθανότητα λάθους είναι πολύ μεγάλη. Έτσι προς το παρόν προτιμήθηκε η χειροκίνητη επιλογή.

Εάν έχει γίνει επιλογή θύρας στην οποία υπάρχει ενεργός μετατροπέας, τότε το σύστημα ανταποκρίνεται με το κατάλληλο μήνυμα στην status bar, στο κάτω μέρος της εφαρμογής.



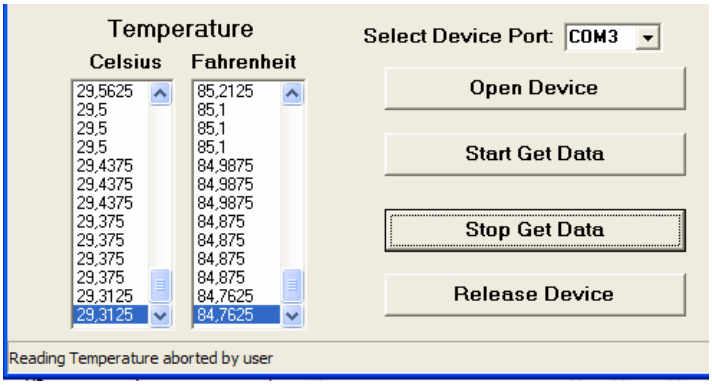
Εικόνα 14: DS1920 Demo Εφαρμογή- Επιλογή συσκευής

Μετά την επιλογή της θύρας, ο χρήστης πατάει το κουμπί "Open Device". Εάν υπάρχει πράγματι συνδεδεμένης το δίκτυο συσκευή DS1920/1820 τότε το σύστημα ανταποκρίνεται με το κατάλληλο μήνυμα στην γραμμή κατάστασης (status bar) και καλεί την συνάρτηση owAcquire() θέτοντας και την Boolean μεταβλητή closed=false.



Εικόνα 15: DS1920 Demo Εφαρμογή- Εμφάνιση δεδομένων θερμοκρασίας πραγματικού χρόνου

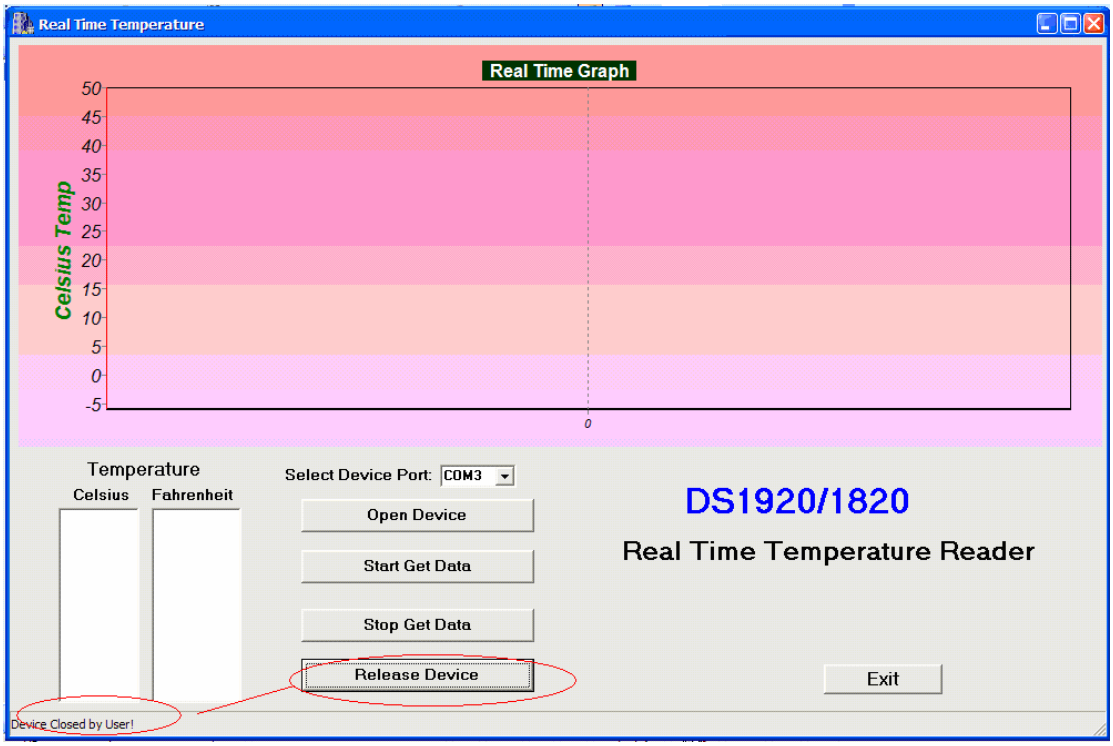
Όταν πατηθεί το κουμπί "Start Get Data" η εφαρμογή εμφανίζει την θερμοκρασία του περιβάλλοντος χώρου, στο αριστερό κάτω μέρος της οθόνης σε δύο πίνακες , εκ των οποίων ο ένας απεικονίζει βαθμούς Κελσίου και ο άλλος βαθμούς Φαρενάιτ. Επίσης η θερμοκρασία σε βαθμούς Κελσίου, εμφανίζεται και σε γράφημα το οποίο ανανεώνεται σε πραγματικό χρόνο στο πάνω μέρος της οθόνης. Η ελαφρά κλίση της πορείας του γραφήματος προς τα πάνω, οφείλεται στο «φύσημα» της συσκευής με ανθρώπινη ανάσα για άμεση μεταβολή της θερμοκρασίας της.



Εικόνα 16: DS1920 Demo Εφαρμογή- Σταμάτημα "κατεβάσματος" δεδομένων

Πατώντας το κουμπί "Stop Get Data" ο χρήστης έχει την επιλογή να διακόψει προσωρινά την ανανέωση των δεδομένων. Κατάλληλο μήνυμα εμφανίζεται στην γραμμή κατάστασης (status bar).

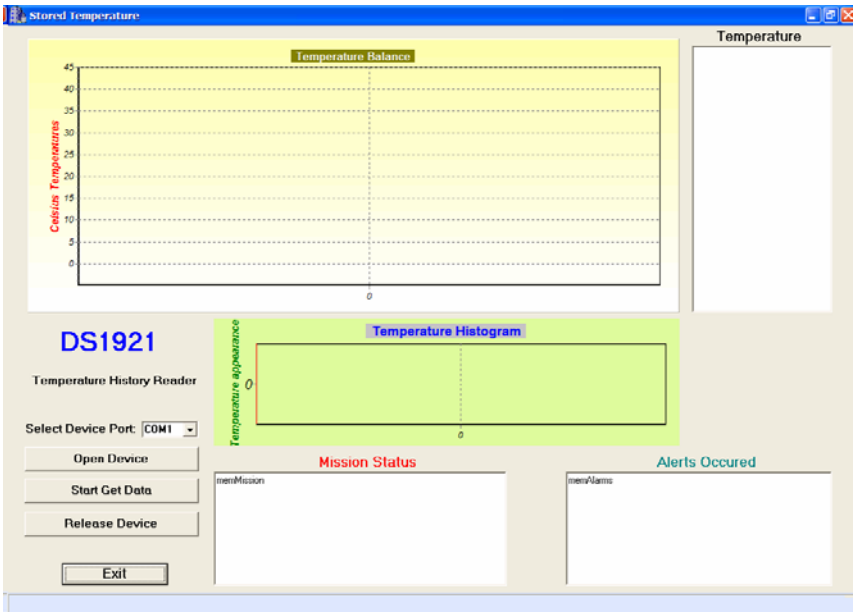
Με αυτό το κουμπί η Boolean μεταβλητή (closed=false) παραμένει στην προηγούμενη κατάσταση και ο χρήστης μπορεί έτσι να πατήσει το κουμπί "Start Get Data" και να ξανά-αρχίσει η εμφάνιση δεδομένων.



Εικόνα 17: DS1920 Demo Εφαρμογή- Απελευθέρωση Συσκευής DS1920

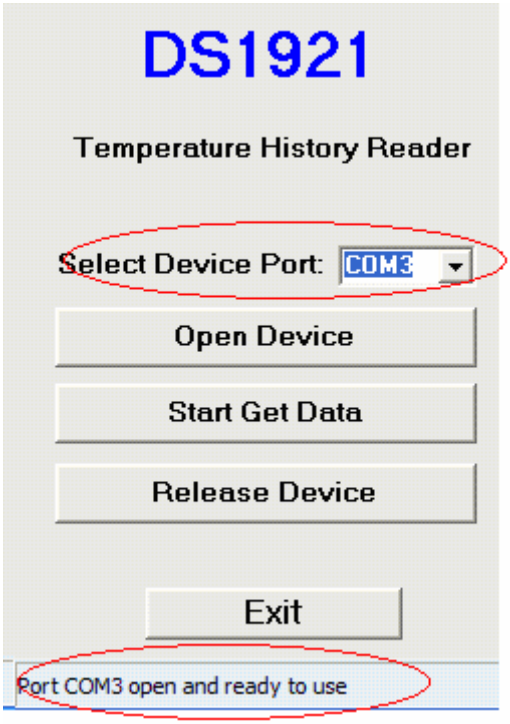
Το πάτημα του κουμπιού "Release Device" απελευθερώνει την συσκευή (καλεί την owRelease()) , «καθαρίζει» το γράφημα και τους δύο πίνακες με τις θερμοκρασίες και αλλάζει τιμή στην δίτιμη μεταβλητή (closed=true). Τώρα πια πρέπει να κληθεί η "Open Device" , εάν θέλουμε να επαναδραστηριοποιήσουμε την συσκευή.

4.7.3. Αποθηκευμένα δείγματα θερμοκρασίας



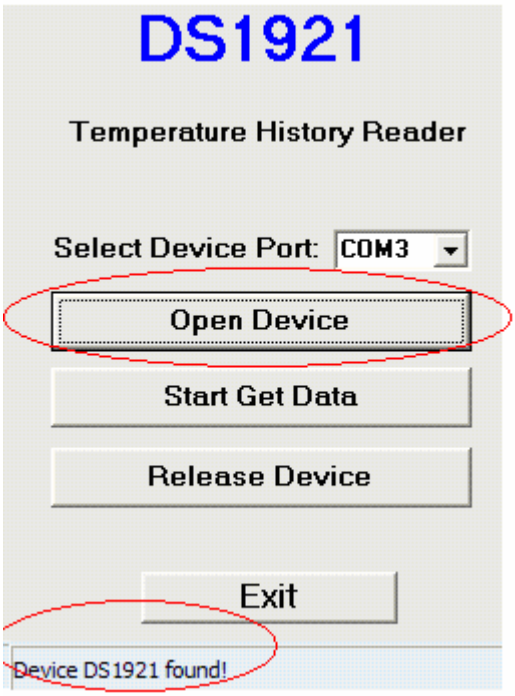
Εικόνα 18: DS1921 Demo Εφαρμογή- Αρχική φόρμα

Στην αντίστοιχη με την προηγούμενη συσκευή φόρμα/οθόνη στην οποία φαίνονται όλες οι λειτουργίες που μπορούμε να επιλέξουμε για την συσκευή, ως αρχικό βήμα πρέπει να επιλέξουμε την θύρα (COM port) στην οποία βρίσκεται συνδεδεμένος ο μετατροπέας (1-Wire adaptor). Με την σωστή επιλογή της πόρτας, αντίστοιχο μήνυμα εμφανίζεται στην γραμμή κατάστασης (status bar) της εφαρμογής. Επίσης μία αντίστοιχη με την προηγούμενη φόρμα, δίτιμη (Boolean) μεταβλητή αλλάζει τιμή (closed=false).



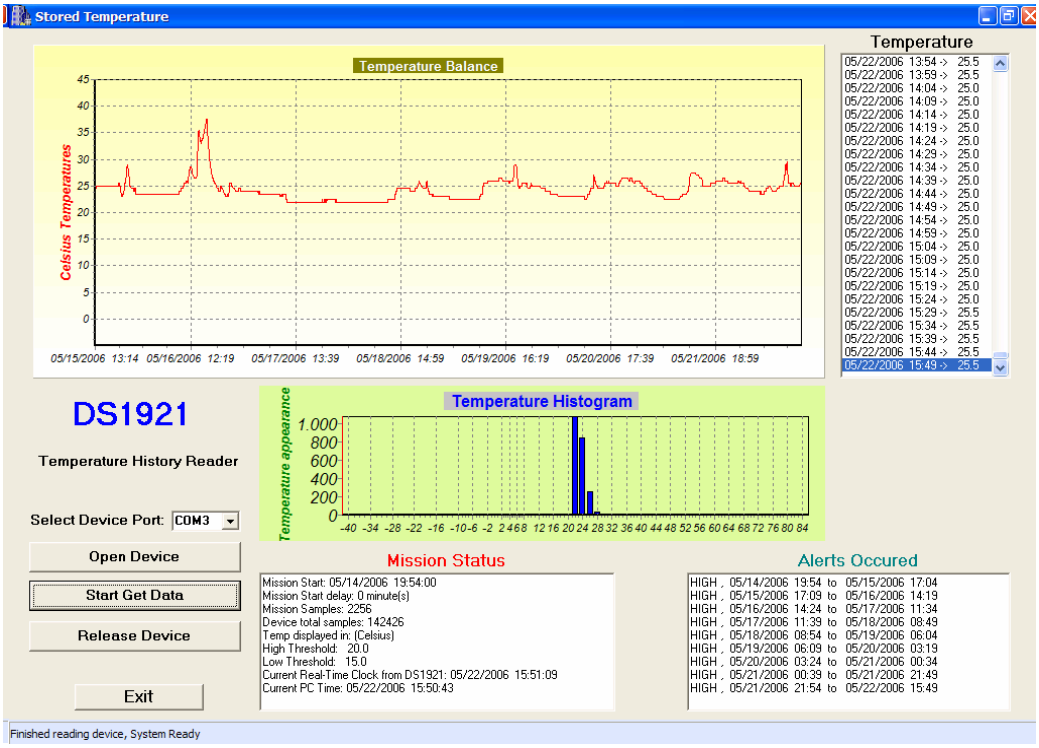
Εικόνα 19: DS1921 Demo Εφαρμογή- Επιλογή θύρας (COM port)

Επιλέγοντας το κουμπί "Open Device" το σύστημα ανταποκρίνεται καλώντας την συνάρτηση owAquire(), και εκτυπώνει κατάλληλο μήνυμα στην γραμμή κατάστασης εάν πράγματι βρεθεί μία συσκευή DS1921 στο δίκτυο.



Εικόνα 20: DS1921 Demo Εφαρμογή- Επιλογή συσκευής DS1921

Όταν ο χρήστης πατήσει το κουμπί "Start Get Data" τότε τα αποθηκευμένα στην συσκευή δεδομένα «κατεβαίνουν» στην φόρμα και εκτυπώνονται στα διαφορετικά της σημεία.



Εικόνα 21: DS1921 Demo Εφαρμογή- "Κατέβασμα" αποθηκευμένων δεδομένων συσκευής

Ο πίνακας "Mission Status" περιέχει τα εξής:

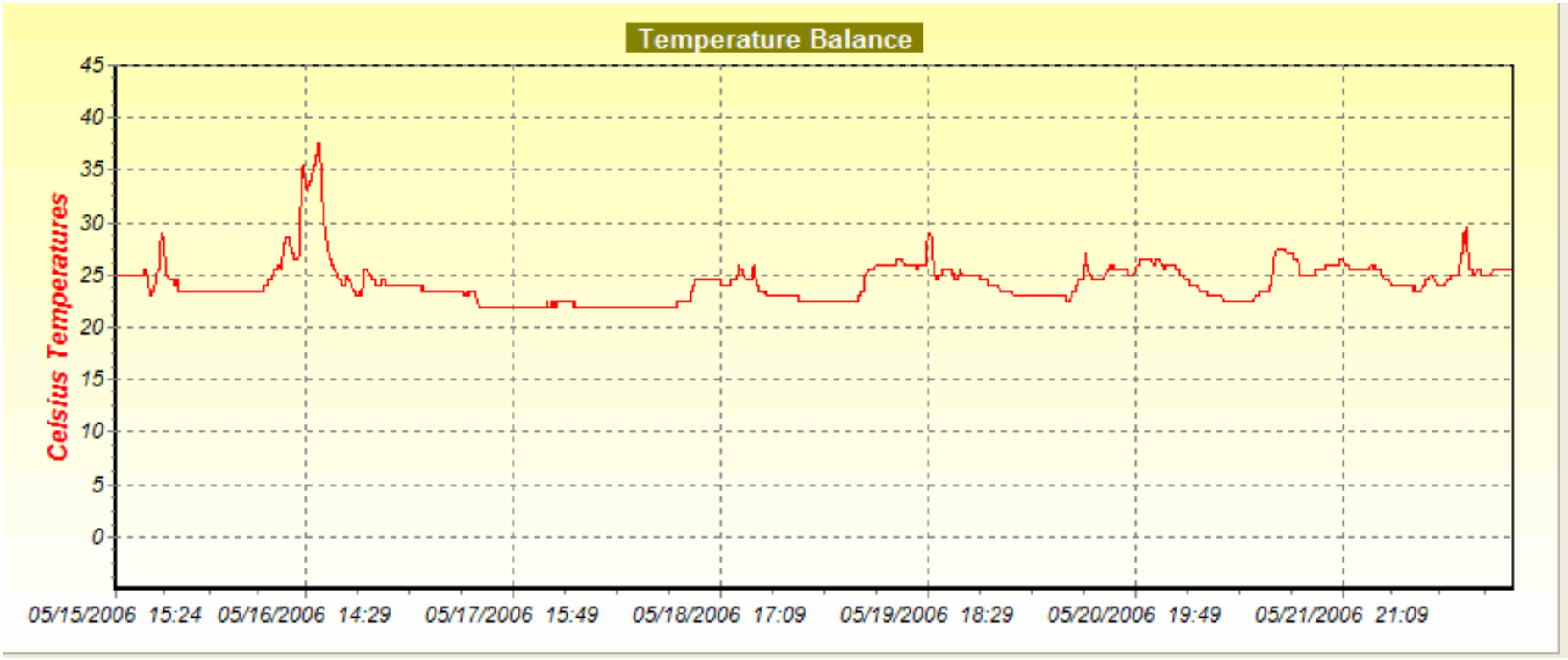
- Πότε άρχισε και πότε τελείωσε (ημερομηνία & ώρα) η καταγραφή των αποθηκεμένων δειγμάτων
- Ποιος ήταν ο ρυθμός δειγματοληψίας (sampling Rate). Μπορεί να παραμετροποιηθεί από ένα πρώτο λεπτό έως μία ώρα.
- Εάν είναι ενεργοποιημένη η επανεγγραφή δεδομένων σε περίπτωση που «γεμίσει» η μνήμη (*δεν πρέπει να ξεχνάμε ότι έχουμε να κάνουμε με μνήμη EPROM, άρα περιορισμένης χωρητικότητας*).
- Εάν συνέβη επανεγγραφή δεδομένων.
- Πόσα συνολικά δείγματα είναι αποθηκευμένα στην μνήμη της συσκευής.
- Ποιες θερμοκρασίες-όρια είχαν τεθεί ως High & Low Threshold *(θερμοκρασίες που αν ξεπεραστούν να καταγραφεί το γεγονός με ημερομηνία, διάρκεια. Έτσι ξέρουμε ότι ευαίσθητες τροφές όπως π.χ. παγωτά ή ψάρια δεν αλλοιώθηκαν λόγω υψηλών θερμοκρασιών)*
- Εάν η θερμοκρασία εμφανίζεται σε βαθμούς Κελσίου ή Φαρενάιτ

- Την ημερομηνία & ώρα της συσκευής σε σύγκρισή με την ημερομηνία & ώρα του συγκεκριμένου Η/Υ.

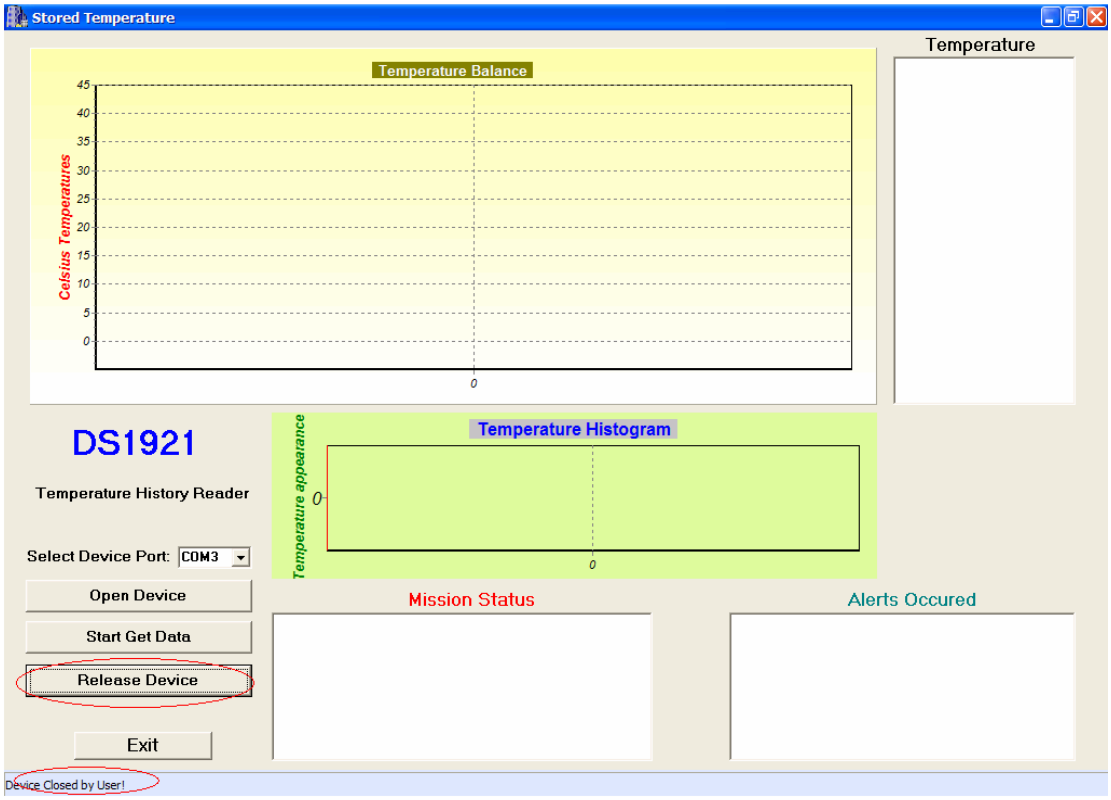
Ο πίνακας "Alarms Occurred" δείχνει αν και πότε συνέβησαν οι υπερβάσεις των ορίων θερμοκρασίας που είχα τεθεί ως όρια (High & Low Threshold).

Ο πίνακας (scroll box) πάνω δεξιά με την επικεφαλίδα "Temperature" δείχνει όλα τα τα καταγεγραμμένα δείγματα με ημερομηνία & ώρα.

Και τέλος το γράφημα "Temperature Balance" μας δείχνει την λεπτομερή διακύμανση των θερμοκρασιών που κατέγραψε η συσκευή.



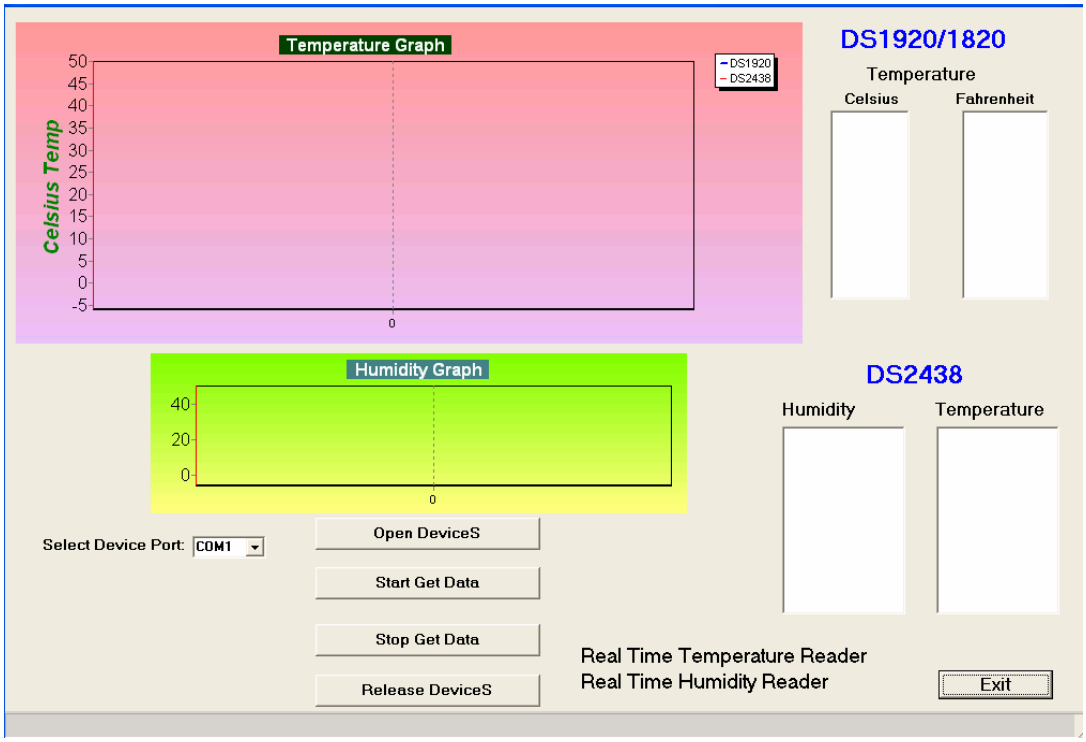
Εικόνα 22: DS1921 Demo Εφαρμογή- Γράφημα καταγεγραμμένων δειγμάτων



Εικόνα 23: Demo Εφαρμογή- «Καθάρισμα» δεδομένων και απελευθέρωση συσκευής

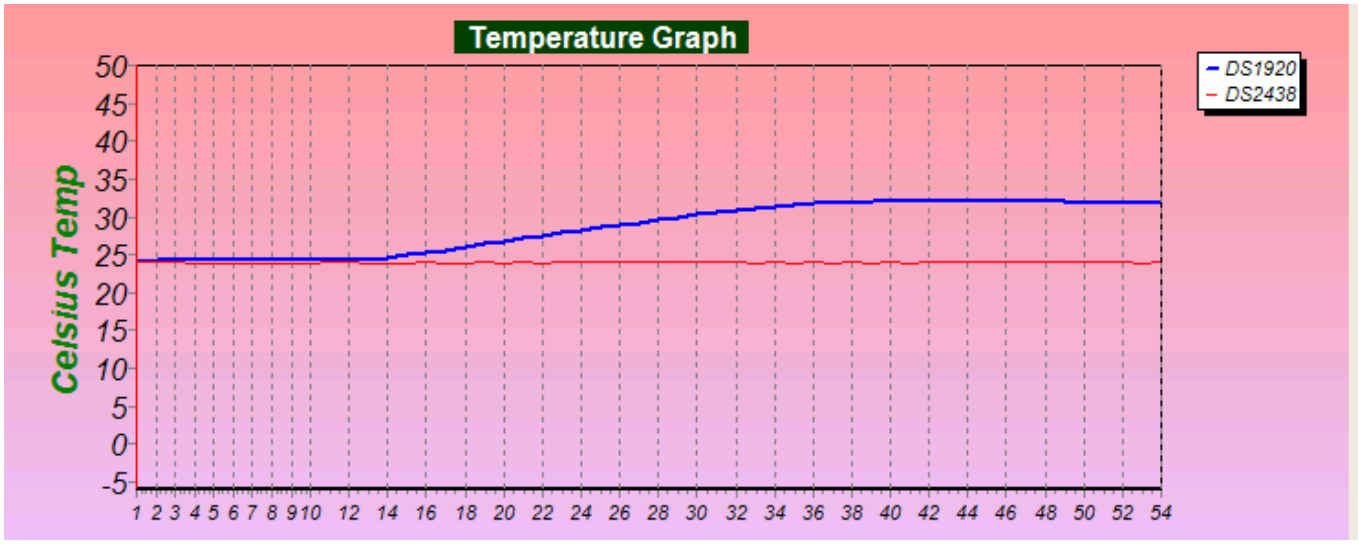
Όταν στην συσκευή πατηθεί το κουμπί "Release Device", καλείται η owAquire() που απελευθερώνει την συσκευή και «καθαρίζουν» όλα τα γραφήματα και οι πίνακες. Επίσης αλλάζει τιμή η δίτιμη μεταβλητή (Closed=true). Το σύστημα είναι έτοιμο να «αναγνώσει» και να εμφανίσει δεδομένα από τυχόν άλλη συσκευή.

4.7.4. Θερμοκρασίες και υγρασία πραγματικού χρόνου



Εικόνα 24: Καταγραφή δεδομένων δύο συσκευών

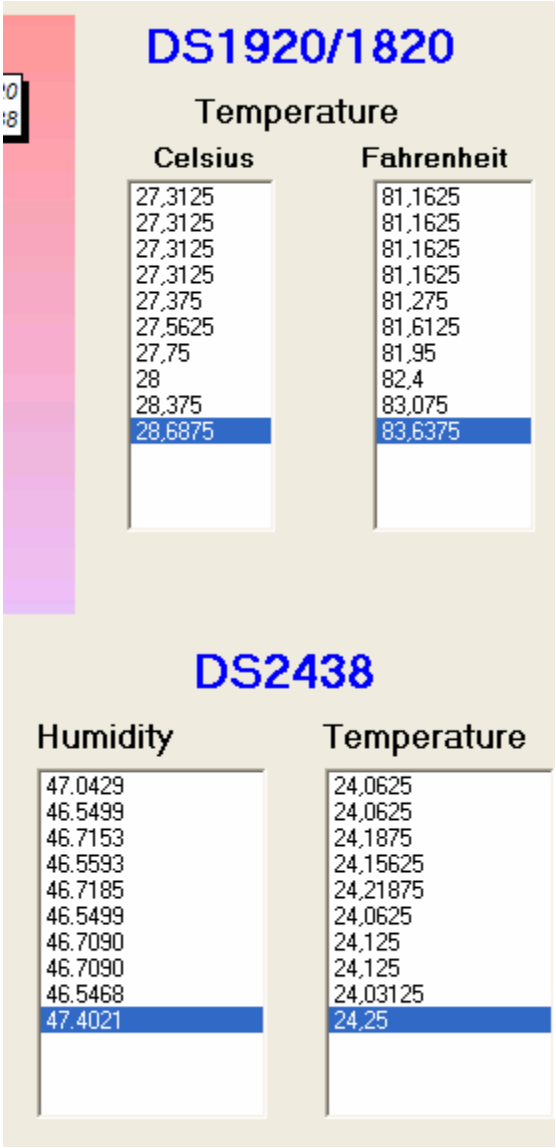
Στην τρίτη φόρμα της εφαρμογής εμφανίζονται σε ένα γράφημα οι θερμοκρασίες πραγματικού χρόνου από τις δύο διαφορετικές συσκευές DS1920 & DS2438.



Εικόνα 25: γράφημα πραγματικού χρόνου δύο δειγμάτων θερμοκρασιών

Στο γράφημα στην Εικόνα 20, φαίνονται οι διαφορετικές θερμοκρασίες που κατέγραψαν οι δύο συσκευές σε πραγματικό χρόνο. Στην Εικόνα 21, βλέπουμε στον πάνω αριστερά πίνακα την θερμοκρασία που κατέγραφε μία δεδομένη χρονική, η συσκευή DS1920 και στον πίνακα κάτω

δεξιά, η θερμοκρασία που κατέγραφε η συσκευή DS2438. Επίσης στον πίνακα κάτω αριστερά εμφανίζεται η υγρασία που έχει καταγράψει η ίδια συσκευή (DS2438).



Εικόνα 26:καταγραφή θερμοκρασιών δύο συσκευών

4.7.5. Αρχεία – βιβλιοθήκες που χρησιμοποιήθηκαν

Όπως προαναφέρθηκε χρησιμοποιήθηκαν οι βιβλιοθήκες που αναπτύχθηκαν στο πλαίσιο της διπλωματικής εργασίας και είναι οι:

HA7Net.c

ds2490.h mownetu.c musbwnet.c

m2480ut.c mowsesu.c musbwses.c

m2490ut.c	mowtran.c	musbwtrn.c
mcrcutil.c	mowtrnu.c	mwn32lnk.c
mds2480.h	mpsesw32.c	mowllu.c
mlpwin32.c	multilnk.c	mownet.c
mowerr.c	multinet.c	mownet.h
multitran.c	multises.c	musbwlnk.c

Όπως αναφέρθηκε στην αρχή του κεφαλαίου χρησιμοποιήθηκε ο κώδικας που εμπεριέχεται σε βιβλιοθήκες demo που παρέχει η εταιρεία κατασκευής των συσκευών. Οι βιβλιοθήκες αυτές είναι:

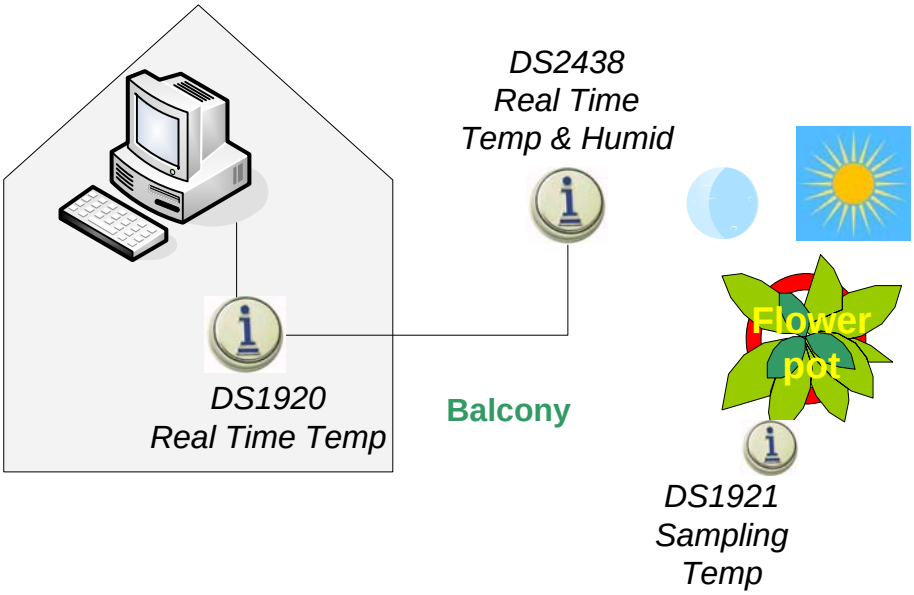
atod20.c
gethumd.c
atod20.h
Temp10.c
Thermo21.c
FindType.c
Temp10.h
Thermo21.h
FindType.h

Και τέλος για τις ανάγκες της εφαρμογής δημιουργηθήκαν οι παρακάτω βιβλιοθήκες οι οποίες εμπεριέχονται και αυτές στο παράρτημα.

Main.c
RealTime.c
Stored.c
About.c
TempHum.c
*Σημείωση: Για λόγους συμβατότητας του Compiler, όλες οι βιβλιοθήκες μετονομάστηκαν σε *.cpp .*

4.8. Δοκιμαστική Λειτουργία της Εφαρμογής

Για την δοκιμή της εφαρμογής και την άντληση στοιχείων προς επεξεργασία εκτελέστηκε το σενάριο που σκιαγραφείται στο σχήμα 23:



Σχήμα 24: Δοκιμαστική Λειτουργία Εφαρμογής

Ο αισθητήρας DS1920 (καταγραφή θερμοκρασίας πραγματικού χρόνου) παρέμεινε συνδεδεμένος στην master συσκευή πάνω στο γραφείο εργασίας του Η/Υ και δίπλα στην οθόνη μακριά από τις ηλιακές ακτίνες.

Ο αισθητήρας DS2438 (καταγραφή θερμοκρασίας και υγρασίας πραγματικού χρόνου) συνδέθηκε με καλώδιο 4 μέτρων περίπου στην master συσκευή και τοποθετήθηκε σε μπαλκόνι, δίπλα σε γλάστρες με φυτά ώστε να έχει άμεση επαφή με τις ακτίνες του ήλιου καθόλη την διάρκεια που αυτές προσπίπτουν πάνω στα φυτά.

Εικόνα 27: Συσκευή DS2438

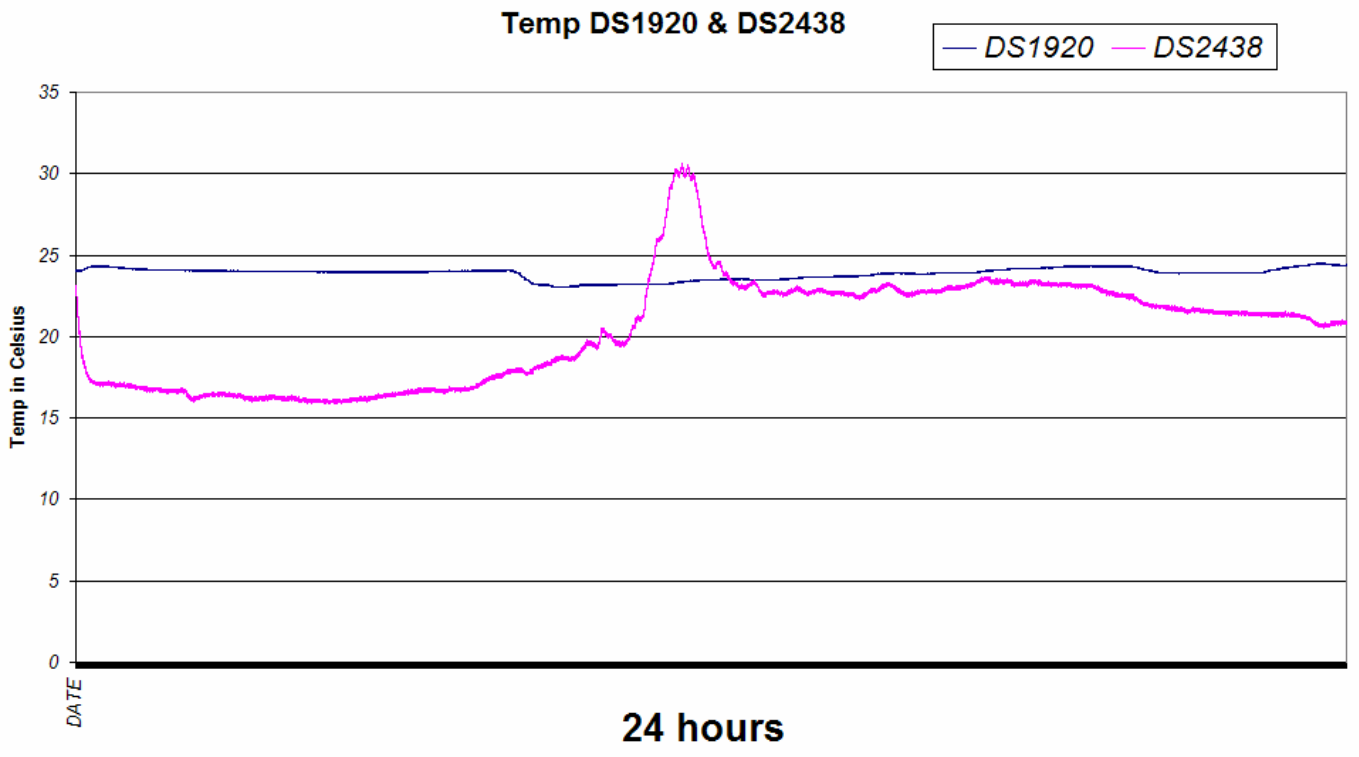
Και τέλος η συσκευή DS1920 προγραμματίστηκε να παίρνει δείγματα θερμοκρασίας κάθε 5 λεπτά της ώρας και να καταγράφει τις χρονικές στιγμές που η θερμοκρασία υπερέβη τους 20 βαθμούς Κελσίου και/ή κατέβηκε κάτω από 15 βαθμούς Κελσίου. Η συσκευή τοποθετήθηκε περίπου 5 cm κάτω από την επιφάνεια του χώματος σε μία από τις γλάστρες στο μπαλκόνι. Στις επόμενες φωτογραφίες βλέπουμε την συσκευή όταν «μπήκε» μέσα στο χώμα της γλάστρας και αντίστοιχα όταν βγήκε και πλύθηκε (μην ξεχνάμε ότι τα iButtons είναι και αδιάβροχα) για να συνδεθεί στο δίκτυο στο οποίο «ξεφόρτωσε» τα δείγματα που είχε συλλέξει. Στις επόμενες φωτογραφίες βλέπουμε τις προαναφερθείσες συσκευές «επί το έργον».

Εικόνα 28: Συσκευή DS1921

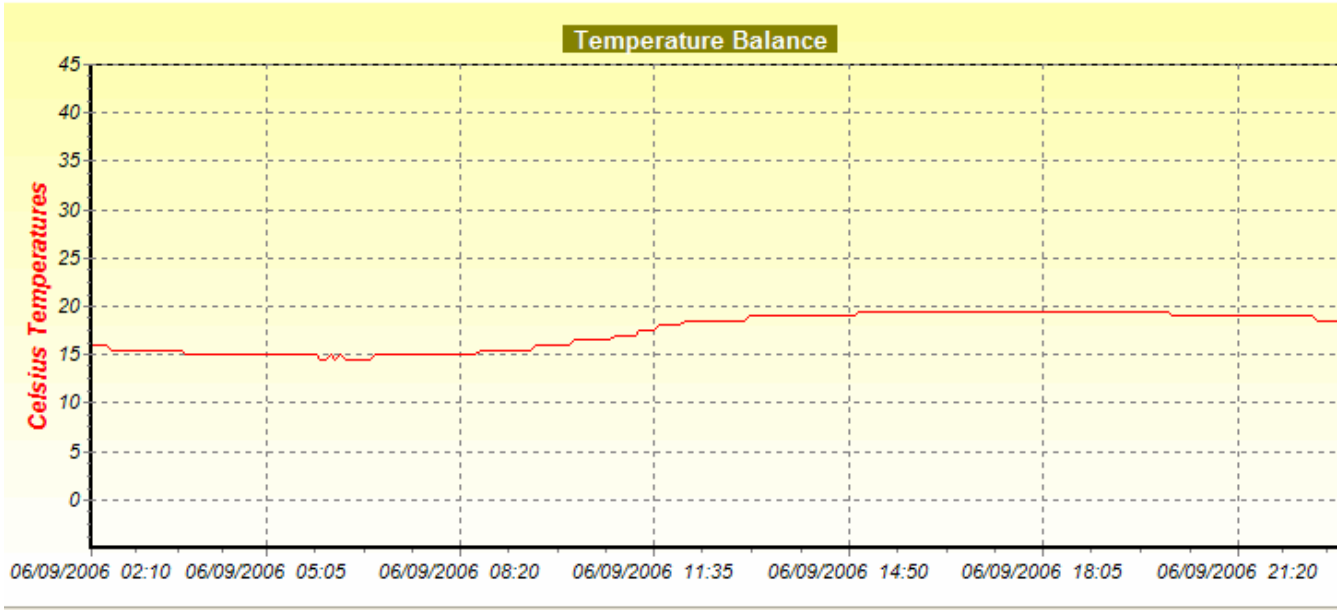
Εικόνα 29: Δίκτυο 1-Wire και συσκευή DS1920

Εικόνα 30: Δίκτυο 1-Wire και συσκευή DS2438

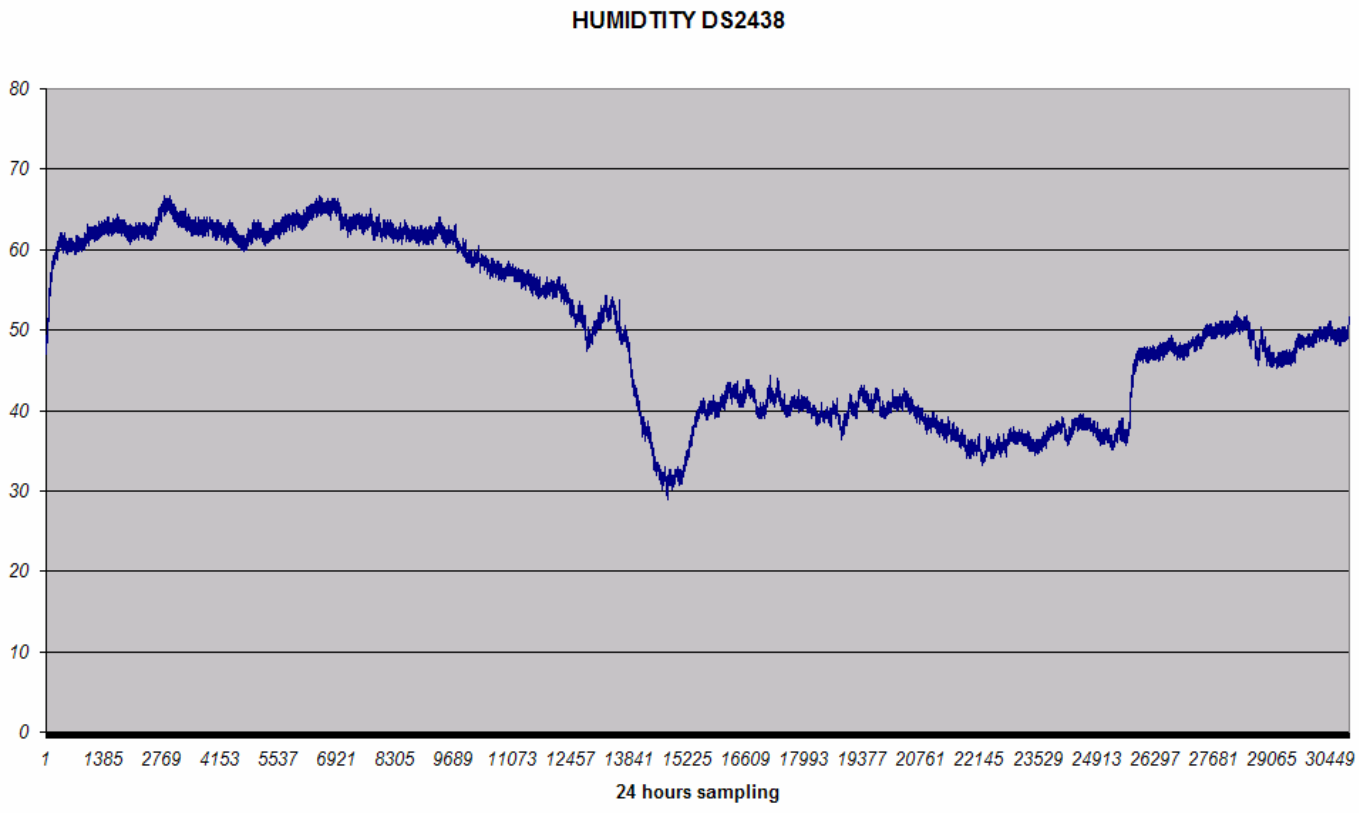
Να σημειωθεί εδώ ότι η DEMO εφαρμογή έχει την δυνατότητα να αποθηκεύει τα δείγματα θερμοκρασίας και υγρασίας και από τις δύο συσκευές (DS1920 & DS2438) σε ξεχωριστά αρχεία. Οι συσκευές αφέθηκαν επί 24ώρο να συλλέξουν δείγματα προς επεξεργασία. Μετά το πέρας του 24ώρου τα αποθηκευμένα δεδομένα των συσκευών δημιούργησαν τα επόμενα γραφήματα στο MsExcel για εξαγωγή συμπερασμάτων.



Σχήμα 25:Γράφημα 2 θερμοκρασιών DS1920 & DS2438



Σχήμα 26:Γράφημα δειγμάτων θερμοκρασίας DS1921



Σχήμα 27: Γράφημα υγρασίας DS2438

Στο Σχήμα 25 βλέπουμε καθαρά την μεγάλη διαφορά θερμοκρασίας μεταξύ εξωτερικού και εσωτερικού χώρου όπως και το πόσο λιγότερη διακύμανση έχει η θερμοκρασία στις ρίζες του φυτού (Σχήμα 26 , δείγματα από DS1921) από την θερμοκρασία στον κορμό και στα φύλλα

που είναι άμεσα εκτεθειμένα στον ήλιο. Επίσης είναι εμφανής στο σχήμα 26, η μεγάλη αύξηση της υγρασίας κατά την διάρκεια της νύχτας.

5. Ανασκόπηση και Συμπεράσματα

Ως προαπαιτούμενο αυτή η εργασία είχε την συστηματική μελέτη και σε βάθος ανάλυση του υπάρχοντος κώδικα, όπως και την αναλυτική κατάταξη του σχετικά με master συσκευές , πρωτόκολλα επικοινωνίας και γλώσσες προγραμματισμού. Αυτό σήμαινε την δημιουργία ενός λεπτομερούς χάρτη με την αναλυτική περιγραφή της υπάρχουσας κατάστασης , για την δημιουργία του οποίου αφιερώθηκε σοβαρός χρόνος. Ο χάρτης αυτός επέφερε και την συστηματοποίηση του υπάρχοντος κώδικα σε επίπεδο επικοινωνίας και χρησιμοποιήθηκε ως οδικός χάρτης για την υλοποίηση του API, αλλά και αυτής της ίδιας της DEMO εφαρμογής.

Στην πορεία προς την συγγραφή του κώδικα, το δυσκολότερο κομμάτι αρχικά ήταν να βρεθούν τα κατάλληλα αρχεία για την δημιουργία του API της εργασίας, μέσα από το site της Dallas Semiconductors. Η όποια πληροφορία υπάρχει στο site, δεν είναι οργανωμένη και πολλές φορές δεν υπάρχει κανενός είδους βοήθεια, ούτε καν ένα αρχείο readme.txt που να αναφέρει τα βασικά. Ακόμη και στις περιπτώσεις που υπάρχουν αρχεία readme.txt είναι κακογραμμένα και ελλιπή. Επίσης υπάρχουν σοβαρά λάθη στα σχόλια μέσα στον κώδικα διάφορων αρχείων (προφανώς από τα αλλεπάλληλα copy & paste) τα οποία διορθώθηκαν. Επίσης έπρεπε να καλυφθούν κάποιες ασυμβατότητες. Π.χ. υπήρχε δήλωση πίνακα ακεραίων με μηδέν στοιχεία (array[0]). Στον συγκεκριμένο compiler που χρησιμοποιήθηκε (Borland C++ Builder 5), θεωρείτο λανθασμένη μία τέτοια δήλωση. (προφανώς κάποιος άλλος compiler ίσως να το θεωρούσε σωστό). Έγινε σαφής ο τρόπος που θα δομηθεί το API, με βασικό γνώρισμα τα 4 βασικά αρχεία:

- multilnk.c
- multinet.C
- multises.c
- multitran.c

που καλούν τις συναρτήσεις από όλα τα άλλα.
Η κλήση των επιμέρους συναρτήσεων με case, κάνει τον κώδικα πολύ συμπαγή, παρουσιάζει μεγάλη ανοχή σε λάθη και παραλείψεις και το κυριότερο, ενώ ίσως να έχει παραπάνω δουλειά για τον προγραμματιστή που πρέπει να προσθέσει ένα καινούριο πρωτόκολλο ή προσαρμογέα, δεν αλλάζει απολύτως τίποτα για τον τελικό χρήστη , ή για τον προγραμματιστή που θα χρησιμοποιήσει το API για υψηλού επιπέδου εφαρμογές.

Καταβλήθηκε μεγάλη προσπάθεια να υπάρξει πλήρης συμβατότητα των καινούριων συναρτήσεων με τις υπάρχουσες. Στις περισσότερες των περιπτώσεων αυτό είναι εφικτό, και ακόμη και πολλές μεταβλητές παρέμειναν ακριβώς για αυτό τον λόγο ίδιες. Στην περίπτωση

όμως των συναρτήσεων για τα 3 άλλα πρωτόκολλα επικοινωνίας (σειριακό, παράλληλο & USB) αυτές είναι πολύ περισσότερες από αυτές που υλοποιεί ο http server. Είτε γιατί δεν υπάρχει η δυνατότητα αφού κάποιες από αυτές τις συναρτήσεις έχουν νόημα μόνο στα συγκεκριμένα πρωτόκολλα επικοινωνίας, είτε επειδή το http πρωτόκολλο επικοινωνίας εγγενώς αδυνατεί να υλοποιήσει συγκεκριμένες εντολές.

Μέχρι τώρα έχει ολοκληρωθεί ο βασικός κορμός του API. Θεωρητικά είναι έτοιμο να διαρθρώσει εφαρμογές που τυχόν θα «χτιστούν» πάνω του. Ως ένα μεγάλο βαθμό έχει υπάρξει αποσφαλμάτωση του σε επίπεδο πρωτογενών λειτουργιών, αφού καθ' όλη την διάρκεια της συγγραφής του κώδικα γινόταν συνεχής μεταγλώττιση (compilation) των βιβλιοθηκών και συναρτήσεων που κατασκευαζόντουσαν. Ακόμη και οι συναρτήσεις για τον http server της παρούσης εργασίας, δοκιμάστηκαν όλες στην πράξη. Επίσης ολοκληρώθηκαν δύο πειραματικά παραθυρικά interface που δημιουργήθηκαν για να ελεγχθούν οι διάφορες συναρτήσεις.

Το πρώτο interface υλοποιήθηκε ώστε να δοκιμαστούν οι συναρτήσεις των 3 προσαρμογέων (USB, LPT Serial). Εμπλουτίστηκε κατάλληλα με σχεδόν όλες τις λειτουργίες ώστε να μπορεί να λειτουργήσει καλώντας συναρτήσεις μέσα από το API.

Το δεύτερο interface υλοποιήθηκε αποκλειστικά για να δοκιμαστούν οι συναρτήσεις του http πρωτοκόλλου. Είναι αρκετά λειτουργικό και ο επόμενος προγραμματιστής έχει ένα απλό και ικανοποιητικό εργαλείο στα χέρια του ώστε να επικεντρωθεί αμέσως στην συγγραφή κώδικα για το πρωτόκολλο και να τον δοκιμάζει αμέσως στην πράξη.

Συμπερασματικά μπορούμε να αναφέρουμε τα εξής:
Το 1-wire είναι ένα σχετικά εύκολο και φτηνό στην υλοποίηση μέσο για να πάρουμε μετρήσεις σε μεγέθη που μπορεί να ενδιαφέρουν μία μεγάλη γκάμα επαγγελμάτων και επιστημονικών πεδίων. Φαίνεται ιδανικό για μετρήσεις μέσα σε θερμοκήπια όπου απαιτείται η συνεχής καταγραφή και παρατήρηση μεγεθών όπως η θερμοκρασία και η υγρασία. Επίσης φαίνεται πως είναι αρκετά αξιόπιστο και η σχετική απλότητα των συσκευών του δίνει την δυνατότητα ακόμη και σε έναν άπειρο προγραμματιστή (με την συμβολή και του δημιουργηθέντος για την εργασία API) να δημιουργήσει εφαρμογές στα μέτρα του πελάτη (custom) που να ανταποκρίνονται πλήρως στις ανάγκες του.

Αναφορές

1. Brelsford M. Harry: *«Τα μυστικά των Windows 2000 Server»* (Μ.Γκιούρδας, Αθήνα 2001)

2. Dallas Semiconductors/ Maxim APP 126 http://www.maxim-ic.com/appnotes.cfm/appnote_number/126, 24.11.05

3. Dallas Semiconductors/ Maxim APP 187 http://pdfserv.maxim-ic.com/en/an/app187.pdf , 28.10.05

4. Dallas Semiconductors/ Maxim APPLICATION NOTE 214 *Using a UART to Implement a 1-Wire Bus Master* http://www.maxim-ic.com/appnotes.cfm/appnote_number/214, 13.11.05

5. Dallas Semiconductors/ Maxim APPLICATION NOTE 244 *Advanced 1-Wire Network Driver* http://www.maxim-ic.com/appnotes.cfm/appnote_number/244, 12.12.05

6. Dallas Semiconductors/ Maxim Maxim/Dallas Product Line Card http://www.maxim-ic.com/sales/pdfs/line_card_4_1_03.pdf , 22.05.06

7. Dallas Semiconductors/ Maxim iButton http://www.maxim-ic.com/products/ibutton/news/images/access.pdf , 22.05.06

8. Dallas Semiconductors/ Maxim APPLICATION NOTE 3808 *What Is an iButton?* http://www.maxim-ic.com/appnotes.cfm/appnote_number/3808 , 22.05.06

9. Dallas Semiconductors/ Maxim APPLICATION NOTE 3438 *Serial Digital Data Networks* http://www.maxim-ic.com/appnotes.cfm/appnote_number/3438 , 20.12.2005

10. Dallas Semiconductors/ Maxim APPLICATION NOTE 155, *1-Wire Software Resource Guide* http://www.maxim-ic.com/appnotes.cfm/appnote_number/155, 22.10.05

11. Dallas Semiconductors/ Maxim *iButton Readers and Adapters* http://www.maxim-ic.com/1-Wire.cfm ,

12. Dallas Semiconductors/ Maxim, *1-Wire Public Domain Kit* http://www.maxim-ic.com/products/ibutton/software/1wire/wirekit.cfm, 22.10.06

13. Dallas Semiconductors/ Maxim APPLICATION NOTE 882, *RS-232 Features Explained*
14. http://www.maxim-ic.com/appnotes.cfm/appnote_number/882, 22.10.05

15. Embedded Data Systems, *Low Level 1-Wire support HA7Net Users Manual*, HA7NetUsersManual.pdf, http://embeddeddatasystems.com/page/EDS/PROD/HA/HA7Net , 29.10.05

16. Jesse Liberty: *«Μάθετε την C++ σε 24 ώρες»*, (2η έκδοση, Μ.Γκιούρδας, Αθήνα, 2000)

17. Stevens Al: *«Οδηγός της C++ με παραδείγματα»,* 6η έκδοση, (Μ.Γκιούρδας, Αθήνα, 2000)
18. Walrand Jean: *«Δίκτυα Επικοινωνιών»*, (Παπασωτηρίου, Αθήνα 1997)

19. Yan Y Song: "*An introduction to FORMAL LANGUACES AND MACHINE COMPUTATION*", (World Scientifc, Singapore 1998)

20. Ziemmerman H. : *"OSI reference model-The OSI model of architecture for open systems interconnection*" (Trans. Commun, 1980)

21. Φούσκας Γιώργος: *«Δίκτυα Υπολογιστών Ι»* (Τόμος Γ' Ελληνικό Ανοιχτό Πανεπιστήμιο, 2002, Πάτρα)

Σημείωση: Η εργασία αυτή αφορά στην ανάπτυξη – σχεδιασμό λογισμικού σχετικού με το πρωτόκολλο επικοινωνίας (1-Wire) το οποίο είναι κατασκευασμένο και ως εκ τούτου αποκλειστικό προϊόν μίας εταιρείας (Dallas Semiconductor). Έτσι, η πλειοψηφία της βιβλιογραφίας αποτελείται από white papers που αφορούν στο συγκεκριμένο πρωτόκολλο και στις εφαρμογές του, καθώς και από συγγράμματα και βιβλία που ασχολούνται με την γλώσσα προγραμματισμού στην οποία υλοποιήθηκε η εργασία (C++).

Παράρτημα Α – Συναρτήσεις API

Τροποποιημένες Συναρτήσεις

Δηλώσεις των συναρτήσεων που εμπεριέχονται σε όλες τις επιμέρους βιβλιοθήκες
Mownet.h

```
//--------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
// ownet.h - Include file for 1-Wire Net library for multi-build.
//
// Version: 3.00
//

//

#ifndef OWNET_H
#define OWNET_H

//--------------------------------------------------------------//
// Common Includes to ownet applications
//--------------------------------------------------------------//
#include <stdlib.h>
#include <stdio.h>
#include <windows.h>


//--------------------------------------------------------------//
// Typedefs
//--------------------------------------------------------------//
#ifndef SMALLINT
   //
   // purpose of smallint is for compile-time changing of formal
   // parameters and return values of functions.  For each target
   // machine, an integer is alleged to represent the most "simple"
   // number representable by that architecture.  This should, in
   // most cases, produce optimal code for that particular arch.
   // BUT...  The majority of compilers designed for embedded
   // processors actually keep an int at 16 bits, although the
   // architecture might only be comfortable with 8 bits.
   // The default size of smallint will be the same as that of
   // an integer, but this allows for easy overriding of that size.
   //
   // NOTE:
   // In all cases where a smallint is used, it is assumed that
   // decreasing the size of this integer to something as low as
   // a single byte _will_not_ change the functionality of the
   // application.  e.g. a loop counter that will iterate through
   // several kilobytes of data should not be SMALLINT.  The most
```

```
   // common place you'll see smallint is for boolean return types.
   //
   #define SMALLINT int
#endif

   // 0x02 = PARAMSET_19200
#define MAX_BAUD 0x02

#ifndef OW_UCHAR
   #define OW_UCHAR
   typedef unsigned char  uchar;
   typedef unsigned short ushort;
   typedef unsigned long  ulong;
#endif

// general defines
#define WRITE_FUNCTION 1
#define READ_FUNCTION  0

// error codes
// todo: investigate these and replace with new Error Handling library
#define READ_ERROR    -1
#define INVALID_DIR   -2
#define NO_FILE       -3
#define WRITE_ERROR   -4
#define WRONG_TYPE    -5
#define FILE_TOO_BIG  -6

// Misc
#define FALSE          0
#define TRUE           1

#ifndef MAX_PORTNUM
   #define MAX_PORTNUM    16
#endif

// mode bit flags
#define MODE_NORMAL                   0x00
#define MODE_OVERDRIVE                0x01
#define MODE_STRONG5                  0x02
#define MODE_PROGRAM                  0x04
#define MODE_BREAK                    0x08

// Output flags
#define LV_ALWAYS          2
#define LV_OPTIONAL        1
#define LV_VERBOSE         0

// TYPES
#define DS1410E            2
#define DS9097U            5
#define DS9490             6

//--------------------------------------------------------------//
// Error handling
//--------------------------------------------------------------//
extern int owGetErrorNum(void);
extern int owHasErrors(void);

//Clears the stack.
#define OWERROR_CLEAR() while(owHasErrors()) owGetErrorNum();

#ifdef DEBUG
   //Raises an exception with extra debug info
   #define OWERROR(err) owRaiseError(err,__LINE__,__FILE__)
   extern void owRaiseError(int,int,char*);
   #define OWASSERT(s,err,ret) if(!(s)){owRaiseError((err),__LINE__,__FILE__);return
(ret);}
#else
   //Raises an exception with just the error code
   #define OWERROR(err) owRaiseError(err)
   extern void owRaiseError(int);
   #define OWASSERT(s,err,ret) if(!(s)){owRaiseError((err));return (ret);}
#endif

#ifdef SMALL_MEMORY_TARGET
```

```
   #define OWERROR_DUMP(fileno) /*no-op*/;
#else
   //Prints the stack out to the given file.
   #define OWERROR_DUMP(fileno) while(owHasErrors()) owPrintErrorMsg(fileno);
   extern void owPrintErrorMsg(FILE *);
   extern void owPrintErrorMsgStd();
   extern char *owGetErrorMsg(int);
#endif

#define OWERROR_NO_ERROR_SET                    0
#define OWERROR_NO_DEVICES_ON_NET               1
#define OWERROR_RESET_FAILED                    2
#define OWERROR_SEARCH_ERROR                    3
#define OWERROR_ACCESS_FAILED                   4
#define OWERROR_DS2480_NOT_DETECTED             5
#define OWERROR_DS2480_WRONG_BAUD               6
#define OWERROR_DS2480_BAD_RESPONSE             7
#define OWERROR_OPENCOM_FAILED                  8
#define OWERROR_WRITECOM_FAILED                 9
#define OWERROR_READCOM_FAILED                  10
#define OWERROR_BLOCK_TOO_BIG                   11
#define OWERROR_BLOCK_FAILED                    12
#define OWERROR_PROGRAM_PULSE_FAILED            13
#define OWERROR_PROGRAM_BYTE_FAILED             14
#define OWERROR_WRITE_BYTE_FAILED               15
#define OWERROR_READ_BYTE_FAILED                16
#define OWERROR_WRITE_VERIFY_FAILED             17
#define OWERROR_READ_VERIFY_FAILED              18
#define OWERROR_WRITE_SCRATCHPAD_FAILED         19
#define OWERROR_COPY_SCRATCHPAD_FAILED          20
#define OWERROR_INCORRECT_CRC_LENGTH            21
#define OWERROR_CRC_FAILED                      22
#define OWERROR_GET_SYSTEM_RESOURCE_FAILED      23
#define OWERROR_SYSTEM_RESOURCE_INIT_FAILED     24
#define OWERROR_DATA_TOO_LONG                   25
#define OWERROR_READ_OUT_OF_RANGE               26
#define OWERROR_WRITE_OUT_OF_RANGE              27
#define OWERROR_DEVICE_SELECT_FAIL              28
#define OWERROR_READ_SCRATCHPAD_VERIFY          29
#define OWERROR_COPY_SCRATCHPAD_NOT_FOUND       30
#define OWERROR_ERASE_SCRATCHPAD_NOT_FOUND      31
#define OWERROR_ADDRESS_READ_BACK_FAILED        32
#define OWERROR_EXTRA_INFO_NOT_SUPPORTED        33
#define OWERROR_PG_PACKET_WITHOUT_EXTRA         34
#define OWERROR_PACKET_LENGTH_EXCEEDS_PAGE      35
#define OWERROR_INVALID_PACKET_LENGTH           36
#define OWERROR_NO_PROGRAM_PULSE                37
#define OWERROR_READ_ONLY                       38
#define OWERROR_NOT_GENERAL_PURPOSE             39
#define OWERROR_READ_BACK_INCORRECT             40
#define OWERROR_INVALID_PAGE_NUMBER             41
#define OWERROR_CRC_NOT_SUPPORTED               42
#define OWERROR_CRC_EXTRA_INFO_NOT_SUPPORTED    43
#define OWERROR_READ_BACK_NOT_VALID             44
#define OWERROR_COULD_NOT_LOCK_REDIRECT         45
#define OWERROR_READ_STATUS_NOT_COMPLETE        46
#define OWERROR_PAGE_REDIRECTION_NOT_SUPPORTED  47
#define OWERROR_LOCK_REDIRECTION_NOT_SUPPORTED  48
#define OWERROR_READBACK_EPROM_FAILED           49
#define OWERROR_PAGE_LOCKED                     50
#define OWERROR_LOCKING_REDIRECTED_PAGE_AGAIN   51
#define OWERROR_REDIRECTED_PAGE                 52
#define OWERROR_PAGE_ALREADY_LOCKED             53
#define OWERROR_WRITE_PROTECTED                 54
#define OWERROR_NONMATCHING_MAC                 55
#define OWERROR_WRITE_PROTECT                   56
#define OWERROR_WRITE_PROTECT_SECRET            57
#define OWERROR_COMPUTE_NEXT_SECRET             58
#define OWERROR_LOAD_FIRST_SECRET               59
#define OWERROR_POWER_NOT_AVAILABLE             60
#define OWERROR_XBAD_FILENAME                   61
#define OWERROR_XUNABLE_TO_CREATE_DIR           62
#define OWERROR_REPEAT_FILE                     63
#define OWERROR_DIRECTORY_NOT_EMPTY             64
#define OWERROR_WRONG_TYPE                      65
#define OWERROR_BUFFER_TOO_SMALL                66
```

```
#define OWERROR_NOT_WRITE_ONCE                 67
#define OWERROR_FILE_NOT_FOUND                 68
#define OWERROR_OUT_OF_SPACE                   69
#define OWERROR_TOO_LARGE_BITNUM               70
#define OWERROR_NO_PROGRAM_JOB                 71
#define OWERROR_FUNC_NOT_SUP                   72
#define OWERROR_HANDLE_NOT_USED                73
#define OWERROR_FILE_WRITE_ONLY                74
#define OWERROR_HANDLE_NOT_AVAIL               75
#define OWERROR_INVALID_DIRECTORY              76
#define OWERROR_HANDLE_NOT_EXIST               77
#define OWERROR_NONMATCHING_SNUM               78
#define OWERROR_NON_PROGRAM_PARTS              79
#define OWERROR_PROGRAM_WRITE_PROTECT          80
#define OWERROR_FILE_READ_ERR                  81
#define OWERROR_ADDFILE_TERMINATED             82
#define OWERROR_READ_MEMORY_PAGE_FAILED        83
#define OWERROR_MATCH_SCRATCHPAD_FAILED        84
#define OWERROR_ERASE_SCRATCHPAD_FAILED        85
#define OWERROR_READ_SCRATCHPAD_FAILED         86
#define OWERROR_SHA_FUNCTION_FAILED            87
#define OWERROR_NO_COMPLETION_BYTE             88
#define OWERROR_WRITE_DATA_PAGE_FAILED         89
#define OWERROR_COPY_SECRET_FAILED             90
#define OWERROR_BIND_SECRET_FAILED             91
#define OWERROR_INSTALL_SECRET_FAILED          92
#define OWERROR_VERIFY_SIG_FAILED              93
#define OWERROR_SIGN_SERVICE_DATA_FAILED       94
#define OWERROR_VERIFY_AUTH_RESPONSE_FAILED    95
#define OWERROR_ANSWER_CHALLENGE_FAILED        96
#define OWERROR_CREATE_CHALLENGE_FAILED        97
#define OWERROR_BAD_SERVICE_DATA               98
#define OWERROR_SERVICE_DATA_NOT_UPDATED       99
#define OWERROR_CATASTROPHIC_SERVICE_FAILURE   100
#define OWERROR_LOAD_FIRST_SECRET_FAILED       101
#define OWERROR_MATCH_SERVICE_SIGNATURE_FAILED 102
#define OWERROR_KEY_OUT_OF_RANGE               103
#define OWERROR_BLOCK_ID_OUT_OF_RANGE          104
#define OWERROR_PASSWORDS_ENABLED              105
#define OWERROR_PASSWORD_INVALID               106
#define OWERROR_NO_READ_ONLY_PASSWORD          107
#define OWERROR_NO_READ_WRITE_PASSWORD         108
#define OWERROR_OW_SHORTED                     109
#define OWERROR_ADAPTER_ERROR                  110
#define OWERROR_EOP_COPY_SCRATCHPAD_FAILED     111
#define OWERROR_EOP_WRITE_SCRATCHPAD_FAILED    112
#define OWERROR_HYGRO_STOP_MISSION_UNNECESSARY 113
#define OWERROR_HYGRO_STOP_MISSION_ERROR       114
#define OWERROR_PORTNUM_ERROR                  115


// external One Wire global from owllu.c
extern SMALLINT FAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE;

//\\//\\ STANDARD //\\//\\
// One Wire functions defined in ownetu.c
SMALLINT  owFirst(int portnum, SMALLINT do_reset, SMALLINT alarm_only);
SMALLINT  owNext(int portnum, SMALLINT do_reset, SMALLINT alarm_only);
void      owSerialNum(int portnum, uchar *serialnum_buf, SMALLINT do_read);
void      owFamilySearchSetup(int portnum, SMALLINT search_family);
void      owSkipFamily(int portnum);
SMALLINT  owAccess(int portnum);
SMALLINT  owVerify(int portnum, SMALLINT alarm_only);
SMALLINT  owOverdriveAccess(int portnum);

// external One Wire functions defined in owsesu.c
int      owAcquireEx(char *port_zstr);
SMALLINT owAcquire(int portnum, char *port_zstr);
void     owRelease(int portnum);

// external One Wire functions from link layer owllu.c
SMALLINT owTouchReset(int portnum);
SMALLINT owTouchBit(int portnum, SMALLINT sendbit);
SMALLINT owTouchByte(int portnum, SMALLINT sendbyte);
SMALLINT owWriteByte(int portnum, SMALLINT sendbyte);
SMALLINT owReadByte(int portnum);
```

```
SMALLINT owSpeed(int portnum, SMALLINT new_speed);
SMALLINT owLevel(int portnum, SMALLINT new_level);
SMALLINT owProgramPulse(int portnum);
SMALLINT owWriteBytePower(int portnum, SMALLINT sendbyte);
SMALLINT owReadBytePower(int portnum);
SMALLINT owHasPowerDelivery(int portnum);
SMALLINT owHasProgramPulse(int portnum);
SMALLINT owHasOverDrive(int portnum);
SMALLINT owReadBitPower(int portnum, SMALLINT applyPowerResponse);

// external One Wire functions from transaction layer in owtrnu.c
SMALLINT owBlock(int portnum, SMALLINT do_reset, uchar *tran_buf, SMALLINT tran_len);
SMALLINT owProgramByte(int portnum, SMALLINT write_byte, int addr, SMALLINT write_cmd,
                       SMALLINT crc_type, SMALLINT do_access);
// link functions
void      msDelay(int len);
long      msGettick(void);

//\\//\\ DS9097U //\\//\\
// One Wire functions defined in ownetu.c
SMALLINT  owFirst_DS9097U(int portnum, SMALLINT do_reset, SMALLINT alarm_only);
SMALLINT  owNext_DS9097U(int portnum, SMALLINT do_reset, SMALLINT alarm_only);
void      owSerialNum_DS9097U(int portnum, uchar *serialnum_buf, SMALLINT do_read);
void      owFamilySearchSetup_DS9097U(int portnum, SMALLINT search_family);
void      owSkipFamily_DS9097U(int portnum);
SMALLINT  owAccess_DS9097U(int portnum);
SMALLINT  owVerify_DS9097U(int portnum, SMALLINT alarm_only);
SMALLINT  owOverdriveAccess_DS9097U(int portnum);

// external One Wire functions defined in owsesu.c
int      owAcquireEx_DS9097U(char *port_zstr);
SMALLINT owAcquire_DS9097U(int portnum, char *port_zstr);
void     owRelease_DS9097U(int portnum);

// external One Wire functions from link layer owllu.c
SMALLINT owTouchReset_DS9097U(int portnum);
SMALLINT owTouchBit_DS9097U(int portnum, SMALLINT sendbit);
SMALLINT owTouchByte_DS9097U(int portnum, SMALLINT sendbyte);
SMALLINT owWriteByte_DS9097U(int portnum, SMALLINT sendbyte);
SMALLINT owReadByte_DS9097U(int portnum);
SMALLINT owSpeed_DS9097U(int portnum, SMALLINT new_speed);
SMALLINT owLevel_DS9097U(int portnum, SMALLINT new_level);
SMALLINT owProgramPulse_DS9097U(int portnum);
SMALLINT owWriteBytePower_DS9097U(int portnum, SMALLINT sendbyte);
SMALLINT owReadBytePower_DS9097U(int portnum);
SMALLINT owHasPowerDelivery_DS9097U(int portnum);
SMALLINT owHasProgramPulse_DS9097U(int portnum);
SMALLINT owHasOverDrive_DS9097U(int portnum);
SMALLINT owReadBitPower_DS9097U(int portnum, SMALLINT applyPowerResponse);

// external One Wire functions from transaction layer in owtrnu.c
SMALLINT owBlock_DS9097U(int portnum, SMALLINT do_reset, uchar *tran_buf, SMALLINT
tran_len);
SMALLINT owProgramByte_DS9097U(int portnum, SMALLINT write_byte, int addr, SMALLINT
write_cmd,
                       SMALLINT crc_type, SMALLINT do_access);
// link functions
void      msDelay_DS9097U(int len);
long      msGettick_DS9097U(void);

//\\//\\ DS9490 //\\//\\
// One Wire functions defined in ownetu.c
SMALLINT  owFirst_DS9490(int portnum, SMALLINT do_reset, SMALLINT alarm_only);
SMALLINT  owNext_DS9490(int portnum, SMALLINT do_reset, SMALLINT alarm_only);
void      owSerialNum_DS9490(int portnum, uchar *serialnum_buf, SMALLINT do_read);
void      owFamilySearchSetup_DS9490(int portnum, SMALLINT search_family);
void      owSkipFamily_DS9490(int portnum);
SMALLINT  owAccess_DS9490(int portnum);
SMALLINT  owVerify_DS9490(int portnum, SMALLINT alarm_only);
SMALLINT  owOverdriveAccess_DS9490(int portnum);

// external One Wire functions defined in owsesu.c
int      owAcquireEx_DS9490(char *port_zstr);
SMALLINT owAcquire_DS9490(int portnum, char *port_zstr);
void     owRelease_DS9490(int portnum);
```

```
// external One Wire functions from link layer owllu.c
SMALLINT owTouchReset_DS9490(int portnum);
SMALLINT owTouchBit_DS9490(int portnum, SMALLINT sendbit);
SMALLINT owTouchByte_DS9490(int portnum, SMALLINT sendbyte);
SMALLINT owWriteByte_DS9490(int portnum, SMALLINT sendbyte);
SMALLINT owReadByte_DS9490(int portnum);
SMALLINT owSpeed_DS9490(int portnum, SMALLINT new_speed);
SMALLINT owLevel_DS9490(int portnum, SMALLINT new_level);
SMALLINT owProgramPulse_DS9490(int portnum);
SMALLINT owWriteBytePower_DS9490(int portnum, SMALLINT sendbyte);
SMALLINT owReadBytePower_DS9490(int portnum);
SMALLINT owHasPowerDelivery_DS9490(int portnum);
SMALLINT owHasProgramPulse_DS9490(int portnum);
SMALLINT owHasOverDrive_DS9490(int portnum);
SMALLINT owReadBitPower_DS9490(int portnum, SMALLINT applyPowerResponse);

// external One Wire functions from transaction layer in owtrnu.c
SMALLINT owBlock_DS9490(int portnum, SMALLINT do_reset, uchar *tran_buf, SMALLINT
tran_len);
SMALLINT owProgramByte_DS9490(int portnum, SMALLINT write_byte, int addr, SMALLINT
write_cmd,
                      SMALLINT crc_type, SMALLINT do_access);

// link functions
void      msDelay_DS9490(int len);
long      msGettick_DS9490(void);

//\\//\\ DS1410E //\\//\\
// One Wire functions defined in ownetu.c
SMALLINT  owFirst_DS1410E(int portnum, SMALLINT do_reset, SMALLINT alarm_only);
SMALLINT  owNext_DS1410E(int portnum, SMALLINT do_reset, SMALLINT alarm_only);
void      owSerialNum_DS1410E(int portnum, uchar *serialnum_buf, SMALLINT do_read);
void      owFamilySearchSetup_DS1410E(int portnum, SMALLINT search_family);
void      owSkipFamily_DS1410E(int portnum);
SMALLINT  owAccess_DS1410E(int portnum);
SMALLINT  owVerify_DS1410E(int portnum, SMALLINT alarm_only);
SMALLINT  owOverdriveAccess_DS1410E(int portnum);

// external One Wire functions defined in owsesu.c
int      owAcquireEx_DS1410E(char *port_zstr);
SMALLINT owAcquire_DS1410E(int portnum, char *port_zstr);
void     owRelease_DS1410E(int portnum);

// external One Wire functions from link layer owllu.c
SMALLINT owTouchReset_DS1410E(int portnum);
SMALLINT owTouchBit_DS1410E(int portnum, SMALLINT sendbit);
SMALLINT owTouchByte_DS1410E(int portnum, SMALLINT sendbyte);
SMALLINT owWriteByte_DS1410E(int portnum, SMALLINT sendbyte);
SMALLINT owReadByte_DS1410E(int portnum);
SMALLINT owSpeed_DS1410E(int portnum, SMALLINT new_speed);
SMALLINT owLevel_DS1410E(int portnum, SMALLINT new_level);
SMALLINT owProgramPulse_DS1410E(int portnum);
SMALLINT owWriteBytePower_DS1410E(int portnum, SMALLINT sendbyte);
SMALLINT owReadBytePower_DS1410E(int portnum);
SMALLINT owHasPowerDelivery_DS1410E(int portnum);
SMALLINT owHasProgramPulse_DS1410E(int portnum);
SMALLINT owHasOverDrive_DS1410E(int portnum);
SMALLINT owReadBitPower_DS1410E(int portnum, SMALLINT applyPowerResponse);

// external One Wire functions from transaction layer in owtrnu.c
SMALLINT owBlock_DS1410E(int portnum, SMALLINT do_reset, uchar *tran_buf, SMALLINT
tran_len);
SMALLINT owProgramByte_DS1410E(int portnum, SMALLINT write_byte, int addr, SMALLINT
write_cmd,
                      SMALLINT crc_type, SMALLINT do_access);
// link functions
void      msDelay_DS1410E(int len);
long      msGettick_DS1410E(void);

// external functions defined in crcutil.c
void setcrc16(int portnum, ushort reset);
ushort docrc16(int portnum, ushort cdata);
void setcrc8(int portnum, uchar reset);
uchar docrc8(int portnum, uchar x);

#endif //OWNET_H
```

Συναρτήσεις επιπέδου Συνόδου

Βιβλιοθήκη που καλεί τις επιμέρους συναρτήσεις των προσαρμογέων για το επίπεδο συνόδου

Multises.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  multises.c - Wrapper to hook all adapter types in the 1-Wire Public
//               Domain API for session functions.
//
//  Version: 3.01
//  added support for the http server HA7Net.com
//  line added in function owAcquire: "case HA7Net:........"
//  line added in function owRelease: "case HA7Net:........"
//

#include "mownet.h"

SMALLINT default_type = 0;

//--------------------------------------------------------------------------
// Attempt to acquire the specified 1-Wire net.
//
// 'port_zstr'     - zero terminated port name.  For this platform
//                   use format {port number, port type}.  The port types
//                   are: 2 LPT/DS1410E, 5 COM/DS2480B, 6 USB/DS9490
//
// Returns: port number or -1 if not successful in setting up the port.
//
SMALLINT owAcquireEx(char *port_zstr)
{
   SMALLINT rt = -1;
   char tmp_port_zstr[16];
   int type, num;

   if (port_zstr)
   {
      if (sscanf(port_zstr,"\\\\.\\DS2490-%d",&num) == 1)
         rt = owAcquire_DS9490(num, port_zstr);
      else if ( (sscanf(port_zstr,"COM%d",&num) == 1) ||
                (sscanf(port_zstr,"com%d",&num) == 1) )
         rt = owAcquire_DS9097U(num, port_zstr);
      else if ( (sscanf(port_zstr,"%d",&num) == 1) &&
                (num>=1 && num <=3) )
```

```
         rt = owAcquire_DS1410E(num, port_zstr);
      else if (sscanf(port_zstr,"{%d,%d}",&num,&type) == 2)
      {
         switch (type)
         {
            case DS9490:
               sprintf(tmp_port_zstr,"\\\\.\\DS2490-%d",num);
               rt = owAcquireEx_DS9490(tmp_port_zstr);
               break;
            case DS1410E:
               sprintf(tmp_port_zstr,"%d",num);
               rt = owAcquireEx_DS1410E(tmp_port_zstr);
               break;
            case DS9097U:
               sprintf(tmp_port_zstr,"COM%d",num);
               rt = owAcquireEx_DS9097U(tmp_port_zstr);
               break;
            default:
               rt = -1;
               break;
         }
      }
   }

   if (rt >=0)
      return ((type << 8) | rt);
   else
      return -1;
}

//--------------------------------------------------------------------------
// Attempt to acquire a 1-Wire net
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'port_zstr'  - zero terminated port name.  Format indicates the adapter
//                type.
//
// Returns: TRUE - success, port opened
//
SMALLINT owAcquire(int portnum, char *port_zstr)
{
   int type, num;

   // legacy call, attempt to discern adapter type from string format
   // if adapter type is not embedded in portnum
   if ((portnum & 0xFF00) == 0 && port_zstr)
   {
      if (sscanf(port_zstr,"\\\\.\\DS2490-%d",&num) == 1)
         default_type = DS9490;
      else if ( (sscanf(port_zstr,"COM%d",&num) == 1) ||
                (sscanf(port_zstr,"com%d",&num) == 1) )
         default_type = DS9097U;
      else if ( (sscanf(port_zstr,"%d",&num) == 1) &&
                (num>=1 && num <=3) )
         default_type = DS1410E;
      else if (sscanf(port_zstr,"{%d,%d}",&num,&type) == 2)
      {
         switch (type)
         {
            case DS9490:
               default_type = DS9490;
               break;
            case DS1410E:
               default_type = DS1410E;
               break;
            default:
            case DS9097U:
               default_type = DS9097U;
               break;
         }
      }
   }

   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
```

```
      case DS9490:
         return owAcquire_DS9490(portnum & 0xFF, port_zstr);
      case DS1410E:
         return owAcquire_DS1410E(portnum & 0xFF, port_zstr);
      default:
      case DS9097U:
         return owAcquire_DS9097U(portnum & 0xFF, port_zstr);
      case HA7Net:
         return owAcquire_HA7Net(portnum & 0xFF, port_zstr);
   }
}

//--------------------------------------------------------------------------
// Release the previously acquired a 1-Wire net.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
//
void owRelease(int portnum)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:
         owRelease_DS9490(portnum & 0xFF);
         break;
      case DS1410E:
         owRelease_DS1410E(portnum & 0xFF);
         break;
      default:
      case DS9097U:
         owRelease_DS9097U(portnum & 0xFF);
         break;
      case HA7Net:
         owRelease_HA7Net(portnum & 0xFF);
         break;
   };
}
```

Επίπεδο συνόδου για τον USB μετατροπέα
Musbwsesu.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  usbwses.c - Acquire and release a Session on the 1-Wire Net using the
//              USB interface DS2490.
//
//  Version: 3.00
//
//  History:
//
```

```
#include "mownet.h"
#include "ds2490.h"

// handles to USB ports
HANDLE usbhnd[MAX_PORTNUM];
static SMALLINT usbhnd_init = 0;

//--------------------------------------------------------------------------
// Attempt to acquire a 1-Wire net using a USB port and a DS2490 based
// adapter.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'port_zstr'  - zero terminated port name.  For this platform
//                use format '\\.\DS2490-X' where X is the port number.
//
// Returns: TRUE - success, COM port opened
//
SMALLINT owAcquire_DS9490(int portnum, char *port_zstr)
{
   if(!usbhnd_init)
   {
      int i;
      for(i=0; i<MAX_PORTNUM; i++)
         usbhnd[0] = 0;
      usbhnd_init = 1;
   }
   OWASSERT( portnum<MAX_PORTNUM && portnum>=0 && !usbhnd[portnum],
             OWERROR_PORTNUM_ERROR, FALSE );

   // get a handle to the device
   usbhnd[portnum] = CreateFile(port_zstr,
                        GENERIC_READ | GENERIC_WRITE,
                        FILE_SHARE_READ,
                        NULL,
                        OPEN_EXISTING,
                        FILE_ATTRIBUTE_NORMAL,
                        NULL);

   if (usbhnd[portnum] != INVALID_HANDLE_VALUE)
   {
      // verify adapter is working
      if (!AdapterRecover(portnum))
      {
         CloseHandle(usbhnd[portnum]);
         return FALSE;
      }
      // reset the 1-Wire
      owTouchReset_DS9490(portnum);
   }
   else
   {
      // could not get resouse
      OWERROR(OWERROR_GET_SYSTEM_RESOURCE_FAILED);
      return FALSE;
   }

   return TRUE;
}

//--------------------------------------------------------------------------
// Attempt to acquire the specified 1-Wire net.
//
// 'port_zstr'     - zero terminated port name.
//
// Returns: port number or -1 if not successful in setting up the port.
//
int owAcquireEx_DS9490(char *port_zstr)
{
   int portnum = 0;

   if(!usbhnd_init)
   {
      int i;
      for(i=0; i<MAX_PORTNUM; i++)
         usbhnd[0] = 0;
```

```
      usbhnd_init = 1;
   }

   // check to find first available handle slot
   for(portnum = 0; portnum<MAX_PORTNUM; portnum++)
   {
      if(!usbhnd[portnum])
         break;
   }
   OWASSERT( portnum<MAX_PORTNUM, OWERROR_PORTNUM_ERROR, -1 );

   // get a handle to the device
   usbhnd[portnum] = CreateFile(port_zstr,
                       GENERIC_READ | GENERIC_WRITE,
                       FILE_SHARE_READ,
                       NULL,
                       OPEN_EXISTING,
                       FILE_ATTRIBUTE_NORMAL,
                       NULL);

   if (usbhnd[portnum] != INVALID_HANDLE_VALUE)
   {
      // verify adapter is working
      if (!AdapterRecover(portnum))
      {
         CloseHandle(usbhnd[portnum]);
         return -1;
      }
      // reset the 1-Wire
      owTouchReset_DS9490(portnum);
   }
   else
   {
      // could not get resouse
      OWERROR(OWERROR_GET_SYSTEM_RESOURCE_FAILED);
      return -1;
   }

   return portnum;
}

//--------------------------------------------------------------------------
// Release the port previously acquired a 1-Wire net.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
//
void owRelease_DS9490(int portnum)
{
   CloseHandle(usbhnd[portnum]);
}
```

Επίπεδο συνόδου για τον παράλληλο μετατροπέα
Mpsesw32.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
```

```
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  w32pses.C - Acquire and release a Session for general 1-Wire Net
//              library for parallel port adapters DS1410E/DS1410D
//
//  Version: 2.00
//

#include "mownet.h"
#include <windows.h>
#include "sacwd32.h"

// local function prototypes
SMALLINT  owAcquire_DS1410E(int,char *);
void owRelease_DS1410E(int);

// external prototypes
extern SMALLINT owTouchReset_DS1410E(int);
extern SMALLINT owWriteByte_DS1410E(int,int sendbyte);
extern SMALLINT owReadByte_DS1410E(int);
extern SMALLINT owSpeed_DS1410E(int,int);

// globals
char portname[MAX_PORTNUM][32];
int  PortNum[MAX_PORTNUM];
SMALLINT PortNum_init = 0;
extern sa_struct SauthGB[MAX_PORTNUM];

//--------------------------------------------------------------------------
// Attempt to acquire a 1-Wire net
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'port_zstr'  - zero terminated port name.
// 'return_msg' - zero terminated return message.
//
// Returns: TRUE - success, port opened
//
SMALLINT owAcquire_DS1410E(int portnum, char *port_zstr)
{
   uchar phys_port;

   if(!PortNum_init)
   {
      int i;
      for(i=0; i<MAX_PORTNUM; i++)
         PortNum[i] = 0;
      PortNum_init = 1;
   }

   OWASSERT( portnum<MAX_PORTNUM && portnum>=0 && !PortNum[portnum],
             OWERROR_PORTNUM_ERROR, FALSE );

   // convert the string in port_zstr to be the port number
   phys_port = atoi(port_zstr);
   if (phys_port > 3)
      return FALSE;

   PortNum[portnum] = phys_port;

   // get permission from sauth
   if (dowcheck())
   {
      setup(phys_port,&SauthGB[phys_port]);
      if (keyopen())
      {
         // looks like we have an adapter
         if (owTouchReset_DS1410E(portnum))
         {
            owWriteByte_DS1410E(portnum,0xF0);// Search
            if (owReadByte_DS1410E(portnum) != 0xFF)
            {
```

```
                sprintf(portname[portnum],"DS1410E/DS1410D on port %d",phys_port);
                return TRUE;
             }
          }

          keyclose();
       }
    }

    PortNum[portnum] = 0;
    return FALSE;
}

//--------------------------------------------------------------------------
// Attempt to acquire a 1-Wire net using a com port and a DS2480 based
// adapter.
//
// 'port_zstr'  - zero terminated port name.  For this platform
//                use format COMX where X is the port number.
//
// Returns: The portnum or -1 if the port wasn't acquired.
//
int owAcquireEx_DS1410E(char *port_zstr)
{
   int portnum;

   if(!PortNum_init)
   {
      int i;
      for(i=0; i<MAX_PORTNUM; i++)
         PortNum[i] = 0;
      PortNum_init = 1;
   }

   // check to find first available handle slot
   for(portnum = 0; portnum<MAX_PORTNUM; portnum++)
   {
      if(!PortNum[portnum])
         break;
   }
   OWASSERT( portnum<MAX_PORTNUM, OWERROR_PORTNUM_ERROR, -1 );

   if(owAcquire_DS1410E(portnum, port_zstr))
      return portnum;
   else
      return -1;
}

//--------------------------------------------------------------------------
// Release the previously acquired a 1-Wire net.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'return_msg' - zero terminated return message.
//
void owRelease_DS1410E(int portnum)
{
   // release for sauth
   keyclose();

   PortNum[portnum] = 0;
}
```

Επίπεδο συνόδου για τον σειριακό μετατροπέα
Mowsesu.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
```

```
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  owSesU.C - Acquire and release a Session on the 1-Wire Net.
//
//  Version: 2.01
//
//  History: 1.03 -> 2.00  Changed 'MLan' to 'ow'. Added support for
//                         multiple ports.
//           2.00 -> 2.01 Added error handling. Added circular-include check.
//           2.01 -> 2.10 Added raw memory error handling and SMALLINT
//           2.10 -> 3.00 Added memory bank functionality
//                        Added file I/O operations
//

#include "mownet.h"
#include "mds2480.h"

//--------------------------------------------------------------------------
// Attempt to acquire a 1-Wire net using a com port and a DS2480 based
// adapter.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number was provided to
//                OpenCOM to indicate the port number.
// 'port_zstr'  - zero terminated port name.  For this platform
//                use format COMX where X is the port number.
//
// Returns: TRUE - success, COM port opened
//
// exportable functions defined in ownetu.c
SMALLINT owAcquire_DS9097U(int portnum, char *port_zstr)
{
   // attempt to open the communications port
   if (OpenCOM(portnum,port_zstr) < 0)
   {
      OWERROR(OWERROR_OPENCOM_FAILED);
      return FALSE;
   }

   // detect DS2480
   if (!DS2480Detect(portnum))
   {
      CloseCOM(portnum);
      OWERROR(OWERROR_DS2480_NOT_DETECTED);
      return FALSE;
   }

   return TRUE;
}

//--------------------------------------------------------------------------
// Attempt to acquire a 1-Wire net using a com port and a DS2480 based
// adapter.
//
// 'port_zstr'  - zero terminated port name.  For this platform
//                use format COMX where X is the port number.
//
// Returns: valid handle, or -1 if an error occurred
//
// exportable functions defined in ownetu.c
//
int owAcquireEx_DS9097U(char *port_zstr)
```

```
{
   int portnum;

   // attempt to open the communications port
   if ((portnum = OpenCOMEx(port_zstr)) < 0)
   {
      OWERROR(OWERROR_OPENCOM_FAILED);
      return -1;
   }

   // detect DS2480
   if (!DS2480Detect(portnum))
   {
      CloseCOM(portnum);
      OWERROR(OWERROR_DS2480_NOT_DETECTED);
      return -1;
   }

   return portnum;
}

//--------------------------------------------------------------------------
// Release the previously acquired a 1-Wire net.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number was provided to
//                OpenCOM to indicate the port number.
//
void owRelease_DS9097U(int portnum)
{
   CloseCOM(portnum);
}
```

Συναρτήσεις επιπέδου διασύνδεσης

Βιβλιοθήκη που καλεί τις επιμέρους συναρτήσεις των προσαρμογέων για το επίπεδο διασύνδεσης

Multilink.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  multilnk.C - Wrapper to hook all adapter types in the 1-Wire Public
//               Domain API for link functions.
//
//  Version: 3.01 owTouchReset
//
```

```
//  added support fer the http server HA7Net.com
//  line added in function owTouchReset: "case HA7Net:........"
//  line added in function owReadBitPower: "case HA7Net:........"
//
//

#include "mownet.h"

// If TRUE, puts a delay in owTouchReset to compensate for alarming clocks.
SMALLINT FAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE = FALSE; // default owTouchReset to
quickest response.

extern SMALLINT default_type;
extern SMALLINT LFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE;
extern SMALLINT SFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE;
extern SMALLINT UFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE;

//--------------------------------------------------------------------------
// Reset all of the devices on the 1-Wire Net and return the result.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns: TRUE(1):  presense pulse(s) detected, device(s) reset
//          FALSE(0): no presense pulses detected
//
SMALLINT owTouchReset(int portnum)
{
   // check global flag does not match specific adapters
   if (FAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE !=
       UFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE)
   {
      UFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE =
         FAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE;
      SFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE =
         FAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE;
      LFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE =
         FAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE;
   }

   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owTouchReset_DS9490(portnum & 0xFF);
      case DS1410E: return owTouchReset_DS1410E(portnum & 0xFF);
      default:
      case DS9097U: return owTouchReset_DS9097U(portnum & 0xFF);
      case HA7Net: return owTouchReset_HA7Net(portnum & 0xFF);
   };
}


//--------------------------------------------------------------------------
// Send 1 bit of communication to the 1-Wire Net and return the
// result 1 bit read from the 1-Wire Net.  The parameter 'sendbit'
// least significant bit is used and the least significant bit
// of the result is the return bit.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
// 'sendbit'     - the least significant bit is the bit to send
//
// Returns: 0:   0 bit read from sendbit
//          1:   1 bit read from sendbit
//
SMALLINT owTouchBit(int portnum, SMALLINT sendbit)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owTouchBit_DS9490(portnum & 0xFF, sendbit);
      case DS1410E: return owTouchBit_DS1410E(portnum & 0xFF, sendbit);
      default:
      case DS9097U: return owTouchBit_DS9097U(portnum & 0xFF, sendbit);
   };
}

//--------------------------------------------------------------------------
```

```
// Send 8 bits of communication to the 1-Wire Net and verify that the
// 8 bits read from the 1-Wire Net is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'sendbyte'   - 8 bits to send (least significant byte)
//
// Returns:  TRUE: bytes written and echo was the same
//           FALSE: echo was not the same
//
SMALLINT owTouchByte(int portnum, SMALLINT sendbyte)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owTouchByte_DS9490(portnum & 0xFF, sendbyte);
      case DS1410E: return owTouchByte_DS1410E(portnum & 0xFF, sendbyte);
      default:
      case DS9097U: return owTouchByte_DS9097U(portnum & 0xFF, sendbyte);
   };
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the MicroLAN and verify that the
// 8 bits read from the MicroLAN is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'sendbyte'   - 8 bits to send (least significant byte)
//
// Returns:  TRUE: bytes written and echo was the same
//           FALSE: echo was not the same
//
SMALLINT owWriteByte(int portnum, SMALLINT sendbyte)
{
   return (owTouchByte(portnum,sendbyte) == sendbyte) ? TRUE : FALSE;
}

//--------------------------------------------------------------------------
// Send 8 bits of read communication to the 1-Wire Net and and return the
// result 8 bits read from the 1-Wire Net.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns:  TRUE:  8 bytes read from 1-Wire Net
//           FALSE: the 8 bytes were not read
//
SMALLINT owReadByte(int portnum)
{
   return owTouchByte(portnum,0xFF);
}

//--------------------------------------------------------------------------
// Set the 1-Wire Net communucation speed.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'new_speed'  - new speed defined as
//                MODE_NORMAL     0x00
//                MODE_OVERDRIVE  0x01
//
// Returns:  current 1-Wire Net speed
//
SMALLINT owSpeed(int portnum, SMALLINT new_speed)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owSpeed_DS9490(portnum & 0xFF, new_speed);
      case DS1410E: return owSpeed_DS1410E(portnum & 0xFF, new_speed);
      default:
      case DS9097U: return owSpeed_DS9097U(portnum & 0xFF, new_speed);
   };
}
```

```
//--------------------------------------------------------------------------
// Set the 1-Wire Net line level.  The values for new_level are
// as follows:
//
// 'portnum'   - number 0 to MAX_PORTNUM-1.  This number is provided to
//               indicate the symbolic port number.
// 'new_level' - new level defined as
//                MODE_NORMAL     0x00
//                MODE_STRONG5    0x02
//                MODE_PROGRAM    0x04
//                MODE_BREAK      0x08 (not supported)
//
// Returns:  current 1-Wire Net level
//
SMALLINT owLevel(int portnum, SMALLINT new_level)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owLevel_DS9490(portnum & 0xFF, new_level);
      case DS1410E: return owLevel_DS1410E(portnum & 0xFF, new_level);
      default:
      case DS9097U: return owLevel_DS9097U(portnum & 0xFF, new_level);
   };
}

//--------------------------------------------------------------------------
// This procedure creates a fixed 480 microseconds 12 volt pulse
// on the 1-Wire Net for programming EPROM iButtons.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns:  TRUE  successful
//           FALSE program voltage not available
//
SMALLINT owProgramPulse(int portnum)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owProgramPulse_DS9490(portnum & 0xFF);
      case DS1410E: return owProgramPulse_DS1410E(portnum & 0xFF);
      default:
      case DS9097U: return owProgramPulse_DS9097U(portnum & 0xFF);
   };
}

//--------------------------------------------------------------------------
//  Description:
//     Delay for at least 'len' ms
//
void msDelay(int len)
{
   Sleep(len);
}

//--------------------------------------------------------------------------
// Get the current millisecond tick count.  Does not have to represent
// an actual time, it just needs to be an incrementing timer.
//
long msGettick(void)
{
   return GetTickCount();
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the 1-Wire Net and verify that the
// 8 bits read from the 1-Wire Net is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.  After the
// 8 bits are sent change the level of the 1-Wire net.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'sendbyte' - 8 bits to send (least significant byte)
//
// Returns:  TRUE: bytes written and echo was the same
//           FALSE: echo was not the same
```

```
//
SMALLINT owWriteBytePower(int portnum, SMALLINT sendbyte)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owWriteBytePower_DS9490(portnum & 0xFF, sendbyte);
      case DS1410E: return owWriteBytePower_DS1410E(portnum & 0xFF, sendbyte);
      default:
      case DS9097U: return owWriteBytePower_DS9097U(portnum & 0xFF, sendbyte);
   };
}

//--------------------------------------------------------------------------
// Send 1 bit of communication to the 1-Wire Net and verify that the
// response matches the 'applyPowerResponse' bit and apply power delivery
// to the 1-Wire net.  Note that some implementations may apply the power
// first and then turn it off if the response is incorrect.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'applyPowerResponse' - 1 bit response to check, if correct then start
//                        power delivery
//
// Returns:  TRUE: bit written and response correct, strong pullup now on
//           FALSE: response incorrect
//
SMALLINT owReadBitPower(int portnum, SMALLINT applyPowerResponse)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owReadBitPower_DS9490(portnum & 0xFF, applyPowerResponse);
      case DS1410E: return owReadBitPower_DS1410E(portnum & 0xFF, applyPowerResponse);
      default:
      case DS9097U: return owReadBitPower_DS9097U(portnum & 0xFF, applyPowerResponse);
      case HA7Net: return owReadBitPower_HA7Net(portnum & 0xFF, applyPowerResponse);
   };
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the 1-Wire Net and verify that the
// 8 bits read from the 1-Wire Net is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.  After the
// 8 bits are sent change the level of the 1-Wire net.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'sendbyte' - 8 bits to send (least significant bit)
//
// Returns:  TRUE: bytes written and echo was the same, strong pullup now on
//           FALSE: echo was not the same
//
SMALLINT owReadBytePower(int portnum)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owReadBytePower_DS9490(portnum & 0xFF);
      case DS1410E: return owReadBytePower_DS1410E(portnum & 0xFF);
      default:
      case DS9097U: return owReadBytePower_DS9097U(portnum & 0xFF);
   };
}

//--------------------------------------------------------------------------
// This procedure indicates wether the adapter can deliver power.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  because all userial adapters have over drive.
//
SMALLINT owHasPowerDelivery(int portnum)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owHasPowerDelivery_DS9490(portnum & 0xFF);
      case DS1410E: return owHasPowerDelivery_DS1410E(portnum & 0xFF);
```

```
      default:
      case DS9097U: return owHasPowerDelivery_DS9097U(portnum & 0xFF);
   };
}

//--------------------------------------------------------------------------
// This procedure indicates wether the adapter can deliver power.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  because all userial adapters have over drive.
//
SMALLINT owHasOverDrive(int portnum)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owHasOverDrive_DS9490(portnum & 0xFF);
      case DS1410E: return owHasOverDrive_DS1410E(portnum & 0xFF);
      default:
      case DS9097U: return owHasOverDrive_DS9097U(portnum & 0xFF);
   };
}

//--------------------------------------------------------------------------
// This procedure creates a fixed 480 microseconds 12 volt pulse
// on the 1-Wire Net for programming EPROM iButtons.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  program volatage available
//           FALSE program voltage not available
SMALLINT owHasProgramPulse(int portnum)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owHasProgramPulse_DS9490(portnum & 0xFF);
      case DS1410E: return owHasProgramPulse_DS1410E(portnum & 0xFF);
      default:
      case DS9097U: return owHasProgramPulse_DS9097U(portnum & 0xFF);
   };
}
```

Επίπεδο διασύνδεσης για τον USB μετατροπέα
Musbwlink.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  usbwlnk.c - USB (DS2490) 1-Wire Public Domain low-level functions
```

```
//            (Requires DS2490.SYS)
//
//  Version: 3.00
//

#include "mownet.h"
#include "ds2490.h"

// globals
// flag for DS1994/DS2404 support
SMALLINT SFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE = TRUE;

// DS2490 state info
extern SMALLINT USBSpeed[MAX_PORTNUM];
extern SMALLINT USBLevel[MAX_PORTNUM];
extern HANDLE usbhnd[MAX_PORTNUM];
extern SMALLINT USBVpp[MAX_PORTNUM];

//--------------------------------------------------------------------------
// Reset all of the devices on the 1-Wire Net and return the result.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns: TRUE(1):  presense pulse(s) detected, device(s) reset
//          FALSE(0): no presense pulses detected
//
SMALLINT owTouchReset_DS9490(int portnum)
{
   SETUP_PACKET setup;
   ULONG nOutput = 0;
   SMALLINT present,vpp;

   // make sure strong pullup is not on
   if (USBLevel[portnum] == MODE_STRONG5)
      owLevel_DS9490(portnum, MODE_NORMAL);

   // construct command
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = COMM_CMD;
   setup.Value = COMM_1_WIRE_RESET | COMM_F | COMM_IM | COMM_SE;
   setup.Index = (USBSpeed[portnum] == MODE_OVERDRIVE) ?
                  ONEWIREBUSSPEED_OVERDRIVE : ONEWIREBUSSPEED_FLEXIBLE;
   setup.Length = 0;
   setup.DataOut = FALSE;
   // call the driver
   if (!DeviceIoControl(usbhnd[portnum],
                                        DS2490_IOCTL_VENDOR,
                                        &setup,
                                        sizeof(SETUP_PACKET),
                                        NULL,
                                        0,
                                        &nOutput,
                                        NULL))
   {
      // failure
      OWERROR(OWERROR_RESET_FAILED);
      AdapterRecover(portnum);
      return FALSE;
   }
   else
   {
      // extra delay for alarming DS1994/DS2404 complience
      if ((SFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE) && (USBSpeed[portnum] !=
MODE_OVERDRIVE))
         Sleep(5);

      // success, check for shorts
      if (DS2490ShortCheck(usbhnd[portnum], &present,&vpp))
      {
         USBVpp[portnum] = vpp;
         return present;
      }
      else
      {
         OWERROR(OWERROR_OW_SHORTED);
```

```
         // short occuring
         msDelay_DS9490(300);
         AdapterRecover(portnum);
         return FALSE;
      }
   }
}


//--------------------------------------------------------------------------
// Send 1 bit of communication to the 1-Wire Net and return the
// result 1 bit read from the 1-Wire Net.  The parameter 'sendbit'
// least significant bit is used and the least significant bit
// of the result is the return bit.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
// 'sendbit'     - the least significant bit is the bit to send
//
// Returns: 0:   0 bit read from sendbit
//          1:   1 bit read from sendbit
//
SMALLINT owTouchBit_DS9490(int portnum, SMALLINT sendbit)
{
   SETUP_PACKET setup;
   ULONG nOutput = 0;
   WORD nBytes;
   BYTE buf[2];

   // make sure strong pullup is not on
   if (USBLevel[portnum] == MODE_STRONG5)
      owLevel_DS9490(portnum, MODE_NORMAL);

   // set to do touchbit
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = COMM_CMD;
   setup.Value = COMM_BIT_IO | COMM_IM | ((sendbit) ? COMM_D : 0);
   setup.Index = 0;
   setup.Length = 0;
   setup.DataOut = FALSE;

   // call the driver
   if (!DeviceIoControl(usbhnd[portnum],
                                          DS2490_IOCTL_VENDOR,
                                          &setup,
                                          sizeof(SETUP_PACKET),
                                          NULL,
                                          0,
                                          &nOutput,
                                          NULL))
   {
      // failure
      OWERROR(OWERROR_ADAPTER_ERROR);
      AdapterRecover(portnum);
      return 0;
   }
   else
   {
      // success, read the result
      nBytes = 1;
      if (DS2490Read(usbhnd[portnum], buf, &nBytes))
         return buf[0];
      else
      {
         OWERROR(OWERROR_ADAPTER_ERROR);
         AdapterRecover(portnum);
         return 0;
      }
   }
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the 1-Wire Net and verify that the
// 8 bits read from the 1-Wire Net is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.
//
```

```
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'sendbyte'   - 8 bits to send (least significant byte)
//
// Returns:  TRUE: bytes written and echo was the same
//           FALSE: echo was not the same
//
SMALLINT owTouchByte_DS9490(int portnum, SMALLINT sendbyte)
{
   SETUP_PACKET setup;
   ULONG nOutput = 0;
   WORD nBytes;
   BYTE buf[2];

   // make sure strong pullup is not on
   if (USBLevel[portnum] == MODE_STRONG5)
      owLevel_DS9490(portnum, MODE_NORMAL);

   // set to do touchbyte
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = COMM_CMD;
   setup.Value = COMM_BYTE_IO | COMM_IM;
   setup.Index = sendbyte & 0xFF;
   setup.Length = 0;
   setup.DataOut = FALSE;

   // call the driver
   if (!DeviceIoControl(usbhnd[portnum],
                                          DS2490_IOCTL_VENDOR,
                                          &setup,
                                          sizeof(SETUP_PACKET),
                                          NULL,
                                          0,
                                          &nOutput,
                                          NULL))
   {
      // failure
      OWERROR(OWERROR_ADAPTER_ERROR);
      AdapterRecover(portnum);
      return 0;
   }
   else
   {
      // success, read the result
      nBytes = 1;
      if (DS2490Read(usbhnd[portnum], buf, &nBytes))
         return buf[0];
      else
      {
         OWERROR(OWERROR_ADAPTER_ERROR);
         AdapterRecover(portnum);
         return 0;
      }
   }
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the MicroLAN and verify that the
// 8 bits read from the MicroLAN is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'sendbyte'   - 8 bits to send (least significant byte)
//
// Returns:  TRUE: bytes written and echo was the same
//           FALSE: echo was not the same
//
SMALLINT owWriteByte_DS9490(int portnum, SMALLINT sendbyte)
{
   return (owTouchByte_DS9490(portnum,sendbyte) == sendbyte) ? TRUE : FALSE;
}

//--------------------------------------------------------------------------
// Send 8 bits of read communication to the 1-Wire Net and and return the
// result 8 bits read from the 1-Wire Net.
```

```
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns:  TRUE:  8 bytes read from 1-Wire Net
//           FALSE: the 8 bytes were not read
//
SMALLINT owReadByte_DS9490(int portnum)
{
   return owTouchByte_DS9490(portnum,0xFF);
}

//--------------------------------------------------------------------------
// Set the 1-Wire Net communucation speed.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'new_speed'  - new speed defined as
//                MODE_NORMAL     0x00
//                MODE_OVERDRIVE  0x01
//
// Returns:  current 1-Wire Net speed
//
SMALLINT owSpeed_DS9490(int portnum, SMALLINT new_speed)
{
   SETUP_PACKET setup;
   ULONG nOutput = 0;

   // set to change the speed
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = MODE_CMD;
   setup.Value = MOD_1WIRE_SPEED;
   setup.Index = ((new_speed == MODE_OVERDRIVE) ? ONEWIREBUSSPEED_OVERDRIVE :
ONEWIREBUSSPEED_FLEXIBLE) & 0x00FF;
   setup.Length = 0x00;
   setup.DataOut = FALSE;
   // call the driver
   if (!DeviceIoControl(usbhnd[portnum],
                                        DS2490_IOCTL_VENDOR,
                                        &setup,
                                        sizeof(SETUP_PACKET),
                                        NULL,
                                        0,
                                        &nOutput,
                                        NULL))
   {
      // failure
      OWERROR(OWERROR_ADAPTER_ERROR);
      AdapterRecover(portnum);
      return USBSpeed[portnum];
   }
   else
   {
      // success, read the result
      USBSpeed[portnum] = new_speed;
      return new_speed;
   }
}

//--------------------------------------------------------------------------
// Set the 1-Wire Net line level.  The values for new_level are
// as follows:
//
// 'portnum'   - number 0 to MAX_PORTNUM-1.  This number is provided to
//               indicate the symbolic port number.
// 'new_level' - new level defined as
//                MODE_NORMAL     0x00
//                MODE_STRONG5    0x02
//                MODE_PROGRAM    0x04 (not supported in this version)
//                MODE_BREAK      0x08 (not supported in chip)
//
// Returns:  current 1-Wire Net level
//
SMALLINT owLevel_DS9490(int portnum, SMALLINT new_level)
{
   SETUP_PACKET setup;
```

```
   ULONG nOutput = 0;

   // Turn off infinite strong pullup?
   if ((new_level == MODE_NORMAL) && (USBLevel[portnum] == MODE_STRONG5))
   {
      if (DS2490HaltPulse(usbhnd[portnum]))
         USBLevel[portnum] = MODE_NORMAL;
   }
   // Turn on infinite strong5 pullup?
   else if ((new_level == MODE_STRONG5) && (USBLevel[portnum] == MODE_NORMAL))
   {
      // assume duration set to infinite during setup of device
      // enable the pulse
      setup.RequestTypeReservedBits = 0x40;
      setup.Request = MODE_CMD;
      setup.Value = MOD_PULSE_EN;
      setup.Index = ENABLEPULSE_SPUE;
      setup.Length = 0x00;
      setup.DataOut = FALSE;
      // call the driver
      if (!DeviceIoControl(usbhnd[portnum],
                                          DS2490_IOCTL_VENDOR,
                                          &setup,
                                          sizeof(SETUP_PACKET),
                                          NULL,
                                          0,
                                          &nOutput,
                                          NULL))
      {
         // failure
         OWERROR(OWERROR_ADAPTER_ERROR);
         AdapterRecover(portnum);
         return USBLevel[portnum];
      }

      // start the pulse
      setup.RequestTypeReservedBits = 0x40;
      setup.Request = COMM_CMD;
      setup.Value = COMM_PULSE | COMM_IM;
      setup.Index = 0;
      setup.Length = 0;
      setup.DataOut = FALSE;
      // call the driver
      if (!DeviceIoControl(usbhnd[portnum],
                                          DS2490_IOCTL_VENDOR,
                                          &setup,
                                          sizeof(SETUP_PACKET),
                                          NULL,
                                          0,
                                          &nOutput,
                                          NULL))
      {
         // failure
         OWERROR(OWERROR_ADAPTER_ERROR);
         AdapterRecover(portnum);
         return USBLevel[portnum];
      }
      else
      {
         // success, read the result
         USBLevel[portnum] = new_level;
         return new_level;
      }
   }
   // unsupported
   else if (new_level != USBLevel[portnum])
   {
      OWERROR(OWERROR_FUNC_NOT_SUP);
      return USBLevel[portnum];
   }

   // success, return the current level
   return USBLevel[portnum];
}

//--------------------------------------------------------------------------
```

```
// This procedure creates a fixed 480 microseconds 12 volt pulse
// on the 1-Wire Net for programming EPROM iButtons.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns:  TRUE  successful
//           FALSE program voltage not available
//
SMALLINT owProgramPulse_DS9490(int portnum)
{
   SETUP_PACKET setup;
   ULONG nOutput = 0;

   // check if Vpp available
   if (!USBVpp[portnum])
      return FALSE;

   // make sure strong pullup is not on
   if (USBLevel[portnum] == MODE_STRONG5)
      owLevel_DS9490(portnum, MODE_NORMAL);

   // send the pulse (already enabled)
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = COMM_CMD;
   setup.Value = COMM_PULSE | COMM_IM | COMM_TYPE;
   setup.Index = 0;
   setup.Length = 0;
   setup.DataOut = FALSE;
   // call the driver
   if (!DeviceIoControl(usbhnd[portnum],
                                        DS2490_IOCTL_VENDOR,
                                        &setup,
                                        sizeof(SETUP_PACKET),
                                        NULL,
                                        0,
                                        &nOutput,
                                        NULL))
   {
      // failure
      OWERROR(OWERROR_ADAPTER_ERROR);
      AdapterRecover(portnum);
      return FALSE;
   }
   else
      return TRUE;
}

//--------------------------------------------------------------------------
//  Description:
//     Delay for at least 'len' ms
//
void msDelay_DS9490(int len)
{
   Sleep(len);
}

//--------------------------------------------------------------------------
// Get the current millisecond tick count.  Does not have to represent
// an actual time, it just needs to be an incrementing timer.
//
long msGettick_DS9490(void)
{
   return GetTickCount();
}

//--------------------------------------------------------------------------
// Send and recieve 8 bits of communication to the 1-Wire Net.
// The parameter 'sendbyte' least significant 8 bits are used.  After the
// 8 bits are sent change the level of the 1-Wire net to strong pullup
// power delivery.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'sendbyte' - 8 bits to send (least significant byte)
//
```

```
// Returns:  Byte read: byte read (or echo if write)
//           0: failure
//
SMALLINT owTouchBytePower(int portnum, SMALLINT sendbyte)
{
   SETUP_PACKET setup;
   ULONG nOutput = 0;
   WORD nBytes;
   BYTE buf[2];

   // make sure strong pullup is not on
   if (USBLevel[portnum] == MODE_STRONG5)
      owLevel_DS9490(portnum, MODE_NORMAL);

   // enable the strong pullup pulse
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = MODE_CMD;
   setup.Value = MOD_PULSE_EN;
   setup.Index = ENABLEPULSE_SPUE;
   setup.Length = 0x00;
   setup.DataOut = FALSE;
   // call the driver
   if (!DeviceIoControl(usbhnd[portnum],
                                        DS2490_IOCTL_VENDOR,
                                        &setup,
                                        sizeof(SETUP_PACKET),
                                        NULL,
                                        0,
                                        &nOutput,
                                        NULL))
   {
      // failure
      OWERROR(OWERROR_ADAPTER_ERROR);
      AdapterRecover(portnum);
      return FALSE;
   }

   // set to do touchbyte with the SPU immediatly after
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = COMM_CMD;
   setup.Value = COMM_BYTE_IO | COMM_IM | COMM_SPU;
   setup.Index = sendbyte & 0xFF;
   setup.Length = 0;
   setup.DataOut = FALSE;

   // call the driver
   if (!DeviceIoControl(usbhnd[portnum],
                                        DS2490_IOCTL_VENDOR,
                                        &setup,
                                        sizeof(SETUP_PACKET),
                                        NULL,
                                        0,
                                        &nOutput,
                                        NULL))
   {
      // failure
      OWERROR(OWERROR_ADAPTER_ERROR);
      AdapterRecover(portnum);
      return FALSE;
   }
   else
   {
      // now strong pullup is enabled
      USBLevel[portnum] = MODE_STRONG5;
      // success, read the result
      nBytes = 1;
      if (DS2490Read(usbhnd[portnum], buf, &nBytes))
         return buf[0];
      else
      {
         OWERROR(OWERROR_ADAPTER_ERROR);
         AdapterRecover(portnum);
         return FALSE;
      }
   }
}
```

```
//--------------------------------------------------------------------------
// Send 8 bits of communication to the 1-Wire net and verify that the
// 8 bits read from the 1-Wire Net is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.  After the
// 8 bits are sent change the level of the 1-Wire net.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'sendbyte' - 8 bits to send (least significant byte)
//
// Returns:  TRUE: bytes written and echo was the same
//           FALSE: echo was not the same
//
SMALLINT owWriteBytePower_DS9490(int portnum, SMALLINT sendbyte)
{
   return (owTouchBytePower(portnum,sendbyte) == sendbyte);
}

//--------------------------------------------------------------------------
// Read 8 bits of communication from the 1-Wire net and provide strong
// pullup power.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  byte read
//
SMALLINT owReadBytePower_DS9490(int portnum)
{
   return owTouchBytePower(portnum,0xFF);
}

//--------------------------------------------------------------------------
// Read 1 bit of communication from the 1-Wire net and verify that the
// response matches the 'applyPowerResponse' bit and apply power delivery
// to the 1-Wire net.  Note that some implementations may apply the power
// first and then turn it off if the response is incorrect.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'applyPowerResponse' - 1 bit response to check, if correct then start
//                        power delivery
//
// Returns:  TRUE: bit written and response correct, strong pullup now on
//           FALSE: response incorrect
//
SMALLINT owReadBitPower_DS9490(int portnum, SMALLINT applyPowerResponse)
{
   SETUP_PACKET setup;
   ULONG nOutput = 0;
   WORD nBytes;
   BYTE buf[2];

   // make sure strong pullup is not on
   if (USBLevel[portnum] == MODE_STRONG5)
      owLevel_DS9490(portnum, MODE_NORMAL);

   // enable the strong pullup pulse
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = MODE_CMD;
   setup.Value = MOD_PULSE_EN;
   setup.Index = ENABLEPULSE_SPUE;
   setup.Length = 0x00;
   setup.DataOut = FALSE;
   // call the driver
   if (!DeviceIoControl(usbhnd[portnum],
                                        DS2490_IOCTL_VENDOR,
                                        &setup,
                                        sizeof(SETUP_PACKET),
                                        NULL,
                                        0,
                                        &nOutput,
                                        NULL))
   {
      // failure
```

```
      OWERROR(OWERROR_ADAPTER_ERROR);
      AdapterRecover(portnum);
      return FALSE;
   }

   // set to do touchbit
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = COMM_CMD;
   setup.Value = COMM_BIT_IO | COMM_IM | COMM_SPU | COMM_D;
   setup.Index = 0;
   setup.Length = 0;
   setup.DataOut = FALSE;

   // call the driver
   if (!DeviceIoControl(usbhnd[portnum],
                                        DS2490_IOCTL_VENDOR,
                                        &setup,
                                        sizeof(SETUP_PACKET),
                                        NULL,
                                        0,
                                        &nOutput,
                                        NULL))
   {
      // failure
      OWERROR(OWERROR_ADAPTER_ERROR);
      AdapterRecover(portnum);
      return FALSE;
   }
   else
   {
      // now strong pullup is enabled
      USBLevel[portnum] = MODE_STRONG5;
      // success, read the result
      nBytes = 1;
      if (DS2490Read(usbhnd[portnum], buf, &nBytes))
      {
         // check response
         if (buf[0] != applyPowerResponse)
         {
            owLevel_DS9490(portnum, MODE_NORMAL);
            return FALSE;
         }
         else
            return TRUE;
      }
      else
      {
         OWERROR(OWERROR_ADAPTER_ERROR);
         AdapterRecover(portnum);
         return FALSE;
      }
   }
}

//--------------------------------------------------------------------------
// This procedure indicates wether the adapter can deliver power.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  because all userial adapters have over drive.
//
SMALLINT owHasPowerDelivery_DS9490(int portnum)
{
   return TRUE;
}

//--------------------------------------------------------------------------
// This procedure indicates wether the adapter can deliver power.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  because all userial adapters have over drive.
//
SMALLINT owHasOverDrive_DS9490(int portnum)
```

```
{
   return TRUE;
}

//--------------------------------------------------------------------------
// This procedure creates a fixed 480 microseconds 12 volt pulse
// on the 1-Wire Net for programming EPROM iButtons.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  program volatage available
//           FALSE program voltage not available
//
SMALLINT owHasProgramPulse_DS9490(int portnum)
{
   owTouchReset_DS9490(portnum);
   return USBVpp[portnum];
}

//--------------------------------------------------------------------------
// Attempt to recover communication with the DS2490
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              indicate the port number.
//
// Returns:  TRUE  DS2490 recover successfull
//           FALSE failed to recover
//
SMALLINT AdapterRecover(int portnum)
{
   // dectect DS2490 and set it up
   if (DS2490Detect(usbhnd[portnum]))
   {
      USBSpeed[portnum] = MODE_NORMAL; // current DS2490 1-Wire Net communication speed
      USBLevel[portnum] = MODE_NORMAL; // current DS2490 1-Wire Net level
      return TRUE;
   }
   else
   {
      OWERROR(OWERROR_SYSTEM_RESOURCE_INIT_FAILED);
      return FALSE;
   }
}
```

Επίπεδο διασύνδεσης για τον παράλληλο μετατροπέα
Mlpwin32.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  Win32P.C - Link Layer functions general 1-Wire driver
```

```
//           implementation for DS1410E/DS1410D parallel adapter.
//
//  Version: 3.00
//
//  History: 2.00 -> 3.00 Added call to setup() before each low level call
//                        to get base port address correct if multiple
//                        ports open

#include "mownet.h"
#include <windows.h>
#include "sacwd32.h"

// exportable link-level functions
SMALLINT owTouchReset_DS1410E(int);
SMALLINT owTouchBit_DS1410E(int,int);
SMALLINT owTouchByte_DS1410E(int,int);
SMALLINT owWriteByte_DS1410E(int,int sendbyte);
SMALLINT owReadByte_DS1410E(int);
SMALLINT owSpeed_DS1410E(int,int);
SMALLINT owLevel_DS1410E(int,int);
SMALLINT owProgramPulse_DS1410E(int);
void msDelay_DS1410E(int);
long msGettick_DS1410E(void);
SMALLINT owWriteBytePower_DS1410E(int,SMALLINT);
SMALLINT owHasPowerDelivery_DS1410E(int);
SMALLINT owHasProgramPulse_DS1410E(int);
SMALLINT owHasOverDrive_DS1410E(int);


// from Sacwd32
extern uchar DOWReset(void);
extern uchar ToggleOverdrive(void);

// globals
SMALLINT Level[MAX_PORTNUM]; // current 1-Wire Net level
SMALLINT Speed[MAX_PORTNUM]; // current 1-Wire Net communication speed
sa_struct SauthGB[MAX_PORTNUM];
extern int  PortNum[MAX_PORTNUM];

// global for DS1994/DS2404/DS1427.  If TRUE, puts a delay in owTouchReset to compensate
for alarming clocks.
SMALLINT LFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE = FALSE; // default owTouchReset to
FALSE


//--------------------------------------------------------------------------
// Reset all of the devices on the 1-Wire Net and return the result.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
//
// Returns: TRUE(1):  presense pulse(s) detected, device(s) reset
//          FALSE(0): no presense pulses detected
//
SMALLINT owTouchReset_DS1410E(int portnum)
{
   int rt;

   // make sure normal level
   owLevel_DS1410E(portnum,MODE_NORMAL);

   // do the reset
   setup((uchar)PortNum[portnum],&SauthGB[PortNum[portnum]]);
   rt = DOWReset();

   // wait at least 8ms (for DS1994/DS2404)
   if (LFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE)
   {
      Sleep(8);
      return TRUE;  // return TRUE as DOWReset will be FALSE
   }

   return rt;
}

//--------------------------------------------------------------------------
```

```
// Send 1 bit of communication to the 1-Wire Net and return the
// result 1 bit read from the 1-Wire Net.  The parameter 'sendbit'
// least significant bit is used and the least significant bit
// of the result is the return bit.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'sendbit'    - the least significant bit is the bit to send
//
// Returns: 0:   0 bit read from sendbit
//          1:   1 bit read from sendbit
//
SMALLINT owTouchBit_DS1410E(int portnum, int sendbit)
{
   // make sure normal level
   owLevel_DS1410E(portnum,MODE_NORMAL);

   return databit((uchar)sendbit,&SauthGB[PortNum[portnum]]);
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the 1-Wire Net and return the
// result 8 bits read from the 1-Wire Net.  The parameter 'sendbyte'
// least significant 8 bits are used and the least significant 8 bits
// of the result is the return byte.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'sendbyte'   - 8 bits to send (least significant byte)
//
// Returns:  8 bytes read from sendbyte
//
SMALLINT owTouchByte_DS1410E(int portnum, int sendbyte)
{
   // make sure normal level
   owLevel_DS1410E(portnum,MODE_NORMAL);

   return databyte((uchar)sendbyte,&SauthGB[PortNum[portnum]]);
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the 1-Wire Net and verify that the
// 8 bits read from the 1-Wire Net is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'sendbyte'   - 8 bits to send (least significant byte)
//
// Returns:  TRUE: bytes written and echo was the same
//           FALSE: echo was not the same
//
SMALLINT owWriteByte_DS1410E(int portnum, int sendbyte)
{
   return (owTouchByte_DS1410E(portnum,sendbyte) == sendbyte) ? TRUE : FALSE;
}

//--------------------------------------------------------------------------
// Send 8 bits of read communication to the 1-Wire Net and and return the
// result 8 bits read from the 1-Wire Net.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
//
// Returns:  8 bytes read from 1-Wire Net
//
SMALLINT owReadByte_DS1410E(int portnum)
{
   return owTouchByte_DS1410E(portnum,0xFF);
}

//--------------------------------------------------------------------------
// Set the 1-Wire Net communucation speed.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
```

```
// 'new_speed'  - new speed defined as
//                MODE_NORMAL     0x00
//                MODE_OVERDRIVE  0x01
//
// Returns:  current 1-Wire Net speed
//
SMALLINT owSpeed_DS1410E(int portnum, int new_speed)
{
   // check if already correct level
   if (new_speed == Speed[portnum])
      return Speed[portnum];

   // normal
   if (new_speed == MODE_NORMAL)
   {
      if (Speed[portnum] != MODE_NORMAL)
      {
         setup((uchar)PortNum[portnum],&SauthGB[PortNum[portnum]]);
         // OverdriveOff
         if(ToggleOverdrive())
            ToggleOverdrive();
      }

      Speed[portnum] = MODE_NORMAL;
   }
   // overdrive
   else if (new_speed == MODE_OVERDRIVE)
   {
      setup((uchar)PortNum[portnum],&SauthGB[PortNum[portnum]]);
      if (!ToggleOverdrive())
        if (!ToggleOverdrive()) // DS1410D
           return MODE_NORMAL;

      Speed[portnum] = MODE_OVERDRIVE;
   }

   return Speed[portnum];
}

//--------------------------------------------------------------------------
// Set the 1-Wire Net line level.  The values for NewLevel are
// as follows:
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'new_level'  - new level defined as
//                MODE_NORMAL     0x00
//                MODE_STRONG5    0x02
//                MODE_PROGRAM    0x04
//                MODE_BREAK      0x08
//
// Returns:  current 1-Wire Net level
//
SMALLINT owLevel_DS1410E(int portnum, int new_level)
{
   // check if already correct level
   if (new_level == Level[portnum])
      return Level[portnum];

   // check if just putting back to normal
   if (new_level == MODE_NORMAL)
   {
      // only turn off strong5 if not using overdrive
      if (Speed[portnum] != MODE_OVERDRIVE)
      {
         // Overdrive Off
         setup((uchar)PortNum[portnum],&SauthGB[PortNum[portnum]]);
         if(ToggleOverdrive())
            ToggleOverdrive();
      }

      Level[portnum] = MODE_NORMAL;
   }
   else if (new_level == MODE_STRONG5)
   {
      // only turn on strong5 if not using overdrive
```

```
      if (Speed[portnum] != MODE_OVERDRIVE)
      {
         setup((uchar)PortNum[portnum],&SauthGB[PortNum[portnum]]);
         if (!ToggleOverdrive())
           if (!ToggleOverdrive()) // DS1410D
              return MODE_NORMAL;
      }

      Level[portnum] = MODE_STRONG5;
   }

   return Level[portnum];
}

//--------------------------------------------------------------------------
// This procedure creates a fixed 480 microseconds 12 volt pulse
// on the 1-Wire Net for programming EPROM iButtons.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
//
// Returns:  TRUE  successful
//           FALSE program voltage not available
//
SMALLINT owProgramPulse_DS1410E(int portnum)
{
   // not supported with this adapter
   return 0;
}

//--------------------------------------------------------------------------
//  Description:
//     Delay for at least 'len' ms
//
void msDelay_DS1410E(int len)
{
   Sleep(len);
}

//--------------------------------------------------------------------------
// Get the current millisecond tick count.  Does not have to represent
// an actual time, it just needs to be an incrementing timer.
//
long msGettick_DS1410E(void)
{
   return GetTickCount();
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the 1-Wire Net and verify that the
// 8 bits read from the 1-Wire Net is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.  After the
// 8 bits are sent change the level of the 1-Wire net.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'sendbyte' - 8 bits to send (least significant byte)
//
// Returns:  TRUE: bytes written and echo was the same
//           FALSE: echo was not the same
//
SMALLINT owWriteBytePower_DS1410E(int portnum, SMALLINT sendbyte)
{
   if (owTouchByte_DS1410E(portnum,sendbyte) != sendbyte)
      return FALSE;

   if (owLevel_DS1410E(portnum,MODE_STRONG5) != MODE_STRONG5)
      return FALSE;

   return TRUE;
}

//--------------------------------------------------------------------------
// Send 1 bit of communication to the 1-Wire Net and verify that the
// response matches the 'applyPowerResponse' bit and apply power delivery
// to the 1-Wire net.  Note that some implementations may apply the power
```

```
// first and then turn it off if the response is incorrect.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'applyPowerResponse' - 1 bit response to check, if correct then start
//                        power delivery
//
// Returns:  TRUE: bit written and response correct, strong pullup now on
//           FALSE: response incorrect
//
SMALLINT owReadBitPower_DS1410E(int portnum, SMALLINT applyPowerResponse)
{
   SMALLINT rt=FALSE;

   if (owTouchBit_DS1410E(portnum, applyPowerResponse) == applyPowerResponse)
   {
      owLevel_DS1410E(portnum,MODE_STRONG5);
      rt = TRUE;
   }
   return rt;
}

//--------------------------------------------------------------------------
// Read 8 bits of communication to the 1-Wire Net and then turn on
// power delivery.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  byte read
//           FALSE: power delivery failed
//
SMALLINT owReadBytePower_DS1410E(int portnum)
{
   SMALLINT getbyte;

   getbyte = owTouchByte_DS1410E(portnum,0xFF);

   if (owLevel_DS1410E(portnum,MODE_STRONG5) != MODE_STRONG5)
      return FALSE;

   return getbyte;
}

//--------------------------------------------------------------------------
// This procedure indicates wether the adapter can deliver power.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  because all userial adapters have over drive.
//
SMALLINT owHasPowerDelivery_DS1410E(int portnum)
{
   return TRUE;
}

//--------------------------------------------------------------------------
// This procedure indicates wether the adapter can deliver power.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  because all userial adapters have over drive.
//
SMALLINT owHasOverDrive_DS1410E(int portnum)
{
   return TRUE;
}
//--------------------------------------------------------------------------
// This procedure creates a fixed 480 microseconds 12 volt pulse
// on the 1-Wire Net for programming EPROM iButtons.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
```

```
// Returns:  TRUE  program volatage available
//           FALSE program voltage not available
SMALLINT owHasProgramPulse_DS1410E(int portnum)
{
   return FALSE;
}
```

Επίπεδο διασύνδεσης για τον σειριακό μετατροπέα
Mowllu.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  owLLU.C - Link Layer 1-Wire Net functions using the DS2480/DS2480B (U)
//            serial interface chip.
//
//  Version: 3.00
//
//  History: 1.00 -> 1.01  DS2480 version number now ignored in
//                         owTouchReset.
//           1.02 -> 1.03  Removed caps in #includes for Linux capatibility
//                         Removed #include <windows.h>
//                         Add #include "mownet.h" to define TRUE,FALSE
//           1.03 -> 2.00  Changed 'MLan' to 'ow'. Added support for
//                         multiple ports.
//           2.00 -> 2.01 Added error handling. Added circular-include check.
//           2.01 -> 2.10 Added raw memory error handling and SMALLINT
//           2.10 -> 3.00 Added memory bank functionality
//                        Added file I/O operations
//                        Added owReadBitPower and owWriteBytePower
//                        Added support for THE LINK
//                        Updated owLevel to match AN192
//

#include "mownet.h"
#include "mds2480.h"

int dodebug=0;

// external globals
extern SMALLINT ULevel[MAX_PORTNUM]; // current DS2480B 1-Wire Net level
extern SMALLINT UBaud[MAX_PORTNUM];  // current DS2480B baud rate
extern SMALLINT UMode[MAX_PORTNUM];  // current DS2480B command or data mode state
extern SMALLINT USpeed[MAX_PORTNUM]; // current DS2480B 1-Wire Net communication speed
extern SMALLINT UVersion[MAX_PORTNUM]; // current DS2480B version

// new global for DS1994/DS2404/DS1427.  If TRUE, puts a delay in owTouchReset to
compensate for alarming clocks.
SMALLINT UFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE = FALSE; // default owTouchReset to
quickest response.
```

```
// local varable flag, true if program voltage available
static SMALLINT ProgramAvailable[MAX_PORTNUM];

//--------------------------------------------------------------------------
// Reset all of the devices on the 1-Wire Net and return the result.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number was provided to
//                OpenCOM to indicate the port number.
//
// Returns: TRUE(1):  presense pulse(s) detected, device(s) reset
//          FALSE(0): no presense pulses detected
//
// WARNING: Without setting the above global
(UFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE)
//          to TRUE, this routine will not function correctly on some
//          Alarm reset types of the DS1994/DS1427/DS2404 with
//          Rev 1,2, and 3 of the DS2480/DS2480B.
//
//
SMALLINT owTouchReset_DS9097U(int portnum)
{
   uchar readbuffer[10],sendpacket[10];
   uchar sendlen=0;

   if (dodebug)
      printf("\nRST ");//??????????????

   // make sure normal level
   owLevel_DS9097U(portnum,MODE_NORMAL);

   // check if correct mode
   if (UMode[portnum] != MODSEL_COMMAND)
   {
      UMode[portnum] = MODSEL_COMMAND;
      sendpacket[sendlen++] = MODE_COMMAND;
   }

   // construct the command
   sendpacket[sendlen++] = (uchar)(CMD_COMM | FUNCTSEL_RESET | USpeed[portnum]);

   // flush the buffers
   FlushCOM(portnum);

   // send the packet
   if (WriteCOM(portnum,sendlen,sendpacket))
   {
      // read back the 1 byte response
      if (ReadCOM(portnum,1,readbuffer) == 1)
      {
         // make sure this byte looks like a reset byte
         if (((readbuffer[0] & RB_RESET_MASK) == RB_PRESENCE) ||
             ((readbuffer[0] & RB_RESET_MASK) == RB_ALARMPRESENCE))
         {
            // check if programming voltage available
            ProgramAvailable[portnum] = ((readbuffer[0] & 0x20) == 0x20);
            UVersion[portnum] = (readbuffer[0] & VERSION_MASK);

            // only check for alarm pulse if DS2404 present and not using THE LINK
            if ((UFAMILY_CODE_04_ALARM_TOUCHRESET_COMPLIANCE) &&
                (UVersion[portnum] != VER_LINK))
            {
               msDelay_DS9097U(5); // delay 5 ms to give DS1994 enough time
               FlushCOM(portnum);
            }
            return TRUE;
         }
         else
            OWERROR(OWERROR_RESET_FAILED);

      }
      else
         OWERROR(OWERROR_READCOM_FAILED);
   }
   else
      OWERROR(OWERROR_WRITECOM_FAILED);
```

```
   // an error occured so re-sync with DS2480
   DS2480Detect(portnum);

   return FALSE;
}

//--------------------------------------------------------------------------
// Send 1 bit of communication to the 1-Wire Net and return the
// result 1 bit read from the 1-Wire Net.  The parameter 'sendbit'
// least significant bit is used and the least significant bit
// of the result is the return bit.
//
// 'portnum' - number 0 to MAX_PORTNUM-1.  This number was provided to
//             OpenCOM to indicate the port number.
// 'sendbit' - the least significant bit is the bit to send
//
// Returns: 0:   0 bit read from sendbit
//          1:   1 bit read from sendbit
//
SMALLINT owTouchBit_DS9097U(int portnum, SMALLINT sendbit)
{
   uchar readbuffer[10],sendpacket[10];
   uchar sendlen=0;

   // make sure normal level
   owLevel_DS9097U(portnum,MODE_NORMAL);

   // check if correct mode
   if (UMode[portnum] != MODSEL_COMMAND)
   {
      UMode[portnum] = MODSEL_COMMAND;
      sendpacket[sendlen++] = MODE_COMMAND;
   }

   // construct the command
   sendpacket[sendlen] = (sendbit != 0) ? BITPOL_ONE : BITPOL_ZERO;
   sendpacket[sendlen++] |= CMD_COMM | FUNCTSEL_BIT | USpeed[portnum];

   // flush the buffers
   FlushCOM(portnum);

   // send the packet
   if (WriteCOM(portnum,sendlen,sendpacket))
   {
      // read back the response
      if (ReadCOM(portnum,1,readbuffer) == 1)
      {
         // interpret the response
         if (((readbuffer[0] & 0xE0) == 0x80) &&
             ((readbuffer[0] & RB_BIT_MASK) == RB_BIT_ONE))
            return 1;
         else
            return 0;
      }
      else
         OWERROR(OWERROR_READCOM_FAILED);
   }
   else
      OWERROR(OWERROR_WRITECOM_FAILED);

   // an error occured so re-sync with DS2480
   DS2480Detect(portnum);

   return 0;
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the 1-Wire Net and verify that the
// 8 bits read from the 1-Wire Net is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'sendbyte' - 8 bits to send (least significant byte)
//
// Returns:  TRUE: bytes written and echo was the same
```

```
//           FALSE: echo was not the same
//
SMALLINT owWriteByte_DS9097U(int portnum, SMALLINT sendbyte)
{
   return (owTouchByte_DS9097U(portnum,sendbyte) == (0xff & sendbyte)) ? TRUE : FALSE;
}


//--------------------------------------------------------------------------
// Send 8 bits of read communication to the 1-Wire Net and and return the
// result 8 bits read from the 1-Wire Net.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  8 bits read from 1-Wire Net
//
SMALLINT owReadByte_DS9097U(int portnum)
{
   return owTouchByte_DS9097U(portnum,(SMALLINT)0xFF);
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the 1-Wire Net and return the
// result 8 bits read from the 1-Wire Net.  The parameter 'sendbyte'
// least significant 8 bits are used and the least significant 8 bits
// of the result is the return byte.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'sendbyte' - 8 bits to send (least significant byte)
//
// Returns:  8 bits read from sendbyte
//
SMALLINT owTouchByte_DS9097U(int portnum, SMALLINT sendbyte)
{
   uchar readbuffer[10],sendpacket[10];
   uchar sendlen=0;

   // make sure normal level
   owLevel_DS9097U(portnum,MODE_NORMAL);

   // check if correct mode
   if (UMode[portnum] != MODSEL_DATA)
   {
      UMode[portnum] = MODSEL_DATA;
      sendpacket[sendlen++] = MODE_DATA;
   }

   // add the byte to send
   sendpacket[sendlen++] = (uchar)sendbyte;

   // check for duplication of data that looks like COMMAND mode
   if (sendbyte ==(SMALLINT)MODE_COMMAND)
      sendpacket[sendlen++] = (uchar)sendbyte;

   // flush the buffers
   FlushCOM(portnum);

   // send the packet
   if (WriteCOM(portnum,sendlen,sendpacket))
   {
      // read back the 1 byte response
      if (ReadCOM(portnum,1,readbuffer) == 1)
      {
         if (dodebug)
            printf("%02X ",readbuffer[0]);//?????????????

          // return the response
          return (int)readbuffer[0];
      }
      else
         OWERROR(OWERROR_READCOM_FAILED);
   }
   else
      OWERROR(OWERROR_WRITECOM_FAILED);
```

```
   // an error occured so re-sync with DS2480
   DS2480Detect(portnum);

   return 0;
}

//--------------------------------------------------------------------------
// Set the 1-Wire Net communucation speed.
//
// 'portnum'   - number 0 to MAX_PORTNUM-1.  This number was provided to
//               OpenCOM to indicate the port number.
// 'new_speed' - new speed defined as
//                MODE_NORMAL     0x00
//                MODE_OVERDRIVE  0x01
//
// Returns:  current 1-Wire Net speed
//
SMALLINT owSpeed_DS9097U(int portnum, SMALLINT new_speed)
{
   uchar sendpacket[5];
   uchar sendlen=0;
   uchar rt = FALSE;

   // check if change from current mode
   if (((new_speed == MODE_OVERDRIVE) &&
        (USpeed[portnum] != SPEEDSEL_OD)) ||
       ((new_speed == MODE_NORMAL) &&
        (USpeed[portnum] != SPEEDSEL_FLEX)))
   {
      if (new_speed == MODE_OVERDRIVE)
      {
         // check for unsupported mode in THE LINK
         if (UVersion[portnum] == VER_LINK)
            OWERROR(OWERROR_FUNC_NOT_SUP);
         // if overdrive then switch to higher baud
         else if (DS2480ChangeBaud(portnum,MAX_BAUD) == MAX_BAUD)
         {
            USpeed[portnum] = SPEEDSEL_OD;
            rt = TRUE;
         }
      }
      else if (new_speed == MODE_NORMAL)
      {
         // else normal so set to 9600 baud
         if (DS2480ChangeBaud(portnum,PARMSET_9600) == PARMSET_9600)
         {
            USpeed[portnum] = SPEEDSEL_FLEX;
            rt = TRUE;
         }

      }

      // if baud rate is set correctly then change DS2480 speed
      if (rt)
      {
         // check if correct mode
         if (UMode[portnum] != MODSEL_COMMAND)
         {
            UMode[portnum] = MODSEL_COMMAND;
            sendpacket[sendlen++] = MODE_COMMAND;
         }

         // proceed to set the DS2480 communication speed
         sendpacket[sendlen++] = CMD_COMM | FUNCTSEL_SEARCHOFF | USpeed[portnum];

         // send the packet
         if (!WriteCOM(portnum,sendlen,sendpacket))
         {
            OWERROR(OWERROR_WRITECOM_FAILED);
            rt = FALSE;
            // lost communication with DS2480 then reset
            DS2480Detect(portnum);
         }
      }
   }
```

```
   // return the current speed
   return (USpeed[portnum] == SPEEDSEL_OD) ? MODE_OVERDRIVE : MODE_NORMAL;
}

//--------------------------------------------------------------------------
// Set the 1-Wire Net line level.  The values for new_level are
// as follows:
//
// 'portnum'   - number 0 to MAX_PORTNUM-1.  This number was provided to
//               OpenCOM to indicate the port number.
// 'new_level' - new level defined as
//                MODE_NORMAL     0x00
//                MODE_STRONG5    0x02
//                MODE_PROGRAM    0x04
//                MODE_BREAK      0x08 (not supported)
//
// Returns:  current 1-Wire Net level
//
SMALLINT owLevel_DS9097U(int portnum, SMALLINT new_level)
{
   uchar sendpacket[10],readbuffer[10];
   uchar sendlen=0;
   uchar rt=FALSE;

   // check if need to change level
   if (new_level != ULevel[portnum])
   {
      // check if correct mode
      if (UMode[portnum] != MODSEL_COMMAND)
      {
         UMode[portnum] = MODSEL_COMMAND;
         sendpacket[sendlen++] = MODE_COMMAND;
      }

      // check if just putting back to normal
      if (new_level == MODE_NORMAL)
      {
         // stop pulse command
         sendpacket[sendlen++] = MODE_STOP_PULSE;

         // add the command to begin the pulse WITHOUT prime
         sendpacket[sendlen++] = CMD_COMM | FUNCTSEL_CHMOD | SPEEDSEL_PULSE | BITPOL_5V
| PRIME5V_FALSE;

         // stop pulse command
         sendpacket[sendlen++] = MODE_STOP_PULSE;

         // flush the buffers
         FlushCOM(portnum);

         // send the packet
         if (WriteCOM(portnum,sendlen,sendpacket))
         {
            // read back the 2 byte response
            if (ReadCOM(portnum,2,readbuffer) == 2)
            {
               // check response byte
               if (((readbuffer[0] & 0xE0) == 0xE0) &&
                   ((readbuffer[1] & 0xE0) == 0xE0))
               {
                  rt = TRUE;
                  ULevel[portnum] = MODE_NORMAL;
               }
            }
            else
               OWERROR(OWERROR_READCOM_FAILED);
         }
         else
            OWERROR(OWERROR_WRITECOM_FAILED);
      }
      // set new level
      else
      {
         // strong 5 volts
         if (new_level == MODE_STRONG5)
```

```
         {
            // set the SPUD time value
            sendpacket[sendlen++] = CMD_CONFIG | PARMSEL_5VPULSE | PARMSET_infinite;
            // add the command to begin the pulse
            sendpacket[sendlen++] = CMD_COMM | FUNCTSEL_CHMOD | SPEEDSEL_PULSE |
BITPOL_5V;
         }
         // 12 volts
         else if (new_level == MODE_PROGRAM)
         {
            // check if programming voltage available
            if (!ProgramAvailable[portnum])
               return MODE_NORMAL;

            // set the PPD time value
            sendpacket[sendlen++] = CMD_CONFIG | PARMSEL_12VPULSE | PARMSET_infinite;
            // add the command to begin the pulse
            sendpacket[sendlen++] = CMD_COMM | FUNCTSEL_CHMOD | SPEEDSEL_PULSE |
BITPOL_12V;
         }

         // flush the buffers
         FlushCOM(portnum);

         // send the packet
         if (WriteCOM(portnum,sendlen,sendpacket))
         {
            // read back the 1 byte response from setting time limit
            if (ReadCOM(portnum,1,readbuffer) == 1)
            {
               // check response byte
               if ((readbuffer[0] & 0x81) == 0)
               {
                  ULevel[portnum] = new_level;
                  rt = TRUE;
               }
            }
            else
               OWERROR(OWERROR_READCOM_FAILED);
         }
         else
            OWERROR(OWERROR_WRITECOM_FAILED);
      }

      // if lost communication with DS2480 then reset
      if (rt != TRUE)
         DS2480Detect(portnum);
   }

   // return the current level
   return ULevel[portnum];
}

//--------------------------------------------------------------------------
// This procedure creates a fixed 480 microseconds 12 volt pulse
// on the 1-Wire Net for programming EPROM iButtons.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  successful
//           FALSE program voltage not available
//
SMALLINT owProgramPulse_DS9097U(int portnum)
{
   uchar sendpacket[10],readbuffer[10];
   uchar sendlen=0;

   // check if programming voltage available
   if (!ProgramAvailable[portnum])
      return FALSE;

   // make sure normal level
   owLevel_DS9097U(portnum,MODE_NORMAL);

   // check if correct mode
```

```
   if (UMode[portnum] != MODSEL_COMMAND)
   {
      UMode[portnum] = MODSEL_COMMAND;
      sendpacket[sendlen++] = MODE_COMMAND;
   }

   // set the SPUD time value
   sendpacket[sendlen++] = CMD_CONFIG | PARMSEL_12VPULSE | PARMSET_512us;

   // pulse command
   sendpacket[sendlen++] = CMD_COMM | FUNCTSEL_CHMOD | BITPOL_12V | SPEEDSEL_PULSE;

   // flush the buffers
   FlushCOM(portnum);

   // send the packet
   if (WriteCOM(portnum,sendlen,sendpacket))
   {
      // read back the 2 byte response
      if (ReadCOM(portnum,2,readbuffer) == 2)
      {
         // check response byte
         if (((readbuffer[0] | CMD_CONFIG) ==
                (CMD_CONFIG | PARMSEL_12VPULSE | PARMSET_512us)) &&
             ((readbuffer[1] & 0xFC) ==
                (0xFC & (CMD_COMM | FUNCTSEL_CHMOD | BITPOL_12V | SPEEDSEL_PULSE))))
            return TRUE;
      }
      else
         OWERROR(OWERROR_READCOM_FAILED);
   }
   else
      OWERROR(OWERROR_WRITECOM_FAILED);

   // an error occured so re-sync with DS2480
   DS2480Detect(portnum);

   return FALSE;
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the 1-Wire Net and verify that the
// 8 bits read from the 1-Wire Net is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.  After the
// 8 bits are sent change the level of the 1-Wire net.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'sendbyte' - 8 bits to send (least significant bit)
//
// Returns:  TRUE: bytes written and echo was the same, strong pullup now on
//           FALSE: echo was not the same
//
SMALLINT owWriteBytePower_DS9097U(int portnum, SMALLINT sendbyte)
{
   uchar sendpacket[10],readbuffer[10];
   uchar sendlen=0;
   uchar rt=FALSE;
   uchar i, temp_byte;

   if (dodebug)
      printf("P%02X ",sendbyte);//??????????????

   // check if correct mode
   if (UMode[portnum] != MODSEL_COMMAND)
   {
      UMode[portnum] = MODSEL_COMMAND;
      sendpacket[sendlen++] = MODE_COMMAND;
   }

   // set the SPUD time value
   sendpacket[sendlen++] = CMD_CONFIG | PARMSEL_5VPULSE | PARMSET_infinite;

   // construct the stream to include 8 bit commands with the last one
   // enabling the strong-pullup
   temp_byte = sendbyte;
```

```
   for (i = 0; i < 8; i++)
   {
      sendpacket[sendlen++] = ((temp_byte & 0x01) ? BITPOL_ONE : BITPOL_ZERO)
                              | CMD_COMM | FUNCTSEL_BIT | USpeed[portnum] |
                              ((i == 7) ? PRIME5V_TRUE : PRIME5V_FALSE);
      temp_byte >>= 1;
   }

   // flush the buffers
   FlushCOM(portnum);

   // send the packet
   if (WriteCOM(portnum,sendlen,sendpacket))
   {
      // read back the 9 byte response from setting time limit
      if (ReadCOM(portnum,9,readbuffer) == 9)
      {
         // check response
         if ((readbuffer[0] & 0x81) == 0)
         {
            // indicate the port is now at power delivery
            ULevel[portnum] = MODE_STRONG5;

            // reconstruct the echo byte
            temp_byte = 0;
            for (i = 0; i < 8; i++)
            {
               temp_byte >>= 1;
               temp_byte |= (readbuffer[i + 1] & 0x01) ? 0x80 : 0;
            }

            if (temp_byte == sendbyte)
               rt = TRUE;
         }
      }
      else
         OWERROR(OWERROR_READCOM_FAILED);
   }
   else
      OWERROR(OWERROR_WRITECOM_FAILED);

   // if lost communication with DS2480 then reset
   if (rt != TRUE)
      DS2480Detect(portnum);

   return rt;
}

//--------------------------------------------------------------------------
// Send 8 bits of communication to the 1-Wire Net and verify that the
// 8 bits read from the 1-Wire Net is the same (write operation).
// The parameter 'sendbyte' least significant 8 bits are used.  After the
// 8 bits are sent change the level of the 1-Wire net.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'sendbyte' - 8 bits to send (least significant bit)
//
// Returns:  TRUE: bytes written and echo was the same, strong pullup now on
//           FALSE: echo was not the same
//
SMALLINT owReadBytePower_DS9097U(int portnum)
{
   uchar sendpacket[10],readbuffer[10];
   uchar sendlen=0;
   uchar rt=FALSE;
   uchar i, temp_byte;

   // check if correct mode
   if (UMode[portnum] != MODSEL_COMMAND)
   {
      UMode[portnum] = MODSEL_COMMAND;
      sendpacket[sendlen++] = MODE_COMMAND;
   }

   // set the SPUD time value
```

```
   sendpacket[sendlen++] = CMD_CONFIG | PARMSEL_5VPULSE | PARMSET_infinite;

   // construct the stream to include 8 bit commands with the last one
   // enabling the strong-pullup
   temp_byte = 0xFF;
   for (i = 0; i < 8; i++)
   {
      sendpacket[sendlen++] = ((temp_byte & 0x01) ? BITPOL_ONE : BITPOL_ZERO)
                            | CMD_COMM | FUNCTSEL_BIT | USpeed[portnum] |
                            ((i == 7) ? PRIME5V_TRUE : PRIME5V_FALSE);
      temp_byte >>= 1;
   }

   // flush the buffers
   FlushCOM(portnum);

   // send the packet
   if (WriteCOM(portnum,sendlen,sendpacket))
   {
      // read back the 9 byte response from setting time limit
      if (ReadCOM(portnum,9,readbuffer) == 9)
      {
         // check response
         if ((readbuffer[0] & 0x81) == 0)
         {
            // indicate the port is now at power delivery
            ULevel[portnum] = MODE_STRONG5;

            // reconstruct the return byte
            temp_byte = 0;
            for (i = 0; i < 8; i++)
            {
               temp_byte >>= 1;
               temp_byte |= (readbuffer[i + 1] & 0x01) ? 0x80 : 0;
            }

            rt = TRUE;
         }
      }
      else
         OWERROR(OWERROR_READCOM_FAILED);
   }
   else
      OWERROR(OWERROR_WRITECOM_FAILED);

   // if lost communication with DS2480 then reset
   if (rt != TRUE)
      DS2480Detect(portnum);

   if (dodebug)
      printf("PFF%02X ",temp_byte);//??????????????

   return temp_byte;
}


//--------------------------------------------------------------------------
// Send 1 bit of communication to the 1-Wire Net and verify that the
// response matches the 'applyPowerResponse' bit and apply power delivery
// to the 1-Wire net.  Note that some implementations may apply the power
// first and then turn it off if the response is incorrect.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'applyPowerResponse' - 1 bit response to check, if correct then start
//                        power delivery
//
// Returns:  TRUE: bit written and response correct, strong pullup now on
//           FALSE: response incorrect
//
SMALLINT owReadBitPower_DS9097U(int portnum, SMALLINT applyPowerResponse)
{
   uchar sendpacket[3],readbuffer[3];
   uchar sendlen=0;
   uchar rt=FALSE;
```

```
   // check if correct mode
   if (UMode[portnum] != MODSEL_COMMAND)
   {
      UMode[portnum] = MODSEL_COMMAND;
      sendpacket[sendlen++] = MODE_COMMAND;
   }

   // set the SPUD time value
   sendpacket[sendlen++] = CMD_CONFIG | PARMSEL_5VPULSE | PARMSET_infinite;

   // enabling the strong-pullup after bit
   sendpacket[sendlen++] = BITPOL_ONE
                         | CMD_COMM | FUNCTSEL_BIT | USpeed[portnum] |
                         PRIME5V_TRUE;
   // flush the buffers
   FlushCOM(portnum);

   // send the packet
   if (WriteCOM(portnum,sendlen,sendpacket))
   {
      // read back the 2 byte response from setting time limit
      if (ReadCOM(portnum,2,readbuffer) == 2)
      {
         // check response to duration set
         if ((readbuffer[0] & 0x81) == 0)
         {
            // indicate the port is now at power delivery
            ULevel[portnum] = MODE_STRONG5;

            // check the response bit
            if ((readbuffer[1] & 0x01) == applyPowerResponse)
               rt = TRUE;
            else
               owLevel_DS9097U(portnum,MODE_NORMAL);

            return rt;
         }
      }
      else
         OWERROR(OWERROR_READCOM_FAILED);
   }
   else
      OWERROR(OWERROR_WRITECOM_FAILED);

   // if lost communication with DS2480 then reset
   if (rt != TRUE)
      DS2480Detect(portnum);

   return rt;
}

//--------------------------------------------------------------------------
// This procedure indicates whether the adapter can deliver power.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  because all userial adapters have over drive.
//
SMALLINT owHasPowerDelivery_DS9097U(int portnum)
{
   return TRUE;
}

//--------------------------------------------------------------------------
// This procedure indicates wether the adapter can deliver power.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  because all userial adapters have over drive.
//
SMALLINT owHasOverDrive_DS9097U(int portnum)
{
   return TRUE;
}
```

```
//--------------------------------------------------------------------------
// This procedure creates a fixed 480 microseconds 12 volt pulse
// on the 1-Wire Net for programming EPROM iButtons.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE  program volatage available
//           FALSE program voltage not available
SMALLINT owHasProgramPulse_DS9097U(int portnum)
{
   return ProgramAvailable[portnum];
}
```

Συναρτήσεις επιπέδου δικτύου

Βιβλιοθήκη που καλεί τις επιμέρους συναρτήσεις των προσαρμογέων για το επίπεδο δικτύου

Multinet.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  multinet.C - Wrapper to hook all adapter types in the 1-Wire Public
//               Domain API for network functions.
//
//  Version: 3.01 George VIolettas
//  added support ofr the http server HA7Net.com
//  line added in owNext: "case HA7Net:............."
//  line added in owFamilySearchSetup: "case HA7Net:............"
//  line added in owVerify: "case HA7Net:............"

#include "mownet.h"

extern SMALLINT default_type;

//--------------------------------------------------------------------------
// The 'owFirst' finds the first device on the 1-Wire Net  This function
// contains one parameter 'alarm_only'.  When
// 'alarm_only' is TRUE (1) the find alarm command 0xEC is
// sent instead of the normal search command 0xF0.
// Using the find alarm command 0xEC will limit the search to only
// 1-Wire devices that are in an 'alarm' state.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'do_reset'   - TRUE (1) perform reset before search, FALSE (0) do not
//                perform reset before search.
// 'alarm_only' - TRUE (1) the find alarm command 0xEC is
//                sent instead of the normal search command 0xF0
```

```
//
// Returns:   TRUE (1) : when a 1-Wire device was found and it's
//                       Serial Number placed in the global SerialNum[portnum]
//            FALSE (0): There are no devices on the 1-Wire Net.
//
SMALLINT owFirst(int portnum, SMALLINT do_reset, SMALLINT alarm_only)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owFirst_DS9490(portnum & 0xFF, do_reset, alarm_only);
      case DS1410E: return owFirst_DS1410E(portnum & 0xFF, do_reset, alarm_only);
      default:
      case DS9097U: return owFirst_DS9097U(portnum & 0xFF, do_reset, alarm_only);
   };
}

//--------------------------------------------------------------------------
// The 'owNext' function does a general search.  This function
// continues from the previos search state. The search state
// can be reset by using the 'owFirst' function.
// This function contains one parameter 'alarm_only'.
// When 'alarm_only' is TRUE (1) the find alarm command
// 0xEC is sent instead of the normal search command 0xF0.
// Using the find alarm command 0xEC will limit the search to only
// 1-Wire devices that are in an 'alarm' state.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'do_reset'   - TRUE (1) perform reset before search, FALSE (0) do not
//                perform reset before search.
// 'alarm_only' - TRUE (1) the find alarm command 0xEC is
//                sent instead of the normal search command 0xF0
//
// Returns:   TRUE (1) : when a 1-Wire device was found and it's
//                       Serial Number placed in the global SerialNum[portnum]
//            FALSE (0): when no new device was found.  Either the
//                       last search was the last device or there
//                       are no devices on the 1-Wire Net.
//
SMALLINT owNext(int portnum, SMALLINT do_reset, SMALLINT alarm_only)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owNext_DS9490(portnum & 0xFF, do_reset, alarm_only);
      case DS1410E: return owNext_DS1410E(portnum & 0xFF, do_reset, alarm_only);
      default:
      case DS9097U: return owNext_DS9097U(portnum & 0xFF, do_reset, alarm_only);
      case HA7Net: return owNext_HA7Net(portnum & 0xFF, do_reset, alarm_only);
   };
}

//--------------------------------------------------------------------------
// The 'owSerialNum' function either reads or sets the SerialNum buffer
// that is used in the search functions 'owFirst' and 'owNext'.
// This function contains two parameters, 'serialnum_buf' is a pointer
// to a buffer provided by the caller.  'serialnum_buf' should point to
// an array of 8 unsigned chars.  The second parameter is a flag called
// 'do_read' that is TRUE (1) if the operation is to read and FALSE
// (0) if the operation is to set the internal SerialNum buffer from
// the data in the provided buffer.
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number is provided to
//                   indicate the symbolic port number.
// 'serialnum_buf' - buffer to that contains the serial number to set
//                   when do_read = FALSE (0) and buffer to get the serial
//                   number when do_read = TRUE (1).
// 'do_read'       - flag to indicate reading (1) or setting (0) the current
//                   serial number.
//
void owSerialNum(int portnum, uchar *serialnum_buf, SMALLINT do_read)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  owSerialNum_DS9490(portnum & 0xFF, serialnum_buf, do_read);
               break;
      case DS1410E: owSerialNum_DS1410E(portnum & 0xFF, serialnum_buf, do_read);
```

```
                break;
      default:
      case DS9097U: owSerialNum_DS9097U(portnum & 0xFF, serialnum_buf, do_read);
                break;
   };
}

//--------------------------------------------------------------------------
// Setup the search algorithm to find a certain family of devices
// the next time a search function is called 'owNext'.
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number was provided to
//                   OpenCOM to indicate the port number.
// 'search_family' - family code type to set the search algorithm to find
//                   next.
//
void owFamilySearchSetup(int portnum, SMALLINT search_family)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  owFamilySearchSetup_DS9490(portnum & 0xFF, search_family);
                break;
      case DS1410E: owFamilySearchSetup_DS1410E(portnum & 0xFF, search_family);
                break;
      default:
      case DS9097U: owFamilySearchSetup_DS9097U(portnum & 0xFF, search_family);
                break;
      case HA7Net: owFamilySearchSetup_HA7Net(portnum & 0xFF, search_family);
                break;
   };
}

//--------------------------------------------------------------------------
// Set the current search state to skip the current family code.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
void owSkipFamily(int portnum)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  owSkipFamily_DS9490(portnum & 0xFF);
                break;
      case DS1410E: owSkipFamily_DS1410E(portnum & 0xFF);
                break;
      default:
      case DS9097U: owSkipFamily_DS9097U(portnum & 0xFF);
                break;
   };
}

//--------------------------------------------------------------------------
// The 'owAccess' function resets the 1-Wire and sends a MATCH Serial
// Number command followed by the current SerialNum code. After this
// function is complete the 1-Wire device is ready to accept device-specific
// commands.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns:   TRUE (1) : reset indicates present and device is ready
//                       for commands.
//            FALSE (0): reset does not indicate presence or echos 'writes'
//                       are not correct.
//
SMALLINT owAccess(int portnum)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owAccess_DS9490(portnum & 0xFF);
      case DS1410E: return owAccess_DS1410E(portnum & 0xFF);
      default:
      case DS9097U: return owAccess_DS9097U(portnum & 0xFF);
   };
}
```

```
//--------------------------------------------------------------------
// The function 'owVerify' verifies that the current device
// is in contact with the 1-Wire Net.
// Using the find alarm command 0xEC will verify that the device
// is in contact with the 1-Wire Net and is in an 'alarm' state.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
// 'alarm_only'  - TRUE (1) the find alarm command 0xEC
//                 is sent instead of the normal search
//                 command 0xF0.
//
// Returns:   TRUE (1) : when the 1-Wire device was verified
//                       to be on the 1-Wire Net
//                       with alarm_only == FALSE
//                       or verified to be on the 1-Wire Net
//                       AND in an alarm state when
//                       alarm_only == TRUE.
//            FALSE (0): the 1-Wire device was not on the
//                       1-Wire Net or if alarm_only
//                       == TRUE, the device may be on the
//                       1-Wire Net but in a non-alarm state.
//
SMALLINT owVerify(int portnum, SMALLINT alarm_only)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owVerify_DS9490(portnum & 0xFF, alarm_only);
      case DS1410E: return owVerify_DS1410E(portnum & 0xFF, alarm_only);
      default:
      case DS9097U: return owVerify_DS9097U(portnum & 0xFF, alarm_only);
      case HA7Net: return owVerify_HA7Net(portnum & 0xFF, alarm_only);
   };
}

//--------------------------------------------------------------------
// Perform a overdrive MATCH command to select the 1-Wire device with
// the address in the ID data register.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns:  TRUE: If the device is present on the 1-Wire Net and
//                 can do overdrive then the device is selected.
//           FALSE: Device is not present or not capable of overdrive.
//
//  *Note: This function could be converted to send DS2480
//         commands in one packet.
//
SMALLINT owOverdriveAccess(int portnum)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owOverdriveAccess_DS9490(portnum & 0xFF);
      case DS1410E: return owOverdriveAccess_DS1410E(portnum & 0xFF);
      default:
      case DS9097U: return owOverdriveAccess_DS9097U(portnum & 0xFF);
   };
}
```

Επίπεδο διασύνδεσης για τον USB μετατροπέα
Musbwnet.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
```

```
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  usbwnet.C - USB (DS2490) 1-Wire Public Domain API for network functions.
//
//  Version: 3.00
//
//

#include "mownet.h"
#include "ds2490.h"
#include <windows.h>


// local functions defined in ownetu.c
static SMALLINT bitacc(SMALLINT,SMALLINT,SMALLINT,uchar *);

// external
extern HANDLE usbhnd[MAX_PORTNUM];

// global variables for this module to hold search state information
static int LastDiscrepancy[MAX_PORTNUM];
static int LastFamilyDiscrepancy[MAX_PORTNUM];
static uchar LastDevice[MAX_PORTNUM];
uchar SerialNum[MAX_PORTNUM][8];

//--------------------------------------------------------------------------
// The 'owFirst' finds the first device on the 1-Wire Net  This function
// contains one parameter 'alarm_only'.  When
// 'alarm_only' is TRUE (1) the find alarm command 0xEC is
// sent instead of the normal search command 0xF0.
// Using the find alarm command 0xEC will limit the search to only
// 1-Wire devices that are in an 'alarm' state.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'do_reset'   - TRUE (1) perform reset before search, FALSE (0) do not
//                perform reset before search.
// 'alarm_only' - TRUE (1) the find alarm command 0xEC is
//                sent instead of the normal search command 0xF0
//
// Returns:   TRUE (1) : when a 1-Wire device was found and it's
//                       Serial Number placed in the global SerialNum[portnum]
//            FALSE (0): There are no devices on the 1-Wire Net.
//
SMALLINT owFirst_DS9490(int portnum, SMALLINT do_reset, SMALLINT alarm_only)
{
   // reset the search state
   LastDiscrepancy[portnum] = 0;
   LastDevice[portnum] = FALSE;
   LastFamilyDiscrepancy[portnum] = 0;

   return owNext_DS9490(portnum, do_reset, alarm_only);
}

//--------------------------------------------------------------------------
// The 'owNext' function does a general search.  This function
// continues from the previos search state. The search state
// can be reset by using the 'owFirst' function.
// This function contains one parameter 'alarm_only'.
// When 'alarm_only' is TRUE (1) the find alarm command
// 0xEC is sent instead of the normal search command 0xF0.
```

```
// Using the find alarm command 0xEC will limit the search to only
// 1-Wire devices that are in an 'alarm' state.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'do_reset'   - TRUE (1) perform reset before search, FALSE (0) do not
//                perform reset before search.
// 'alarm_only' - TRUE (1) the find alarm command 0xEC is
//                sent instead of the normal search command 0xF0
//
// Returns:   TRUE (1) : when a 1-Wire device was found and it's
//                       Serial Number placed in the global SerialNum[portnum]
//            FALSE (0): when no new device was found.  Either the
//                       last search was the last device or there
//                       are no devices on the 1-Wire Net.
//
SMALLINT owNext_DS9490(int portnum, SMALLINT do_reset, SMALLINT alarm_only)
{
   SETUP_PACKET setup;
   short rt=FALSE,i,ResetSearch=FALSE;
   BYTE rom_buf[16];
   BYTE ret_buf[16];
   WORD buf_len;
   uchar lastcrc8;
   WORD  nBytes = 8;
   ULONG nOutput = 0;
   long limit;
   STATUS_PACKET status;
   BYTE  nResult;

   // if the last call was the last one
   if (LastDevice[portnum])
   {
      // reset the search
      LastDiscrepancy[portnum] = 0;
      LastDevice[portnum] = FALSE;
      LastFamilyDiscrepancy[portnum] = 0;
      return FALSE;
   }

   // check if reset first is requested
   if (do_reset)
   {
      // reset the 1-wire
      // extra reset if last part was a DS1994/DS2404 (due to alarm)
      if ((SerialNum[portnum][0] & 0x7F) == 0x04)
         owTouchReset_DS9490(portnum);
      // if there are no parts on 1-wire, return FALSE
      if (!owTouchReset_DS9490(portnum))
      {
         // reset the search
         LastDiscrepancy[portnum] = 0;
         LastFamilyDiscrepancy[portnum] = 0;
         OWERROR(OWERROR_NO_DEVICES_ON_NET);
         return FALSE;
      }
   }

   // build the rom number to put to the USB chip
   for (i = 0; i < 8; i++)
      rom_buf[i] = SerialNum[portnum][i];

   // take into account LastDiscrepancy
   if (LastDiscrepancy[portnum] != 0xFF)
   {
      if (LastDiscrepancy[portnum] > 0)
         bitacc(WRITE_FUNCTION,1,(short)(LastDiscrepancy[portnum] - 1),rom_buf);

      for (i = LastDiscrepancy[portnum]; i < 64; i++)
         bitacc(WRITE_FUNCTION,0,i,rom_buf);
   }

   // put the ROM ID in EP2
   nBytes = 8;
   if (!DS2490Write(usbhnd[portnum], rom_buf, &nBytes))
   {
```

```
      AdapterRecover(portnum);
      return FALSE;
   }

   // setup for search command call
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = COMM_CMD;
   setup.Value = COMM_SEARCH_ACCESS | COMM_IM | COMM_SM | COMM_F | COMM_RTS;
   // the number of devices to read (1) with the search command
   setup.Index = 0x0100 | (((alarm_only) ? 0xEC : 0xF0) & 0x00FF);
   setup.Length = 0;
   setup.DataOut = FALSE;
   // call the driver
   if (!DeviceIoControl(usbhnd[portnum],
                                          DS2490_IOCTL_VENDOR,
                                          &setup,
                                          sizeof(SETUP_PACKET),
                                          NULL,
                                          0,
                                          &nOutput,
                                          NULL))
   {
      // failure
      AdapterRecover(portnum);
      return FALSE;
   }

   // set a time limit
   limit = msGettick_DS9490() + 200;
   // loop to wait on EP3 ready (device will go idle)
   do
   {
      // read the status to see if the pulse has been stopped
      if (!DS2490GetStatus(usbhnd[portnum], &status, &nResult))
      {
         // failure
         break;
      }
      else
      {
             // look for any fail conditions
             for (i = 0; i < nResult; i++)
             {
               // only check for error conditions when the condition is not a
ONEWIREDEVICEDETECT
               if (status.CommResultCodes[i] != ONEWIREDEVICEDETECT)
               {
              // failure
              break;
          }
         }
      }
   }
   while (((status.StatusFlags & STATUSFLAGS_IDLE) == 0) &&
          (limit > msGettick_DS9490()));

   // check results of the waite for idel
   if ((status.StatusFlags & STATUSFLAGS_IDLE) == 0)
   {
      AdapterRecover(portnum);
      return FALSE;
   }

   // check for data
   if (status.ReadBufferStatus > 0)
   {
      // read the load
      buf_len = 16;
           if(!DS2490Read(usbhnd[portnum], ret_buf, &buf_len))
      {
         AdapterRecover(portnum);
                  return FALSE;
      }

      // success, get rom and desrepancy
      LastDevice[portnum] = (buf_len == 8);
```

```
      // extract the ROM (and check crc)
      setcrc8(portnum,0);
      for (i = 0; i < 8; i++)
      {
         SerialNum[portnum][i] = ret_buf[i];
         lastcrc8 = docrc8(portnum,ret_buf[i]);
      }
      // crc OK and family code is not 0
      if (!lastcrc8 && SerialNum[portnum][0])
      {
         // loop through the descrepancy to get the pointers
         for (i = 0; i < 64; i++)
         {
            // if descrepancy
            if (bitacc(READ_FUNCTION,0,i,&ret_buf[8]) &&
                (bitacc(READ_FUNCTION,0,i,&ret_buf[0]) == 0))
            {
               LastDiscrepancy[portnum] = i + 1;
            }
         }
         rt = TRUE;
      }
      else
      {
         ResetSearch = TRUE;
         rt = FALSE;
      }
   }

   // check if need to reset search
   if (ResetSearch)
   {
      LastDiscrepancy[portnum] = 0xFF;
      LastDevice[portnum] = FALSE;
      for (i = 0; i < 8; i++)
         SerialNum[portnum][i] = 0;
   }

   return rt;
}

//--------------------------------------------------------------------------
// The 'owSerialNum' function either reads or sets the SerialNum buffer
// that is used in the search functions 'owFirst' and 'owNext'.
// This function contains two parameters, 'serialnum_buf' is a pointer
// to a buffer provided by the caller.  'serialnum_buf' should point to
// an array of 8 unsigned chars.  The second parameter is a flag called
// 'do_read' that is TRUE (1) if the operation is to read and FALSE
// (0) if the operation is to set the internal SerialNum buffer from
// the data in the provided buffer.
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number is provided to
//                   indicate the symbolic port number.
// 'serialnum_buf' - buffer to that contains the serial number to set
//                   when do_read = FALSE (0) and buffer to get the serial
//                   number when do_read = TRUE (1).
// 'do_read'       - flag to indicate reading (1) or setting (0) the current
//                   serial number.
//
void owSerialNum_DS9490(int portnum, uchar *serialnum_buf, SMALLINT do_read)
{
   uchar i;

   // read the internal buffer and place in 'serialnum_buf'
   if (do_read)
   {
      for (i = 0; i < 8; i++)
         serialnum_buf[i] = SerialNum[portnum][i];
   }
   // set the internal buffer from the data in 'serialnum_buf'
   else
   {
      for (i = 0; i < 8; i++)
         SerialNum[portnum][i] = serialnum_buf[i];
   }
}
```

```
//--------------------------------------------------------------------------
// Setup the search algorithm to find a certain family of devices
// the next time a search function is called 'owNext'.
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number was provided to
//                   OpenCOM to indicate the port number.
// 'search_family' - family code type to set the search algorithm to find
//                   next.
//
void owFamilySearchSetup_DS9490(int portnum, SMALLINT search_family)
{
   uchar i;

   // set the search state to find search_family type devices
   SerialNum[portnum][0] = search_family;
   for (i = 1; i < 8; i++)
      SerialNum[portnum][i] = 0;
   LastDiscrepancy[portnum] = 64;
   LastFamilyDiscrepancy[portnum] = 0;
   LastDevice[portnum] = FALSE;
}

//--------------------------------------------------------------------------
// Set the current search state to skip the current family code.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
void owSkipFamily_DS9490(int portnum)
{
   // set the Last discrepancy to last family discrepancy
   LastDiscrepancy[portnum] = LastFamilyDiscrepancy[portnum];

   // clear the last family discrpepancy
   LastFamilyDiscrepancy[portnum] = 0;

   // check for end of list
   if (LastDiscrepancy[portnum] == 0)
      LastDevice[portnum] = TRUE;
}

//--------------------------------------------------------------------------
// The 'owAccess' function resets the 1-Wire and sends a MATCH Serial
// Number command followed by the current SerialNum code. After this
// function is complete the 1-Wire device is ready to accept device-specific
// commands.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns:   TRUE (1) : reset indicates present and device is ready
//                       for commands.
//            FALSE (0): reset does not indicate presence or echos 'writes'
//                       are not correct.
//
SMALLINT owAccess_DS9490(int portnum)
{
   uchar sendpacket[9];
   uchar i;

   // reset the 1-wire
   if (owTouchReset_DS9490(portnum))
   {
      // create a buffer to use with block function
      // match Serial Number command 0x55
      sendpacket[0] = 0x55;
      // Serial Number
      for (i = 1; i < 9; i++)
         sendpacket[i] = SerialNum[portnum][i-1];

      // send/recieve the transfer buffer
      if (owBlock_DS9490(portnum,FALSE,sendpacket,9))
      {
         // verify that the echo of the writes was correct
         for (i = 1; i < 9; i++)
```

```
            if (sendpacket[i] != SerialNum[portnum][i-1])
               return FALSE;
         if (sendpacket[0] != 0x55)
         {
            OWERROR(OWERROR_WRITE_VERIFY_FAILED);
            return FALSE;
         }
         else
            return TRUE;
      }
      else
         OWERROR(OWERROR_BLOCK_FAILED);
   }
   else
      OWERROR(OWERROR_NO_DEVICES_ON_NET);

   // reset or match echo failed
   return FALSE;
}

//--------------------------------------------------------------------------
// The function 'owVerify' verifies that the current device
// is in contact with the 1-Wire Net.
// Using the find alarm command 0xEC will verify that the device
// is in contact with the 1-Wire Net and is in an 'alarm' state.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
// 'alarm_only'  - TRUE (1) the find alarm command 0xEC
//                 is sent instead of the normal search
//                 command 0xF0.
//
// Returns:   TRUE (1) : when the 1-Wire device was verified
//                       to be on the 1-Wire Net
//                       with alarm_only == FALSE
//                       or verified to be on the 1-Wire Net
//                       AND in an alarm state when
//                       alarm_only == TRUE.
//            FALSE (0): the 1-Wire device was not on the
//                       1-Wire Net or if alarm_only
//                       == TRUE, the device may be on the
//                       1-Wire Net but in a non-alarm state.
//
SMALLINT owVerify_DS9490(int portnum, SMALLINT alarm_only)
{
   uchar i,sendlen=0,goodbits=0,cnt=0,s,tst;
   uchar sendpacket[50];

   // construct the search rom
   if (alarm_only)
      sendpacket[sendlen++] = 0xEC; // issue the alarming search command
   else
      sendpacket[sendlen++] = 0xF0; // issue the search command
   // set all bits at first
   for (i = 1; i <= 24; i++)
      sendpacket[sendlen++] = 0xFF;
   // now set or clear apropriate bits for search
   for (i = 0; i < 64; i++)

bitacc(WRITE_FUNCTION,bitacc(READ_FUNCTION,0,i,&SerialNum[portnum][0]),(int)((i+1)*3-
1),&sendpacket[1]);

   // send/recieve the transfer buffer
   if (owBlock_DS9490(portnum,TRUE,sendpacket,sendlen))
   {
      // check results to see if it was a success
      for (i = 0; i < 192; i += 3)
      {
         tst = (bitacc(READ_FUNCTION,0,i,&sendpacket[1]) << 1) |
                bitacc(READ_FUNCTION,0,(int)(i+1),&sendpacket[1]);

         s = bitacc(READ_FUNCTION,0,cnt++,&SerialNum[portnum][0]);

         if (tst == 0x03)  // no device on line
         {
              goodbits = 0;    // number of good bits set to zero
```

```
            break;     // quit
         }

         if (((s == 0x01) && (tst == 0x02)) ||
             ((s == 0x00) && (tst == 0x01))    )  // correct bit
            goodbits++;  // count as a good bit
      }

      // check too see if there were enough good bits to be successful
      if (goodbits >= 8)
         return TRUE;
   }
   else
      OWERROR(OWERROR_BLOCK_FAILED);

   // block fail or device not present
   return FALSE;
}

//--------------------------------------------------------------------------
// Perform a overdrive MATCH command to select the 1-Wire device with
// the address in the ID data register.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns:  TRUE: If the device is present on the 1-Wire Net and
//                 can do overdrive then the device is selected.
//           FALSE: Device is not present or not capable of overdrive.
//
//  *Note: This function could be converted to send DS2480
//         commands in one packet.
//
SMALLINT owOverdriveAccess_DS9490(int portnum)
{
   uchar sendpacket[8];
   uchar i, bad_echo = FALSE;

   // make sure normal level
   owLevel_DS9490(portnum,MODE_NORMAL);

   // force to normal communication speed
   owSpeed_DS9490(portnum,MODE_NORMAL);

   // call the 1-Wire Net reset function
   if (owTouchReset_DS9490(portnum))
   {
      // send the match command 0x69
      if (owWriteByte_DS9490(portnum,0x69))
      {
         // switch to overdrive communication speed
         owSpeed_DS9490(portnum,MODE_OVERDRIVE);

         // create a buffer to use with block function
         // Serial Number
         for (i = 0; i < 8; i++)
            sendpacket[i] = SerialNum[portnum][i];

         // send/recieve the transfer buffer
         if (owBlock_DS9490(portnum,FALSE,sendpacket,8))
         {
            // verify that the echo of the writes was correct
            for (i = 0; i < 8; i++)
               if (sendpacket[i] != SerialNum[portnum][i])
                  bad_echo = TRUE;
            // if echo ok then success
            if (!bad_echo)
               return TRUE;
            else
               OWERROR(OWERROR_WRITE_VERIFY_FAILED);
         }
         else
            OWERROR(OWERROR_BLOCK_FAILED);
      }
      else
         OWERROR(OWERROR_WRITE_BYTE_FAILED);
```

```
   }
   else
      OWERROR(OWERROR_NO_DEVICES_ON_NET);

   // failure, force back to normal communication speed
   owSpeed_DS9490(portnum,MODE_NORMAL);

   return FALSE;
}

//--------------------------------------------------------------------------
// Bit utility to read and write a bit in the buffer 'buf'.
//
// 'op'    - operation (1) to set and (0) to read
// 'state' - set (1) or clear (0) if operation is write (1)
// 'loc'   - bit number location to read or write
// 'buf'   - pointer to array of bytes that contains the bit
//           to read or write
//
// Returns: 1   if operation is set (1)
//          0/1 state of bit number 'loc' if operation is reading
//
static SMALLINT bitacc(SMALLINT op, SMALLINT state, SMALLINT loc, uchar *buf)
{
   SMALLINT nbyt,nbit;

   nbyt = (loc / 8);
   nbit = loc - (nbyt * 8);

   if (op == WRITE_FUNCTION)
   {
      if (state)
         buf[nbyt] |= (0x01 << nbit);
      else
         buf[nbyt] &= ~(0x01 << nbit);

      return 1;
   }
   else
      return ((buf[nbyt] >> nbit) & 0x01);
}
```

Επίπεδο διασύνδεσης για τον παράλληλο μετατροπέα
Mownet.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  ownet.C - Network functions for 1-Wire net devices.
//
```

```
//  Version: 3.00
//
//  History: 1.00 -> 1.01  Change to owFamilySearchSetup, LastDiscrepancy[portnum]
//                         was set to 64 instead of 8 to enable devices with
//                         early contention to go in the '0' direction first.
//           1.02 -> 1.03  Initialized goodbits in  owVerify
//           1.03 -> 2.00  Changed 'MLan' to 'ow'. Added support for
//                         multiple ports.
//           2.00 -> 2.01  Added support for owError library
//           2.01 -> 3.00  Make search functions consistent with AN187
//

#include <stdio.h>
#include "mownet.h"

// exportable functions defined in ownet.c
SMALLINT bitacc(SMALLINT,SMALLINT,SMALLINT,uchar *);

// global variables for this module to hold search state information
static SMALLINT LastDiscrepancy[MAX_PORTNUM];
static SMALLINT LastFamilyDiscrepancy[MAX_PORTNUM];
static SMALLINT LastDevice[MAX_PORTNUM];
uchar SerialNum[MAX_PORTNUM][8];

//--------------------------------------------------------------------------
// The 'owFirst' finds the first device on the 1-Wire Net  This function
// contains one parameter 'alarm_only'.  When
// 'alarm_only' is TRUE (1) the find alarm command 0xEC is
// sent instead of the normal search command 0xF0.
// Using the find alarm command 0xEC will limit the search to only
// 1-Wire devices that are in an 'alarm' state.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'do_reset'   - TRUE (1) perform reset before search, FALSE (0) do not
//                perform reset before search.
// 'alarm_only' - TRUE (1) the find alarm command 0xEC is
//                sent instead of the normal search command 0xF0
//
// Returns:   TRUE (1) : when a 1-Wire device was found and it's
//                       Serial Number placed in the global SerialNum[portnum]
//            FALSE (0): There are no devices on the 1-Wire Net.
//
SMALLINT owFirst_DS1410E(int portnum, SMALLINT do_reset, SMALLINT alarm_only)
{
   // reset the search state
   LastDiscrepancy[portnum] = 0;
   LastDevice[portnum] = FALSE;
   LastFamilyDiscrepancy[portnum] = 0;

   return owNext_DS1410E(portnum,do_reset,alarm_only);
}

//--------------------------------------------------------------------------
// The 'owNext' function does a general search.  This function
// continues from the previos search state. The search state
// can be reset by using the 'owFirst' function.
// This function contains one parameter 'alarm_only'.
// When 'alarm_only' is TRUE (1) the find alarm command
// 0xEC is sent instead of the normal search command 0xF0.
// Using the find alarm command 0xEC will limit the search to only
// 1-Wire devices that are in an 'alarm' state.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'do_reset'   - TRUE (1) perform reset before search, FALSE (0) do not
//                perform reset before search.
// 'alarm_only' - TRUE (1) the find alarm command 0xEC is
//                sent instead of the normal search command 0xF0
//
// Returns:   TRUE (1) : when a 1-Wire device was found and it's
//                       Serial Number placed in the global SerialNum[portnum]
//            FALSE (0): when no new device was found.  Either the
//                       last search was the last device or there
//                       are no devices on the 1-Wire Net.
//
```

```
SMALLINT owNext_DS1410E(int portnum, SMALLINT do_reset, SMALLINT alarm_only)
{
   uchar bit_test, search_direction, bit_number;
   uchar last_zero, serial_byte_number, next_result;
   uchar serial_byte_mask;
   uchar lastcrc8;

   // initialize for search
   bit_number = 1;
   last_zero = 0;
   serial_byte_number = 0;
   serial_byte_mask = 1;
   next_result = 0;
   setcrc8(portnum,0);

   // if the last call was not the last one
   if (!LastDevice[portnum])
   {
      // check if reset first is requested
      if (do_reset)
      {
         // reset the 1-wire
         // if there are no parts on 1-wire, return FALSE
         if (!owTouchReset_DS1410E(portnum))
         {
            // printf("owTouchReset failed\r\n");
            // reset the search
            LastDiscrepancy[portnum] = 0;
            LastFamilyDiscrepancy[portnum] = 0;
            OWERROR(OWERROR_NO_DEVICES_ON_NET);
            return FALSE;
         }
      }

      // If finding alarming devices issue a different command
      if (alarm_only)
         owWriteByte_DS1410E(portnum,0xEC);  // issue the alarming search command
      else
         owWriteByte_DS1410E(portnum,0xF0);  // issue the search command

      //pause before beginning the search
      //usDelay(100);

      // loop to do the search
      do
      {
         // read a bit and its compliment
         bit_test = owTouchBit_DS1410E(portnum,1) << 1;
         bit_test |= owTouchBit_DS1410E(portnum,1);

         // check for no devices on 1-wire
         if (bit_test == 3)
            break;
         else
         {
            // all devices coupled have 0 or 1
            if (bit_test > 0)
              search_direction = !(bit_test & 0x01);  // bit write value for search
            else
            {
               // if this discrepancy if before the Last Discrepancy
               // on a previous next then pick the same as last time
               if (bit_number < LastDiscrepancy[portnum])
                  search_direction = ((SerialNum[portnum][serial_byte_number] &
serial_byte_mask) > 0);
               else
                  // if equal to last pick 1, if not then pick 0
                  search_direction = (bit_number == LastDiscrepancy[portnum]);

               // if 0 was picked then record its position in LastZero
               if (search_direction == 0)
               {
                  last_zero = bit_number;

                  // check for Last discrepancy in family
                  if (last_zero < 9)
```

```
                    LastFamilyDiscrepancy[portnum] = last_zero;
                }
            }

            // set or clear the bit in the SerialNum[portnum] byte serial_byte_number
            // with mask serial_byte_mask
            if (search_direction == 1)
              SerialNum[portnum][serial_byte_number] |= serial_byte_mask;
            else
              SerialNum[portnum][serial_byte_number] &= ~serial_byte_mask;

            // serial number search direction write bit
            owTouchBit_DS1410E(portnum,search_direction);

            // increment the byte counter bit_number
            // and shift the mask serial_byte_mask
            bit_number++;
            serial_byte_mask <<= 1;

            // if the mask is 0 then go to new SerialNum[portnum] byte
serial_byte_number
            // and reset mask
            if (serial_byte_mask == 0)
            {
                lastcrc8 = docrc8(portnum,SerialNum[portnum][serial_byte_number]);  //
accumulate the CRC
                serial_byte_number++;
                serial_byte_mask = 1;
            }
         }
      }
      while(serial_byte_number < 8);  // loop until through all SerialNum[portnum] bytes
0-7

      // if the search was successful then
      if (!((bit_number < 65) || lastcrc8))
      {
         // search successful so set
LastDiscrepancy[portnum],LastDevice[portnum],next_result
         LastDiscrepancy[portnum] = last_zero;
         LastDevice[portnum] = (LastDiscrepancy[portnum] == 0);
         next_result = TRUE;
      }
   }

   // if no device found then reset counters so next 'next' will be
   // like a first
   if (!next_result || !SerialNum[portnum][0])
   {
      LastDiscrepancy[portnum] = 0;
      LastDevice[portnum] = FALSE;
      LastFamilyDiscrepancy[portnum] = 0;
      next_result = FALSE;
   }

   return next_result;
}

//--------------------------------------------------------------------------
// The 'owSerialNum' function either reads or sets the SerialNum buffer
// that is used in the search functions 'owFirst' and 'owNext'.
// This function contains two parameters, 'serialnum_buf' is a pointer
// to a buffer provided by the caller.  'serialnum_buf' should point to
// an array of 8 unsigned chars.  The second parameter is a flag called
// 'do_read' that is TRUE (1) if the operation is to read and FALSE
// (0) if the operation is to set the internal SerialNum buffer from
// the data in the provided buffer.
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number is provided to
//                   indicate the symbolic port number.
// 'serialnum_buf' - buffer to that contains the serial number to set
//                   when do_read = FALSE (0) and buffer to get the serial
//                   number when do_read = TRUE (1).
// 'do_read'       - flag to indicate reading (1) or setting (0) the current
//                   serial number.
//
```

```
void owSerialNum_DS1410E(int portnum, uchar *serialnum_buf, SMALLINT do_read)
{
   uchar i;

   // read the internal buffer and place in 'serialnum_buf'
   if (do_read)
   {
      for (i = 0; i < 8; i++)
         serialnum_buf[i] = SerialNum[portnum][i];
   }
   // set the internal buffer from the data in 'serialnum_buf'
   else
   {
      for (i = 0; i < 8; i++)
         SerialNum[portnum][i] = serialnum_buf[i];
   }
}

//--------------------------------------------------------------------------
// Setup the search algorithm to find a certain family of devices
// the next time a search function is called 'owNext'.
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number was provided to
//                   OpenCOM to indicate the port number.
// 'search_family' - family code type to set the search algorithm to find
//                   next.
//
void owFamilySearchSetup_DS1410E(int portnum, SMALLINT search_family)
{
   uchar i;

   // set the search state to find SearchFamily type devices
   SerialNum[portnum][0] = search_family;
   for (i = 1; i < 8; i++)
      SerialNum[portnum][i] = 0;
   LastDiscrepancy[portnum] = 64;
   LastDevice[portnum] = FALSE;
}

//--------------------------------------------------------------------------
// Set the current search state to skip the current family code.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
void owSkipFamily_DS1410E(int portnum)
{
   // set the Last discrepancy to last family discrepancy
   LastDiscrepancy[portnum] = LastFamilyDiscrepancy[portnum];
   LastFamilyDiscrepancy[portnum] = 0;

   // check for end of list
   if (LastDiscrepancy[portnum] == 0)
      LastDevice[portnum] = TRUE;
}

//--------------------------------------------------------------------------
// The 'owAccess' function resets the 1-Wire and sends a MATCH Serial
// Number command followed by the current SerialNum code. After this
// function is complete the 1-Wire device is ready to accept device-specific
// commands.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns:   TRUE (1) : reset indicates present and device is ready
//                       for commands.
//            FALSE (0): reset does not indicate presence or echos 'writes'
//                       are not correct.
//
SMALLINT owAccess_DS1410E(int portnum)
{
   uchar sendpacket[9];
   uchar i;

   // reset the 1-wire
```

```
   if (owTouchReset_DS1410E(portnum))
   {
      // create a buffer to use with block function
      // match Serial Number command 0x55
      sendpacket[0] = 0x55;
      // Serial Number
      for (i = 1; i < 9; i++)
         sendpacket[i] = SerialNum[portnum][i-1];

      // send/recieve the transfer buffer
      if (owBlock_DS1410E(portnum,FALSE,sendpacket,9))
      {
         // verify that the echo of the writes was correct
         for (i = 1; i < 9; i++)
            if (sendpacket[i] != SerialNum[portnum][i-1])
               return FALSE;
         if (sendpacket[0] != 0x55)
         {
            OWERROR(OWERROR_WRITE_VERIFY_FAILED);
            return FALSE;
         }
         else
            return TRUE;
      }
      else
         OWERROR(OWERROR_BLOCK_FAILED);
   }
   else
      OWERROR(OWERROR_NO_DEVICES_ON_NET);

   // reset or match echo failed
   return FALSE;
}

//--------------------------------------------------------------------------
// The function 'owVerify' verifies that the current device
// is in contact with the 1-Wire Net.
// Using the find alarm command 0xEC will verify that the device
// is in contact with the 1-Wire Net and is in an 'alarm' state.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
// 'alarm_only'  - TRUE (1) the find alarm command 0xEC
//                 is sent instead of the normal search
//                 command 0xF0.
//
// Returns:   TRUE (1) : when the 1-Wire device was verified
//                       to be on the 1-Wire Net
//                       with alarm_only == FALSE
//                       or verified to be on the 1-Wire Net
//                       AND in an alarm state when
//                       alarm_only == TRUE.
//            FALSE (0): the 1-Wire device was not on the
//                       1-Wire Net or if alarm_only
//                       == TRUE, the device may be on the
//                       1-Wire Net but in a non-alarm state.
//
SMALLINT owVerify_DS1410E(int portnum, SMALLINT alarm_only)
{
   uchar i,sendlen=0,goodbits=0,cnt=0,s,tst;
   uchar sendpacket[50];

   // construct the search
   if (alarm_only)
      sendpacket[sendlen++] = 0xEC; // issue the alarming search command
   else
      sendpacket[sendlen++] = 0xF0; // issue the search command
   // set all bits at first
   for (i = 1; i <= 24; i++)
      sendpacket[sendlen++] = 0xFF;
   // now set or clear apropriate bits for search
   for (i = 0; i < 64; i++)

bitacc(WRITE_FUNCTION,bitacc(READ_FUNCTION,0,i,&SerialNum[portnum][0]),(int)((i+1)*3-
1),&sendpacket[1]);
```

```
   // send/recieve the transfer buffer
   if (owBlock_DS1410E(portnum,TRUE,sendpacket,sendlen))
   {
      // check results to see if it was a success
      for (i = 0; i < 192; i += 3)
      {
         tst = (bitacc(READ_FUNCTION,0,i,&sendpacket[1]) << 1) |
                bitacc(READ_FUNCTION,0,(int)(i+1),&sendpacket[1]);

         s = bitacc(READ_FUNCTION,0,cnt++,&SerialNum[portnum][0]);

         if (tst == 0x03)  // no device on line
         {
              goodbits = 0;    // number of good bits set to zero
              break;     // quit
         }

         if (((s == 0x01) && (tst == 0x02)) ||
             ((s == 0x00) && (tst == 0x01))    )  // correct bit
            goodbits++;  // count as a good bit
      }

      // check too see if there were enough good bits to be successful
      if (goodbits >= 8)
         return TRUE;
   }
   else
      OWERROR(OWERROR_BLOCK_FAILED);

   // block fail or device not present
   return FALSE;
}

//--------------------------------------------------------------------------
// Perform a overdrive MATCH command to select the 1-Wire device with
// the address in the ID data register.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
//
// Returns:  TRUE: If the device is present on the 1-Wire Net and
//                 can do overdrive then the device is selected.
//           FALSE: Device is not present or not capable of overdrive.
//
//  *Note: This function could be converted to send DS2480
//         commands in one packet.
//
SMALLINT owOverdriveAccess_DS1410E(int portnum)
{
   uchar sendpacket[8];
   uchar i, bad_echo = FALSE;

   // make sure normal level
   owLevel_DS1410E(portnum,MODE_NORMAL);

   // force to normal communication speed
   owSpeed_DS1410E(portnum,MODE_NORMAL);

   // call the 1-Wire Net reset function
   if (owTouchReset_DS1410E(portnum))
   {
      // send the match command 0x69
      if (owWriteByte_DS1410E(portnum,0x69))
      {
         // switch to overdrive communication speed
         owSpeed_DS1410E(portnum,MODE_OVERDRIVE);

         // create a buffer to use with block function
         // Serial Number
         for (i = 0; i < 8; i++)
            sendpacket[i] = SerialNum[portnum][i];

         // send/recieve the transfer buffer
         if (owBlock_DS1410E(portnum,FALSE,sendpacket,8))
         {
            // verify that the echo of the writes was correct
```

```
            for (i = 0; i < 8; i++)
               if (sendpacket[i] != SerialNum[portnum][i])
                  bad_echo = TRUE;
            // if echo ok then success
            if (!bad_echo)
               return TRUE;
            else
               OWERROR(OWERROR_WRITE_VERIFY_FAILED);
         }
         else
            OWERROR(OWERROR_BLOCK_FAILED);
      }
      else
         OWERROR(OWERROR_WRITE_BYTE_FAILED);
   }
   else
      OWERROR(OWERROR_NO_DEVICES_ON_NET);

   // failure, force back to normal communication speed
   owSpeed_DS1410E(portnum,MODE_NORMAL);

   return FALSE;
}

//--------------------------------------------------------------------------
// Bit utility to read and write a bit in the buffer 'buf'.
//
// 'op'    - operation (1) to set and (0) to read
// 'state' - set (1) or clear (0) if operation is write (1)
// 'loc'   - bit number location to read or write
// 'buf'   - pointer to array of bytes that contains the bit
//           to read or write
//
// Returns: 1   if operation is set (1)
//          0/1 state of bit number 'loc' if operation is reading
//
SMALLINT bitacc(SMALLINT op, SMALLINT state, SMALLINT loc, uchar *buf)
{
   SMALLINT nbyt,nbit;

   nbyt = (loc / 8);
   nbit = loc - (nbyt * 8);

   if (op == WRITE_FUNCTION)
   {
      if (state)
         buf[nbyt] |= (0x01 << nbit);
      else
         buf[nbyt] &= ~(0x01 << nbit);

      return 1;
   }
   else
      return ((buf[nbyt] >> nbit) & 0x01);
}
```

Επίπεδο δικτύου για τον σειριακό μετατροπέα
Mownetu.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
```

```
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  owNetU.C - Network functions for 1-Wire Net devices
//             using the DS2480/DS2480B (U) serial interface chip.
//
//  Version: 2.01
//
//           1.02 -> 1.03  Removed caps in #includes for Linux capatibility
//           1.03 -> 2.00  Changed 'MLan' to 'ow'. Added support for
//                         multiple ports.
//          2.00 -> 2.01 Added error handling. Added circular-include check.
//          2.01 -> 2.10 Added raw memory error handling and SMALLINT
//          2.10 -> 3.00 Added memory bank functionality
//                       Added file I/O operations
//                       Updated search functions to be consistent with AN192
//

#include "mownet.h"
#include "mds2480.h"

// local functions defined in ownetu.c
static SMALLINT bitacc(SMALLINT,SMALLINT,SMALLINT,uchar *);

// global variables for this module to hold search state information
static int LastDiscrepancy[MAX_PORTNUM];
static int LastFamilyDiscrepancy[MAX_PORTNUM];
static uchar LastDevice[MAX_PORTNUM];
uchar SerialNum[MAX_PORTNUM][8];

// external globals
extern int UMode[MAX_PORTNUM];
extern uchar USpeed[MAX_PORTNUM];

//--------------------------------------------------------------------------
// The 'owFirst' finds the first device on the 1-Wire Net  This function
// contains one parameter 'alarm_only'.  When
// 'alarm_only' is TRUE (1) the find alarm command 0xEC is
// sent instead of the normal search command 0xF0.
// Using the find alarm command 0xEC will limit the search to only
// 1-Wire devices that are in an 'alarm' state.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'do_reset'   - TRUE (1) perform reset before search, FALSE (0) do not
//                perform reset before search.
// 'alarm_only' - TRUE (1) the find alarm command 0xEC is
//                sent instead of the normal search command 0xF0
//
// Returns:   TRUE (1) : when a 1-Wire device was found and it's
//                       Serial Number placed in the global SerialNum
//            FALSE (0): There are no devices on the 1-Wire Net.
//
SMALLINT owFirst_DS9097U(int portnum, SMALLINT do_reset, SMALLINT alarm_only)
{
   // reset the search state
   LastDiscrepancy[portnum] = 0;
   LastDevice[portnum] = FALSE;
   LastFamilyDiscrepancy[portnum] = 0;

   return owNext_DS9097U(portnum, do_reset, alarm_only);
}

//--------------------------------------------------------------------------
// The 'owNext' function does a general search.  This function
// continues from the previos search state. The search state
// can be reset by using the 'owFirst' function.
// This function contains one parameter 'alarm_only'.
// When 'alarm_only' is TRUE (1) the find alarm command
```

```
// 0xEC is sent instead of the normal search command 0xF0.
// Using the find alarm command 0xEC will limit the search to only
// 1-Wire devices that are in an 'alarm' state.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number was provided to
//                OpenCOM to indicate the port number.
// 'do_reset'   - TRUE (1) perform reset before search, FALSE (0) do not
//                perform reset before search.
// 'alarm_only' - TRUE (1) the find alarm command 0xEC is
//                sent instead of the normal search command 0xF0
//
// Returns:   TRUE (1) : when a 1-Wire device was found and it's
//                       Serial Number placed in the global SerialNum
//            FALSE (0): when no new device was found.  Either the
//                       last search was the last device or there
//                       are no devices on the 1-Wire Net.
//
SMALLINT owNext_DS9097U(int portnum, SMALLINT do_reset, SMALLINT alarm_only)
{
   uchar last_zero,pos;
   uchar tmp_serial_num[8];
   uchar readbuffer[20],sendpacket[40];
   uchar i,sendlen=0;
   uchar lastcrc8;

   // if the last call was the last one
   if (LastDevice[portnum])
   {
      // reset the search
      LastDiscrepancy[portnum] = 0;
      LastDevice[portnum] = FALSE;
      LastFamilyDiscrepancy[portnum] = 0;
      return FALSE;
   }

   // check if reset first is requested
   if (do_reset)
   {
      // reset the 1-wire
      // if there are no parts on 1-wire, return FALSE
      if (!owTouchReset_DS9097U(portnum))
      {
         // reset the search
         LastDiscrepancy[portnum] = 0;
         LastFamilyDiscrepancy[portnum] = 0;
         OWERROR(OWERROR_NO_DEVICES_ON_NET);
         return FALSE;
      }
   }

   // build the command stream
   // call a function that may add the change mode command to the buff
   // check if correct mode
   if (UMode[portnum] != MODSEL_DATA)
   {
      UMode[portnum] = MODSEL_DATA;
      sendpacket[sendlen++] = MODE_DATA;
   }

   // search command
   if (alarm_only)
      sendpacket[sendlen++] = 0xEC; // issue the alarming search command
   else
      sendpacket[sendlen++] = 0xF0; // issue the search command

   // change back to command mode
   UMode[portnum] = MODSEL_COMMAND;
   sendpacket[sendlen++] = MODE_COMMAND;

   // search mode on
   sendpacket[sendlen++] = (uchar)(CMD_COMM | FUNCTSEL_SEARCHON | USpeed[portnum]);

   // change back to data mode
   UMode[portnum] = MODSEL_DATA;
   sendpacket[sendlen++] = MODE_DATA;
```

```
   // set the temp Last Descrep to none
   last_zero = 0;

   // add the 16 bytes of the search
   pos = sendlen;
   for (i = 0; i < 16; i++)
      sendpacket[sendlen++] = 0;

   // only modify bits if not the first search
   if (LastDiscrepancy[portnum] != 0)
   {
      // set the bits in the added buffer
      for (i = 0; i < 64; i++)
      {
         // before last discrepancy
         if (i < (LastDiscrepancy[portnum] - 1))
               bitacc(WRITE_FUNCTION,
                   bitacc(READ_FUNCTION,0,i,&SerialNum[portnum][0]),
                   (short)(i * 2 + 1),
                   &sendpacket[pos]);
         // at last discrepancy
         else if (i == (LastDiscrepancy[portnum] - 1))
                bitacc(WRITE_FUNCTION,1,
                   (short)(i * 2 + 1),
                   &sendpacket[pos]);
         // after last discrepancy so leave zeros
      }
   }

   // change back to command mode
   UMode[portnum] = MODSEL_COMMAND;
   sendpacket[sendlen++] = MODE_COMMAND;

   // search OFF
   sendpacket[sendlen++] = (uchar)(CMD_COMM | FUNCTSEL_SEARCHOFF | USpeed[portnum]);

   // flush the buffers
   FlushCOM(portnum);

   // send the packet
   if (WriteCOM(portnum,sendlen,sendpacket))
   {
      // read back the 1 byte response
      if (ReadCOM(portnum,17,readbuffer) == 17)
      {
         // interpret the bit stream
         for (i = 0; i < 64; i++)
         {
            // get the SerialNum bit
            bitacc(WRITE_FUNCTION,
                   bitacc(READ_FUNCTION,0,(short)(i * 2 + 1),&readbuffer[1]),
                   i,
                   &tmp_serial_num[0]);
            // check LastDiscrepancy
            if ((bitacc(READ_FUNCTION,0,(short)(i * 2),&readbuffer[1]) == 1) &&
                (bitacc(READ_FUNCTION,0,(short)(i * 2 + 1),&readbuffer[1]) == 0))
            {
               last_zero = i + 1;
               // check LastFamilyDiscrepancy
               if (i < 8)
                  LastFamilyDiscrepancy[portnum] = i + 1;
            }
         }

         // do dowcrc
         setcrc8(portnum,0);
         for (i = 0; i < 8; i++)
            lastcrc8 = docrc8(portnum,tmp_serial_num[i]);

         // check results
         if ((lastcrc8 != 0) || (LastDiscrepancy[portnum] == 63) || (tmp_serial_num[0]
== 0))
         {
            // error during search
            // reset the search
            LastDiscrepancy[portnum] = 0;
```

```
            LastDevice[portnum] = FALSE;
            LastFamilyDiscrepancy[portnum] = 0;
            OWERROR(OWERROR_SEARCH_ERROR);
            return FALSE;
         }
         // successful search
         else
         {
            // set the last discrepancy
            LastDiscrepancy[portnum] = last_zero;

            // check for last device
            if (LastDiscrepancy[portnum] == 0)
               LastDevice[portnum] = TRUE;

            // copy the SerialNum to the buffer
            for (i = 0; i < 8; i++)
               SerialNum[portnum][i] = tmp_serial_num[i];

            // set the count
            return TRUE;
         }
      }
      else
         OWERROR(OWERROR_READCOM_FAILED);
   }
   else
      OWERROR(OWERROR_WRITECOM_FAILED);

   // an error occured so re-sync with DS2480
   DS2480Detect(portnum);

   // reset the search
   LastDiscrepancy[portnum] = 0;
   LastDevice[portnum] = FALSE;
   LastFamilyDiscrepancy[portnum] = 0;

   return FALSE;
}

//--------------------------------------------------------------------------
// The 'owSerialNum' function either reads or sets the SerialNum buffer
// that is used in the search functions 'owFirst' and 'owNext'.
// This function contains two parameters, 'serialnum_buf' is a pointer
// to a buffer provided by the caller.  'serialnum_buf' should point to
// an array of 8 unsigned chars.  The second parameter is a flag called
// 'do_read' that is TRUE (1) if the operation is to read and FALSE
// (0) if the operation is to set the internal SerialNum buffer from
// the data in the provided buffer.
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number was provided to
//                   OpenCOM to indicate the port number.
// 'serialnum_buf' - buffer to that contains the serial number to set
//                   when do_read = FALSE (0) and buffer to get the serial
//                   number when do_read = TRUE (1).
// 'do_read'       - flag to indicate reading (1) or setting (0) the current
//                   serial number.
//
void owSerialNum_DS9097U(int portnum, uchar *serialnum_buf, SMALLINT do_read)
{
   uchar i;

   // read the internal buffer and place in 'serialnum_buf'
   if (do_read)
   {
      for (i = 0; i < 8; i++)
         serialnum_buf[i] = SerialNum[portnum][i];
   }
   // set the internal buffer from the data in 'serialnum_buf'
   else
   {
      for (i = 0; i < 8; i++)
         SerialNum[portnum][i] = serialnum_buf[i];
   }
}
```

```
//--------------------------------------------------------------------------
// Setup the search algorithm to find a certain family of devices
// the next time a search function is called 'owNext'.
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number was provided to
//                   OpenCOM to indicate the port number.
// 'search_family' - family code type to set the search algorithm to find
//                   next.
//
void owFamilySearchSetup_DS9097U(int portnum, SMALLINT search_family)
{
   uchar i;

   // set the search state to find search_family type devices
   SerialNum[portnum][0] = search_family;
   for (i = 1; i < 8; i++)
      SerialNum[portnum][i] = 0;
   LastDiscrepancy[portnum] = 64;
   LastFamilyDiscrepancy[portnum] = 0;
   LastDevice[portnum] = FALSE;
}

//--------------------------------------------------------------------------
// Set the current search state to skip the current family code.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
void owSkipFamily_DS9097U(int portnum)
{
   // set the Last discrepancy to last family discrepancy
   LastDiscrepancy[portnum] = LastFamilyDiscrepancy[portnum];

   // clear the last family discrpepancy
   LastFamilyDiscrepancy[portnum] = 0;

   // check for end of list
   if (LastDiscrepancy[portnum] == 0)
      LastDevice[portnum] = TRUE;
}

//--------------------------------------------------------------------------
// The 'owAccess' function resets the 1-Wire and sends a MATCH Serial
// Number command followed by the current SerialNum code. After this
// function is complete the 1-Wire device is ready to accept device-specific
// commands.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:   TRUE (1) : reset indicates present and device is ready
//                       for commands.
//            FALSE (0): reset does not indicate presence or echos 'writes'
//                       are not correct.
//
SMALLINT owAccess_DS9097U(int portnum)
{
   uchar sendpacket[9];
   uchar i;

   // reset the 1-wire
   if (owTouchReset_DS9097U(portnum))
   {
      // create a buffer to use with block function
      // match Serial Number command 0x55
      sendpacket[0] = 0x55;
      // Serial Number
      for (i = 1; i < 9; i++)
         sendpacket[i] = SerialNum[portnum][i-1];

      // send/recieve the transfer buffer
      if (owBlock_DS9097U(portnum,FALSE,sendpacket,9))
      {
         // verify that the echo of the writes was correct
         for (i = 1; i < 9; i++)
            if (sendpacket[i] != SerialNum[portnum][i-1])
```

```
            return FALSE;
         if (sendpacket[0] != 0x55)
         {
            OWERROR(OWERROR_WRITE_VERIFY_FAILED);
            return FALSE;
         }
         else
            return TRUE;
      }
      else
         OWERROR(OWERROR_BLOCK_FAILED);
   }
   else
      OWERROR(OWERROR_NO_DEVICES_ON_NET);

   // reset or match echo failed
   return FALSE;
}

//--------------------------------------------------------------------------
// The function 'owVerify' verifies that the current device
// is in contact with the 1-Wire Net.
// Using the find alarm command 0xEC will verify that the device
// is in contact with the 1-Wire Net and is in an 'alarm' state.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number was provided to
//                OpenCOM to indicate the port number.
// 'alarm_only' - TRUE (1) the find alarm command 0xEC
//                         is sent instead of the normal search
//                         command 0xF0.
//
// Returns:   TRUE (1) : when the 1-Wire device was verified
//                       to be on the 1-Wire Net
//                       with alarm_only == FALSE
//                       or verified to be on the 1-Wire Net
//                       AND in an alarm state when
//                       alarm_only == TRUE.
//            FALSE (0): the 1-Wire device was not on the
//                       1-Wire Net or if alarm_only
//                       == TRUE, the device may be on the
//                       1-Wire Net but in a non-alarm state.
//
SMALLINT owVerify_DS9097U(int portnum, SMALLINT alarm_only)
{
   uchar i,sendlen=0,goodbits=0,cnt=0,s,tst;
   uchar sendpacket[50];

   // construct the search rom
   if (alarm_only)
      sendpacket[sendlen++] = 0xEC; // issue the alarming search command
   else
      sendpacket[sendlen++] = 0xF0; // issue the search command
   // set all bits at first
   for (i = 1; i <= 24; i++)
      sendpacket[sendlen++] = 0xFF;
   // now set or clear apropriate bits for search
   for (i = 0; i < 64; i++)

bitacc(WRITE_FUNCTION,bitacc(READ_FUNCTION,0,i,&SerialNum[portnum][0]),(int)((i+1)*3-
1),&sendpacket[1]);

   // send/recieve the transfer buffer
   if (owBlock_DS9097U(portnum,TRUE,sendpacket,sendlen))
   {
      // check results to see if it was a success
      for (i = 0; i < 192; i += 3)
      {
         tst = (bitacc(READ_FUNCTION,0,i,&sendpacket[1]) << 1) |
               bitacc(READ_FUNCTION,0,(int)(i+1),&sendpacket[1]);

         s = bitacc(READ_FUNCTION,0,cnt++,&SerialNum[portnum][0]);

         if (tst == 0x03)  // no device on line
         {
            goodbits = 0;    // number of good bits set to zero
            break;     // quit
```

```
         }

         if (((s == 0x01) && (tst == 0x02)) ||
             ((s == 0x00) && (tst == 0x01))    )  // correct bit
            goodbits++;  // count as a good bit
      }

      // check too see if there were enough good bits to be successful
      if (goodbits >= 8)
         return TRUE;
   }
   else
      OWERROR(OWERROR_BLOCK_FAILED);

   // block fail or device not present
   return FALSE;
}

//--------------------------------------------------------------------------
// Perform a overdrive MATCH command to select the 1-Wire device with
// the address in the ID data register.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
// Returns:  TRUE: If the device is present on the 1-Wire Net and
//                 can do overdrive then the device is selected.
//           FALSE: Device is not present or not capable of overdrive.
//
//  *Note: This function could be converted to send DS2480
//         commands in one packet.
//
SMALLINT owOverdriveAccess_DS9097U(int portnum)
{
   uchar sendpacket[8];
   uchar i, bad_echo = FALSE;

   // make sure normal level
   owLevel_DS9097U(portnum,MODE_NORMAL);

   // force to normal communication speed
   owSpeed_DS9097U(portnum,MODE_NORMAL);

   // call the 1-Wire Net reset function
   if (owTouchReset_DS9097U(portnum))
   {
      // send the match command 0x69
      if (owWriteByte_DS9097U(portnum,0x69))
      {
         // switch to overdrive communication speed
         owSpeed_DS9097U(portnum,MODE_OVERDRIVE);

         // create a buffer to use with block function
         // Serial Number
         for (i = 0; i < 8; i++)
            sendpacket[i] = SerialNum[portnum][i];

         // send/recieve the transfer buffer
         if (owBlock_DS9097U(portnum,FALSE,sendpacket,8))
         {
            // verify that the echo of the writes was correct
            for (i = 0; i < 8; i++)
               if (sendpacket[i] != SerialNum[portnum][i])
                  bad_echo = TRUE;
            // if echo ok then success
            if (!bad_echo)
               return TRUE;
            else
               OWERROR(OWERROR_WRITE_VERIFY_FAILED);
         }
         else
            OWERROR(OWERROR_BLOCK_FAILED);
      }
      else
         OWERROR(OWERROR_WRITE_BYTE_FAILED);
   }
```

```
   else
      OWERROR(OWERROR_NO_DEVICES_ON_NET);

   // failure, force back to normal communication speed
   owSpeed_DS9097U(portnum,MODE_NORMAL);

   return FALSE;
}

//--------------------------------------------------------------------------
// Bit utility to read and write a bit in the buffer 'buf'.
//
// 'op'    - operation (1) to set and (0) to read
// 'state' - set (1) or clear (0) if operation is write (1)
// 'loc'   - bit number location to read or write
// 'buf'   - pointer to array of bytes that contains the bit
//           to read or write
//
// Returns: 1   if operation is set (1)
//          0/1 state of bit number 'loc' if operation is reading
//
static SMALLINT bitacc(SMALLINT op, SMALLINT state, SMALLINT loc, uchar *buf)
{
   SMALLINT nbyt,nbit;

   nbyt = (loc / 8);
   nbit = loc - (nbyt * 8);

   if (op == WRITE_FUNCTION)
   {
      if (state)
         buf[nbyt] |= (0x01 << nbit);
      else
         buf[nbyt] &= ~(0x01 << nbit);

      return 1;
   }
   else
      return ((buf[nbyt] >> nbit) & 0x01);
}
```

Συναρτήσεις επιπέδου μεταφοράς

Βιβλιοθήκη που καλεί τις επιμέρους συναρτήσεις των προσαρμογέων για το επίπεδο μεταφοράς

Multiran.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
```

```
//--------------------------------------------------------------------------
//
//  multitran.c - Wrapper to hook all adapter types in the 1-Wire Public
//                Domain API for transport functions.
//
//  Version: 3.01
//
//  name on line above changed to correct multitran.c instead of multinet.c
//  added support ofr the http server HA7Net.com
//  line added in function owBlock: "case HA7Net:........"
//

#include "mownet.h"

extern SMALLINT default_type;

//--------------------------------------------------------------------------
// The 'owBlock' transfers a block of data to and from the
// 1-Wire Net with an optional reset at the begining of communication.
// The result is returned in the same buffer.
//
// 'do_reset' - cause a owTouchReset to occure at the begining of
//              communication TRUE(1) or not FALSE(0)
// 'tran_buf' - pointer to a block of unsigned
//              chars of length 'TranferLength' that will be sent
//              to the 1-Wire Net
// 'tran_len' - length in bytes to transfer
// Supported devices: all
//
// Returns:   TRUE (1) : The optional reset returned a valid
//                       presence (do_reset == TRUE) or there
//                       was no reset required.
//            FALSE (0): The reset did not return a valid prsence
//                       (do_reset == TRUE).
//
//  The maximum tran_len is 64
//
SMALLINT owBlock(int portnum, SMALLINT do_reset, uchar *tran_buf, SMALLINT tran_len)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owBlock_DS9490(portnum & 0xFF, do_reset, tran_buf, tran_len);
      case DS1410E: return owBlock_DS1410E(portnum & 0xFF, do_reset, tran_buf,
tran_len);
      default:
      case DS9097U: return owBlock_DS9097U(portnum & 0xFF, do_reset, tran_buf,
tran_len);
      case HA7Net:  return owBlock_HA7Net(portnum & 0xFF, do_reset, tran_buf, tran_len);
   };
}

//--------------------------------------------------------------------------
// Write a byte to an EPROM 1-Wire device.
//
// Supported devices: crc_type=0(CRC8)
//                        DS1982
//                    crc_type=1(CRC16)
//                        DS1985, DS1986, DS2407
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'write_byte' - byte to program
// 'addr'       - address of byte to program
// 'write_cmd'  - command used to write (0x0F reg mem, 0x55 status)
// 'crc_type'   - CRC used (0 CRC8, 1 CRC16)
// 'do_access'  - Flag to access device for each byte
//                (0 skip access, 1 do the access)
//                WARNING, only use do_access=0 if programing the NEXT
//                byte immediatly after the previous byte.
//
// Returns: >=0   success, this is the resulting byte from the program
//                effort
//          -1    error, device not connected or program pulse voltage
//                not available
//
SMALLINT owProgramByte(int portnum, SMALLINT write_byte, int addr, SMALLINT write_cmd,
```

```
                        SMALLINT crc_type, SMALLINT do_access)
{
   switch ((default_type) ? default_type : (portnum >> 8) & 0xFF)
   {
      case DS9490:  return owProgramByte_DS9490(portnum & 0xFF, write_byte, addr,
write_cmd, crc_type, do_access);
      case DS1410E: return owProgramByte_DS1410E(portnum & 0xFF, write_byte, addr,
write_cmd, crc_type, do_access);
      default:
      case DS9097U: return owProgramByte_DS9097U(portnum & 0xFF, write_byte, addr,
write_cmd, crc_type, do_access);
   };
}
```

Επίπεδο μεταφοράς για τον USB μετατροπέα
musbwtrn.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  usbwtran.C - USB (DS2490) transport functions for 1-Wire Public Domain
//
//  Version: 3.00
//

#include "mownet.h"
#include "ds2490.h"
#include <windows.h>

// handles to USB ports
extern HANDLE usbhnd[MAX_PORTNUM];

//--------------------------------------------------------------------------
// The 'owBlock' transfers a block of data to and from the
// 1-Wire Net with an optional reset at the begining of communication.
// The result is returned in the same buffer.
//
// 'do_reset' - cause a owTouchReset to occure at the begining of
//              communication TRUE(1) or not FALSE(0)
// 'tran_buf' - pointer to a block of unsigned
//              chars of length 'TranferLength' that will be sent
//              to the 1-Wire Net
// 'tran_len' - length in bytes to transfer
// Supported devices: all
//
// Returns:   TRUE (1) : The optional reset returned a valid
//                       presence (do_reset == TRUE) or there
//                       was no reset required.
//            FALSE (0): The reset did not return a valid prsence
//                       (do_reset == TRUE).
//
```

```
//  The maximum tran_length is (160)
//
SMALLINT owBlock_DS9490(int portnum, SMALLINT do_reset, uchar *tran_buf, SMALLINT
tran_len)
{
   SETUP_PACKET setup;
   ULONG nOutput = 0;
        STATUS_PACKET status;
        BOOL    command_issued = FALSE;
        WORD    bytes_index=0;
        WORD    nBytesLeft = tran_len;
        WORD    block_len;
        WORD    block_write;
        WORD    loopcount;

   // check if need to do a owTouchReset first
   if (do_reset)
   {
      if (!owTouchReset_DS9490(portnum))
         return FALSE;
   }

   // check for a block too big
   if (tran_len > 192)
   {
      OWERROR(OWERROR_BLOCK_TOO_BIG);
      return FALSE;
   }

   // loop to write in blocks of 64
        while (nBytesLeft > 0)
        {
      // Put the bytes of data in EP2 (Warning: do not overfill EP2 FIFO)
      // Let's begin by sending 64 bytes to EP2, then send the vendor command (EP0),
      // followed by additional data (while monitoring EP2 FIFO).
      if (nBytesLeft > 64)
         block_len = 64;
      else
         block_len = nBytesLeft;

      // fill EP2
      block_write = block_len;
      if (!DS2490Write(usbhnd[portnum], &tran_buf[bytes_index], &block_write))
      {
         OWERROR(OWERROR_ADAPTER_ERROR);
         AdapterRecover(portnum);
         return FALSE;
      }

      // check for write incomplete
      if (block_write != block_len)
      {
         OWERROR(OWERROR_ADAPTER_ERROR);
         AdapterRecover(portnum);
         return FALSE;
      }

      // check for one time send of block IO command
      if (!command_issued)
      {
         // send vendor command
         setup.RequestTypeReservedBits = 0x40;
         setup.Request = COMM_CMD;
         setup.Value = COMM_BLOCK_IO | COMM_IM | COMM_F;
         setup.Index = tran_len;
         setup.Length = 0;
         setup.DataOut = FALSE;
         // call the driver
         if (!DeviceIoControl(usbhnd[portnum],
                                              DS2490_IOCTL_VENDOR,
                                              &setup,
                                              sizeof(SETUP_PACKET),
                                              NULL,
                                              0,
                                              &nOutput,
                                              NULL))
```

```
         {
            // failure
            OWERROR(OWERROR_ADAPTER_ERROR);
            AdapterRecover(portnum);
            return FALSE;
         }

         command_issued = TRUE;
      }

      // check whether device idle, block_len bytes ready for read
      // and wait for byte's arrival if necessary
      loopcount = 200;
      do
      {
         if (!DS2490GetStatus(usbhnd[portnum], &status, NULL))
         {
            AdapterRecover(portnum);
              return FALSE;
         }
         msDelay_DS9490(1);
      }
      while (!(status.StatusFlags & STATUSFLAGS_IDLE)
                  && (status.ReadBufferStatus < block_len)
                  && (loopcount-- > 0));

      // check for timeout
      if ((loopcount == 0) && (status.ReadBufferStatus < block_len))
      {
         OWERROR(OWERROR_ADAPTER_ERROR);
         AdapterRecover(portnum);
         return FALSE;
      }

      // read response on previous write
      block_write = block_len;
      if (!DS2490Read(usbhnd[portnum], &tran_buf[bytes_index], &block_write))
      {
         AdapterRecover(portnum);
         return FALSE;
      }

      // check for read incomplete
      if (block_write != block_len)
      {
         OWERROR(OWERROR_ADAPTER_ERROR);
         AdapterRecover(portnum);
         return FALSE;
      }

      // adjust buffer's pointers
      nBytesLeft =  nBytesLeft - block_len;
      bytes_index = bytes_index + block_len;
        }

   return TRUE;
}

//--------------------------------------------------------------------------
// Write a byte to an EPROM 1-Wire device.
//
// Supported devices: crc_type=0(CRC8)
//                        DS1982
//                    crc_type=1(CRC16)
//                        DS1985, DS1986, DS2407
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'write_byte' - byte to program
// 'addr'       - address of byte to program
// 'write_cmd'  - command used to write (0x0F reg mem, 0x55 status)
// 'crc_type'   - CRC used (0 CRC8, 1 CRC16)
// 'do_access'  - Flag to access device for each byte
//                (0 skip access, 1 do the access)
//                WARNING, only use do_access=0 if programing the NEXT
//                byte immediatly after the previous byte.
```

```
//
// Returns: >=0   success, this is the resulting byte from the program
//                effort
//          -1    error, device not connected or program pulse voltage
//                not available
//
SMALLINT owProgramByte_DS9490(int portnum, SMALLINT write_byte, int addr, SMALLINT
write_cmd,
                      SMALLINT crc_type, SMALLINT do_access)
{
   ushort lastcrc16;
   uchar lastcrc8;

   // optionally access the device
   if (do_access)
   {
      if (!owAccess_DS9490(portnum))
      {
         OWERROR(OWERROR_ACCESS_FAILED);
         return -1;
      }

      // send the write command
      if (!owWriteByte_DS9490(portnum,write_cmd))
      {
         OWERROR(OWERROR_WRITE_BYTE_FAILED);
         return -1;
      }

      // send the address
      if (!owWriteByte_DS9490(portnum,addr & 0xFF) || !owWriteByte_DS9490(portnum,addr
>> 8))
      {
         OWERROR(OWERROR_WRITE_BYTE_FAILED);
         return -1;
      }
   }

   // send the data to write
   if (!owWriteByte_DS9490(portnum,write_byte))
   {
      OWERROR(OWERROR_WRITE_BYTE_FAILED);
      return -1;
   }

   // read the CRC
   if (crc_type == 0)
   {
      // calculate CRC8
      if (do_access)
      {
         setcrc8(portnum,0);
         docrc8(portnum,(uchar)write_cmd);
         docrc8(portnum,(uchar)(addr & 0xFF));
         docrc8(portnum,(uchar)(addr >> 8));
      }
      else
         setcrc8(portnum,(uchar)(addr & 0xFF));

      docrc8(portnum,(uchar)write_byte);
      // read and calculate the read crc
      lastcrc8 = docrc8(portnum,(uchar)owReadByte_DS9490(portnum));
      // crc should now be 0x00
      if (lastcrc8 != 0)
      {
         OWERROR(OWERROR_CRC_FAILED);
         return -1;
      }
   }
   else
   {
      // CRC16
      if (do_access)
      {
         setcrc16(portnum,0);
         docrc16(portnum,(ushort)write_cmd);
```

```
         docrc16(portnum,(ushort)(addr & 0xFF));
         docrc16(portnum,(ushort)(addr >> 8));
      }
      else
         setcrc16(portnum,(ushort)addr);
      docrc16(portnum,(ushort)write_byte);
      // read and calculate the read crc
      docrc16(portnum,(ushort)owReadByte_DS9490(portnum));
      lastcrc16 = docrc16(portnum,(ushort)owReadByte_DS9490(portnum));
      // crc should now be 0xB001
      if (lastcrc16 != 0xB001)
         return -1;
   }

   // send the program pulse
   if (!owProgramPulse_DS9490(portnum))
   {
      OWERROR(OWERROR_PROGRAM_PULSE_FAILED);
      return -1;
   }

   // read back and return the resulting byte
   return owReadByte_DS9490(portnum);
}
```

Επίπεδο μεταφοράς για τον παράλληλο μετατροπέα
Mowtran.c

```
//--------------------------------------------------------------------------
// Copyright (C) 1999 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  owTran.C - Transport functions for 1-Wire devices.
//
//  Version: 2.01
//
//  History: 1.03 -> 2.00  Changed 'MLan' to 'ow'. Added support for
//                         multiple ports.
//           2.00 -> 2.01  Added support for owError library
//

#include "mownet.h"

//--------------------------------------------------------------------------
// The 'owBlock' transfers a block of data to and from the
// 1-Wire Net with an optional reset at the begining of communication.
// The result is returned in the same buffer.
//
// 'do_reset' - cause a owTouchReset to occure at the begining of
//              communication TRUE(1) or not FALSE(0)
// 'tran_buf' - pointer to a block of unsigned
//              chars of length 'TranferLength' that will be sent
```

```
//              to the 1-Wire Net
// 'tran_len' - length in bytes to transfer
// Supported devices: all
//
// Returns:   TRUE (1) : The optional reset returned a valid
//                       presence (do_reset == TRUE) or there
//                       was no reset required.
//            FALSE (0): The reset did not return a valid prsence
//                       (do_reset == TRUE).
//
//  The maximum tran_len is 160
//
SMALLINT owBlock_DS1410E(int portnum, SMALLINT do_reset, uchar *tran_buf, SMALLINT
tran_len)
{
   uchar i;

   // check for a block too big
   if (tran_len > 160)
   {
      OWERROR(OWERROR_BLOCK_TOO_BIG);
      return FALSE;
   }

   // check if need to do a owTouchReset first
   if (do_reset)
   {
      if (!owTouchReset_DS1410E(portnum))
      {
         OWERROR(OWERROR_NO_DEVICES_ON_NET);
         return FALSE;
      }
   }

   // send and receive the buffer
   for (i = 0; i < tran_len; i++)
      tran_buf[i] = (uchar)owTouchByte_DS1410E(portnum,tran_buf[i]);

   return TRUE;
}

//--------------------------------------------------------------------------
// Write a byte to an EPROM 1-Wire device.
//
// Supported devices: crc_type=0(CRC8)
//                        DS1982
//                    crc_type=1(CRC16)
//                        DS1985, DS1986, DS2407
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'write_byte' - byte to program
// 'addr'       - address of byte to program
// 'write_cmd'  - command used to write (0x0F reg mem, 0x55 status)
// 'crc_type'   - CRC used (0 CRC8, 1 CRC16)
// 'do_access'  - Flag to access device for each byte
//                (0 skip access, 1 do the access)
//                WARNING, only use do_access=0 if programing the NEXT
//                byte immediatly after the previous byte.
//
// Returns: >=0   success, this is the resulting byte from the program
//                effort
//          -1    error, device not connected or program pulse voltage
//                not available
//
SMALLINT owProgramByte_DS1410E(int portnum, SMALLINT write_byte, int addr, SMALLINT
write_cmd,
                       SMALLINT crc_type, SMALLINT do_access)
{
   ushort lastcrc16;
   uchar lastcrc8;

   // optionally access the device
   if (do_access)
   {
      if (!owAccess_DS1410E(portnum))
```

```
      {
         OWERROR(OWERROR_ACCESS_FAILED);
         return -1;
      }

      // send the write command
      if (!owWriteByte_DS1410E(portnum,write_cmd))
      {
         OWERROR(OWERROR_WRITE_BYTE_FAILED);
         return -1;
      }

      // send the address
      if (!owWriteByte_DS1410E(portnum,addr & 0xFF) || !owWriteByte_DS1410E(portnum,addr
>> 8))
      {
         OWERROR(OWERROR_WRITE_BYTE_FAILED);
         return -1;
      }
   }

   // send the data to write
   if (!owWriteByte_DS1410E(portnum,write_byte))
   {
      OWERROR(OWERROR_WRITE_BYTE_FAILED);
      return -1;
   }

   // read the CRC
   if (crc_type == 0)
   {
      // calculate CRC8
      if (do_access)
      {
         setcrc8(portnum,0);
         docrc8(portnum,(uchar)write_cmd);
         docrc8(portnum,(uchar)(addr & 0xFF));
         docrc8(portnum,(uchar)(addr >> 8));
      }
      else
         setcrc8(portnum,(uchar)(addr & 0xFF));

      docrc8(portnum,(uchar)write_byte);
      // read and calculate the read crc
      lastcrc8 = docrc8(portnum,(uchar)owReadByte_DS1410E(portnum));
      // crc should now be 0x00
      if (lastcrc8 != 0)
      {
         OWERROR(OWERROR_CRC_FAILED);
         return -1;
      }
   }
   else
   {
      // CRC16
      if (do_access)
      {
         setcrc16(portnum,0);
         docrc16(portnum,(ushort)write_cmd);
         docrc16(portnum,(ushort)(addr & 0xFF));
         docrc16(portnum,(ushort)(addr >> 8));
      }
      else
         setcrc16(portnum,(ushort)addr);
      docrc16(portnum,(ushort)write_byte);
      // read and calculate the read crc
      docrc16(portnum,(ushort)owReadByte_DS1410E(portnum));
      lastcrc16 = docrc16(portnum,(ushort)owReadByte_DS1410E(portnum));
      // crc should now be 0xB001
      if (lastcrc16 != 0xB001)
         return -1;
   }

   // send the program pulse
   if (!owProgramPulse_DS1410E(portnum))
   {
```

```
      OWERROR(OWERROR_PROGRAM_PULSE_FAILED);
      return -1;
   }

   // read back and return the resulting byte
   return owReadByte_DS1410E(portnum);
}
```

Επίπεδο μεταφοράς για τον σειριακό μετατροπέα
Mowtrnu.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  owTranU.C - Transport functions for 1-Wire Net
//              using the DS2480B (U) serial interface chip.
//
//  Version: 2.01
//
//  History: 1.02 -> 1.03  Removed caps in #includes for Linux capatibility
//           1.03 -> 2.00  Changed 'MLan' to 'ow'. Added support for
//                         multiple ports.
//           2.00 -> 2.01  Added support for owError library
//           2.01 -> 2.10  Added SMALLINT for small processors and error
//                         handling plus the raw memory utilities.
//           2.10 -> 3.00 Added memory bank functionality
//                        Added file I/O operations
//

#include "mownet.h"
#include "mds2480.h"
// external defined in ds2480ut.c
extern SMALLINT UBaud[MAX_PORTNUM];
extern SMALLINT UMode[MAX_PORTNUM];
extern SMALLINT USpeed[MAX_PORTNUM];
extern uchar SerialNum[MAX_PORTNUM][8];

// local static functions
static SMALLINT Write_Scratchpad(int,uchar *,int,SMALLINT);
static SMALLINT Copy_Scratchpad(int,int,SMALLINT);

//--------------------------------------------------------------------------
// The 'owBlock' transfers a block of data to and from the
// 1-Wire Net with an optional reset at the begining of communication.
// The result is returned in the same buffer.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number is provided to
//              indicate the symbolic port number.
// 'do_reset' - cause a owTouchReset to occure at the begining of
//              communication TRUE(1) or not FALSE(0)
// 'tran_buf' - pointer to a block of unsigned
//              chars of length 'tran_len' that will be sent
```

```
//              to the 1-Wire Net
// 'tran_len' - length in bytes to transfer

// Supported devices: all
//
// Returns:   TRUE (1) : The optional reset returned a valid
//                       presence (do_reset == TRUE) or there
//                       was no reset required.
//            FALSE (0): The reset did not return a valid prsence
//                       (do_reset == TRUE).
//
//  The maximum tran_length is (160)
//
SMALLINT owBlock_DS9097U(int portnum, SMALLINT do_reset, uchar *tran_buf, SMALLINT
tran_len)
{
   uchar sendpacket[320];
   uchar sendlen=0,pos,i;

   // check for a block too big
   if (tran_len > 160)
   {
      OWERROR(OWERROR_BLOCK_TOO_BIG);
      return FALSE;
   }

   // check if need to do a owTouchReset first
   if (do_reset)
   {
      if (!owTouchReset_DS9097U(portnum))
      {
         OWERROR(OWERROR_NO_DEVICES_ON_NET);
         return FALSE;
      }
   }

   // construct the packet to send to the DS2480
   // check if correct mode
   if (UMode[portnum] != MODSEL_DATA)
   {
      UMode[portnum] = MODSEL_DATA;
      sendpacket[sendlen++] = MODE_DATA;
   }

   // add the bytes to send
   pos = sendlen;
   for (i = 0; i < tran_len; i++)
   {
      sendpacket[sendlen++] = tran_buf[i];

      // check for duplication of data that looks like COMMAND mode
      if (tran_buf[i] == MODE_COMMAND)
         sendpacket[sendlen++] = tran_buf[i];
   }

   // flush the buffers
   FlushCOM(portnum);

   // send the packet
   if (WriteCOM(portnum,sendlen,sendpacket))
   {
      // read back the response
      if (ReadCOM(portnum,tran_len,tran_buf) == tran_len)
         return TRUE;
      else
         OWERROR(OWERROR_READCOM_FAILED);
   }
   else
      OWERROR(OWERROR_WRITECOM_FAILED);

   // an error occured so re-sync with DS2480
   DS2480Detect(portnum);

   return FALSE;
}
```

```
//--------------------------------------------------------------------------
// Read a Universal Data Packet from a standard NVRAM iButton
// and return it in the provided buffer. The page that the
// packet resides on is 'start_page'.  Note that this function is limited
// to single page packets. The buffer 'read_buf' must be at least
// 29 bytes long.
//
// The Universal Data Packet always start on page boundaries but
// can end anywhere.  The length is the number of data bytes not
// including the length byte and the CRC16 bytes.  There is one
// length byte. The CRC16 is first initialized to the starting
// page number.  This provides a check to verify the page that
// was intended is being read.  The CRC16 is then calculated over
// the length and data bytes.  The CRC16 is then inverted and stored
// low byte first followed by the high byte.
//
// Supported devices: DS1992, DS1993, DS1994, DS1995, DS1996, DS1982,
//                    DS1985, DS1986, DS2407, and DS1971.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'do_access'  - flag to indicate if an 'owAccess' should be
//                peformed at the begining of the read.  This may
//                be FALSE (0) if the previous call was to read the
//                previous page (start_page-1).
// 'start_page' - page number to start the read from
// 'read_buf'   - pointer to a location to store the data read
//
// Returns:  >=0 success, number of data bytes in the buffer
//           -1  failed to read a valid UDP
//
//
SMALLINT owReadPacketStd(int portnum, SMALLINT do_access, int start_page, uchar
*read_buf)
{
   uchar i,length,sendlen=0,head_len=0;
   uchar sendpacket[50];
   ushort lastcrc16;

   // check if access header is done
   // (only use if in sequention read with one access at begining)
   if (do_access)
   {
      // match command
      sendpacket[sendlen++] = 0x55;
      for (i = 0; i < 8; i++)
         sendpacket[sendlen++] = SerialNum[portnum][i];
      // read memory command
      sendpacket[sendlen++] = 0xF0;
      // write the target address
      sendpacket[sendlen++] = ((start_page << 5) & 0xFF);
      sendpacket[sendlen++] = (start_page >> 3);
      // check for DS1982 exception (redirection byte)
      if (SerialNum[portnum][0] == 0x09)
         sendpacket[sendlen++] = 0xFF;
      // record the header length
      head_len = sendlen;
   }
   // read the entire page length byte
   for (i = 0; i < 32; i++)
      sendpacket[sendlen++] = 0xFF;

   // send/recieve the transfer buffer
   if (owBlock_DS9097U(portnum,do_access,sendpacket,sendlen))
   {
      // seed crc with page number
      setcrc16(portnum,(ushort)start_page);

      // attempt to read UDP from sendpacket
      length = sendpacket[head_len];
      docrc16(portnum,(ushort)length);

      // verify length is not too large
      if (length <= 29)
      {
         // loop to read packet including CRC
```

```
         for (i = 0; i < length; i++)
         {
             read_buf[i] = sendpacket[i+1+head_len];
             docrc16(portnum,read_buf[i]);
         }

         // read and compute the CRC16
         docrc16(portnum,sendpacket[i+1+head_len]);
         lastcrc16 = docrc16(portnum,sendpacket[i+2+head_len]);

         // verify the CRC16 is correct
         if (lastcrc16 == 0xB001)
           return length;        // return number of byte in record
         else
            OWERROR(OWERROR_CRC_FAILED);
      }
      else
         OWERROR(OWERROR_INCORRECT_CRC_LENGTH);
   }
   else
      OWERROR(OWERROR_BLOCK_FAILED);

   // failed block or incorrect CRC
   return -1;
}

//--------------------------------------------------------------------------
// Write a Universal Data Packet onto a standard NVRAM 1-Wire device
// on page 'start_page'.  This function is limited to UDPs that
// fit on one page.  The data to write is provided as a buffer
// 'write_buf' with a length 'write_len'.
//
// The Universal Data Packet always start on page boundaries but
// can end anywhere.  The length is the number of data bytes not
// including the length byte and the CRC16 bytes.  There is one
// length byte. The CRC16 is first initialized to the starting
// page number.  This provides a check to verify the page that
// was intended is being read.  The CRC16 is then calculated over
// the length and data bytes.  The CRC16 is then inverted and stored
// low byte first followed by the high byte.
//
// Supported devices: is_eprom=0
//                        DS1992, DS1993, DS1994, DS1995, DS1996
//                    is_eprom=1, crc_type=0(CRC8)
//                        DS1982
//                    is_eprom=1, crc_type=1(CRC16)
//                        DS1985, DS1986, DS2407
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'start_page' - page number to write packet to
// 'write_buf'  - pointer to buffer containing data to write
// 'write_len'  - number of data byte in write_buf
// 'is_eprom'   - flag set if device is an EPROM (1 EPROM, 0 NVRAM)
// 'crc_type'   - if is_eprom=1 then indicates CRC type
//                (0 CRC8, 1 CRC16)
//
// Returns: TRUE(1)  success, packet written
//          FALSE(0) failure to write, contact lost or device locked
//
SMALLINT owWritePacketStd(int portnum, int start_page, uchar *write_buf,
                          SMALLINT write_len, SMALLINT is_eprom, SMALLINT crc_type)
{
   uchar construct_buffer[32];
   uchar i,buffer_cnt=0,start_address,do_access;
   ushort lastcrc16;

   // check to see if data too long to fit on device
   if (write_len > 29)
     return FALSE;

   // seed crc with page number
   setcrc16(portnum,(ushort)start_page);

   // set length byte
   construct_buffer[buffer_cnt++] = (uchar)(write_len);
```

```
   docrc16(portnum,(ushort)write_len);

   // fill in the data to write
   for (i = 0; i < write_len; i++)
   {
     lastcrc16 = docrc16(portnum,write_buf[i]);
     construct_buffer[buffer_cnt++] = write_buf[i];
   }

   // add the crc
   construct_buffer[buffer_cnt++] = (uchar)(~(lastcrc16 & 0xFF));
   construct_buffer[buffer_cnt++] = (uchar)(~((lastcrc16 & 0xFF00) >> 8));

   // check if not EPROM
   if (!is_eprom)
   {
      // write the page
      if (!Write_Scratchpad(portnum,construct_buffer,start_page,buffer_cnt))
      {
         OWERROR(OWERROR_WRITE_SCRATCHPAD_FAILED);
         return FALSE;
      }

      // copy the scratchpad
      if (!Copy_Scratchpad(portnum,start_page,buffer_cnt))
      {
         OWERROR(OWERROR_COPY_SCRATCHPAD_FAILED);
         return FALSE;
      }

      // copy scratch pad was good then success
      return TRUE;
   }
   // is EPROM
   else
   {
      // calculate the start address
      start_address = ((start_page >> 3) << 8) | ((start_page << 5) & 0xFF);
      do_access = TRUE;
      // loop to program each byte
      for (i = 0; i < buffer_cnt; i++)
      {
         if (owProgramByte_DS9097U(portnum,construct_buffer[i], start_address + i,
             0x0F, crc_type, do_access) != construct_buffer[i])
         {
            OWERROR(OWERROR_PROGRAM_BYTE_FAILED);
            return FALSE;
         }
         do_access = FALSE;
      }
      return TRUE;
   }
}

//--------------------------------------------------------------------------
// Write a byte to an EPROM 1-Wire device.
//
// Supported devices: crc_type=0(CRC8)
//                        DS1982
//                    crc_type=1(CRC16)
//                        DS1985, DS1986, DS2407
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'write_byte' - byte to program
// 'addr'       - address of byte to program
// 'write_cmd'  - command used to write (0x0F reg mem, 0x55 status)
// 'crc_type'   - CRC used (0 CRC8, 1 CRC16)
// 'do_access'  - Flag to access device for each byte
//                (0 skip access, 1 do the access)
//                WARNING, only use do_access=0 if programing the NEXT
//                byte immediatly after the previous byte.
//
// Returns: >=0   success, this is the resulting byte from the program
//                effort
//          -1    error, device not connected or program pulse voltage
```

```
//               not available
//
SMALLINT owProgramByte_DS9097U(int portnum, SMALLINT write_byte, int addr, SMALLINT
write_cmd,
                   SMALLINT crc_type, SMALLINT do_access)
{
   ushort lastcrc16;
   uchar lastcrc8;

   // optionally access the device
   if (do_access)
   {
      if (!owAccess_DS9097U(portnum))
      {
         OWERROR(OWERROR_ACCESS_FAILED);
         return -1;
      }

      // send the write command
      if (!owWriteByte_DS9097U(portnum,write_cmd))
      {
         OWERROR(OWERROR_WRITE_BYTE_FAILED);
         return -1;
      }

      // send the address
      if (!owWriteByte_DS9097U(portnum,addr & 0xFF) || !owWriteByte_DS9097U(portnum,addr
>> 8))
      {
         OWERROR(OWERROR_WRITE_BYTE_FAILED);
         return -1;
      }
   }

   // send the data to write
   if (!owWriteByte_DS9097U(portnum,write_byte))
   {
      OWERROR(OWERROR_WRITE_BYTE_FAILED);
      return -1;
   }

   // read the CRC
   if (crc_type == 0)
   {
      // calculate CRC8
      if (do_access)
      {
         setcrc8(portnum,0);
         docrc8(portnum,(uchar)write_cmd);
         docrc8(portnum,(uchar)(addr & 0xFF));
         docrc8(portnum,(uchar)(addr >> 8));
      }
      else
         setcrc8(portnum,(uchar)(addr & 0xFF));

      docrc8(portnum,(uchar)write_byte);
      // read and calculate the read crc
      lastcrc8 = docrc8(portnum,(uchar)owReadByte_DS9097U(portnum));
      // crc should now be 0x00
      if (lastcrc8 != 0)
      {
         OWERROR(OWERROR_CRC_FAILED);
         return -1;
      }
   }
   else
   {
      // CRC16
      if (do_access)
      {
         setcrc16(portnum,0);
         docrc16(portnum,(ushort)write_cmd);
         docrc16(portnum,(ushort)(addr & 0xFF));
         docrc16(portnum,(ushort)(addr >> 8));
      }
      else
```

```
         setcrc16(portnum,(ushort)addr);
         docrc16(portnum,(ushort)write_byte);
         // read and calculate the read crc
         docrc16(portnum,(ushort)owReadByte_DS9097U(portnum));
         lastcrc16 = docrc16(portnum,(ushort)owReadByte_DS9097U(portnum));
         // crc should now be 0xB001
         if (lastcrc16 != 0xB001)
         {
            OWERROR(OWERROR_CRC_FAILED);
            return -1;
         }
      }

      // send the program pulse
      if (!owProgramPulse_DS9097U(portnum))
      {
         OWERROR(OWERROR_PROGRAM_PULSE_FAILED);
         return -1;
      }

      // read back and return the resulting byte
      return owReadByte_DS9097U(portnum);
   }

   //--------------------------------------------------------------------------
   // Write the scratchpad of a standard NVRam device such as the DS1992,3,4
   // and verify its contents.
   //
   // 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
   //                indicate the symbolic port number.
   // 'write_buf'  - pointer to buffer containing data to write
   // 'start_page'     - page number to write packet to
   // 'write_len'  - number of data byte in write_buf
   //
   // Returns: TRUE(1)  success, the data was written and verified
   //          FALSE(0) failure, the data could not be written
   //
   //
   SMALLINT Write_Scratchpad(int portnum, uchar *write_buf, int start_page, SMALLINT
   write_len)
   {
      uchar i,sendlen=0;
      uchar sendpacket[50];

      // match command
      sendpacket[sendlen++] = 0x55;
      for (i = 0; i < 8; i++)
         sendpacket[sendlen++] = SerialNum[portnum][i];
      // write scratchpad command
      sendpacket[sendlen++] = 0x0F;
      // write the target address
      sendpacket[sendlen++] = ((start_page << 5) & 0xFF);
      sendpacket[sendlen++] = (start_page >> 3);

      // write packet bytes
      for (i = 0; i < write_len; i++)
         sendpacket[sendlen++] = write_buf[i];

      // send/recieve the transfer buffer
      if (owBlock_DS9097U(portnum,TRUE,sendpacket,sendlen))
      {
         // now attempt to read back to check
         sendlen = 0;
         // match command
         sendpacket[sendlen++] = 0x55;
         for (i = 0; i < 8; i++)
            sendpacket[sendlen++] = SerialNum[portnum][i];
         // read scratchpad command
         sendpacket[sendlen++] = 0xAA;
         // read the target address, offset and data
         for (i = 0; i < (write_len + 3); i++)
            sendpacket[sendlen++] = 0xFF;

         // send/recieve the transfer buffer
         if (owBlock_DS9097U(portnum,TRUE,sendpacket,sendlen))
         {
```

```
         // check address and offset of scratchpad read
         if ((sendpacket[10] != ((start_page << 5) & 0xFF)) ||
             (sendpacket[11] != (start_page >> 3)) ||
             (sendpacket[12] != (write_len - 1)))
         {
            OWERROR(OWERROR_READ_VERIFY_FAILED);
            return FALSE;
         }

         // verify each data byte
         for (i = 0; i < write_len; i++)
            if (sendpacket[i+13] != write_buf[i])
            {
               OWERROR(OWERROR_WRITE_VERIFY_FAILED);
               return FALSE;
            }

         // must have verified
         return TRUE;
      }
      else
         OWERROR(OWERROR_BLOCK_FAILED);
   }
   else
      OWERROR(OWERROR_BLOCK_FAILED);

   // failed a block tranfer
   return FALSE;
}

//----------------------------------------------------------------------------
// Copy the contents of the scratchpad to its intended nv ram page.  The
// page and length of the data is needed to build the authorization bytes
// to copy.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'start_page' - page number to write packet to
// 'write_len'  - number of data bytes that are being copied
//
// Returns: TRUE(1)  success
//          FALSE(0) failure
//
SMALLINT Copy_Scratchpad(int portnum, int start_page, SMALLINT write_len)
{
   uchar i,sendlen=0;
   uchar sendpacket[50];

   // match command
   sendpacket[sendlen++] = 0x55;
   for (i = 0; i < 8; i++)
      sendpacket[sendlen++] = SerialNum[portnum][i];
   // copy scratchpad command
   sendpacket[sendlen++] = 0x55;
   // write the target address
   sendpacket[sendlen++] = ((start_page << 5) & 0xFF);
   sendpacket[sendlen++] = (start_page >> 3);
   sendpacket[sendlen++] = write_len - 1;
   // read copy result
   sendpacket[sendlen++] = 0xFF;

   // send/recieve the transfer buffer
   if (owBlock_DS9097U(portnum,TRUE,sendpacket,sendlen))
   {
      // check address and offset of scratchpad read
      if ((sendpacket[10] != ((start_page << 5) & 0xFF)) ||
          (sendpacket[11] != (start_page >> 3)) ||
          (sendpacket[12] != (write_len - 1)) ||
          (sendpacket[13] & 0xF0))
      {
         OWERROR(OWERROR_READ_VERIFY_FAILED);
         return FALSE;
      }
      else
         return TRUE;
   }
```

```
   else
      OWERROR(OWERROR_BLOCK_FAILED);

   // failed a block tranfer
   return FALSE;
}
```

Υπάρχουσες κοινές βιβλιοθήκες

Οι παρακάτω βιβλιοθήκες χρησιμοποιήθηκαν αυτούσιες. Υλοποιούν επιμέρους λειτουργίες, όπως έλεγχος λαθών, εμφάνιση μυνημάτων λαθών κτλ. Επίσης εδώ εμφανίζονται οι επιμέρους βιβλιοθήκες του κάθε προσαρμογέα ανάλογα με το πρωτόκολλο υλοποιήσης του. Π.χ. ο σειριακός μετατροπέας έχει ειδικές συναρτήσεις χειραγώγησης της σειριακής θύρας κτλ.

Έλεγχος λειτουργιών CRC και πρόληψη λαθών

mcrcutil.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  crcutil.c - Keeps track of the CRC for 16 and 8 bit operations
//  version 2.00

// Include files
#include "mownet.h"

// Local global variables
ushort utilcrc16[MAX_PORTNUM];
uchar utilcrc8[MAX_PORTNUM];
static short oddparity[16] = { 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0 };
static uchar dscrc_table[] = {
        0, 94,188,226, 97, 63,221,131,194,156,126, 32,163,253, 31, 65,
      157,195, 33,127,252,162, 64, 30, 95,  1,227,189, 62, 96,130,220,
       35,125,159,193, 66, 28,254,160,225,191, 93,  3,128,222, 60, 98,
      190,224,  2, 92,223,129, 99, 61,124, 34,192,158, 29, 67,161,255,
       70, 24,250,164, 39,121,155,197,132,218, 56,102,229,187, 89,  7,
      219,133,103, 57,186,228,  6, 88, 25, 71,165,251,120, 38,196,154,
      101, 59,217,135,  4, 90,184,230,167,249, 27, 69,198,152,122, 36,
      248,166, 68, 26,153,199, 37,123, 58,100,134,216, 91,  5,231,185,
      140,210, 48,110,237,179, 81, 15, 78, 16,242,172, 47,113,147,205,
       17, 79,173,243,112, 46,204,146,211,141,111, 49,178,236, 14, 80,
      175,241, 19, 77,206,144,114, 44,109, 51,209,143, 12, 82,176,238,
       50,108,142,208, 83, 13,239,177,240,174, 76, 18,145,207, 45,115,
```

```
      202,148,118, 40,171,245, 23, 73,  8, 86,180,234,105, 55,213,139,
       87,  9,235,181, 54,104,138,212,149,203, 41,119,244,170, 72, 22,
      233,183, 85, 11,136,214, 52,106, 43,117,151,201, 74, 20,246,168,
      116, 42,200,150, 21, 75,169,247,182,232, 10, 84,215,137,107, 53};

//--------------------------------------------------------------------------
// Reset crc16 to the value passed in
//
// 'reset' - data to set crc16 to.
//
void setcrc16(int portnum, ushort reset)
{
   utilcrc16[portnum] = reset;
   return;
}

//--------------------------------------------------------------------------
// Reset crc8 to the value passed in
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number is provided to
//              indicate the symbolic port number.
// 'reset'    - data to set crc8 to
//
void setcrc8(int portnum, uchar reset)
{
   utilcrc8[portnum] = reset;
   return;
}

//--------------------------------------------------------------------------
// Calculate a new CRC16 from the input data short.  Return the current
// CRC16 and also update the global variable CRC16.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number is provided to
//              indicate the symbolic port number.
// 'data'     - data to perform a CRC16 on
//
// Returns: the current CRC16
//
ushort docrc16(int portnum, ushort cdata)
{
   cdata = (cdata ^ (utilcrc16[portnum] & 0xff)) & 0xff;
   utilcrc16[portnum] >>= 8;

   if (oddparity[cdata & 0xf] ^ oddparity[cdata >> 4])
     utilcrc16[portnum] ^= 0xc001;

   cdata <<= 6;
   utilcrc16[portnum]   ^= cdata;
   cdata <<= 1;
   utilcrc16[portnum]   ^= cdata;

   return utilcrc16[portnum];
}

//--------------------------------------------------------------------------
// Update the Dallas Semiconductor One Wire CRC (utilcrc8) from the global
// variable utilcrc8 and the argument.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number is provided to
//              indicate the symbolic port number.
// 'x'        - data byte to calculate the 8 bit crc from
//
// Returns: the updated utilcrc8.
//
uchar docrc8(int portnum, uchar x)
{
   utilcrc8[portnum] = dscrc_table[utilcrc8[portnum] ^ x];
   return utilcrc8[portnum];
}
```

Αναλυτική υλοποιήση όλων των λαθών που μπορούν να εμφανιστούν σε ένα δίκτυο 1-Wire

mowerr.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
// owerr.c - Library functions for error handling with 1-Wire library
//
// Version: 1.00
//

#include <string.h>
#ifndef _WIN32_WCE
#include <stdio.h>
#endif
#include "mownet.h"

#ifndef SIZE_OWERROR_STACK
   #ifdef SMALL_MEMORY_TARGET
      //for small memory, only hole 1 error
      #define SIZE_OWERROR_STACK 1
   #else
      #define SIZE_OWERROR_STACK 10
   #endif
#endif

//--------------------------------------------------------------------------
// Variables
//--------------------------------------------------------------------------

// Error Struct for holding error information.
// In DEBUG, this will also hold the line number and filename.
typedef struct
{
   int owErrorNum;
#ifdef DEBUG
   int lineno;
   char *filename;
#endif
} owErrorStruct;

// Ring-buffer used for stack.
// In case of overflow, deepest error is over-written.
static owErrorStruct owErrorStack[SIZE_OWERROR_STACK];

// Stack pointer to top-most error.
static int owErrorPointer = 0;


//--------------------------------------------------------------------------
// Functions Definitions
//--------------------------------------------------------------------------
int owGetErrorNum(void);
void owClearError(void);
int owHasErrors(void);
#ifdef DEBUG
```

```
   void owRaiseError(int,int,char*);
#else
   void owRaiseError(int);
#endif
#ifndef SMALL_MEMORY_TARGET
   void owPrintErrorMsg(FILE *);
   void owPrintErrorMsgStd();
   char *owGetErrorMsg(int);
#endif


//--------------------------------------------------------------------------
// The 'owGetErroNum' returns the error code of the top-most error on the
// error stack.  NOTE: This function has the side effect of popping the
// current error off the stack.  All successive calls to 'owGetErrorNum'
// will further clear the error stack.
//
// For list of error codes, see 'ownet.h'
//
// Returns:   int :  The error code of the top-most error on the stack
//
int owGetErrorNum(void)
{
   int i = owErrorStack[ owErrorPointer ].owErrorNum;
   owErrorStack[ owErrorPointer ].owErrorNum = 0;
   if(!owErrorPointer)
      owErrorPointer = SIZE_OWERROR_STACK - 1;
   else
      owErrorPointer = (owErrorPointer - 1);
   return i;
}

//--------------------------------------------------------------------------
// The 'owClearError' clears all the errors.
//
void owClearError(void)
{
   owErrorStack[ owErrorPointer ].owErrorNum = 0;
}

//--------------------------------------------------------------------------
// The 'owHasErrors' is a boolean test function which tests whether or not
// a valid error is waiting on the stack.
//
// Returns:   TRUE (1) : When there are errors on the stack.
//            FALSE (0): When stack's errors are set to 0, or NO_ERROR_SET.
//
int owHasErrors(void)
{
   if(owErrorStack[ owErrorPointer ].owErrorNum)
      return 1; //TRUE
   else
      return 0; //FALSE
}

#ifdef DEBUG
   //--------------------------------------------------------------------------
   // The 'owRaiseError' is the method for raising an error onto the error
   // stack.
   //
   // Arguments:  int err - the error code you wish to raise.
   //             int lineno - DEBUG only - the line number where it was raised
   //             char* filename - DEBUG only - the file name where it occured.
   //
   void owRaiseError(int err, int lineno, char* filename)
   {
      owErrorPointer = (owErrorPointer + 1) % SIZE_OWERROR_STACK;
      owErrorStack[ owErrorPointer ].owErrorNum = err;
      owErrorStack[ owErrorPointer ].lineno = lineno;
      owErrorStack[ owErrorPointer ].filename = filename;
   }
#else
   //--------------------------------------------------------------------------
   // The 'owRaiseError' is the method for raising an error onto the error
   // stack.
   //
```

```
   // Arguments:  int err - the error code you wish to raise.
   //
   void owRaiseError(int err)
   {
      owErrorPointer = (owErrorPointer + 1) % SIZE_OWERROR_STACK;
      owErrorStack[ owErrorPointer ].owErrorNum = err;
   }
#endif


// SMALL_MEMORY_TARGET - embedded microcontrollers, where these
// messaging functions might not make any sense.
#ifndef SMALL_MEMORY_TARGET
   //Array of meaningful error messages to associate with codes.
   //Not used on targets with low memory (i.e. PIC).
   static char *owErrorMsg[117] =
   {
   /*000*/ "No Error Was Set",
   /*001*/ "No Devices found on 1-Wire Network",
   /*002*/ "1-Wire Net Reset Failed",
   /*003*/ "Search ROM Error: Couldn't locate next device on 1-Wire",
   /*004*/ "Access Failed: Could not select device",
   /*005*/ "DS2480B Adapter Not Detected",
   /*006*/ "DS2480B: Wrong Baud",
   /*007*/ "DS2480B: Bad Response",
   /*008*/ "Open COM Failed",
   /*009*/ "Write COM Failed",
   /*010*/ "Read COM Failed",
   /*011*/ "Data Block Too Large",
   /*012*/ "Block Transfer failed",
   /*013*/ "Program Pulse Failed",
   /*014*/ "Program Byte Failed",
   /*015*/ "Write Byte Failed",
   /*016*/ "Read Byte Failed",
   /*017*/ "Write Verify Failed",
   /*018*/ "Read Verify Failed",
   /*019*/ "Write Scratchpad Failed",
   /*020*/ "Copy Scratchpad Failed",
   /*021*/ "Incorrect CRC Length",
   /*022*/ "CRC Failed",
   /*023*/ "Failed to acquire a necessary system resource",
   /*024*/ "Failed to initialize system resource",
   /*025*/ "Data too long to fit on specified device.",
   /*026*/ "Read exceeds memory bank end.",
   /*027*/ "Write exceeds memory bank end.",
   /*028*/ "Device select failed",
   /*029*/ "Read Scratch Pad verify failed.",
   /*030*/ "Copy scratchpad complete not found",
   /*031*/ "Erase scratchpad complete not found",
   /*032*/ "Address read back from scrachpad was incorrect",
   /*033*/ "Read page with extra-info not supported by this memory bank",
   /*034*/ "Read page packet with extra-info not supported by this memory bank",
   /*035*/ "Length of packet requested exceeds page size",
   /*036*/ "Invalid length in packet",
   /*037*/ "Program pulse required but not available",
   /*038*/ "Trying to access a read-only memory bank",
   /*039*/ "Current bank is not general purpose memory",
   /*040*/ "Read back from write compare is incorrect, page may be locked",
   /*041*/ "Invalid page number for this memory bank",
   /*042*/ "Read page with CRC not supported by this memory bank",
   /*043*/ "Read page with CRC and extra-info not supported by this memory bank",
   /*044*/ "Read back from write incorrect, could not lock page",
   /*045*/ "Read back from write incorrect, could not lock redirect byte",
   /*046*/ "The read of the status was not completed.",
   /*047*/ "Page redirection not supported by this memory bank",
   /*048*/ "Lock Page redirection not supported by this memory bank",
   /*049*/ "Read back byte on EPROM programming did not match.",
   /*050*/ "Can not write to a page that is locked.",
   /*051*/ "Can not lock a redirected page that has already been locked.",
   /*052*/ "Trying to redirect a locked redirected page.",
   /*053*/ "Trying to lock a page that is already locked.",
   /*054*/ "Trying to write to a memory bank that is write protected.",
   /*055*/ "Error due to not matching MAC.",
   /*056*/ "Memory Bank is write protected.",
   /*057*/ "Secret is write protected, can not Load First Secret.",
   /*058*/ "Error in Reading Scratchpad after Computing Next Secret.",
```

```
   /*059*/ "Load Error from Loading First Secret.",
   /*060*/ "Power delivery required but not available",
   /*061*/ "Not a valid file name.",
   /*062*/ "Unable to Create a Directory in this part.",
   /*063*/ "That file already exists.",
   /*064*/ "The directory is not empty.",
   /*065*/ "The wrong type of part for this operation.",
   /*066*/ "The max len for this file is too small.",
   /*067*/ "This is not a write once bank.",
   /*068*/ "The file can not be found.",
   /*069*/ "There is not enough space availabe.",
   /*070*/ "There is not a page to match that bit in the bitmap.",
   /*071*/ "There are no jobs for EPROM parts.",
   /*072*/ "Function not supported to modify attributes.",
   /*073*/ "Handle is not in use.",
   /*074*/ "Tring to read a write only file.",
   /*075*/ "There is no handle available for use.",
   /*076*/ "The directory provided is an invalid directory.",
   /*077*/ "Handle does not exist.",
   /*078*/ "Serial Number did not match with current job.",
   /*079*/ "Can not program EPROM because a non-EPROM part on the network.",
   /*080*/ "Write protect redirection byte is set.",
   /*081*/ "There is an inappropriate directory length.",
   /*082*/ "The file has already been terminated.",
   /*083*/ "Failed to read memory page of iButton part.",
   /*084*/ "Failed to match scratchpad of iButton part.",
   /*085*/ "Failed to erase scratchpad of iButton part.",
   /*086*/ "Failed to read scratchpad of iButton part.",
   /*087*/ "Failed to execute SHA function on SHA iButton.",
   /*088*/ "SHA iButton did not return a status completion byte.",
   /*089*/ "Write data page failed.",
   /*090*/ "Copy secret into secret memory pages failed.",
   /*091*/ "Bind unique secret to iButton failed.",
   /*092*/ "Could not install secret into user token.",
   /*093*/ "Transaction Incomplete: signature did not match.",
   /*094*/ "Transaction Incomplete: could not sign service data.",
   /*095*/ "User token did not provide a valid authentication response.",
   /*096*/ "Failed to answer a challenge on the user token.",
   /*097*/ "Failed to create a challenge on the coprocessor.",
   /*098*/ "Transaction Incomplete: service data was not valid.",
   /*099*/ "Transaction Incomplete: service data was not updated.",
   /*100*/ "Unrecoverable, catastrophic service failure occured.",
   /*101*/ "Load First Secret from scratchpad data failed.",
   /*102*/ "Failed to match signature of user's service data.",
   /*103*/ "Subkey out of range for the DS1991.",
   /*104*/ "Block ID out of range for the DS1991",
   /*105*/ "Password is enabled",
   /*106*/ "Password is invalid",
   /*107*/ "This memory bank has no read only password",
   /*108*/ "This memory bank has no read/write password",
   /*109*/ "1-Wire is shorted",
   /*110*/ "Error communicating with 1-Wire adapter",
   /*111*/ "CopyScratchpad failed: Ending Offset must go to end of page",
   /*112*/ "WriteScratchpad failed: Ending Offset must go to end of page",
   /*113*/ "Mission can not be stopped while one is not in progress",
   /*114*/ "Error stopping the mission",
   /*115*/ "Port number is outside (0,MAX_PORTNUM) interval",
   /*116*/ "Level of the 1-Wire was not changed"
   };

   char *owGetErrorMsg(int err)
   {
      return owErrorMsg[err];
   }

#ifndef __C51__
   //--------------------------------------------------------------------------
   // The 'owPrintErrorMsg' is the method for printing an error from the stack.
   // The destination for the print is specified by the argument, fileno, which
   // can be stderr, stdout, or a log file.  In non-debug mode, the output is
   // of the form:
   // Error num: Error msg
   //
   // In debug-mode, the output is of the form:
   // Error num: filename line#: Error msg
   //
```

```
   // NOTE: This function has the side-effect of popping the error off the stack.
   //
   // Arguments:  FILE*: the destination for printing.
   //
   void owPrintErrorMsg(FILE *filenum)
   {
   #ifdef DEBUG
      int l = owErrorStack[ owErrorPointer ].lineno;
      char *f = owErrorStack[ owErrorPointer ].filename;
      int err = owGetErrorNum();
      fprintf(filenum,"Error %d: %s line %d: %s\r\n",err,f,l,owErrorMsg[err]);
   #else
      int err = owGetErrorNum();
      fprintf(filenum,"Error %d: %s\r\n",err,owErrorMsg[err]);
   #endif
   }
#endif //__C51__

   // Same as above, except uses default printf output
   void owPrintErrorMsgStd()
   {
   #ifdef DEBUG
      int l = owErrorStack[ owErrorPointer ].lineno;
      char *f = owErrorStack[ owErrorPointer ].filename;
      int err = owGetErrorNum();
      printf("Error %d: %s line %d: %s\r\n",err,f,l,owErrorMsg[err]);
   #else
      int err = owGetErrorNum();
      printf("Error %d: %s\r\n",err,owErrorMsg[err]);
   #endif
   }
#endif
```

Υπάρχουσες βιβλιοθήκες μετατροπέων

Βιβλιοθήκες για τον σειριακό μετατροπέα DS2480

Mwnwlnk32.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  Win32Lnk.C - COM functions using Win32 to be used as a test
//               for DS2480 based Universal Serial Adapter 'U'
//               functions.
//
//  Version: 2.01
//
```

```
//  History: 1.00 -> 1.01  Added function msDelay.
//
//           1.01 -> 1.02  Changed to generic OpenCOM/CloseCOM for easier
//                         use with other platforms.
//
//           1.02 -> 1.03  Add function msGettick_DS9097U()
//
//           1.03 -> 2.00  Support for multiple ports.
//           2.00 -> 2.01 Added error handling. Added circular-include check.
//           2.01 -> 2.10 Added raw memory error handling and SMALLINT
//           2.10 -> 3.00 Added memory bank functionality
//                        Added file I/O operations
//

#include "mownet.h"
#include "mds2480.h"
#include <windows.h>
#include <stdio.h>

// Win32 globals needed
static HANDLE ComID[MAX_PORTNUM];
static OVERLAPPED osRead[MAX_PORTNUM],osWrite[MAX_PORTNUM];
static SMALLINT ComID_init = 0;

//--------------------------------------------------------------------------
//-------- COM required functions for MLANU
//--------------------------------------------------------------------------

//--------------------------------------------------------------------------
// Attempt to open a com port.  Keep the handle in ComID.
// Set the starting baud rate to 9600.
//
// 'portnum'   - number 0 to MAX_PORTNUM-1.  This number provided will
//               be used to indicate the port number desired when calling
//               all other functions in this library.
//
//
// Returns: the port number if it was succesful otherwise -1
//
int OpenCOMEx(char *port_zstr)
{
   int portnum;

   if(!ComID_init)
   {
      int i;
      for(i=0; i<MAX_PORTNUM; i++)
         ComID[i] = 0;
      ComID_init = 1;
   }

   // check to find first available handle slot
   for(portnum = 0; portnum<MAX_PORTNUM; portnum++)
   {
      if(!ComID[portnum])
         break;
   }
   OWASSERT( portnum<MAX_PORTNUM, OWERROR_PORTNUM_ERROR, -1 );

   if(!OpenCOM(portnum, port_zstr))
   {
      return -1;
   }

   return portnum;
}

//--------------------------------------------------------------------------
// Attempt to open a com port.  Keep the handle in ComID.
// Set the starting baud rate to 9600.
//
// 'portnum'   - number 0 to MAX_PORTNUM-1.  This number provided will
//               be used to indicate the port number desired when calling
//               all other functions in this library.
//
// 'port_zstr' - zero terminate port name.  For this platform
```

```
//              use format COMX where X is the port number.
//
//
// Returns: TRUE(1)  - success, COM port opened
//          FALSE(0) - failure, could not open specified port
//
SMALLINT OpenCOM(int portnum, char *port_zstr)
{
   char tempstr[80];
   short fRetVal;
   COMMTIMEOUTS CommTimeOuts;
   DCB dcb;

   if(!ComID_init)
   {
      int i;
      for(i=0; i<MAX_PORTNUM; i++)
         ComID[i] = 0;
      ComID_init = 1;
   }

   OWASSERT( portnum<MAX_PORTNUM && portnum>=0 && !ComID[portnum],
             OWERROR_PORTNUM_ERROR, FALSE );

   // open COMM device
   if ((ComID[portnum] =
      CreateFile( port_zstr, GENERIC_READ | GENERIC_WRITE,
                  0,
                  NULL,                 // no security attrs
                  OPEN_EXISTING,
                  FILE_FLAG_OVERLAPPED, // overlapped I/O
                  NULL )) == (HANDLE) -1 )
   {
      ComID[portnum] = 0;
      OWERROR(OWERROR_GET_SYSTEM_RESOURCE_FAILED);
      return (FALSE) ;
   }
   else
   {
      // create events for detection of reading and write to com port
      sprintf(tempstr,"COMM_READ_OVERLAPPED_EVENT_FOR_%s",port_zstr);
      osRead[portnum].hEvent = CreateEvent(NULL,TRUE,FALSE,tempstr);
      sprintf(tempstr,"COMM_WRITE_OVERLAPPED_EVENT_FOR_%s",port_zstr);
      osWrite[portnum].hEvent = CreateEvent(NULL,TRUE,FALSE,tempstr);

      // get any early notifications
      SetCommMask(ComID[portnum], EV_RXCHAR | EV_TXEMPTY | EV_ERR | EV_BREAK);

      // setup device buffers
      SetupComm(ComID[portnum], 2048, 2048);

      // purge any information in the buffer
      PurgeComm(ComID[portnum], PURGE_TXABORT | PURGE_RXABORT |
                           PURGE_TXCLEAR | PURGE_RXCLEAR );

      // set up for overlapped non-blocking I/O
      CommTimeOuts.ReadIntervalTimeout = 0;
      CommTimeOuts.ReadTotalTimeoutMultiplier = 20;
      CommTimeOuts.ReadTotalTimeoutConstant = 40;
      CommTimeOuts.WriteTotalTimeoutMultiplier = 20;
      CommTimeOuts.WriteTotalTimeoutConstant = 40;
      SetCommTimeouts(ComID[portnum], &CommTimeOuts);

      // setup the com port
      GetCommState(ComID[portnum], &dcb);

      dcb.BaudRate = CBR_9600;              // current baud rate
      dcb.fBinary = TRUE;                   // binary mode, no EOF check
      dcb.fParity = FALSE;                  // enable parity checking
      dcb.fOutxCtsFlow = FALSE;             // CTS output flow control
      dcb.fOutxDsrFlow = FALSE;             // DSR output flow control
      dcb.fDtrControl = DTR_CONTROL_ENABLE; // DTR flow control type
      dcb.fDsrSensitivity = FALSE;          // DSR sensitivity
      dcb.fTXContinueOnXoff = TRUE;         // XOFF continues Tx
      dcb.fOutX = FALSE;                    // XON/XOFF out flow control
      dcb.fInX = FALSE;                     // XON/XOFF in flow control
```

```
      dcb.fErrorChar = FALSE;                // enable error replacement
      dcb.fNull = FALSE;                     // enable null stripping
      dcb.fRtsControl = RTS_CONTROL_ENABLE;  // RTS flow control
      dcb.fAbortOnError = FALSE;             // abort reads/writes on error
      dcb.XonLim = 0;                        // transmit XON threshold
      dcb.XoffLim = 0;                       // transmit XOFF threshold
      dcb.ByteSize = 8;                      // number of bits/byte, 4-8
      dcb.Parity = NOPARITY;                 // 0-4=no,odd,even,mark,space
      dcb.StopBits = ONESTOPBIT;             // 0,1,2 = 1, 1.5, 2
      dcb.XonChar = 0;                       // Tx and Rx XON character
      dcb.XoffChar = 1;                      // Tx and Rx XOFF character
      dcb.ErrorChar = 0;                     // error replacement character
      dcb.EofChar = 0;                       // end of input character
      dcb.EvtChar = 0;                       // received event character

      fRetVal = SetCommState(ComID[portnum], &dcb);
   }

   // check if successfull
   if (!fRetVal)
   {
      CloseHandle(ComID[portnum]);
      CloseHandle(osRead[portnum].hEvent);
      CloseHandle(osWrite[portnum].hEvent);
      ComID[portnum] = 0;
      OWERROR(OWERROR_SYSTEM_RESOURCE_INIT_FAILED);
   }

   return (fRetVal);
}

//--------------------------------------------------------------------------
// Closes the connection to the port.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
void CloseCOM(int portnum)
{
   // disable event notification and wait for thread
   // to halt
   SetCommMask(ComID[portnum], 0);

   // drop DTR
   EscapeCommFunction(ComID[portnum], CLRDTR);

   // purge any outstanding reads/writes and close device handle
   PurgeComm(ComID[portnum], PURGE_TXABORT | PURGE_RXABORT |
                    PURGE_TXCLEAR | PURGE_RXCLEAR );
   CloseHandle(ComID[portnum]);
   CloseHandle(osRead[portnum].hEvent);
   CloseHandle(osWrite[portnum].hEvent);
   ComID[portnum] = 0;
}

//--------------------------------------------------------------------------
// Flush the rx and tx buffers
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
void FlushCOM(int portnum)
{
   // purge any information in the buffer
   PurgeComm(ComID[portnum], PURGE_TXABORT | PURGE_RXABORT |
                    PURGE_TXCLEAR | PURGE_RXCLEAR );
}

//--------------------------------------------------------------------------
// Write an array of bytes to the COM port, verify that it was
// sent out.  Assume that baud rate has been set.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'outlen'   - number of bytes to write to COM port
// 'outbuf'   - pointer ot an array of bytes to write
```

```
//
// Returns:  TRUE(1)  - success
//           FALSE(0) - failure
//
SMALLINT WriteCOM(int portnum, int outlen, uchar *outbuf)
{
   BOOL fWriteStat;
   DWORD dwBytesWritten=0;
   DWORD ler=0,to;

   // calculate a timeout
   to = 20 * outlen + 60;

   // reset the write event
   ResetEvent(osWrite[portnum].hEvent);

   // write the byte
   fWriteStat = WriteFile(ComID[portnum], (LPSTR) &outbuf[0],
                outlen, &dwBytesWritten, &osWrite[portnum] );

   // check for an error
   if (!fWriteStat)
      ler = GetLastError();

   // if not done writting then wait
   if (!fWriteStat && ler == ERROR_IO_PENDING)
   {
      WaitForSingleObject(osWrite[portnum].hEvent,to);

      // verify all is written correctly
      fWriteStat = GetOverlappedResult(ComID[portnum], &osWrite[portnum],
                   &dwBytesWritten, FALSE);

   }

   // check results of write
   if (!fWriteStat || (dwBytesWritten != (DWORD)outlen))
      return 0;
   else
      return 1;
}

//--------------------------------------------------------------------------
// Read an array of bytes to the COM port, verify that it was
// sent out.  Assume that baud rate has been set.
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
// 'inlen'     - number of bytes to read from COM port
// 'inbuf'     - pointer to a buffer to hold the incomming bytes
//
// Returns: number of characters read
//
int ReadCOM(int portnum, int inlen, uchar *inbuf)
{
   DWORD dwLength=0;
   BOOL fReadStat;
   DWORD ler=0,to;

   // calculate a timeout
   to = 20 * inlen + 60;

   // reset the read event
   ResetEvent(osRead[portnum].hEvent);

   // read
   fReadStat = ReadFile(ComID[portnum], (LPSTR) &inbuf[0],
                       inlen, &dwLength, &osRead[portnum]) ;

   // check for an error
   if (!fReadStat)
      ler = GetLastError();

   // if not done writing then wait
   if (!fReadStat && ler == ERROR_IO_PENDING)
   {
```

```
      // wait until everything is read
      WaitForSingleObject(osRead[portnum].hEvent,to);

      // verify all is read correctly
      fReadStat = GetOverlappedResult(ComID[portnum], &osRead[portnum],
                  &dwLength, FALSE);
   }

   // check results
   if (fReadStat)
      return dwLength;
   else
      return 0;
}

//--------------------------------------------------------------------------
// Send a break on the com port for at least 2 ms
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number was provided to
//              OpenCOM to indicate the port number.
//
void BreakCOM(int portnum)
{
   // start the reset pulse
   SetCommBreak(ComID[portnum]);

   // sleep
   Sleep(2);

   // clear the break
   ClearCommBreak(ComID[portnum]);
}

//--------------------------------------------------------------------------
// Set the baud rate on the com port.
//
// 'portnum'   - number 0 to MAX_PORTNUM-1.  This number was provided to
//               OpenCOM to indicate the port number.
// 'new_baud'  - new baud rate defined as
//                PARMSET_9600     0x00
//                PARMSET_19200    0x02
//                PARMSET_57600    0x04
//                PARMSET_115200   0x06
//
void SetBaudCOM(int portnum, uchar new_baud)
{
   DCB dcb;

   // get the current com port state
   GetCommState(ComID[portnum], &dcb);

   // change just the baud rate
   switch (new_baud)
   {
      case PARMSET_115200:
         dcb.BaudRate = CBR_115200;
         break;
      case PARMSET_57600:
         dcb.BaudRate = CBR_57600;
         break;
      case PARMSET_19200:
         dcb.BaudRate = CBR_19200;
         break;
      case PARMSET_9600:
      default:
         dcb.BaudRate = CBR_9600;
         break;
   }

   // restore to set the new baud rate
   SetCommState(ComID[portnum], &dcb);
}

//--------------------------------------------------------------------------
//  Description:
//     Delay for at least 'len' ms
```

```
//
void msDelay_DS9097U(int len)
{
   Sleep(len);
}

//--------------------------------------------------------------------------
// Get the current millisecond tick count.  Does not have to represent
// an actual time, it just needs to be an incrementing timer.
//
long msGettick_DS9097U(void)
{
   return GetTickCount();
}
```

m2480ut.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  ds2480ut.c - DS2480B utility functions.
//
//  Version: 2.01
//
//  History: 1.00 -> 1.01  Default PDSRC changed from 0.83 to 1.37V/us
//                         in DS2480Detect. Changed to use msDelay instead
//                         of Delay.
//           1.01 -> 1.02  Changed global declarations from 'uchar' to 'int'.
//                         Changed DSO/WORT from 7 to 10us in DS2480Detect.
//           1.02 -> 1.03  Removed caps in #includes for Linux capatibility
//           1.03 -> 2.00  Changed 'MLan' to 'ow'. Added support for
//                         multiple ports.  Changed W1LT to 8us.
//           2.00 -> 2.01 Added error handling. Added circular-include check.
//           2.01 -> 2.10 Added raw memory error handling and SMALLINT
//           2.10 -> 3.00 Added memory bank functionality
//                        Added file I/O operations
//

#include "mownet.h"
#include "mds2480.h"

// global DS2480B state
SMALLINT ULevel[MAX_PORTNUM]; // current DS2480B 1-Wire Net level
SMALLINT UBaud[MAX_PORTNUM];  // current DS2480B baud rate
SMALLINT UMode[MAX_PORTNUM];  // current DS2480B command or data mode state
SMALLINT USpeed[MAX_PORTNUM]; // current DS2480B 1-Wire Net communication speed
SMALLINT UVersion[MAX_PORTNUM]; // current DS2480B version

//--------------------------------------------------------------------------
// Attempt to resyc and detect a DS2480B
```

```
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number was provided to
//                OpenCOM to indicate the port number.
//
// Returns:  TRUE  - DS2480B detected successfully
//           FALSE - Could not detect DS2480B
//
SMALLINT DS2480Detect(int portnum)
{
   uchar sendpacket[10],readbuffer[10];
   uchar sendlen=0;

   // reset modes
   UMode[portnum] = MODSEL_COMMAND;
   UBaud[portnum] = PARMSET_9600;
   USpeed[portnum] = SPEEDSEL_FLEX;

   // set the baud rate to 9600
   SetBaudCOM(portnum,(uchar)UBaud[portnum]);

   // send a break to reset the DS2480
   BreakCOM(portnum);

   // delay to let line settle
   msDelay_DS9097U(2);

   // flush the buffers
   FlushCOM(portnum);

   // send the timing byte
   sendpacket[0] = 0xC1;
   if (WriteCOM(portnum,1,sendpacket) != 1)
   {
      OWERROR(OWERROR_WRITECOM_FAILED);
      return FALSE;
   }

   // delay to let line settle
   msDelay_DS9097U(4);

   // set the FLEX configuration parameters
   // default PDSRC = 1.37Vus
   sendpacket[sendlen++] = CMD_CONFIG | PARMSEL_SLEW | PARMSET_Slew1p37Vus;
   // default W1LT = 10us
   sendpacket[sendlen++] = CMD_CONFIG | PARMSEL_WRITE1LOW | PARMSET_Write10us;
   // default DSO/WORT = 8us
   sendpacket[sendlen++] = CMD_CONFIG | PARMSEL_SAMPLEOFFSET | PARMSET_SampOff8us;

   // construct the command to read the baud rate (to test command block)
   sendpacket[sendlen++] = CMD_CONFIG | PARMSEL_PARMREAD | (PARMSEL_BAUDRATE >> 3);

   // also do 1 bit operation (to test 1-Wire block)
   sendpacket[sendlen++] = CMD_COMM | FUNCTSEL_BIT | UBaud[portnum] | BITPOL_ONE;

   // flush the buffers
   FlushCOM(portnum);

   // send the packet
   if (WriteCOM(portnum,sendlen,sendpacket))
   {
      // read back the response
      if (ReadCOM(portnum,5,readbuffer) == 5)
      {
         // look at the baud rate and bit operation
         // to see if the response makes sense
         if (((readbuffer[3] & 0xF1) == 0x00) &&
             ((readbuffer[3] & 0x0E) == UBaud[portnum]) &&
             ((readbuffer[4] & 0xF0) == 0x90) &&
             ((readbuffer[4] & 0x0C) == UBaud[portnum]))
            return TRUE;
         else
            OWERROR(OWERROR_DS2480_BAD_RESPONSE);
      }
      else
         OWERROR(OWERROR_READCOM_FAILED);
   }
```

```
   else
      OWERROR(OWERROR_WRITECOM_FAILED);

   return FALSE;
}

//--------------------------------------------------------------------------
// Change the DS2480B from the current baud rate to the new baud rate.
//
// 'portnum' - number 0 to MAX_PORTNUM-1.  This number was provided to
//             OpenCOM to indicate the port number.
// 'newbaud' - the new baud rate to change to, defined as:
//               PARMSET_9600     0x00
//               PARMSET_19200    0x02
//               PARMSET_57600    0x04
//               PARMSET_115200   0x06
//
// Returns:  current DS2480B baud rate.
//
SMALLINT DS2480ChangeBaud(int portnum, uchar newbaud)
{
   uchar rt=FALSE;
   uchar readbuffer[5],sendpacket[5],sendpacket2[5];
   uchar sendlen=0,sendlen2=0;

   // see if diffenent then current baud rate
   if (UBaud[portnum] == newbaud)
      return UBaud[portnum];
   else
   {
      // build the command packet
      // check if correct mode
      if (UMode[portnum] != MODSEL_COMMAND)
      {
         UMode[portnum] = MODSEL_COMMAND;
         sendpacket[sendlen++] = MODE_COMMAND;
      }
      // build the command
      sendpacket[sendlen++] = CMD_CONFIG | PARMSEL_BAUDRATE | newbaud;

      // flush the buffers
      FlushCOM(portnum);

      // send the packet
      if (!WriteCOM(portnum,sendlen,sendpacket))
      {
         OWERROR(OWERROR_WRITECOM_FAILED);
         rt = FALSE;
      }
      else
      {
         // make sure buffer is flushed
         msDelay_DS9097U(5);

         // change our baud rate
         SetBaudCOM(portnum,newbaud);
         UBaud[portnum] = newbaud;

         // wait for things to settle
         msDelay_DS9097U(5);

         // build a command packet to read back baud rate
         sendpacket2[sendlen2++] = CMD_CONFIG | PARMSEL_PARMREAD | (PARMSEL_BAUDRATE >>
3);

         // flush the buffers
         FlushCOM(portnum);

         // send the packet
         if (WriteCOM(portnum,sendlen2,sendpacket2))
         {
            // read back the 1 byte response
            if (ReadCOM(portnum,1,readbuffer) == 1)
            {
               // verify correct baud
               if (((readbuffer[0] & 0x0E) == (sendpacket[sendlen-1] & 0x0E)))
```

```
                  rt = TRUE;
               else
                  OWERROR(OWERROR_DS2480_WRONG_BAUD);
            }
            else
               OWERROR(OWERROR_READCOM_FAILED);
         }
         else
            OWERROR(OWERROR_WRITECOM_FAILED);
      }
   }

   // if lost communication with DS2480 then reset
   if (rt != TRUE)
      DS2480Detect(portnum);

   return UBaud[portnum];
}
```

Mds2480.h

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  DS2480.H - This file contains the DS2480B constants
//
//  Version: 2.00
//
//  History: 1.02 -> 1.03 Make sure uchar is not defined twice.
//
//

#include "mownet.h"

#ifndef __MC68K__
#include <stdlib.h>
#ifndef _WIN32_WCE
#include <stdio.h>
#endif
#endif

// Mode Commands
#define MODE_DATA                    0xE1
#define MODE_COMMAND                 0xE3
#define MODE_STOP_PULSE              0xF1

// Return byte value
#define RB_CHIPID_MASK               0x1C
#define RB_RESET_MASK                0x03
#define RB_1WIRESHORT                0x00
#define RB_PRESENCE                  0x01
#define RB_ALARMPRESENCE             0x02
```

```
#define RB_NOPRESENCE                   0x03

#define RB_BIT_MASK                     0x03
#define RB_BIT_ONE                      0x03
#define RB_BIT_ZERO                     0x00

// Masks for all bit ranges
#define CMD_MASK                        0x80
#define FUNCTSEL_MASK                   0x60
#define BITPOL_MASK                     0x10
#define SPEEDSEL_MASK                   0x0C
#define MODSEL_MASK                     0x02
#define PARMSEL_MASK                    0x70
#define PARMSET_MASK                    0x0E
#define VERSION_MASK                    0x1C

// Command or config bit
#define CMD_COMM                        0x81
#define CMD_CONFIG                      0x01

// Function select bits
#define FUNCTSEL_BIT                    0x00
#define FUNCTSEL_SEARCHON               0x30
#define FUNCTSEL_SEARCHOFF              0x20
#define FUNCTSEL_RESET                  0x40
#define FUNCTSEL_CHMOD                  0x60

// Bit polarity/Pulse voltage bits
#define BITPOL_ONE                      0x10
#define BITPOL_ZERO                     0x00
#define BITPOL_5V                       0x00
#define BITPOL_12V                      0x10

// One Wire speed bits
#define SPEEDSEL_STD                    0x00
#define SPEEDSEL_FLEX                   0x04
#define SPEEDSEL_OD                     0x08
#define SPEEDSEL_PULSE                  0x0C

// Data/Command mode select bits
#define MODSEL_DATA                     0x00
#define MODSEL_COMMAND                  0x02

// 5V Follow Pulse select bits (If 5V pulse
// will be following the next byte or bit.)
#define PRIME5V_TRUE                    0x02
#define PRIME5V_FALSE                   0x00

// Parameter select bits
#define PARMSEL_PARMREAD                0x00
#define PARMSEL_SLEW                    0x10
#define PARMSEL_12VPULSE                0x20
#define PARMSEL_5VPULSE                 0x30
#define PARMSEL_WRITE1LOW               0x40
#define PARMSEL_SAMPLEOFFSET            0x50
#define PARMSEL_ACTIVEPULLUPTIME        0x60
#define PARMSEL_BAUDRATE                0x70

// Pull down slew rate.
#define PARMSET_Slew15Vus               0x00
#define PARMSET_Slew2p2Vus              0x02
#define PARMSET_Slew1p65Vus             0x04
#define PARMSET_Slew1p37Vus             0x06
#define PARMSET_Slew1p1Vus              0x08
#define PARMSET_Slew0p83Vus             0x0A
#define PARMSET_Slew0p7Vus              0x0C
#define PARMSET_Slew0p55Vus             0x0E

// 12V programming pulse time table
#define PARMSET_32us                    0x00
#define PARMSET_64us                    0x02
#define PARMSET_128us                   0x04
#define PARMSET_256us                   0x06
#define PARMSET_512us                   0x08
#define PARMSET_1024us                  0x0A
#define PARMSET_2048us                  0x0C
```

```
#define PARMSET_infinite              0x0E

// 5V strong pull up pulse time table
#define PARMSET_16p4ms                0x00
#define PARMSET_65p5ms                0x02
#define PARMSET_131ms                 0x04
#define PARMSET_262ms                 0x06
#define PARMSET_524ms                 0x08
#define PARMSET_1p05s                 0x0A
#define PARMSET_2p10s                 0x0C
#define PARMSET_infinite              0x0E

// Write 1 low time
#define PARMSET_Write8us              0x00
#define PARMSET_Write9us              0x02
#define PARMSET_Write10us             0x04
#define PARMSET_Write11us             0x06
#define PARMSET_Write12us             0x08
#define PARMSET_Write13us             0x0A
#define PARMSET_Write14us             0x0C
#define PARMSET_Write15us             0x0E

// Data sample offset and Write 0 recovery time
#define PARMSET_SampOff3us            0x00
#define PARMSET_SampOff4us            0x02
#define PARMSET_SampOff5us            0x04
#define PARMSET_SampOff6us            0x06
#define PARMSET_SampOff7us            0x08
#define PARMSET_SampOff8us            0x0A
#define PARMSET_SampOff9us            0x0C
#define PARMSET_SampOff10us           0x0E

// Active pull up on time
#define PARMSET_PullUp0p0us           0x00
#define PARMSET_PullUp0p5us           0x02
#define PARMSET_PullUp1p0us           0x04
#define PARMSET_PullUp1p5us           0x06
#define PARMSET_PullUp2p0us           0x08
#define PARMSET_PullUp2p5us           0x0A
#define PARMSET_PullUp3p0us           0x0C
#define PARMSET_PullUp3p5us           0x0E

// Baud rate bits
#define PARMSET_9600                  0x00
#define PARMSET_19200                 0x02
#define PARMSET_57600                 0x04
#define PARMSET_115200                0x06

// DS2480B program voltage available
#define DS2480PROG_MASK               0x20

// mode bit flags
#define MODE_NORMAL                   0x00
#define MODE_OVERDRIVE                0x01
#define MODE_STRONG5                  0x02
#define MODE_PROGRAM                  0x04
#define MODE_BREAK                    0x08

// Versions of DS2480
#define VER_LINK                      0x1C
#define VER_DS2480                    0x08
#define VER_DS2480B                   0x0C

// exportable functions defined in ds2480ut.c
SMALLINT DS2480Detect(int portnum);
SMALLINT DS2480ChangeBaud(int portnum, uchar newbaud);

// link functions from win32lnk.c or other link files
SMALLINT  OpenCOM(int portnum, char *port_zstr);
int       OpenCOMEx(char *port_zstr);
void      CloseCOM(int portnum);
void      FlushCOM(int portnum);
SMALLINT  WriteCOM(int portnum, int outlen, uchar *outbuf);
int       ReadCOM(int portnum, int inlen, uchar *inbuf);
void      BreakCOM(int portnum);
void      SetBaudCOM(int portnum, uchar new_baud);
```

Βιβλιοθήκες για τον USB μετατροπέα DS2490

m2490ut.c

```
//---------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//---------------------------------------------------------------------------
//
//  ds2490ut.c - DS2490 utility functions.
//             (Requires DS2490.SYS)
//
//  Version:
//

#include "mownet.h"
#include "ds2490.h"

// handles for the USB ports
extern HANDLE usbhnd[MAX_PORTNUM];

// global DS2490 state
SMALLINT USBLevel[MAX_PORTNUM];
SMALLINT USBSpeed[MAX_PORTNUM];
SMALLINT USBVersion[MAX_PORTNUM];
SMALLINT USBVpp[MAX_PORTNUM];

//---------------------------------------------------------------------------
// Attempt to resyc and detect a DS2490
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number was provided to
//                OpenCOM to indicate the port number.
//
// Returns:  TRUE  - DS2490 detected successfully
//           FALSE - Could not detect DS2490
//
SMALLINT DS2490Detect(HANDLE hDevice)
{
   SMALLINT present,vpp;
   SETUP_PACKET setup;
   ULONG nOutput = 0;

   // reset the DS2490
   DS2490Reset(hDevice);

   // set the strong pullup duration to infinite
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = COMM_CMD;
   setup.Value = COMM_SET_DURATION | COMM_IM;
   setup.Index = 0x0000;
```

```
   setup.Length = 0;
   setup.DataOut = FALSE;
   // call the driver
   DeviceIoControl(hDevice,
                                       DS2490_IOCTL_VENDOR,
                                       &setup,
                                       sizeof(SETUP_PACKET),
                                       NULL,
                                       0,
                                       &nOutput,
                                       NULL);

   // set the 12V pullup duration to 512us
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = COMM_CMD;
   setup.Value = COMM_SET_DURATION | COMM_IM | COMM_TYPE;
   setup.Index = 0x0040;
   setup.Length = 0;
   setup.DataOut = FALSE;
   // call the driver
   DeviceIoControl(hDevice,
                                       DS2490_IOCTL_VENDOR,
                                       &setup,
                                       sizeof(SETUP_PACKET),
                                       NULL,
                                       0,
                                       &nOutput,
                                       NULL);

   // disable strong pullup, but leave program pulse enabled (faster)
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = MODE_CMD;
   setup.Value = MOD_PULSE_EN;
   setup.Index = ENABLEPULSE_PRGE;
   setup.Length = 0x00;
   setup.DataOut = FALSE;
   // call the driver
   DeviceIoControl(hDevice,
                                       DS2490_IOCTL_VENDOR,
                                       &setup,
                                       sizeof(SETUP_PACKET),
                                       NULL,
                                       0,
                                       &nOutput,
                                       NULL);

   // return result of short check
   return DS2490ShortCheck(hDevice,&present,&vpp);
}


//---------------------------------------------------------------------------
// Check to see if there is a short on the 1-Wire bus. Used to stop
// communication with the DS2490 while the short is in effect to not
// overrun the buffers.
//
// '*present' - flag set (1) if device presence detected
// '*vpp' - flag set (1) if Vpp programming voltage detected
//
// Returns:  TRUE  - DS2490 1-Wire is NOT shorted
//           FALSE - Could not detect DS2490 or 1-Wire shorted
//
SMALLINT DS2490ShortCheck(HANDLE hDevice, SMALLINT *present, SMALLINT *vpp)
{
   STATUS_PACKET status;
   BYTE nResultRegisters;
   BYTE i;

   // get the result registers (if any)
   if (!DS2490GetStatus(hDevice, &status, &nResultRegisters))
      return FALSE;

   // get vpp present flag
   *vpp = ((status.StatusFlags & STATUSFLAGS_12VP) != 0);

   //  Check for short
```

```
   if(status.CommBufferStatus != 0)
   {
      return FALSE;
   }
   else
   {
      // check for short
           for (i = 0; i < nResultRegisters; i++)
           {
         // check for SH bit (0x02), ignore 0xA5
         if (status.CommResultCodes[i] & COMMCMDERRORRESULT_SH)
         {
            // short detected
            return FALSE;
         }
      }
   }

   // check for No 1-Wire device condition
   *present = TRUE;
   // loop through result registers
   for (i = 0; i < nResultRegisters; i++)
   {
      // only check for error conditions when the condition is not a ONEWIREDEVICEDETECT
      if (status.CommResultCodes[i] != ONEWIREDEVICEDETECT)
      {
         // check for NRS bit (0x01)
         if (status.CommResultCodes[i] & COMMCMDERRORRESULT_NRS)
         {
            // empty bus detected
            *present = FALSE;
         }
      }
   }

   return TRUE;
}


//---------------------------------------------------------------------------
// Stop any on-going pulses
//
// Returns:  TRUE  - pulse stopped
//           FALSE - Could not stop pulse
//
SMALLINT DS2490HaltPulse(HANDLE hDevice)
{
   STATUS_PACKET status;
   BYTE nResultRegisters;
   SETUP_PACKET setup;
   ULONG nOutput = 0;
   long limit;


   // set a time limit
   limit = msGettick_DS9490() + 300;
   // loop until confirm pulse has ended or timeout
   do
   {
      // HalExecWhenIdle, ResumExecution to stop an infinite pulse

      // HalExecWhenIdle
      setup.RequestTypeReservedBits = 0x40;
      setup.Request = CONTROL_CMD;
      setup.Value = CTL_HALT_EXE_IDLE;
      setup.Index = 0x00;
      setup.Length = 0x00;
      setup.DataOut = FALSE;
      // call the driver
      if (!DeviceIoControl(hDevice,
                                          DS2490_IOCTL_VENDOR,
                                          &setup,
                                          sizeof(SETUP_PACKET),
                                          NULL,
                                          0,
                                          &nOutput,
```

```
                                        NULL))
      {
         // failure
         break;
      }

      // ResumExecution
      setup.RequestTypeReservedBits = 0x40;
      setup.Request = CONTROL_CMD;
      setup.Value = CTL_RESUME_EXE;
      setup.Index = 0x00;
      setup.Length = 0x00;
      setup.DataOut = FALSE;

      // call the driver
      if (!DeviceIoControl(hDevice,
                                        DS2490_IOCTL_VENDOR,
                                        &setup,
                                        sizeof(SETUP_PACKET),
                                        NULL,
                                        0,
                                        &nOutput,
                                        NULL))
      {
         // failure
         break;
      }

      // read the status to see if the pulse has been stopped
      if (!DS2490GetStatus(hDevice, &status, &nResultRegisters))
      {
         // failure
         break;
      }
      else
      {
         // check the SPU flag
         if ((status.StatusFlags & STATUSFLAGS_SPUA) == 0)
         {
            // success
            // disable both pulse types
            setup.RequestTypeReservedBits = 0x40;
            setup.Request = MODE_CMD;
            setup.Value = MOD_PULSE_EN;
            setup.Index = 0;
            setup.Length = 0x00;
            setup.DataOut = FALSE;
            // call the driver
            DeviceIoControl(hDevice,
                                              DS2490_IOCTL_VENDOR,
                                              &setup,
                                              sizeof(SETUP_PACKET),
                                              NULL,
                                              0,
                                              &nOutput,
                                              NULL);

            return TRUE;
         }
      }
   }
   while (limit > msGettick_DS9490());

   return FALSE;
}

//---------------------------------------------------------------------------
// Description: Gets the status of the DS2490 device
// Input:       hDevice - the handle to the DS2490 device
//              pStatus - the Status Packet to be filled with data
//              pResultSize - the number of result register codes returned
//                can be NULL
// Returns:     FALSE on failure, TRUE on success
// Error Codes: DS2490COMM_ERR_USBDEVICE
SMALLINT DS2490GetStatus(HANDLE hDevice, STATUS_PACKET *status, BYTE *pResultSize)
{
```

```
   BYTE *bufOutput = (BYTE *)status;
   ULONG nOutput = sizeof(STATUS_PACKET);

   // Call device IO Control interface (DS2490_IOCTL_STATUS) in driver
   if (!DeviceIoControl(hDevice,
                                          DS2490_IOCTL_STATUS,
                                          NULL,
                                          0,
                                          bufOutput,
                                          sizeof(STATUS_PACKET),
                                          &nOutput,
                                          NULL)
         )
   {
      OWERROR(OWERROR_ADAPTER_ERROR);
      return FALSE;
   }

   // any bytes returned after the 16th byte are place in the CommResultCodes field
   if (pResultSize != NULL)
   {
      *pResultSize = (BYTE) nOutput - 16;
   }

   return TRUE;
}


//---------------------------------------------------------------------------
// Description: Perfroms a hardware reset of the DS2490 equivalent to a
//              power-on reset
// Input:       hDevice - the handle to the DS2490 device
// Returns:     FALSE on failure, TRUE on success
// Error Codes: DS2490COMM_ERR_USBDEVICE
//
SMALLINT DS2490Reset(HANDLE hDevice)
{
   SETUP_PACKET setup;
   ULONG nOutput = 0;

   // setup for reset
   setup.RequestTypeReservedBits = 0x40;
   setup.Request = CONTROL_CMD;
   setup.Value = CTL_RESET_DEVICE;
   setup.Index = 0x00;
   setup.Length = 0x00;
   setup.DataOut = FALSE;
   // call the driver
   return (!DeviceIoControl(hDevice,
                                       DS2490_IOCTL_VENDOR,
                                       &setup,
                                       sizeof(SETUP_PACKET),
                                       NULL,
                                       0,
                                       &nOutput,
                                       NULL));
}


//---------------------------------------------------------------------------
// Description: Reads data from EP3
// Input:       hDevice - the handle to the DS2490 device
//              buffer - the size must be >= nBytes
//              pnBytes - the number of bytes to read from the device
// Returns:     FALSE on failure, TRUE on success
//
SMALLINT DS2490Read(HANDLE hDevice, BYTE *buffer, WORD *pnBytes)
{

   // Synchronous read:
        ULONG   nBytes = *pnBytes;

        if      (!ReadFile(hDevice,  // handle
                  buffer,   // buffer to read
                  nBytes,   // number of bytes to read
                  &nBytes,  // number of bytes read
```

```
                  NULL   // overlap
                                              ))
        {
       OWERROR(OWERROR_ADAPTER_ERROR);
                return FALSE;
        }
        else
        {
                *pnBytes = (WORD)nBytes;
                return TRUE;
        }
}

//---------------------------------------------------------------------------
// Description: Writes data to EP2
// Input:       hDevice - the handle to the DS2490 device
//              buffer - the size must be >= nBytes
//              pnBytes - the number of bytes to write to the device
// Returns:     FALSE on failure, TRUE on success
//
SMALLINT DS2490Write(HANDLE hDevice, BYTE *buffer, WORD *pnBytes)
{
   // Synchronous write:
   // assume enough room for write
   ULONG       nBytes = *pnBytes;

   if  (!WriteFile(hDevice,    // handle
                 buffer,     // buffer to read
                 *pnBytes,   // number of bytes to read
                 &nBytes,    // number of bytes read
                 NULL))      // overlap
   {
      OWERROR(OWERROR_ADAPTER_ERROR);
      return FALSE;
   }
   else
   {
      *pnBytes = (WORD)nBytes;
      return TRUE;
   }
}
```

d2490u.h

```
//---------------------------------------------------------------------------
// Copyright (C) 2003 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//---------------------------------------------------------------------------
//

#ifndef _ds2490comm_h_
#define _ds2490comm_h_

#include <windows.h>
```

```
#include <winioctl.h>

// single byte alignment on structures
#pragma pack(push, 1)

// Error codes
#define DS2490COMM_ERR_NOERROR               0   // an error has not yet been
encountered
#define DS2490COMM_ERR_GETLASTERROR          1   // use GetLastError() for more
information
#define DS2490COMM_ERR_RESULTREGISTERS       2   // use
DS2490COMM_GetLastResultRegister() for more info
#define DS2490COMM_ERR_USBDEVICE             3   // error from USB device driver
#define DS2490COMM_ERR_READWRITE             4   // an I/O error occurred while
communicating w/ device
#define DS2490COMM_ERR_TIMEOUT               5   // an operation timed out before
completion
#define DS2490COMM_ERR_INCOMPLETEWRITE       6   // not all data could be sent for
output
#define DS2490COMM_ERR_INCOMPLETEREAD        7   // not all data could be received for
an input
#define DS2490COMM_ERR_INITTOUCHBYTE         8   // the touch byte thread could not be
started
#define g_DS2490COMM_LAST_SHORTEDBUS         9   // 1Wire bus shorted on
#define g_DS2490COMM_LAST_EMPTYBUS           10  // 1Wire bus empty

#define IOCTL_INBUF_SIZE              512
#define IOCTL_OUTBUF_SIZE             512
#define TIMEOUT_PER_BYTE            15    //1000 modified 4/27/00 BJV

// Definition is taken from DDK
typedef struct _USB_DEVICE_DESCRIPTOR {
    UCHAR bLength;
    UCHAR bDescriptorType;
    USHORT bcdUSB;
    UCHAR bDeviceClass;
    UCHAR bDeviceSubClass;
    UCHAR bDeviceProtocol;
    UCHAR bMaxPacketSize0;
    USHORT idVendor;
    USHORT idProduct;
    USHORT bcdDevice;
    UCHAR iManufacturer;
    UCHAR iProduct;
    UCHAR iSerialNumber;
    UCHAR bNumConfigurations;
} USB_DEVICE_DESCRIPTOR, *PUSB_DEVICE_DESCRIPTOR;

// IOCTL codes
#define DS2490_IOCTL_VENDOR CTL_CODE(FILE_DEVICE_UNKNOWN, 0x800, METHOD_BUFFERED,
FILE_ANY_ACCESS)
#define DS2490_IOCTL_STATUS CTL_CODE(FILE_DEVICE_UNKNOWN, 0x801, METHOD_BUFFERED,
FILE_ANY_ACCESS)
#define DS2490_IOCTL_DEVICE CTL_CODE(FILE_DEVICE_UNKNOWN, 0x802, METHOD_BUFFERED,
FILE_ANY_ACCESS)


// Device Information is put here
typedef struct _USB_DEVICE_INFO
{
       ULONG   DriverVersion;
       USB_DEVICE_DESCRIPTOR  Descriptor;
       UCHAR   Unit;
} USB_DEVICE_INFO, *PUSB_DEVICE_INFO;

// Setup packet
typedef struct _SETUP_PACKET
{
   // Only for vendor specifc bits for COMM commands.
       UCHAR   RequestTypeReservedBits;
    // The Request byte
       UCHAR   Request;
    // The Value byte
       USHORT  Value;
    // The Index byte
       USHORT  Index;
```

```
    // The length of extra Data In or Out
        USHORT  Length;
   // Does the extra data go In, or Out?
   BOOLEAN DataOut;
   // If there is extra data In, it goes here.
   UCHAR   DataInBuffer[0];
} SETUP_PACKET, *PSETUP_PACKET;

// Request byte, Command Type Code Constants
#define CONTROL_CMD                 0x00
#define COMM_CMD                            0x01
#define MODE_CMD                            0x02
#define TEST_CMD                            0x03

//
// Value field, Control commands
//
// Control Command Code Constants
#define CTL_RESET_DEVICE                    0x0000
#define CTL_START_EXE                       0x0001
#define CTL_RESUME_EXE                      0x0002
#define CTL_HALT_EXE_IDLE                   0x0003
#define CTL_HALT_EXE_DONE                   0x0004
#define CTL_CANCEL_CMD                      0x0005
#define CTL_CANCEL_MACRO                    0x0006
#define CTL_FLUSH_COMM_CMDS         0x0007
#define CTL_FLUSH_RCV_BUFFER        0x0008
#define CTL_FLUSH_XMT_BUFFER        0x0009
#define CTL_GET_COMM_CMDS                   0x000A

//
// Value field COMM Command options
//
// COMM Bits (bitwise or into COMM commands to build full value byte pairs)
// Byte 1
#define COMM_TYPE                           0x0008
#define COMM_SE                     0x0008
#define COMM_D                      0x0008
#define COMM_Z                      0x0008
#define COMM_CH                     0x0008
#define COMM_SM                     0x0008
#define COMM_R                      0x0008
#define COMM_IM                     0x0001

// Byte 2
#define COMM_PS                     0x4000
#define COMM_PST                            0x4000
#define COMM_CIB                    0x4000
#define COMM_RTS                    0x4000
#define COMM_DT                     0x2000
#define COMM_SPU                            0x1000
#define COMM_F                              0x0800
#define COMM_ICP                            0x0200
#define COMM_RST                            0x0100

// Read Straight command, special bits
#define COMM_READ_STRAIGHT_NTF          0x0008
#define COMM_READ_STRAIGHT_ICP          0x0004
#define COMM_READ_STRAIGHT_RST          0x0002
#define COMM_READ_STRAIGHT_IM           0x0001

//
// Value field COMM Command options (0-F plus assorted bits)
//
#define COMM_ERROR_ESCAPE               0x0601
#define COMM_SET_DURATION               0x0012
#define COMM_BIT_IO                     0x0020
#define COMM_PULSE                      0x0030
#define COMM_1_WIRE_RESET               0x0042
#define COMM_BYTE_IO                    0x0052
#define COMM_MATCH_ACCESS               0x0064
#define COMM_BLOCK_IO                   0x0074
#define COMM_READ_STRAIGHT              0x0080
#define COMM_DO_RELEASE                 0x6092
#define COMM_SET_PATH                   0x00A2
#define COMM_WRITE_SRAM_PAGE            0x00B2
```

```
#define COMM_WRITE_EPROM                0x00C4
#define COMM_READ_CRC_PROT_PAGE         0x00D4
#define COMM_READ_REDIRECT_PAGE_CRC     0x21E4
#define COMM_SEARCH_ACCESS              0x00F4

// Mode Command Code Constants
// Enable Pulse Constants
#define ENABLEPULSE_PRGE                      0x01  // strong pull-up
#define ENABLEPULSE_SPUE                      0x02  // programming pulse

// 1Wire Bus Speed Setting Constants
#define ONEWIREBUSSPEED_REGULAR        0x00
#define ONEWIREBUSSPEED_FLEXIBLE       0x01
#define ONEWIREBUSSPEED_OVERDRIVE      0x02

//
// Value field Mode Commands options
//
#define MOD_PULSE_EN                   0x0000
#define MOD_SPEED_CHANGE_EN            0x0001
#define MOD_1WIRE_SPEED                0x0002
#define MOD_STRONG_PU_DURATION         0x0003
#define MOD_PULLDOWN_SLEWRATE          0x0004
#define MOD_PROG_PULSE_DURATION        0x0005
#define MOD_WRITE1_LOWTIME             0x0006
#define MOD_DSOW0_TREC                 0x0007

//
// This is the status structure as returned by DS2490_IOCTL_STATUS.
//
typedef struct _STATUS_PACKET
{
       UCHAR   EnableFlags;
       UCHAR   OneWireSpeed;
       UCHAR   StrongPullUpDuration;
       UCHAR   ProgPulseDuration;
       UCHAR   PullDownSlewRate;
       UCHAR   Write1LowTime;
       UCHAR   DSOW0RecoveryTime;
       UCHAR   Reserved1;
       UCHAR   StatusFlags;
       UCHAR   CurrentCommCmd1;
       UCHAR   CurrentCommCmd2;
       UCHAR   CommBufferStatus;  // Buffer for COMM commands
       UCHAR   WriteBufferStatus; // Buffer we write to
       UCHAR   ReadBufferStatus;  // Buffer we read from
       UCHAR   Reserved2;
       UCHAR   Reserved3;
   // There may be up to 16 bytes here, or there may not.
   UCHAR   CommResultCodes[16];
} STATUS_PACKET, *PSTATUS_PACKET;


//
// STATUS FLAGS
//
// Enable Flags
#define ENABLEFLAGS_SPUE                        0x01  // if set Strong Pull-up to 5V
enabled
#define ENABLEFLAGS_PRGE                        0x02  // if set 12V programming pulse
enabled
#define ENABLEFLAGS_SPCE                        0x04  // if set a dynamic 1-Wire bus
speed change through Comm. Cmd. enabled

// Device Status Flags
#define STATUSFLAGS_SPUA                    0x01  // if set Strong Pull-up is active
#define STATUSFLAGS_PRGA                    0x02  // if set a 12V programming pulse is
being generated
#define STATUSFLAGS_12VP                    0x04  // if set the external 12V
programming voltage is present
#define STATUSFLAGS_PMOD                    0x08  // if set the DS2490 powered from USB
and external sources
#define STATUSFLAGS_HALT                    0x10  // if set the DS2490 is currently
halted
#define STATUSFLAGS_IDLE                    0x20  // if set the DS2490 is currently
idle
```

```
// Result Registers
#define ONEWIREDEVICEDETECT               0xA5  // 1-Wire device detected on bus
#define COMMCMDERRORRESULT_NRS            0x01  // if set 1-WIRE RESET did not reveal a
Presence Pulse or SET PATH did not get a Presence Pulse from the branch to be connected
#define COMMCMDERRORRESULT_SH             0x02  // if set 1-WIRE RESET revealed a short
on the 1-Wire bus or the SET PATH couln not connect a branch due to short
#define COMMCMDERRORRESULT_APP            0x04  // if set a 1-WIRE RESET revealed an
Alarming Presence Pulse
#define COMMCMDERRORRESULT_VPP            0x08  // if set during a PULSE with TYPE=1 or
WRITE EPROM command the 12V programming pulse not seen on 1-Wire bus
#define COMMCMDERRORRESULT_CMP            0x10  // if set there was an error reading
confirmation byte of SET PATH or WRITE EPROM was unsuccessful
#define COMMCMDERRORRESULT_CRC            0x20  // if set a CRC occurred for one of the
commands: WRITE SRAM PAGE, WRITE EPROM, READ EPROM, READ CRC PROT PAGE, or READ REDIRECT
PAGE W/CRC
#define COMMCMDERRORRESULT_RDP            0x40  // if set READ REDIRECT PAGE WITH CRC
encountered a redirected page
#define COMMCMDERRORRESULT_EOS            0x80  // if set SEARCH ACCESS with SM=1 ended
sooner than expected with too few ROM IDs

// Strong Pullup
#define SPU_MULTIPLE_MS                   128
#define SPU_DEFAULT_CODE                  512 / SPU_MULTIPLE_MS   // default Strong
pullup value

// Programming Pulse
#define PRG_MULTIPLE_US                   8               // Programming Pulse Time
Multiple (Time = PRG_MULTIPLE_US * DurationCode)
#define PRG_DEFAULT_CODE                  512 / PRG_MULTIPLE_US   // default Programming
pulse value

// Support functions
SMALLINT DS2490Detect(HANDLE hDevice);
SMALLINT DS2490GetStatus(HANDLE hDevice, STATUS_PACKET *status, BYTE *pResultSize);
SMALLINT DS2490ShortCheck(HANDLE hDevice, SMALLINT *present,SMALLINT *vpp);
SMALLINT DS2490Reset(HANDLE hDevice);
SMALLINT DS2490Read(HANDLE hDevice, BYTE *buffer, WORD *pnBytes);
SMALLINT DS2490Write(HANDLE hDevice, BYTE *buffer, WORD *pnBytes);
SMALLINT DS2490HaltPulse(HANDLE hDevice);
SMALLINT AdapterRecover(int portnum);

#endif // _ds2490comm_h_
```

Παράρτημα Β - HA7Net http Server

Υπάρχουσα βιβλιοθήκη για τον HA7net

(Embedded Data Systems, Low Level 1-Wire support HA7Net Users Manual, 29.10.05)

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------



//  HA7Net.c - One Library supporting ALL the functions for the http server
//              HA7Net.
//              Version: 1.0

//--------------------------------------------------------------------------
// CONDITION FOR ALL HTTP COMMANDS
// As soon as the http protocol is selected, in order to achive compatibility
// with the other interfaces, it will be addressed at port 9x (COM90-COM99)
//--------------------------------------------------------------------------------

public:

// the 4 octets of the IP Address will be returned by the user interface
// initialized to -1 so if the IP is not provided, the name of the HTTP
// server will be used insead

   int FirstIPOctet=-1;
   int SecIPOctet=-1;
   int ThirdIPOctet=-1;
   int ForthIPOctet=-1;


// Returns the IP Address of the HA7Net http server device.
// in case that no address is provided or found, then the default name of the
// server will be used (HA7Net.com).
// Because of the varius problems of C++ with http format, the address will
// be returned in 4 contigius blocks of integers without the dots in between.
// E.G. the http address 192.168.101.145 will be returned as 192 168 101 145

   int GetIPAddress(int FirstIPOctet, int SecIPOctet, int ThirdIPOctet,
         int ForthIPOctet){
   // get IP Address from User Interface

   // set FirstIPOctet;
```

```
   // set SecIPOctet;
   // set ThirdIPOctet;
   // set ForthIPOctet;
   }

// Formats for the functions the IP Address of the HA7Net http server device.
// in case that no address is provided or found, then it returns -1, so the
// name of server will be used (HA7Net.com).
   int SetIPAddress(){
         while (FirstIPOctet && SecIPOctet && ThirdIPOctet &&
         ForthIPOctet)!=-1)
            //return  IPAddress_Prop_formated
         else return -1;
   }

// basic function for all devices. it executes the Search command
// which is the equivalent of owNext for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found

   int owNext_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/Search.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/Search.html
          }

      }

// basic function for all devices. it executes the family search command
// which is the equivalent of owFamilySearchSetup for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found

   int owAccess_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/Search.html & familyCode
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/Search.html & familyCode
          }

      }


// basic function for all devices. it executes the address device command
// which is the equivalent of owAccess for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found

   int owAccess_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/AddressDevice.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/AddressDevice.html
          }

      }

// basic function for all devices. it executes the Match ROM command
// which is the equivalent of owVerify for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found
```

```
   int owVerify_HA7Net(int portnum)
      while (portnum==90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/MatchRom.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/MatchRom.html
          }

      }

// basic function for all devices. it executes the Match ROM command
// which is the equivalent of owVerify for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found

   int owtouchReset_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/Reset.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/Reset.html
          }

      }

// basic function for all devices. it executes the Power Down command

   void owPowerDown_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/PowerDownBus.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/PowerDownBus.html
          }

      }

// basic function for all devices. it executes the Match ROM command
// which is the equivalent of owVerify for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found

   int owVerify_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/MatchRom.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/MatchRom.html
          }

      }

// basic function for all devices. it executes the Write Block command
// which is the equivalent of owBlock for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found

   int owBlock_HA7Net(int portnum)
```

```
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/WriteBlock.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/WriteBlock.html
          }

      }


// basic function for all devices. it executes the Write bit command


   int owWriteBit_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/WriteBit.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/WriteBlock.html
          }

      }

// basic function for all devices. it executes the Read Bit command

   int owReadBit_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/ReadBit.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/ReadBit.html
          }

      }

// basic function for all devices. it executes the Write/Read command
// which is the equivalent of owReadBitPower for the 1-Wire Networks
// it returns 'ResultData'–Table that contains the data read back from
// the 1-Wire bus. This value is stored in a text field named 'ResultData_0'

   int owBlock_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

          //  http://IPAddress_Prop_formated/1Wire/WriteBit.html
          //  http://IPAddress_Prop_formated/1Wire/ReadBit.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/WriteBlock.html
         //  http://HA7Net.com/1Wire/ReadBit.html
          }

      }

// basic function for all devices. it executes the Read Pages command

   int owReadPages_HA7Net(int portnum)
      while (portnum>=90) {
```

```
            while (SetIPAddress()!=-1)
            // if there is an IP Address use it to issue the command

            //  http://IPAddress_Prop_formated/1Wire/ReadPages.html
             else
             // if there is NOT an IP Address use the default name of the
             // http server to issue the command

            //  http://HA7Net.com/1Wire/ReadPages.html
             }

         }


   // basic function for all devices. it executes the Read File Records command
   // it returns 'Records'-Table that contains the records read from the 1-Wire
   // device. This value is stored as hex in a text field named 'Record_x',
   // where x is a 0 based sequentially numbered integer.

      int owReadFileRecords_HA7Net(int portnum)
         while (portnum>=90) {
            while (SetIPAddress()!=-1)
            // if there is an IP Address use it to issue the command

            //  http://IPAddress_Prop_formated/1Wire/ReadFileRecords.html
             else
             // if there is NOT an IP Address use the default name of the
             // http server to issue the command

            //  http://HA7Net.com/1Wire/ReadFileRecords.html
             }

         }

   // basic function for all devices. it executes the Write File Records command
   // This command does not return anything other than the page statistics and
   // an exception, if applicable.

      void owWriteFileRecords_HA7Net(int portnum)
         while (portnum>=90) {
            while (SetIPAddress()!=-1)
            // if there is an IP Address use it to issue the command

            //  http://IPAddress_Prop_formated/1Wire/WriteFileRecords.html
             else
             // if there is NOT an IP Address use the default name of the
             // http server to issue the command

            //  http://HA7Net.com/1Wire/WriteFileRecords.html
             }

         }

   // basic function for all devices. it executes the GetLock command

   // it returns the LOck ID of the network in question

      int owAquire_HA7Net(int portnum)
         while (portnum>=90) {
            while (SetIPAddress()!=-1)
            // if there is an IP Address use it to issue the command

            //  http://IPAddress_Prop_formated/1Wire/GetLock.html
             else
             // if there is NOT an IP Address use the default name of the
             // http server to issue the command

            //  http://HA7Net.com/1Wire/GetLock.html
             }

         }

   // basic function for all devices. it executes the Release Lock command
   // which is the equivalent of owAquire for the 1-Wire Networks
   // it returns the LOck ID of the network in question
```

```
int owAquire_HA7Net(int portnum)
   while (portnum>=90) {
      while (SetIPAddress()!=-1)
      // if there is an IP Address use it to issue the command

      //  http://IPAddress_Prop_formated/1Wire/ReleaseLock.html
       else
       // if there is NOT an IP Address use the default name of the
       // http server to issue the command

      //  http://HA7Net.com/1Wire/ReleaseLock.html
       }

   }
```

Βιβλιοθήκη για τον εξυπηρετητή http HA7net

Σημείωση: Αυτή η βιβλιοθήκη ουσιαστικά έχει απλώς το ίδιο όνομα με την προηγούμενη. Είναι γραμμένη εξ' ολοκλήρου από την αρχή και μάλιστα με αρκετά διαφορετική φιλοσοφία από την προηγούμενη. Όλες οι συναρτήσεις εμπεριέχονται μέσα σε ένα μοναδικό αρχείο

HA7net.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------



//  HA7Net.c - One Library supporting ALL the functions for the http server
//             HA7Net.
//             Version: 1.18
//
// George Violettas-Greek Open University, 7 major changes to library structure
//
//  1. owtouchReset_HA7Net notes changed to .. of owtouchReset for the 1-Wire..
//
//  2. after function owPowerDown the second instance of owVerify_HA7Net
//  was deleted
```

```
//
//  3. last function name changed from int owAquire_HA7Net(..
//  to int owRelease_HA7Net
//
//  4. 3 global variables definitions added:
//  LastDiscrepancy, LastFamilyDiscrepancy, LastDevice, SerialNum
//
//  5. the 1st line of the owtouchReset_HA7Net function description,
//  was chanced from  "....it executes the Match ROM command.... to
//  "....it executes the Reset command...." . Also at the same function
//  the line // it returns 'Adresses" the 8 byte unique code of the devices found
//  was deleted, the particular function returns nothing.
//
//  6. after the owNext_HA7Net function, the function owAccess_HA7Net
//  was repeated twice. The first one was substituted with the correct
//  owFamilySearchSetup_HA7Net.
//
//  7. a new function was added named int owTouchBit_HA7Net(int portnum,
//  SMALLINT sendbit). It is almost the same with thw owWriteBit_HA7Net,
//  it was added for compatibility reasons.

//--------------------------------------------------------------------------
// CONDITION FOR ALL HTTP COMMANDS
// As soon as the http protocol is selected, in order to achive compatibility
// with the other interfaces, it will be addressed at port 9x (COM90-COM99)
//--------------------------------------------------------------------------

//---------------------------------------------------------------
//following lines were added in order to store the results of the
// varius http commands
//??????????????????????????????????????????
// local functions defined in ownetu.c

#include<string.h>
#define MAX_DEVICES 100

static SMALLINT bitacc(SMALLINT,SMALLINT,SMALLINT,uchar *);

// global variables for this module to hold search state information
static int LastDiscrepancy[MAX_PORTNUM];
static int LastFamilyDiscrepancy[MAX_PORTNUM];
static uchar LastDevice[MAX_PORTNUM];

AnsiString Pages[256];
AnsiString Records[256];
AnsiString SerialNum;
AnsiString SerialNums[MAX_DEVICES];
int DevNum;
int CurrDev;
AnsiString LockID="";
//---------------------------------------------------------------

public:

// the 4 octets of the IP Address will be returned by the user interface
// initialized to -1 so if the IP is not provided, the name of the HTTP
// server will be used insead

   int FirstIPOctet=-1;
   int SecIPOctet=-1;
   int ThirdIPOctet=-1;
   int ForthIPOctet=-1;
   String IPAddress_Prop_formated;

// Returns the IP Address of the HA7Net http server device.
// in case that no address is provided or found, then the default name of the
// server will be used (HA7Net.com).
// Because of the varius problems of C++ with http format, the address will
// be returned in 4 contigius blocks of integers without the dots in between.
// E.G. the http address 192.168.101.145 will be returned as 192 168 101 145

   int GetIPAddress(int FirstIPOctet, int SecIPOctet, int ThirdIPOctet,
         int ForthIPOctet){

         IPAddress_Prop_formated = "155.207.13.16"
   // get IP Address from User Interface
```

```
   // set FirstIPOctet;
   // set SecIPOctet;
   // set ThirdIPOctet;
   // set ForthIPOctet;
   }

// Formats for the functions the IP Address of the HA7Net http server device.
// in case that no address is provided or found, then it returns -1, so the
// name of server will be used (HA7Net.com).
   int SetIPAddress(){
         while (FirstIPOctet && SecIPOctet && ThirdIPOctet &&
         ForthIPOctet)!=-1)
            //return  IPAddress_Prop_formated
         else return -1;
   }

// George Violettas-Greek Open University, added:

// basic function for searching inside http pages
// it always finds all the: VALUE="........." 16 characters word
// returned by the HTTP Server as the result to various GET Commands

   int SearchHttpPage(AnsiString IPAddress_Prop_formated, AnsiString HttpSpecCommand)
   {
      AnsiString Command;
      AnsiString Values[100];
      int n=0,pos,start, end;
      //execute the specific command
      NMHTTP1->Get("http://"+IPAddress_Prop_formated+"/1Wire/"+HttpSpecCommand);
      //collect the body of the http page
      Command = NMHTTP1->Body;
      //narrow the search starting from the field "Addresses"
      start = Command.AnsiPos("Addresses");
      //dont search further than the field "Statistics"
      end = Command.AnsiPos("Statistics");
      Command = Command.SubString(start,end - start);
      //if it finds the string "VALUE" with capital letters
      while((pos = Command.AnsiPos("VALUE")) != 0) {
      //there is a hex number of two bytes (16 bits)
         pos=pos+7;
         Values[n] = Command.SubString(pos,16);
         Command = Command.SubString(pos+16,Command.Length());
         n++;
         }
         //if there is no VALUE=".." then return 0. else return 1
         if(n==0)
            return 0;
         else
            return 1;
   }


// Find Network Devices
int owFindDevices(AnsiString IPAddress_Prop_formated, AnsiString HttpSpecCommand)
   {
      AnsiString Command;
      AnsiString Values[100];
      int n=0,pos,start, end;

      //execute the specific command
      NMHTTP1-
>Get("http://"+IPAddress_Prop_formated+"/1Wire/Search.html"+HttpSpecCommand);
      //collect the body of the http page
      Command = NMHTTP1->Body;
      //narrow the search starting from the field "Addresses"
      start = Command.AnsiPos("Addresses");
      //dont search further than the field "Statistics"
      end = Command.AnsiPos("Statistics");
      Command = Command.SubString(start,end - start);
      //if it finds the string "VALUE" with capital letters
      while((pos = Command.AnsiPos("VALUE")) != 0) {
      //there is a hex number of two bytes (16 bits)
         pos=pos+7;
         SerialNums[n] = Command.SubString(pos,16);
         Command = Command.SubString(pos+16,Command.Length());
```

```
         n++;
      }
      //if there is no VALUE=".." then return 0. else return 1
      if(n==0){
         DevNum = -1;
         CurrDev = -1;
         return 0;
      }
      else {
         return 1;
      }
   }

// George Violettas-Greek Open University, added:

// this function was added for compatibility reasons with the
// other protocols. it is exactly the same in all 3 protocols.

//--------------------------------------------------------------------------
// The 'owFirst' finds the first device on the 1-Wire Net  This function
// contains one parameter 'alarm_only'.  When
// 'alarm_only' is TRUE (1) the find alarm command 0xEC is
// sent instead of the normal search command 0xF0.
// Using the find alarm command 0xEC will limit the search to only
// 1-Wire devices that are in an 'alarm' state.
//
// 'portnum'    - number 0 to MAX_PORTNUM-1.  This number is provided to
//                indicate the symbolic port number.
// 'do_reset'   - TRUE (1) perform reset before search, FALSE (0) do not
//                perform reset before search.
// 'alarm_only' - TRUE (1) the find alarm command 0xEC is
//                sent instead of the normal search command 0xF0
//
// Returns:   TRUE (1) : when a 1-Wire device was found and it's
//                       Serial Number placed in the global SerialNum
//            FALSE (0): There are no devices on the 1-Wire Net.
//
SMALLINT owFirst_HA7Net(AnsiString IPAddress_Prop_formated, SMALLINT do_reset, SMALLINT
alarm_only)
{

   owFindDevices(IPAddress_Prop_formated,"");
   CurrDev=-1;

   return owNext_HA7Net(IPAddress_Prop_formated, do_reset, alarm_only);
}

// basic function for all devices. it executes the Search command
// which is the equivalent of owNext for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found
int owNext_HA7Net(AnsiString IPAddress_Prop_formated)

    CurrDev= CurrDev + 1
    if (CurrDev < DevNum){
       SerialNum= SerialNums[CurrDev];
       return 1;
    }
    else {
       LastDevice[portnum] = TRUE;
       CurrDev= CurrDev - 1
       return 0; //last device
    }
}


// basic function for all devices. it executes the family search command
// which is the equivalent of owFamilySearchSetup for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found

   int owFamilySearchSetup_HA7Net(int portnum, SMALLINT search_family) {

      return owFindDevices(IPAddress_Prop_formated,(" & "+AnsiString(search_family)));

      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command
```

```
            //  http://IPAddress_Prop_formated/1Wire/Search.html & familyCode
             else
             // if there is NOT an IP Address use the default name of the
             // http server to issue the command

            //  http://HA7Net.com/1Wire/Search.html & familyCode
             }

         }
      }

// basic function for all devices. it executes the address device command
// which is the equivalent of owAccess for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found

   int owAccess_HA7Net(int portnum){
      AnsiString Command;


      //execute the specific command
      NMHTTP1-
>Get("http://"+IPAddress_Prop_formated+"/1Wire/AddressDevice.html?Address="+SerialNum);
      //collect the body of the http page
      Command = NMHTTP1->Body;
      if(Command == "")
         return 0;
      else
         return 1;
   }

  /*  ???  while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/AddressDevice.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/AddressDevice.html
          }

      }  */

// basic function for all devices. it executes the Match ROM command
// which is the equivalent of owVerify for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found

   int owVerify_HA7Net(int portnum) {

       NMHTTP1->Get("http://"+IPAddress_Prop_formated+"/1Wire/MatchRom.html");


      while (portnum==90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/MatchRom.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/MatchRom.html
          }

      }
   }

// basic function for all devices. it executes the Reset command
// which is the equivalent of owtouchReset for the 1-Wire Networks

   int owtouchReset_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command
```

```
            //  http://IPAddress_Prop_formated/1Wire/Reset.html
            NMHTTP1->Get("http://"+IPAddress_Prop_formated+"/1Wire/Reset.html");

             else
             // if there is NOT an IP Address use the default name of the
             // http server to issue the command

            //  http://HA7Net.com/1Wire/Reset.html
            NMHTTP1->Get("http://HA7Net.com/1Wire/Reset.html");
             }

         }

// basic function for all devices. it executes the Power Down command

   void owPowerDown_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/PowerDownBus.html
         NMHTTP1->Get("http://"+IPAddress_Prop_formated+"/1Wire/PowerDownBus.html");
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/PowerDownBus.html
         NMHTTP1->Get("http://HA7Net.com/1Wire/PowerDownBus.html");
          }

      }



// basic function for all devices. it executes the Write Block command
// which is the equivalent of owBlock for the 1-Wire Networks
// it returns 'Adresses" the 8 byte unique code of the devices found

   int owBlock_HA7Net(int portnum, AnsiString data)

      AnsiString Command;
      AnsiString Values[100];
      int n=0,pos,start, end;
      AnsiString retData;

      //execute the specific command
      NMHTTP1-
>Get("http://"+IPAddress_Prop_formated+"/1Wire/WriteBlock.html"+"?Address="+SerialNum+"&
Data="+data);
      //collect the body of the http page
      Command = NMHTTP1->Body;
      //narrow the search starting from the field "Addresses"
      start = Command.AnsiPos("ResultaData_0");
      //dont search further than the field "Statistics"     //???
      end = Command.AnsiPos("Statistics");
      Command = Command.SubString(start,end - start);
      //if it finds the string "VALUE" with capital letters
      retData = Command.SubString(pos,16);
      if(retData == data)
         return 1;
      else
         return 0;

      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/WriteBlock.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/WriteBlock.html
          }
```

```
      }


// basic function for all devices. it executes the Write bit command


   int owWriteBit_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/WriteBit.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/WriteBlock.html
          }

      }

// basic function for all devices. it executes the Read Bit command

   int owReadBit_HA7Net(int portnum)
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/ReadBit.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/ReadBit.html
          }

      }

// George Violettas-Greek Open University, added:

//-------------------------------------------------------------------------
// Send 1 bit of communication to the 1-Wire Net and return the
// result 1 bit read from the 1-Wire Net.  The parameter 'sendbit'
// least significant bit is used and the least significant bit
// of the result is the return bit.
//
// 'portnum' - number 0 to MAX_PORTNUM-1.  This number was provided to
//             OpenCOM to indicate the port number.
// 'sendbit' - the least significant bit is the bit to send
//
// Returns: 0:   0 bit read from sendbit
//          1:   1 bit read from sendbit
//
   SMALLINT owTouchBit_HA7Net(int portnum, SMALLINT sendbit) {
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/WriteBit.html
         NMHTTP1-
>Get("http://"+IPAddress_Prop_formated+"/1Wire/WriteBit.html?"+AnsiString(sendbit));
         // check the returnig bit if it is thw same with the sendbit
         //????????????????????????????????????????????????????????

          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/WriteBit.html
         NMHTTP1->Get("http://HA7Net.com/1Wire/WriteBit.html?"+sendbit);
      }

      //collect the body of the http page
      Command = NMHTTP1->Body;
      //narrow the search starting from the field
      start = Command.AnsiPos("ResultaData_0");
```

```
      //dont search further than the field "Statistics"     //???
      end = Command.AnsiPos("Statistics");
      Command = Command.SubString(start,end - start);
      //if it finds the string "VALUE" with capital letters
      retData = Command.SubString(pos,16);
      if(retData == AnsiString(sendbit))
         return 1;
      else
         return 0;

  }

// basic function for all devices. it executes the Write/Read command
// which is the equivalent of owReadBitPower for the 1-Wire Networks
// it returns 'ResultData'–Table that contains the data read back from
// the 1-Wire bus. This value is stored in a text field named 'ResultData_0'

   int owBlock_HA7Net(int portnum) {

      return owTouchBit_HA7Net(int portnum, SMALLINT sendbit);

   }
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

          //  http://IPAddress_Prop_formated/1Wire/WriteBit.html
          //  http://IPAddress_Prop_formated/1Wire/ReadBit.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/WriteBit.html
         //  http://HA7Net.com/1Wire/ReadBit.html
          }

      }

// basic function for all devices. it executes the Read Pages command

   int owReadPages_HA7Net(int portnum,int startp, int numofpages) {
      AnsiString Command;
      int n=0,pos,start, end;

      //execute the specific command
      NMHTTP1-
>Get("http://"+IPAddress_Prop_formated+"/1Wire/ReadPages.html?Address="+SerialNum+"&Star
tPage="+AnsiString(startp)+"&PagesToRead="+AnsiString(numofpaegs));
      //collect the body of the http page
      Command = NMHTTP1->Body;
      //narrow the search starting from the field "Addresses"
      start = Command.AnsiPos("Page_");
      //dont search further than the field "Statistics"
      end = Command.AnsiPos("???");
      Command = Command.SubString(start,end - start);
      //if it finds the string "VALUE" with capital letters
      while((pos = Command.AnsiPos(("Pages_"+AnsiString(n)) ) ) != 0) {
      //there is a hex number of two bytes (16 bits)
         pos=pos+7;
         Pages[n] = Command.SubString(pos,????);  // teloes prin to end tou page
         Command = Command.SubString(pos+16,Command.Length());
         n++;
      }
      //if there is no PAGES=".." then return 0. else return 1
      if(n==0){
         return 0;
      }
      else {
         return 1;
      }
   }



   }
```

```
      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/ReadPages.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/ReadPages.html
          }

      }


// basic function for all devices. it executes the Read File Records command
// it returns 'Records'–Table that contains the records read from the 1-Wire
// device. This value is stored as hex in a text field named 'Record_x',
// where x is a 0 based sequentially numbered integer.

   int owReadFileRecords_HA7Net(int portnum,int startrec, int numofrecs)

       AnsiString Command;
      int n=0,pos,start, end;

      //execute the specific command
      NMHTTP1-
>Get("http://"+IPAddress_Prop_formated+"/1Wire/ReadFileRecords.html?Address="+SerialNum+
"&StartRecord="+AnsiString(startrec)+"&RecordsToRead="+AnsiString(numofrecs));
      //collect the body of the http page
      Command = NMHTTP1->Body;
      //narrow the search starting from the field "Addresses"
      start = Command.AnsiPos("Records_");
      //dont search further than the field "Statistics"
      end = Command.AnsiPos("???");
      Command = Command.SubString(start,end - start);
      //if it finds the string "RECORDS" with capital letters
      while((pos = Command.AnsiPos(("Records_"+AnsiString(n)) ) ) != 0) {
      //there is a hex number of two bytes (16 bits)
         pos=pos+7;
         Records[n] = Command.SubString(pos,????);  // teloes prin to end tou Record
         Command = Command.SubString(pos+16,Command.Length());
         n++;
      }
      //if there is no RECORDS=".." then return 0. else return 1
      if(n==0){
         return 0;
      }
      else {
         return 1;
      }
   }



      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/ReadFileRecords.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/ReadFileRecords.html
          }

      }

// basic function for all devices. it executes the Write File Records command
// This command does not return anything other than the page statistics and
// an exception, if applicable.

   void owWriteFileRecords_HA7Net(int portnum, int recnum,  char hexdata[])  {
```

```
        AnsiString Command;
      int n=0,pos,start, end;

      //execute the specific command
      NMHTTP1-
>Get("http://"+IPAddress_Prop_formated+"/1Wire/WriteFileRecords.html?Address="+SerialNum
+"&RecordNumber="+AnsiString(recnum)+"&Data="+AnsiString(hexdata));
      //collect the body of the http page
      Command = NMHTTP1->Body;

      if(Command == "")
         return 0;
      else
         return 1;

   }


      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/WriteFileRecords.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/WriteFileRecords.html
          }

      }

// basic function for all devices. it executes the GetLock command

// it returns the LOck ID of the network in question

   int owAquire_HA7Net(int portnum)  {
      AnsiString Command;
      AnsiString Values[100];
      int n=0,pos,start, end;
      AnsiString retData;

      //execute the specific command
      NMHTTP1->Get("http://"+IPAddress_Prop_formated+"/1Wire/GetLock.html");
      //collect the body of the http page
      Command = NMHTTP1->Body;
      //narrow the search starting from the field "Addresses"
      start = Command.AnsiPos("ResultaData_0");
      //dont search further than the field "Statistics"     //???
      end = Command.AnsiPos("Statistics");
      Command = Command.SubString(start,end - start);
      //if it finds the string "VALUE" with capital letters
      LockID = Command.SubString(pos,16);
      if(LockID != "")
         return 1;
      else
         return 0;

   }


      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/GetLock.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/GetLock.html
          }

      }

// basic function for all devices. it executes the Release Lock command
```

```
// which is the equivalent of owAquire for the 1-Wire Networks
// it returns the LOck ID of the network in question

   int owRelease_HA7Net(int portnum) {

       NMHTTP1-
>Get("http://"+IPAddress_Prop_formated+"/1Wire/ReleaseLock.html?Lockid="+LockID);

   }

      while (portnum>=90) {
         while (SetIPAddress()!=-1)
         // if there is an IP Address use it to issue the command

         //  http://IPAddress_Prop_formated/1Wire/ReleaseLock.html
          else
          // if there is NOT an IP Address use the default name of the
          // http server to issue the command

         //  http://HA7Net.com/1Wire/ReleaseLock.html
          }

      }
```

Εντολές που υποστηρίζονται από τον HA7net Server

(Embedded Data Systems, Low Level 1-Wire support HA7Net Users Manual, 29.10.05)

Search ROM Command

Base URL:

http://HA7Net.com/1Wire/Search.html

Parameters:

Optional LockID Ten byte decimal number previously returned by GetLock.
Optional FamilyCode Two byte hex number used to restrict the results to devices of a given family.
Optional Conditional A '0'or '1', with '1'indicating

Examples:

To retrieve a list of every device on the 1-Wire bus:
http://HA7Net.com/1Wire/Search.html
To retrieve a list of all DS18S20 temperature sensors on the 1-Wire bus:
http://HA7Net.com/1Wire/Search.html?FamilyCode=10
To retrieve a list of all devices on the 1-Wire bus in a conditional state:
http://HA7Net.com/1Wire/Search.html?Conditional=1
To retrieve a list of all DS18S20 temperature sensors on the 1-Wire bus that are in an alarm state:
http://HA7Net.com/1Wire/Search.html?FamilyCode=10&Conditional=True

Address Device Command

Base URL:

http://HA7Net.com/1Wire/AddressDevice.html

Parameters:

Required *Address* 8 byte hex 1-Wire ROM Address

Optional *LockID* Ten byte decimal number previously returned by GetLock.

Examples:
To address a 1-Wire temperature sensor having ROM code '280000003042C210':
http://HA7Net.com/1Wire/AddressDevice.html?Address=280000003042C210
This address might have been previously discovered using the 'Search'command, stored in a database from an initial system installation,

Match ROM Command
Base URL:
http://HA7Net.com/1Wire/MatchRom.html
Parameters:
Optional LockID **Ten byte decimal number previously returned by GetLock.**

Examples:
To address a 1-Wire temperature sensor having ROM code '280000003042C210', perform some actions on the bus, then reset the bus and reselect the same sensor:
http://HA7Net.com/1Wire/AddressDevice.html?Address=280000003042C210
… Do some work with the Sensor here…
http://HA7Net.com/1Wire/MatchRom.html

Reset Command
Base URL:
http://HA7Net.com/1Wire/Reset.html
Parameters:
Optional LockID **Ten byte decimal number previously returned by GetLock.**

Examples:
To reset the 1-Wire bus:
http://HA7Net.com/1Wire/Reset.html

Power Down Bus Command
Base URL:
http://HA7Net.com/1Wire/PowerDownBus.html
Parameters:
Optional LockID **Ten byte decimal number previously returned by GetLock.**

Examples:
To power down the 1-Wire bus:
http://HA7Net.com/1Wire/PowerDownBus.html

Write Block Command
Base URL:
http://HA7Net.com/1Wire/WriteBlock.html
Parameters:
Optional LockID Ten byte decimal number previously returned by GetLock.
Optional Address 8 byte hex 1-Wire ROM Address
Required Data 1-32 bytes of data formatted as HEX

Examples:
To tell the 1-Wire temperature sensor having ROM code '280000003042C210'to perform a temperature conversion:
http://HA7Net.com/1Wire/WriteBlock.html?Address=280000003042C210&Data=44
To read the 9 byte scratchpad from the temperature sensor above, in order to obtain the temperature information:
http://HA7Net.com/1Wire/WriteBlock.html?Address=280000003042C210&Data=BEFFFFFFFFFFFFFFFF
Notice the 9 'FF'bytes. This will create the necessary time slots on the 1 wire bus in order for the sensor to write back the response, as discussed in the description above.
To select the DS2406 with ROMcode 2400000007377212, issue the Channel Access command and read the Channel Info Byte which contains the input latches, output latches, and sensed levels of the two IO lines PIOA and PIOB:
http://HA7Net.com/1Wire/WriteBlock.html?Address=2400000007377212&Data=F5CFFFFF
Note that 4 bytes were written, and 4 bytes were read.

Read Bit Command
Base URL:
http://HA7Net.com/1Wire/ReadBit.html
Parameters:
Optional LockID **Ten byte decimal number previously returned by GetLock.**

Examples:
To read a single bit from the 1-Wire bus:
http://HA7Net.com/1Wire/ReadBit.html

Write Bit Command
Base URL:
http://HA7Net.com/1Wire/WriteBit.html
Parameters:
Optional LockID Ten byte decimal number previously returned by GetLock.
Required Bit Bit to write to the 1-Wire bus (Should be either 0 or 1)

Examples:
To read a single bit from the 1-Wire bus:
http://HA7Net.com/1Wire/ReadBit.html

Read Pages Command
Base URL:
http://HA7Net.com/1Wire/ReadPages.html
Parameters:
Optional LockID Ten byte decimal number previously returned by GetLock.
Required StartPage Page number to begin reading on. Should be an integral value between 0 and 255.
Optional PagesToRead Number of pages to read. Defaults to 1 if not specified.
Optional Address 8 byte hex 1-Wire ROM Address

Examples:
To read a 4 consecutive memory pages beginning with page 1 from the from the 1-Wire device having ROM code '2400000007377212':
http://HA7Net.com/1Wire/ReadPages.html?Address=2400000007377212&StartPage=1&PagesToRead=4

Read File Records Command

Base URL:

http://HA7Net.com/1Wire/ReadFileRecords.html

Parameters:

Optional LockID Ten byte decimal number previously returned by GetLock.
Required StartRecord Page number to reading the record from. Should be an integral value between 0 and 255.
Optional RecordsToRead Number of records to read. Defaults to 1 if not specified.
Optional Address 8 byte hex 1-Wire ROM Address

Examples:

To read a 4 consecutive file records beginning with the record located at page 1 from the from the 1-Wire device having ROM code '2400000007377212':
http://HA7Net.com/1Wire/ReadFileRecords.html?Address=2400000007377212&StartRecord=1&RecordsToRead=4

Write File Record Command

Base URL:

http://HA7Net.com/1Wire/WriteFileRecord.html

Parameters:

Optional LockID Ten byte decimal number previously returned by GetLock.
Required RecordNumber Page number to write the record to. Should be an integral value between 0 and 255.
Required Data Up to 28 bytes in hex (56 hex characters)
Optional Address 8 byte hex 1-Wire ROM Address

Examples:

To write the record "HA7 is Easy to USE"into the file that contains page 21h of the DS1996 with ROMcode EF00000003B7890C:
http://HA7Net.com/1Wire/WriteFileRecord.html?Address=EF00000003B7890C&RecordNumber=21&Data=484137206973204561737920544F20555345

Παράρτημα Γ- DEMO Εφαρμογή

Κώδικας βιβλιοθηκών

Gethumd.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
```

```
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  gethumd.c - This utility gets the Volts for pins Vad and Vdd from the DS2438.
//
//  Version:   2.00
//  History:

#include <stdio.h>
#include "ownet.h"
#include "atod26.h"
#include "findtype.h"

#define MAXDEVICES 5
#define ONEKBITADD 0x89


//--------------------------------------------------------------------
//  This is the Main routine for debit
//
int main(short argc, char **argv)
{
   char msg[200];
   int portnum = 0;
   float Vdd,Vad;
   double humid,temp;
   int i;
   int numbat,cnt=0;
   uchar famvolt[MAXDEVICES][8];

   // check for required port name
   if (argc != 2)
   {
      sprintf(msg,"1-Wire Net name required on command line!\n"
                  " (example: \"COM1\" (Win32 DS2480),\"/dev/cua0\" "
                  "(Linux DS2480),\"1\" (Win32 TMEX)\n");
      printf("%s",msg);
```

```
   return 0;
}

if((portnum = owAcquireEx(argv[1])) < 0)
{
   printf("Failed to acquire port.\n");
   return 0;
}
else
{

   do
   {
      numbat = FindDevices(portnum,&famvolt[0],SBATTERY_FAM,MAXDEVICES);

      if(numbat == 0)
      {
         if(cnt > 1000)
         {
            cnt = 0;
            printf("No humidity buttons found.\n");
         }
         else
         {
            cnt++;
         }
      }
      else
      {
         for(i=0;i<numbat;i++)
         {
            Vdd = ReadAtoD(portnum,TRUE,&famvolt[0][0]);
            if(Vdd > 5.8)
            {
               Vdd = (float)5.8;
            }
            else if(Vdd < 4.0)
            {
               Vdd = (float) 4.0;
            }

            Vad = ReadAtoD(portnum,FALSE,&famvolt[0][0]);

            temp = Get_Temperature(portnum,&famvolt[0][0]);

            humid = (((Vad/Vdd) - 0.16)/0.0062)/(1.0546 - 0.00216 * temp);
            if(humid > 100)
            {
               humid = 100;
            }
            else if(humid < 0)
```

```
                {
                   humid = 0;
                }

                printf("\n");
                printf("The humidity is:  %4.4f\n", humid);
                printf("Given that the temp was:   %2.2f\n", temp);
                printf("and the volt supply was:   %2.2f\n", Vdd);
                printf("with the volt output was:  %2.2f\n", Vad);
                printf("\n");

             }//for loop
          }

       }while(!key_abort());

       owRelease(portnum);
       printf("Port released.\n");

    }

    return 1;
}
```

Atod20.c

```
//---------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
```

```
//--------------------------------------------------------------------------
//
//  atod20.c - Module(s) to do conversion, setup, read, and write
//             on the DS2450 - 1-Wire Quad A/D Converter.
//
//
//  Version: 2.00
//
//
// --------------------------------------------------------------------------

#include <stdio.h>
#include "ownet.h"
#include "atod20.h"

// --------------------------------------------------------------------------
// Setup A to D control data.  This is hardcoded to 5.12Volt scale at
// 8 bits, but it could be read from a file.
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number is provided to
//                   indicate the symbolic port number.
// 'SerialNum'     - Serial Number of device
// 'ctrl'          - pointer to conrtol data to set (fixed info for now)
// 'msg'           - poniter to string to return message
//
int SetupAtoDControl(int portnum, uchar *SerialNum, uchar *ctrl, char *msg)
{
   uchar i;

   // set the device serial number to the DS2450 device
   owSerialNum(portnum,SerialNum,FALSE);

   // setup CONTROL register
   for (i = 0; i < 8; i += 2)
   {
      ctrl[i] = NO_OUTPUT | BITS_8;
      ctrl[i + 1] = RANGE_512 | ALARM_HIGH_DISABLE | ALARM_LOW_DISABLE;
   }

   // setup ALARM register
   for (i = 8; i < 16; i++)
      ctrl[i] = 0;

   // set return value
   sprintf(msg, "All channels set to 5.12V range at 8 bits");

   return TRUE;
}

//--------------------------------------------------------------------------
// Write A to D with provided buffer.
```

```
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number is provided to
//                   indicate the symbolic port number.
// 'try_overdrive' - True(1) if want to try to use overdrive
// 'SerialNum'     - Serial Number of device
// 'ctrl'          - pointer to conrtol data to write
// 'start_address' - start address to write
// 'end_address'   - end of address to write
//
// Returns: TRUE, success, results read ok
//          FALSE, failure to read
//
int WriteAtoD(int portnum, int try_overdrive, uchar *SerialNum,
               uchar *ctrl, int start_address, int end_address)
{
   uchar send_block[50];
   uchar i;
   short send_cnt=0,flag,address;
   ushort lastcrc16;

   // build the first part of the packet to write
   // reset CRC
   setcrc16(portnum,0);
   send_cnt = 0;
   // write memory command
   send_block[send_cnt++] = 0x55;
   lastcrc16 = docrc16(portnum,0x55);
   // address
   send_block[send_cnt] = (uchar) (start_address & 0xFF);
   lastcrc16 = docrc16(portnum,send_block[send_cnt++]);
   send_block[send_cnt] = (uchar)((start_address >> 8) & 0xFF);
   lastcrc16 = docrc16(portnum,send_block[send_cnt++]);

   // set the device serial number to the DS2450 device
   owSerialNum(portnum,SerialNum,FALSE);

   // select the device
   if (Select(portnum,try_overdrive))
   {
      // loop to write each byte
      for (address = start_address; address <= end_address; address++)
      {
         // build the packet to write
         // init CRC on all but first pass
         if (address != start_address)
            setcrc16(portnum,address);
         // data to write
         send_block[send_cnt] = ctrl[address - start_address];
         lastcrc16 = docrc16(portnum,send_block[send_cnt++]);
         // read CRC16
         send_block[send_cnt++] = 0xFF;
```

```
         send_block[send_cnt++] = 0xFF;
         // echo of byte
         send_block[send_cnt++] = 0xFF;

         // send the block
         flag = owBlock(portnum,FALSE, send_block, send_cnt);

         // check results of block
         if (flag == TRUE)
         {
            // perform crc16 on last 2 bytes
            for (i = (send_cnt - 3); i <= (send_cnt - 2); i++)
               lastcrc16 = docrc16(portnum,send_block[i]);

            // verify crc16 is correct
            if ((lastcrc16 != 0xB001) ||
                (send_block[send_cnt-1] != ctrl[address - start_address]))
               return FALSE;
         }
         else
            return FALSE;

         // reset the packet
         setcrc16(portnum,0);
         send_cnt = 0;
      }
   }
   // no device
   else
      return FALSE;

   return TRUE;
}

// --------------------------------------------------------------------------
// Attempt to do an A to D conversion
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number is provided to
//                   indicate the symbolic port number.
// 'try_overdrive' - True(1) if want to try to use overdrive
// 'SerialNum'     - Serial Number of device
//
//
// Returns: TRUE, success, conversion ok
//          FALSE, failure to do conversion
//
int DoAtoDConversion(int portnum, int try_overdrive, uchar *SerialNum)
{
   uchar send_block[50];
   int i;
   short send_cnt=0;
```

```
ushort lastcrc16;

// set the device serial number to the DS2450 device
owSerialNum(portnum,SerialNum,FALSE);

// access the device
if (Select(portnum,try_overdrive))
{
   setcrc16(portnum,0);
   // create a block to send
   send_block[send_cnt] = 0x3C;
   lastcrc16 = docrc16(portnum,send_block[send_cnt++]);

   // input select mask (all channels)
   send_block[send_cnt] = 0x0F;
   lastcrc16 = docrc16(portnum,send_block[send_cnt++]);

   // read-out control
   send_block[send_cnt] = 0x00;
   lastcrc16 = docrc16(portnum,send_block[send_cnt++]);

   // read CRC16
   send_block[send_cnt++] = 0xFF;
   send_block[send_cnt++] = 0xFF;

   // now send the block
   if (owBlock(portnum,FALSE, send_block, (send_cnt)))
   {
      // check the CRC
      for (i = send_cnt - 2; i < (send_cnt); i++)
         lastcrc16 = docrc16(portnum,send_block[i]);

      // verify CRC16 is correct
      if (lastcrc16 != 0xB001)
      {
         OWERROR(OWERROR_CRC_FAILED);
         return FALSE;
      }

      // if success then apply the strong pullup
      if(!owWriteBytePower(portnum,((send_cnt-1) & 0x1F)))
         return FALSE;
      // delay the max time, 6ms
      msDelay(6);
      // set the pullup back to normal
      if(MODE_NORMAL != owLevel(portnum,MODE_NORMAL))
      {
         OWERROR(OWERROR_LEVEL_FAILED);
         return FALSE;
      }
```

```
         // check conversion over
         if (owReadByte(portnum) == 0xFF)
            return TRUE;
      }
   }

   return FALSE;
}

//--------------------------------------------------------------------------
// Read A to D results
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number is provided to
//                   indicate the symbolic port number.
// 'try_overdrive' - True(1) if want to try to use overdrive
// 'SerialNum'     - Serial Number of device
// 'prslt'         - pointer to array of 4 floats to return voltage results
// 'ctrl'          - pointer to conrtol data for reference when calculating
//                   results.
//
// Returns: TRUE, success, results read ok
//          FALSE, failure to read
//
int ReadAtoDResults(int portnum, int try_overdrive, uchar *SerialNum,
                        float *prslt, uchar *ctrl)
{
   uchar i, send_cnt=0;
   ulong templong;
   uchar rt=FALSE;
   uchar send_block[30];
   ushort lastcrc16;

   setcrc16(portnum,0);

   // set the device serial number to the DS2450 device
   owSerialNum(portnum,SerialNum,FALSE);

   // access the device
   if (Select(portnum,try_overdrive))
   {
      // create a block to send that reads the DS2450
      send_block[send_cnt++] = 0xAA;
      lastcrc16 = docrc16(portnum,0xAA);

      // address block
      send_block[send_cnt++] = 0x00;
      send_block[send_cnt++] = 0x00;
      lastcrc16 = docrc16(portnum,0x00);
      lastcrc16 = docrc16(portnum,0x00);

      // read the bytes
```

```
      for (i = 0; i < 10; i++)
         send_block[send_cnt++] = 0xFF;

      // send the block
      if (owBlock(portnum,FALSE, send_block, send_cnt))
      {
         // perform CRC16
         for (i = send_cnt - 10; i < send_cnt; i++)
            lastcrc16 = docrc16(portnum,send_block[i]);

         // verify CRC16 is correct
         if (lastcrc16 == 0xB001)
         {
            // success
            rt = TRUE;
         }

      }

      // verify read ok
      if (rt == TRUE)
      {
         // convert the value read to floats
         for (i = 3; i < 11; i += 2)  // (1.01)
         {
            templong = ((send_block[i + 1] << 8) | send_block[i]) & 0x0000FFFF;
            prslt[(i - 3) / 2] = (float)((float)(templong / 65535.0) *
               ((ctrl[(i - 3) + 1] & 0x01) ? 5.12 : 2.56)); // (1.01)
         }
      }
   }

   return rt;
}

//--------------------------------------------------------------------------
// Select the current device and attempt overdrive if possible.
//
// 'portnum'       - number 0 to MAX_PORTNUM-1.  This number is provided to
//                   indicate the symbolic port number.
//
// 'try_overdrive' - True(1) if want to try to use overdrive
//
//  Returns: TRUE(1)  current device selected
//           FALSE(0) device not present
//
int Select(int portnum, int try_overdrive)
{
   static uchar current_speed = MODE_NORMAL;

   // attempt to do overdrive
```

```
   if (try_overdrive)
   {
      // verify device is in overdrive
      if (current_speed == MODE_OVERDRIVE)
      {
         if (owAccess(portnum))
            return TRUE;
      }

      if (owOverdriveAccess(portnum))
         current_speed = MODE_OVERDRIVE;
      else
         current_speed = MODE_NORMAL;
   }

   return (owAccess(portnum));
}
```

Atod20.h

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  atod20.h - header file for the Module(s) to do conversion, setup, read,
//             and write on the DS2450 - 1-Wire Quad A/D Converter.
// -------------------------------------------------------------------------
```

```
// functions defined in atod20.c
int SetupAtoDControl(int,uchar *,uchar *,char *);
int WriteAtoD(int,int,uchar *,uchar *,int,int);
int DoAtoDConversion(int,int,uchar *);
int ReadAtoDResults(int,int,uchar *,float *,uchar *);
int Select(int,int);

// family codes of device(s)
#define ATOD_FAM           0x20

// setup
#define NO_OUTPUT          0x00
#define OUTPUT_ON          0x80
#define OUTPUT_OFF         0xC0
#define RANGE_512          0x01
#define RANGE_256          0x00
#define ALARM_HIGH_ENABLE  0x08
#define ALARM_LOW_ENABLE   0x04
#define ALARM_HIGH_DISABLE 0x00
#define ALARM_LOW_DISABLE  0x00
#define BITS_16            0x00
#define BITS_8             0x08
```

Temp10.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  temp10.C - Module to read the DS1920/DS1820 - temperature measurement.
//
//  Version: 2.00
//
//--------------------------------------------------------------------------
//
//
#include "multinet.cpp"
#include "temp10.h"
```

```
//--------------------------------------------------------------------------
// Read the temperature of a DS1920/DS1820
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number was provided to
//                 OpenCOM to indicate the port number.
// 'SerialNum'   - Serial Number of DS1920/DS1820 to read temperature from
// 'Temp '       - pointer to variable where that temperature will be
//                 returned
//
// Returns: TRUE(1)  temperature has been read and verified
//          FALSE(0) could not read the temperature, perhaps device is not
//                   in contact
//
int ReadTemperature(int portnum, uchar *SerialNum, float *Temp)
{
   uchar rt=FALSE;
   uchar send_block[30],lastcrc8;
   int send_cnt, tsht, i, loop=0;
   float tmp,cr,cpc;

   // set the device serial number to the counter device
   owSerialNum(portnum,SerialNum,FALSE);

   for (loop = 0; loop < 2; loop ++)
   {
      // access the device
      if (owAccess(portnum))
      {
         // send the convert command and start power delivery
         if (!owWriteBytePower(portnum,0x44))
            return FALSE;

         // sleep for 1 second
         msDelay(1000);

         // turn off the 1-Wire Net strong pull-up
         if (owLevel(portnum,MODE_NORMAL) != MODE_NORMAL)
            return FALSE;

         // access the device
         if (owAccess(portnum))
         {
            // create a block to send that reads the temperature
            // read scratchpad command
            send_cnt = 0;
            send_block[send_cnt++] = 0xBE;
            // now add the read bytes for data bytes and crc8
            for (i = 0; i < 9; i++)
               send_block[send_cnt++] = 0xFF;

            // now send the block
            if (owBlock(portnum,FALSE,send_block,send_cnt))
            {
               // initialize the CRC8
               setcrc8(portnum,0);
               // perform the CRC8 on the last 8 bytes of packet
               for (i = send_cnt - 9; i < send_cnt; i++)
                  lastcrc8 = docrc8(portnum,send_block[i]);

               // verify CRC8 is correct
               if (lastcrc8 == 0x00)
               {
                  // calculate the high-res temperature
                  tsht = send_block[1]/2;
                  if (send_block[2] & 0x01)
                     tsht |= -128;
                  tmp = (float)(tsht);
                  cr = send_block[7];
                  cpc = send_block[8];
                  if (((cpc - cr) == 1) && (loop == 0))
                     continue;
                  if (cpc == 0)
                     return FALSE;
                  else
                     tmp = tmp - (float)0.25 + (cpc - cr)/cpc;
```

```
                    *Temp = tmp;
                    // success
                    rt = TRUE;
                    break;
                }
             }
          }
       }

   }

   // return the result flag rt
   return rt;
}
```

Thermo21.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  thermo21.c - Thermochron iButton utility functions
//
//  Version: 2.00
//
//    History:
//           1.03 -> 2.00  Reorganization of Public Domain Kit
//                         Convert to global CRC utility functions
//                         Y2K fix.

#include "multinet.cpp"
#include "thermo21.h"
#include <time.h>
#include <stdio.h>

// Include files for the Palm and Visor
#ifdef __MC68K__
#include <DateTime.h>
#include <TimeMgr.h>
#else
#include <string.h>
#endif

// Local Function Prototypes
static int ThermoStep(int,ThermoStateType *,ThermoScript *,int *,int *,int *,char *);
static int ReadPages(int,int,int,int *,uchar *);
static int WriteScratch(int,uchar *,int,int);
static int CopyScratch(int,int,int);
static int WriteMemory(int,uchar *, int, int);

// global state information
static int current_speed[MAX_PORTNUM];
```

```
//--------------------------------------------------------------------------
// The 'DownloadThermo' downloads the specified Thermochron in 'SerialNum'
// and puts the data in the state variable 'ThermoState'.  Progress output
// is printed to the specified file 'fp'.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
// 'SerialNum'   - Device serial number to download
// 'ThermoState' - pointer to a structure type that holds the raw and
//                 translated Thermochron data.
// 'fp'          - file pointer to print status information to
//
// Returns:   TRUE (1) : Thermochron download with raw data in ThermoState
//            FALSE (0): not downloaded.  Abort due to repeated errors
//                       or user keypress.
//
int DownloadThermo(int portnum, uchar *SerialNum,
                                ThermoStateType *ThermoState, FILE *fp)
{
   // set the serial num
   owSerialNum(portnum, SerialNum, FALSE);

   // run the script and download thermochron
   return RunThermoScript(portnum,ThermoState,Download,fp);
}

//--------------------------------------------------------------------------
// The 'ReadThermoStatus' reads the Thermochron status in 'SerialNum'
// and puts the data in the state variable 'ThermoState'.  Progress output
// is printed to the specified file 'fp'.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
// 'SerialNum'   - Device serial number to download
// 'ThermoState' - pointer to a structure type that holds the raw and
//                 translated Thermochron data.
// 'fp'          - file pointer to print status information to
//
// Returns:   TRUE (1) : Thermochron status read with raw data in ThermoState
//            FALSE (0): status not read.  Abort due to repeated errors
//                       or user keypress.
//
int ReadThermoStatus(int portnum, uchar *SerialNum,
                                ThermoStateType *ThermoState, FILE *fp)
{

   // set the serial num
   owSerialNum(portnum, SerialNum, FALSE);

   // run the script and read status of thermochron
   return RunThermoScript(portnum,ThermoState,GetStatus,fp);
}

//--------------------------------------------------------------------------
// The 'MissionThermo' starts a new Thermochron mission on 'SerialNum'
// from the state information provided in 'ThermoState'. Progress output
// is printed to the specified file 'fp'.
//
// 'portnum'     - number 0 to MAX_PORTNUM-1.  This number is provided to
//                 indicate the symbolic port number.
// 'SerialNum'   - Device serial number to download
// 'ThermoState' - pointer to a structure type that holds the raw and
//                 translated Thermochron data.
// 'fp'          - file pointer to print status information to
//
// Returns:   TRUE (1) : Thermochron missioned
//            FALSE (0): not missioned.  Abort due to repeated errors
//                       or user keypress.
//
int MissionThermo(int portnum, uchar *SerialNum, ThermoStateType *ThermoState,FILE *fp)
{
   // set the serial num
     owSerialNum(portnum, SerialNum, FALSE);

   // run the script and mission thermochron
```

```
   return RunThermoScript(portnum,ThermoState,Mission,fp);
}

//--------------------------------------------------------------------------
// Run the specified script.  Return TRUE if all steps completed else FALSE.
// Status is printed to file 'fp'.
//
int RunThermoScript(int portnum, ThermoStateType *ThermoState,
                    ThermoScript script[], FILE *fp)
{
   char msg[256],LastDescription[256],LastMsg[256];
   int StepCount,SubStep,ErrorCount,Status;
   int last_clear_step=0;

   // reset the step to the begining
   StepCount = 0;
   SubStep = 0;
   ErrorCount = 0;
   Status = STATUS_INPROGRESS;

   LastDescription[0] = 0;
   LastMsg[0] = 0;

   // loop to perform all of the steps to download the Thermochron
   do
   {
      // switch on the status of the last step done
      switch(Status)
      {
         // step complete so go to the next
         case STATUS_STEP_COMPLETE:
            StepCount++;
            SubStep = 0;
            ErrorCount = 0;
            Status = STATUS_INPROGRESS;
            LastDescription[0] = 0;
            LastMsg[0] = 0;
            break;
         // in progress so call again
         case STATUS_INPROGRESS:
            // record the step position of the last memory clear
            // this is in case we need to attempt a clear again
            if (script[StepCount].Step == ST_CLEAR_SETUP)
               last_clear_step = StepCount;

            // print step description if different
            if (strcmp(LastDescription,
                script[StepCount].StepDescription) != 0)
            {
               fprintf(fp,"%s --> ",script[StepCount].StepDescription);
               sprintf(LastDescription,"%s",script[StepCount].StepDescription);
            }

            // perform a step in the job
            Status = ThermoStep(portnum,ThermoState,&script[StepCount],
                                &SubStep,&Status, &ErrorCount, msg);

            // print results if different
            if (strcmp(LastMsg,msg) != 0)
            {
               fprintf(fp,"%s\n",msg);
               sprintf(LastMsg,"%s",msg);
            }
            else
               fprintf(fp,".");

            break;
         // encountered a transient error
         case STATUS_ERROR_TRANSIENT:
            // check if transient error is a memory clear
            if (script[StepCount].Step == ST_CLEAR_VERIFY)
            {
               // put back to starting clear over again
               StepCount = last_clear_step;
               SubStep = 0;
               ErrorCount = 0;
```

```
                Status = STATUS_INPROGRESS;
                break;
             }
             // if 20 tansient errors in a row then abort
             if (ErrorCount > 20)
                Status = STATUS_ERROR_HALT;
             else
                Status = STATUS_INPROGRESS;
             break;
          // all steps complete
          case STATUS_COMPLETE:
             fprintf(fp,"End script normally\n");
             return TRUE;
          // non-recoverable error
          case STATUS_ERROR_HALT:
             fprintf(fp,"Aborting script due to non-recoverable error\n");
             return FALSE;
       }
    }
    while (1);
//   while (!key_abort());

    // key abort
    fprintf(fp,"Aborting script due to key press\n");
    return FALSE;
}


//---------------------------------------------------------------------------
//  Use the script to perform a step and return.
//
int ThermoStep(int portnum, ThermoStateType *ThermoState,
               ThermoScript *StateScript, int *SubStep,
               int *Status, int *ErrorCount, char *msg)
{
   short  rslt;
   static int read_page_num, read_pages, write_addr, write_len;
   static uchar *read_buf, *write_buf;
   static uchar tbuf[5];

   // do the current step
   switch (StateScript->Step)
   {
      // the operation is complete
      case ST_FINISH:
         sprintf(msg,"Operation complete");
         *Status = STATUS_COMPLETE;
         break;

      // read the mission status page
      case ST_READ_STATUS:
         read_page_num = STATUS_PAGE;
         read_pages = 1;
         read_buf = ThermoState->MissStat.status_raw;
         sprintf(msg,"Ready to read status page %d",
                     read_page_num);
         *Status = STATUS_STEP_COMPLETE;
         break;

      // set up to read the alarm registers
      case ST_READ_ALARM:
         read_page_num = 17;
         read_pages = 3;
         read_buf = ThermoState->AlarmData.alarm_raw;
         sprintf(msg,"Ready to read alarm pages %d to %d",
                     read_page_num, read_page_num + read_pages - 1);
         *Status = STATUS_STEP_COMPLETE;
         break;

      // set up to read the histogram data
      case ST_READ_HIST:
         read_page_num = 64;
         read_pages = 4;
         read_buf = ThermoState->HistData.hist_raw;
         sprintf(msg,"Ready to read histogram pages %d to %d",
                     read_page_num, read_page_num + read_pages - 1);
```

```
            *Status = STATUS_STEP_COMPLETE;
            break;

         // set up to read the log data
         case ST_READ_LOG:
            read_page_num = 128;
            read_pages = 64;
            read_buf = ThermoState->LogData.log_raw;
            sprintf(msg,"Ready to read log pages %d to %d",
                        read_page_num, read_page_num + read_pages - 1);
            *Status = STATUS_STEP_COMPLETE;
            break;

         // read the specified pages
         case ST_READ_PAGES:
            // check for last page
            if (*SubStep == 0)
               // set the sub-step to the current page being read
               *SubStep =  read_page_num;
            // read the status page
            rslt = ReadPages(portnum, read_page_num, read_pages, SubStep, read_buf);
            if (rslt == FALSE)
            {
               sprintf(msg,"Thermochron not on 1-Wire Net");
#ifdef __MC68K__
               *Status = STATUS_ERROR_HALT;
#else
                           *Status = STATUS_INPROGRESS;
#endif
            }
            else
            {
               sprintf(msg,"Pages read from Thermochron");
               *Status = STATUS_STEP_COMPLETE;
            }
            break;

         // setup the clear memory
         case ST_CLEAR_SETUP:
            // create a small buff to write to start the clear memory
            tbuf[0] = 0x40;
            write_buf = &tbuf[0];
            write_len = 1;
            write_addr = 0x20E;
            sprintf(msg,"Write to setup clear memory");
            *Status = STATUS_STEP_COMPLETE;
            break;

         // clear the memory
         case ST_CLEAR_MEM:
            // set the clear memory command (not check return because verify)
            if(!owAccess(portnum))
            {
               OWERROR(OWERROR_ACCESS_FAILED);
            }
            if(!owWriteByte(portnum,0x3C))
            {
               OWERROR(OWERROR_WRITE_BYTE_FAILED);
            }
            msDelay(3);
            if(!owTouchReset(portnum))
            {
               OWERROR(OWERROR_RESET_FAILED);
            }
            sprintf(msg,"Clear memory command sent");
            *Status = STATUS_STEP_COMPLETE;
            break;

         // clear the memory
         case ST_CLEAR_VERIFY:
            // look at the memory clear bit
            if ((ThermoState->MissStat.status_raw[0x14] & 0x40) == 0x40)
            {
               sprintf(msg,"Memory is clear");
               *Status = STATUS_STEP_COMPLETE;
            }
```

```
            else
            {
               sprintf(msg,"Memory did NOT clear");
               *Status = STATUS_ERROR_TRANSIENT;
            }
            break;

         // setup write time, clock alarm, control, trips
         case ST_WRITE_TIME:
            // create the write buffer
            FormatMission(&ThermoState->MissStat);
            write_buf = &ThermoState->MissStat.status_raw[0x00];
            write_len = 13;
            write_addr = 0x200;
            sprintf(msg,"Write time, clock alarm, and trips setup");
            *Status = STATUS_STEP_COMPLETE;
            break;

         // write the control, mission delay and clear flags
         case ST_WRITE_CONTROL:
            write_buf = &ThermoState->MissStat.status_raw[0x0E];
            write_len = 7;
            write_addr = 0x20E;
            sprintf(msg,"Write control, mission delay, clear flags setup");
            *Status = STATUS_STEP_COMPLETE;
            break;

         case ST_WRITE_RATE:
            write_buf = &ThermoState->MissStat.status_raw[0x0D];
            write_len = 1;
            write_addr = 0x20D;
            sprintf(msg,"Write sample rate setup");
            *Status = STATUS_STEP_COMPLETE;
            break;

         // write the specified memory location
         case ST_WRITE_MEM:
            if (WriteMemory(portnum, write_buf, write_len, write_addr))
            {
               sprintf(msg,"Memory written to Thermochron");
               *Status = STATUS_STEP_COMPLETE;
            }
            else
            {
               sprintf(msg,"Thermochron not on 1-Wire Net");
               *Status = STATUS_INPROGRESS;
            }

         default:
              break;
      }

      return *Status;
   }

   //--------------------------------------------------------------------------
   //  Read a specified number of pages in overdrive
   //
   // 'portnum'  - number 0 to MAX_PORTNUM-1.  This number is provided to
   //              indicate the symbolic port number.
   //
   int ReadPages(int portnum, int start_pg, int num_pgs, int *last_pg, uchar *finalbuf)
   {
      int skip_overaccess = 0, skip_access = 0;
      uchar pkt[60];
      int len,i;
      uchar  SerialNumber[8];
      ushort lastcrc16;

      // read the rom number
      owSerialNum(portnum,SerialNumber,TRUE);

   #ifndef __MC68K__
      // verify device is in overdrive
      if (current_speed[portnum] == MODE_OVERDRIVE)
      {
```

```
      if (owVerify(portnum,FALSE))
         skip_overaccess = 1;
   }

   if (!skip_overaccess)
   {
      if (owOverdriveAccess(portnum))
         current_speed[portnum] = MODE_OVERDRIVE;
      else
         current_speed[portnum] = MODE_NORMAL;
   }
#endif

   // loop while there is pages to read
   do
   {
      // create a packet to read a page
      len = 0;
      setcrc16(portnum,0);
      // optional skip access on subsequent pages
      if (!skip_access)
      {
         // match
         pkt[len++] = 0x55;
         // rom number
         for (i = 0; i < 8; i++)
            pkt[len++] = SerialNumber[i];
         // read memory with crc command
         pkt[len] = 0xA5;
         lastcrc16 = docrc16(portnum,pkt[len++]);
         // address
         pkt[len] = (uchar)((*last_pg << 5) & 0xFF);
         lastcrc16 = docrc16(portnum,pkt[len++]);
         pkt[len] = (uchar)(*last_pg >> 3);
         lastcrc16 = docrc16(portnum,pkt[len++]);
      }

      // set 32 reads for data and 2 for crc
      for (i = 0; i < 34; i++)
         pkt[len++] = 0xFF;

      // send the bytes
      if (owBlock(portnum,!skip_access,pkt,len))
      {
         // calucate the CRC over the last 34 bytes
         for (i = 0; i < 34; i++)
            lastcrc16 = docrc16(portnum,pkt[len - 34 + i]);

         // check crc
         if (lastcrc16 == 0xB001)
         {
            // copy the data into the buffer
            for (i = 0; i < 32; i++)
               finalbuf[i + (*last_pg - start_pg) * 32] = pkt[len - 34 + i];

            // change number of pages
            *last_pg = *last_pg + 1;

            // now skip access
            skip_access = TRUE;
         }
         else
            return FALSE;

      }
      else
         return FALSE;

   }
   while ((*last_pg - start_pg) < num_pgs);

   return TRUE;
}

//--------------------------------------------------------------------------}
// Write a memory location. Data must all be on the same page
```

```
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number is provided to
//              indicate the symbolic port number.
//
int WriteMemory(int portnum, uchar *Buf, int ln, int adr)
{
   // write to scratch and then copy
   if (WriteScratch(portnum,Buf,ln,adr))
      return CopyScratch(portnum,ln,adr);

   return FALSE;
}

//--------------------------------------------------------------------------}
// Write the scratch pad
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number is provided to
//              indicate the symbolic port number.
//
int WriteScratch(int portnum, uchar *Buf, int ln, int adr)
{
   int i;
   uchar pbuf[80];

   // check for alarm indicator
   if (owAccess(portnum))
   {
      // construct a packet to send
      pbuf[0] = 0x0F; // write scratch command
      pbuf[1] = (adr & 0xFF); // address 1
      pbuf[2] = ((adr >> 8) & 0xFF); // address 2

      // the write bytes
      for (i = 0; i < ln; i++)
        pbuf[3 + i] = (uchar)(Buf[i]); // data

      // perform the block
      if (!owBlock(portnum,FALSE,pbuf,ln+3))
         return FALSE;

      // Now read back the scratch
      if (owAccess(portnum))
      {
         // construct a packet to send
         pbuf[0] = 0xAA; // read scratch command
         pbuf[1] = 0xFF; // address 1
         pbuf[2] = 0xFF; // address 2
         pbuf[3] = 0xFF; // offset

         // the write bytes
         for (i = 0; i < ln; i++)
            pbuf[4 + i] = 0xFF; // data

         // perform the block
         if (!owBlock(portnum,FALSE,pbuf,ln+4))
            return FALSE;

         // read address 1
         if (pbuf[1] != (adr & 0xFF))
            return FALSE;
         // read address 2
         if (pbuf[2] != ((adr >> 8) & 0xFF))
            return FALSE;
         // read the offset
         if (pbuf[3] != ((adr + ln - 1) & 0x1F))
            return FALSE;
         // read and compare the contents
         for (i = 0; i < ln; i++)
         {
            if (pbuf[4 + i] != Buf[i])
              return FALSE;
         }
         // success
         return TRUE;
      }
   }
```

```
   OWERROR(OWERROR_ACCESS_FAILED);
   return FALSE;
}


//--------------------------------------------------------------------------}
// Copy the scratch pad
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number is provided to
//              indicate the symbolic port number.
//
int CopyScratch(int portnum, int ln, int adr)
{
   int i;
   uchar pbuf[50];

   // check for alarm indicator
   if (owAccess(portnum))
   {
      // construct a packet to send
      pbuf[0] = 0x55;                  // copy scratch command
      pbuf[1] = (adr & 0xFF);          // address 1
      pbuf[2] = ((adr >> 8) & 0xFF);   // address 2
      pbuf[3] = (adr + ln - 1) & 0x1F; // offset
      for (i = 0; i <= 9; i++)
         pbuf[4 + i] = 0xFF;           // result of copy

      // perform the block
      if (owBlock(portnum,FALSE,pbuf,14))
      {
         if ((pbuf[13] == 0x55) ||
             (pbuf[13] == 0xAA))
           return TRUE;
      }
   }

   OWERROR(OWERROR_ACCESS_FAILED);
   return FALSE;
}

//--------------------------------------------------------------------------
//  Interpret the Status by looking at the 'raw' portion of the
//  mission status structure.
//
void InterpretStatus(MissionStatus *mstatus)
{
   timedate td,tdtmp;
   int offset;
   ulong tmtmp;

#ifndef __MC68K__
   time_t tlong;
   time_t temp;
   struct tm *tstruct;
#endif


   // mission in progress flag
   mstatus->mission_in_progress = (0x20 & mstatus->status_raw[0x14]) >> 5;

   // sample rate
   mstatus->sample_rate = mstatus->status_raw[0x0D];

   // rollover enabled
   mstatus->rollover_enable = (0x08 & mstatus->status_raw[0x0E]) >> 3;

   // startdelay
   mstatus->start_delay = (mstatus->status_raw[0x13] << 8) |
                           mstatus->status_raw[0x12];

   // number of samples in this mission
#ifdef __MC68K__
   mstatus->mission_samples = (((ulong) mstatus->status_raw[0x1C]) * 65536) +
                                                    (((ulong) mstatus-
>status_raw[0x1B]) * 256) +
```

```
                                                    ((ulong) mstatus-
>status_raw[0x1A]);
#else
   mstatus->mission_samples = (mstatus->status_raw[0x1C] << 16) |
                              (mstatus->status_raw[0x1B] << 8) |
                               mstatus->status_raw[0x1A];
#endif



   // total number of samples
#ifdef __MC68K__
   mstatus->samples_total   = (((ulong) mstatus->status_raw[0x1F]) * 65536) +
                                                    (((ulong) mstatus-
>status_raw[0x1E]) * 256) +
                                                    ((ulong) mstatus-
>status_raw[0x1D]);
#else
   mstatus->samples_total = (mstatus->status_raw[0x1F] << 16) |
                            (mstatus->status_raw[0x1E] << 8) |
                             mstatus->status_raw[0x1D];
#endif

   // temperature thresholds
   mstatus->high_threshold = mstatus->status_raw[0x0C];
   mstatus->low_threshold = mstatus->status_raw[0x0B];

   // rollover occurred
   if ((mstatus->mission_samples > 2048) && mstatus->rollover_enable)
      mstatus->rollover_occurred = 1;
   else
      mstatus->rollover_occurred = 0;

   // current real-time clock value
   offset = 0x00;
   td.second = BCDToBin((uchar)(mstatus->status_raw[offset] & 0x7F));
   td.minute = BCDToBin((uchar)(mstatus->status_raw[offset + 1] & 0x7F));
   // check for 12 hour mode
   if (mstatus->status_raw[offset + 2] & 0x40)
   {
      td.hour = BCDToBin((uchar)(mstatus->status_raw[offset + 2] & 0x1F));
      // check for PM
      if (mstatus->status_raw[offset + 2] & 0x20)
         td.hour += 12;
   }
   else
      td.hour = BCDToBin((uchar)(mstatus->status_raw[offset + 2] & 0x3F));
   td.day = BCDToBin((uchar)(mstatus->status_raw[offset + 4] & 0x3F));
   td.month = BCDToBin((uchar)(mstatus->status_raw[offset + 5] & 0x1F));
   td.year = BCDToBin(mstatus->status_raw[offset + 6]) + 1900;
   // check for century bit
   if (mstatus->status_raw[offset + 5] & 0x80)
      td.year = BCDToBin(mstatus->status_raw[offset + 6]) + 2000; // (2.00)
   // convert to seconds since 1970
   mstatus->current_time = DateToSeconds(&td);

   // date/time when mission started
   offset = 0x15;
   td.second = (uchar)0;
   td.minute = BCDToBin((uchar)(mstatus->status_raw[offset] & 0x7F));
   // check for 12 hour mode
   if (mstatus->status_raw[offset + 1] & 0x40)
   {
      td.hour = BCDToBin((uchar)(mstatus->status_raw[offset + 1] & 0x1F));
      // check for PM
      if (mstatus->status_raw[offset + 1] & 0x20)
         td.hour += 12;
   }
   else
      td.hour = BCDToBin((uchar)(mstatus->status_raw[offset + 1] & 0x3F));
   td.day = BCDToBin((uchar)(mstatus->status_raw[offset + 2] & 0x3F));
   td.month = BCDToBin((uchar)(mstatus->status_raw[offset + 3] & 0x1F));
   td.year = BCDToBin((uchar)(mstatus->status_raw[offset + 4])); // (2.00)
   // (2.00) logic to decide on century of mission stamp
   // check if century bit set in mission stamp
   if (mstatus->status_raw[offset + 3] & 0x80)
```

```
      td.year += 2000;
   // check in mission in progress
   else if (mstatus->mission_in_progress)
   {
      // calculate the mission start year back from real time clock
      tmtmp = mstatus->current_time -
            (mstatus->sample_rate * mstatus->mission_samples * 60);
      SecondsToDate(&tdtmp,tmtmp);
      td.year = tdtmp.year;
   }
   else
   {
      // mission stopped so get century by year window
      if (td.year <= 70)
         td.year += 2000;
      else
         td.year += 1900;
   }
   // convert to seconds since 1970
   if ((td.month == 0) || (td.day == 0))
      mstatus->mission_start_time = 0;
   else
      mstatus->mission_start_time = DateToSeconds(&td);

   // download stations time of reading
#ifdef __MC68K__
   mstatus->download_time = TimGetSeconds();
#else
   temp = time(&tlong);
   tstruct = localtime(&tlong);
   td.day = tstruct->tm_mday;
   td.month = tstruct->tm_mon + 1;  // (1.01)
   td.year = tstruct->tm_year + 1900;
   td.hour = tstruct->tm_hour;
   td.minute = tstruct->tm_min;
   td.second = tstruct->tm_sec;
   mstatus->download_time = DateToSeconds(&td);
#endif

   // skip alarm modes and status for now
}

//--------------------------------------------------------------------------
// Take the Mission Status structure and create new raw data to start
// a new mission.
//
void FormatMission(MissionStatus *mstatus)
{
   int i;
   time_t tlong;
   time_t temp;
#ifndef __MC68K__
   struct tm *tstruct;
#else
   DateTimePtr tstruct = NULL;
#endif
   // clear the buffer
   for (i = 0; i < 32; i++)
      mstatus->status_raw[i] = 0;

   // Real Time Clock
#ifdef __MC68K__
   tlong = TimGetSeconds();
#else
   temp = time(&tlong);
#endif
   tlong++;  // add 1 second

#ifdef __MC68K__
   TimSecondsToDateTime(tlong, tstruct);
#else
   tstruct = localtime(&tlong);
#endif

   // convert to BCD
#ifdef __MC68K__
```

```
   mstatus->status_raw[0x00] = ToBCD((short)tstruct->second);
   mstatus->status_raw[0x01] = ToBCD((short)tstruct->minute);
   mstatus->status_raw[0x02] = ToBCD((short)tstruct->hour);
   mstatus->status_raw[0x03] = ToBCD((short)(tstruct->weekDay + 1));
   mstatus->status_raw[0x04] = ToBCD((short)tstruct->day);
   mstatus->status_raw[0x05] = ToBCD((short)(tstruct->month + 1));
   if (tstruct->year >= 100)
      mstatus->status_raw[0x05] |= 0x80;
   mstatus->status_raw[0x06] = ToBCD((short)(tstruct->year % 100));
#else
   mstatus->status_raw[0x00] = ToBCD((short)tstruct->tm_sec);
   mstatus->status_raw[0x01] = ToBCD((short)tstruct->tm_min);
   mstatus->status_raw[0x02] = ToBCD((short)tstruct->tm_hour);
   mstatus->status_raw[0x03] = ToBCD((short)(tstruct->tm_wday + 1));
   mstatus->status_raw[0x04] = ToBCD((short)tstruct->tm_mday);
   mstatus->status_raw[0x05] = ToBCD((short)(tstruct->tm_mon + 1));
   if (tstruct->tm_year >= 100)
      mstatus->status_raw[0x05] |= 0x80;
   mstatus->status_raw[0x06] = ToBCD((short)(tstruct->tm_year % 100));
#endif

   // Real Time clock Alarm (leave 0's)
   // Low temp alarm
   mstatus->status_raw[0x0B] = mstatus->low_threshold;
   // High temp alarm
   mstatus->status_raw[0x0C] = mstatus->high_threshold;
   // sample rate
   mstatus->status_raw[0x0D] = mstatus->sample_rate;
   // control
   mstatus->status_raw[0x0E] = 0x40;
   if (mstatus->rollover_enable)
      mstatus->status_raw[0x0E] |= 0x08;
   // mission start delay
   mstatus->status_raw[0x12] = mstatus->start_delay & 0xFF;
   mstatus->status_raw[0x13] = (mstatus->start_delay >> 8) & 0xFF;
}

//---------------------------------------------------------------------------
// Convert an integer to a 1 Byte BCD number (99 max)
//
uchar ToBCD(short num)
{
   uchar rtbyte;

   rtbyte = (num - ((num / 10) * 10)) & 0x0F;
   rtbyte = rtbyte | ((num / 10) << 4);

   return rtbyte;
}


//---------------------------------------------------------------------------
// Take the Mission Status structure and convert to string format
//
void MissionStatusToString(MissionStatus *mstatus, int ConvertToF, char *str)
{
   int cnt=0,i;
   timedate td;
#ifndef __MC68K__
   time_t tlong;
   time_t temp;
   struct tm *tstruct;
#else
   DateTimePtr tstruct = NULL;
#endif

   // title
#ifdef __MC68K__
   cnt += sprintf(&str[cnt],"Stat For DS1921:");
#else
   cnt += sprintf(&str[cnt],"Mission State\n-------------\n");
   cnt += sprintf(&str[cnt],"Serial Number of DS1921: ");
#endif

   // serial number
   for (i = 7; i >= 0; i--)
```

```
            cnt += sprintf(&str[cnt],"%02X",mstatus->serial_num[i]);

      // mission state
      if (mstatus->mission_in_progress)
         cnt += sprintf(&str[cnt],"\nMission is in progress\n");
      else
         cnt += sprintf(&str[cnt],"\nMission is ended\n");

      // sample rate
      cnt += sprintf(&str[cnt],"Sample rate: %d minute(s)\n",mstatus->sample_rate);

      // rollover
      cnt += sprintf(&str[cnt],"Roll-Over Enabled: ");
      if (mstatus->rollover_enable)
         cnt += sprintf(&str[cnt],"yes\n");
      else
         cnt += sprintf(&str[cnt],"no\n");

      cnt += sprintf(&str[cnt],"Roll-Over Occurred: ");
      if (mstatus->rollover_occurred)
         cnt += sprintf(&str[cnt],"yes\n");
      else
         cnt += sprintf(&str[cnt],"no\n");

      // mission start time
      if (mstatus->start_delay == 0)
      {
         SecondsToDate(&td,mstatus->mission_start_time);
         if (mstatus->mission_start_time == 0)
            cnt += sprintf(&str[cnt],"Mission Start time: not started yet\n");
         else
            cnt += sprintf(&str[cnt],"Mission Start: %02d/%02d/%04d  %02d:%02d:%02d\n",
               td.month,td.day,td.year,td.hour,td.minute,td.second);

      }
      else
         cnt += sprintf(&str[cnt],"Mission Start time: na\n");

      // mission start delay
      cnt += sprintf(&str[cnt],"Mission Start delay: %d minute(s)\n",mstatus->start_delay);

      // mission samples
      cnt += sprintf(&str[cnt],"Mission Samples: %ld\n",mstatus->mission_samples);

      // device total samples
      cnt += sprintf(&str[cnt],"Device total samples: %ld\n",mstatus->samples_total);

      // temperature display mode
      cnt += sprintf(&str[cnt],"Temp displayed in: ");
      if (ConvertToF)
         cnt += sprintf(&str[cnt],"(Fahrenheit)\n");
      else
         cnt += sprintf(&str[cnt],"(Celsius)\n");

      // thresholds
      cnt += sprintf(&str[cnt],"High Threshold: %6.1f\n",
             TempToFloat(mstatus->high_threshold,ConvertToF));

      cnt += sprintf(&str[cnt],"Low Threshold: %6.1f\n",
             TempToFloat(mstatus->low_threshold,ConvertToF));

      // time from D1921
      SecondsToDate(&td,mstatus->current_time);
      cnt += sprintf(&str[cnt],"Current Real-Time Clock from DS1921: %02d/%02d/%04d
%02d:%02d:%02d\n",
          td.month,td.day,td.year,td.hour,td.minute,td.second);

#ifndef __MC68K__
      // current PC time
      temp = time(&tlong);
      tstruct = localtime(&tlong);

      cnt += sprintf(&str[cnt],"Current PC Time: %02d/%02d/%04d  %02d:%02d:%02d\n",
          tstruct->tm_mon + 1,tstruct->tm_mday,tstruct->tm_year + 1900,
          tstruct->tm_hour,tstruct->tm_min,tstruct->tm_sec);
#endif
```

```
   // zero terminate string
   str[cnt] = 0;
}

//--------------------------------------------------------------------
//  Interpret the Histogram by looking at the 'raw' portion of the
//  Histogram structure.  Store the temperature range values in Celsius.
//
void InterpretHistogram(Histogram *hist)
{
   int i;

   // loop through each bin value
   for (i = 0; i < 126; i += 2) // (2.00)
   {
      // get the bin value
      hist->bin_count[i / 2] = hist->hist_raw[i] | (hist->hist_raw[i + 1] << 8);

      // start value for this bin
      hist->start_range[i / 2] = TempToFloat((uchar)((i / 2) << 2),FALSE);

      // end value for this bin
      hist->end_range[i / 2] = TempToFloat((uchar)(((i / 2) << 2) | 0x03),FALSE);
   }
}

//-----------------------------------------------------------------------
// Take the Histogram structure and convert to string format
//
void HistogramToString(Histogram *hist, int ConvertToF, char *str)
{
   int cnt=0,i;

   // title
   cnt += sprintf(&str[cnt],"Temperature Histogram\n---------------------\n"
                            "Format: [Temp Range, Count] ");

   if (ConvertToF)
      cnt += sprintf(&str[cnt],"(Fahrenheit)\n");
   else
      cnt += sprintf(&str[cnt],"(Celsius)\n");

   // loop through bins
   for (i = 0; i < 63; i++) // (2.00)
   {
      cnt += sprintf(&str[cnt],"%6.1f to %6.1f, %d\n",
                     (ConvertToF) ? CToF(hist->start_range[i]): hist->start_range[i],
                     (ConvertToF) ? CToF(hist->end_range[i]): hist->end_range[i],
                      hist->bin_count[i]);
   }

   // zero terminate string
   str[cnt] = 0;
}

//--------------------------------------------------------------------
//  Interpret the Temperature Alarm Event data by looking at the 'raw'
//  portion of the TempAlarmEvents structure.  Mission Status is needed
//  to interpret the events.
//
void InterpretAlarms(TempAlarmEvents *alarm, MissionStatus *mstatus)
{
   int i;
   ulong event_mission_count;
   uchar duration;

   // low events
   alarm->num_low = 0;
   for (i = 0; i < 48; i += 4)
   {
      // get the mission start count of this event
      event_mission_count = (alarm->alarm_raw[i + 2] << 16) |
                            (alarm->alarm_raw[i + 1] << 8) |
                             alarm->alarm_raw[i];
```

```
      // check if done with low events
      if (!event_mission_count)
         break;

      // get the duration
      duration = alarm->alarm_raw[i + 3];

      // calculate the start time
      alarm->low_start_time[alarm->num_low] =
          mstatus->mission_start_time +
          (event_mission_count - 1) * (mstatus->sample_rate * 60);

      // calculate the end time
      alarm->low_end_time[alarm->num_low] =
          alarm->low_start_time[alarm->num_low] +
          (duration - 1) * (mstatus->sample_rate * 60);

      // increment number of low events
      alarm->num_low++;
   }

   // high events
   alarm->num_high = 0;
   for (i = 48; i < 96; i += 4)
   {
      // get the mission start count of this event
      event_mission_count = (alarm->alarm_raw[i + 2] << 16) |
                            (alarm->alarm_raw[i + 1] << 8) |
                             alarm->alarm_raw[i];

      // check if done with low events
      if (!event_mission_count)
         break;

      // get the duration
      duration = alarm->alarm_raw[i + 3];

      // calculate the start time
      alarm->high_start_time[alarm->num_high] =
          mstatus->mission_start_time +
          (event_mission_count - 1) * (mstatus->sample_rate * 60);

      // calculate the end time
      alarm->high_end_time[alarm->num_high] =
          alarm->high_start_time[alarm->num_high] +
          (duration - 1) * (mstatus->sample_rate * 60);

      // increment number of low events
      alarm->num_high++;
   }
}

//-----------------------------------------------------------------------------
// Take the Temperature Alarms Events structure and convert to string
// format
//
void AlarmsToString(TempAlarmEvents *alarm, char *str)
{
   int i, cnt=0;
   timedate td;

   // title
   cnt += sprintf(&str[cnt],"Temperature Alarms\n------------------\n"
                            "Format: [(HIGH/LOW), Time/Date Range]\n");

   // loop through each low alarm
   for (i = 0; i < alarm->num_low; i++)
   {
      cnt += sprintf(&str[cnt],"LOW  , ");
      // start time
      SecondsToDate(&td,alarm->low_start_time[i]);
      cnt += sprintf(&str[cnt]," %02d/%02d/%04d  %02d:%02d  to  ",
         td.month,td.day,td.year,td.hour,td.minute);
      // end time
      SecondsToDate(&td,alarm->low_end_time[i]);
      cnt += sprintf(&str[cnt]," %02d/%02d/%04d  %02d:%02d\n",
```

```
            td.month,td.day,td.year,td.hour,td.minute);
      }

      // loop through each high alarm
      for (i = 0; i < alarm->num_high; i++)
      {
         cnt += sprintf(&str[cnt],"HIGH , ");
         // start time
         SecondsToDate(&td,alarm->high_start_time[i]);
         cnt += sprintf(&str[cnt]," %02d/%02d/%04d  %02d:%02d  to  ",
            td.month,td.day,td.year,td.hour,td.minute);
         // end time
         SecondsToDate(&td,alarm->high_end_time[i]);
         cnt += sprintf(&str[cnt]," %02d/%02d/%04d  %02d:%02d\n",
            td.month,td.day,td.year,td.hour,td.minute);
      }

      // zero terminate string
      str[cnt] = 0;
   }

   //---------------------------------------------------------------------
   //  Interpret the Log data by looking at the 'raw'
   //  portion of the Log structure.  Mission Status is needed
   //  to interpret when the logs occurred.
   //
   void InterpretLog(Log *log, MissionStatus *mstatus)
   {
      ulong loops=0,overlap=0,lastlog=2048,i;
      int logcnt=0;

      // check if wrap occurred
      if (mstatus->rollover_occurred)
      {
         // calculate the number loops
         loops = (mstatus->mission_samples / 2048) - 1;
         // calculate the number of overlap
         overlap = mstatus->mission_samples % 2048;
         log->num_log = 2048;
      }
      else
      {
         log->start_time = mstatus->mission_start_time;
         if (mstatus->mission_samples > 2048)  // (1.02)
            lastlog = 2048;
         else
            lastlog = mstatus->mission_samples;
         log->num_log = (int)lastlog;
      }

      // set the interval
      log->interval = mstatus->sample_rate * 60;

      // caluclate the start time of the first log value
      log->start_time = mstatus->mission_start_time +
         loops * 2048 * log->interval + overlap * log->interval;

      // loop to fill in the remainder first
      for (i = overlap; i < lastlog; i++)
         log->temp[logcnt++] = TempToFloat(log->log_raw[i],FALSE);

      // loop to get the overlap
      for (i = 0; i < overlap; i++)
         log->temp[logcnt++] = TempToFloat(log->log_raw[i],FALSE);
   }

   //---------------------------------------------------------------------------
   // Take the Log structure and convert to string
   // format
   //
   void LogToString(Log *log, int ConvertToF, char *str)
   {
      int i,cnt=0;
      ulong logtime;
      timedate td;
```

```
   // title
   cnt += sprintf(&str[cnt],"Log Data\n--------\n"
                            "Format: [Time/Date , Temperature] ");
   if (ConvertToF)
      cnt += sprintf(&str[cnt],"(Fahrenheit)\n");
   else
      cnt += sprintf(&str[cnt],"(Celsius)\n");

   // loop through the logs
   logtime = log->start_time;
   for (i = 0; i < log->num_log; i++)
   {
      // time
      SecondsToDate(&td,logtime);
      cnt += sprintf(&str[cnt],"%02d/%02d/%04d  %02d:%02d ,",
         td.month,td.day,td.year,td.hour,td.minute);
      // temp
      cnt += sprintf(&str[cnt],"%6.1f\n",
             (ConvertToF) ? CToF(log->temp[i]): log->temp[i]);

      // increment the time
      logtime += log->interval;
   }

   // zero terminate string
   str[cnt] = 0;
}

//---------------------------------------------------------------------------
// Convert the raw debug data to a string
//
void DebugToString(MissionStatus *mstatus, TempAlarmEvents *alarm,
                   Histogram *hist, Log *log, char *str)
{
   int i,cnt=0;

   // title
   cnt += sprintf(&str[cnt],"Debug Dump\n----------\nRegister Page:\n");

   // reg
   for (i = 0; i < 32; i++)
   {
      cnt += sprintf(&str[cnt],"%02X ",mstatus->status_raw[i]);
      if (i && (((i + 1) % 16) == 0))
         cnt += sprintf(&str[cnt],"\n");
   }

   // alarms
   cnt += sprintf(&str[cnt],"Alarms:\n");
   for (i = 0; i < 96; i++)
   {
      cnt += sprintf(&str[cnt],"%02X ",alarm->alarm_raw[i]);
      if (i && (((i + 1) % 16) == 0))
         cnt += sprintf(&str[cnt],"\n");
   }

   // histogram
   cnt += sprintf(&str[cnt],"Histogram:\n");
   for (i = 0; i < 128; i++)
   {
      cnt += sprintf(&str[cnt],"%02X ",hist->hist_raw[i]);
      if (i && (((i + 1) % 16) == 0))
         cnt += sprintf(&str[cnt],"\n");
   }


   // log
   cnt += sprintf(&str[cnt],"Log:\n");
   for (i = 0; i < ((log->num_log > 2048) ? 2048 : log->num_log); i++)
   {
      cnt += sprintf(&str[cnt],"%02X ",log->log_raw[i]);
      if (i && (((i + 1) % 16) == 0))
         cnt += sprintf(&str[cnt],"\n");
   }

   // zero terminate string
```

```
   str[cnt] = 0;
}

//--------------------------------------------------------------------------
// Take one byte BCD value and return binary value
//
uchar BCDToBin(uchar bcd)
{
   return (((bcd & 0xF0) >> 4) * 10) + (bcd & 0x0F);
}


//--------------------------------------------------------------------------
// Take a 4 byte long string and convert it into a timedata structure.
//
static int dm[] = { 0,0,31,59,90,120,151,181,212,243,273,304,334,365 };

void SecondsToDate(timedate *td, ulong x)
{
   short tmp,i,j;
   ulong y;

   // check to make sure date is not over 2070 (sanity check)
   if (x > 0xBBF81E00L)
      x = 0;

   y = x/60;  td->second = (ushort)(x-60*y);
   x = y/60;  td->minute = (ushort)(y-60*x);
   y = x/24;  td->hour   = (ushort)(x-24*y);
   x = 4*(y+731);  td->year = (ushort)(x/1461);
   i = (int)((x-1461*(ulong)(td->year))/4);  td->month = 13;

   do
   {
      td->month -= 1;
      tmp = (td->month > 2) && ((td->year & 3)==0) ? 1 : 0;
      j = dm[td->month]+tmp;

   } while (i < j);

   td->day = i-j+1;

   // slight adjustment to algorithm
   if (td->day == 0)
      td->day = 1;

   td->year = (td->year < 32)  ? td->year + 68 + 1900: td->year - 32 + 2000;
}

//--------------------------------------------------------------------------
// DateToSeconds takes a time/date structure and converts it into the
// number of seconds since 1970
//
ulong DateToSeconds(timedate *td)
{
   ulong Sv,Bv,Xv;

   // convert the date/time values into the 5 byte format used in the touch
   if (td->year >= 2000)
      Sv = td->year + 32 - 2000;
   else
      Sv = td->year - 68 - 1900;

   if ((td->month > 2) && ( (Sv & 3) == 0))
     Bv = 1;
   else
     Bv = 0;

   Xv = 365 * (Sv-2) + (Sv-1)/4 + dm[td->month] + td->day + Bv - 1;

   Xv = 86400 * Xv + (ulong)(td->second) + 60*((ulong)(td->minute) + 60*(ulong)(td-
>hour));

   return Xv;
}
```

```
//--------------------------------------------------------------------------
// Convert from DS1921 termature format to a float
//
//
float TempToFloat(uchar tmp, int ConvertToF)
{
   float tfloat;

   tfloat = (float)((tmp / 2.0) - 40.0);

   if (ConvertToF)
      return (float)(tfloat * 9.0 / 5.0 + 32.0);
   else
      return tfloat;
}

//--------------------------------------------------------------------------
// Convert from Celsius to Fahrenheit
//
float CToF(float CVal)
{
   return (float)(CVal * 9.0 / 5.0 + 32.0);
}
```

FindType.c

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  findtype.c - Test module to find all devices of one type.
//
//  Version: 2.00
//
//--------------------------------------------------------------------------
//
//
#include "multinet.cpp"
#include "findtype.h"

//--------------------------------------------------------------------------
// Search for devices
//
// 'portnum'  - number 0 to MAX_PORTNUM-1.  This number is provided to
//              indicate the symbolic port number.
// 'FamilySN' - an array of all the serial numbers with the matching
//              family code
// 'family_code' - the family code of the devices to search for on the
//                 1-Wire Net
// 'MAXDEVICES'  - the maximum number of devices to look for with the
//                 family code passed.
```

```
//
// Returns: TRUE(1)  success, device type found
//          FALSE(0) device not found
//
SMALLINT FindDevices(int portnum, uchar FamilySN[][8], SMALLINT family_code, int
MAXDEVICES)
{
   int NumDevices=0;

   // find the devices
   // set the search to first find that family code
   owFamilySearchSetup(portnum,family_code);

   // loop to find all of the devices up to MAXDEVICES
   NumDevices = 0;
   do
   {
      // perform the search
      if (!owNext(portnum,TRUE, FALSE))
         break;

      owSerialNum(portnum,FamilySN[NumDevices], TRUE);
      if ((FamilySN[NumDevices][0] & 0x7F) == (family_code & 0x7F))
      {
         NumDevices++;
      }
   }
   while (NumDevices < (MAXDEVICES - 1));

   // check if not at least 1 device
   return NumDevices;
}
```

Temp10.h

```
//---------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
// --------------------------------------------------------------------------
//
//  temp10.h - Header to read the DS1920/DS1820 - temperature measurement.
//
//  Version: 2.00
//
// --------------------------------------------------------------------------

int ReadTemperature(int,uchar *,float *);
```

Thermo21.h

```
//--------------------------------------------------------------------------
// Copyright (C) 2000 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//--------------------------------------------------------------------------
//
//  thermo.h - Include file for Thermochron demo.
//
//  Version: 2.00
//
//    History:
//           1.03 -> 2.00  Reorganization of Public Domain Kit


#ifndef THERMO_TYPES

#define THERMO_TYPES

// defines
#define STATUS_PAGE    16
#define THERMO_FAM     0x21

#include <stdlib.h>

// Typedefs
#ifndef OW_UCHAR
   #define OW_UCHAR
   typedef unsigned char  uchar;
   #ifdef WIN32
      typedef unsigned short ushort;
      typedef unsigned long  ulong;
   #endif
#endif

// structure to hold the mission status
typedef struct
{
   uchar serial_num[8];          // serial number of thermochron

   uchar mission_in_progress;    // 1 mission in progres, 0 mission over

   uchar sample_rate;            // minutes between samples

   uchar rollover_enable;        // 1 if roll-over enabled
   uchar rollover_occurred;      // 1 if roll-over occurred

   ushort start_delay;           // minutes before mission starts

   ulong mission_start_time;     // date/time when mission started
   ulong current_time;           // current real-time clock value
   ulong download_time;          // download stations time of reading

   ulong mission_samples;        // number of samples in this mission
   ulong samples_total;          // total number of samples taken by device

   uchar high_threshold;         // raw temp of high threshold
```

```
   uchar low_threshold;          // raw temp of low threshold

   // skip alarm modes and status for now

   uchar status_raw[32];

} MissionStatus;

// structure to hold the histogram data
typedef struct
{
   ushort bin_count[63];    // counter per bin 0 to 62
   float  start_range[63];  // start temp range (C) in bin 0 to 62
   float  end_range[63];    // end temp range (C) in bin 0 to 62

   uchar  hist_raw[128];    // raw data for histogram

} Histogram;

// structure to hold the histogram data
typedef struct
{
   int num_low;               // number of low events
   ulong low_start_time[12];  // start time of event 0 to 12
   ulong low_end_time[12];    // end time of event 0 to 12
   int num_high;              // number of high events
   ulong high_start_time[12]; // start time of event 0 to 12
   ulong high_end_time[12];   // end time of event 0 to 12

   uchar  alarm_raw[96];     // raw data for alarm events

} TempAlarmEvents;

// structure to hold the log data
typedef struct
{
   int   num_log;             // number of logs
   float temp[2048];          // temperature log in (C)
   ulong start_time;          // start time of log
   int   interval;            // interval in seconds between logs

   uchar log_raw[2048];       // raw data for log

} Log;

// structure to hold all of the thermochron data state
typedef struct
{
   MissionStatus MissStat;    // mission state
   Histogram HistData;        // histogram data
   TempAlarmEvents AlarmData; // temperature alarm event data
   Log LogData;               // log data

} ThermoStateType;

// type structure to holde time/date
typedef struct
{
     ushort  second;
     ushort  minute;
     ushort  hour;
     ushort  day;
     ushort  month;
     ushort  year;
} timedate;

// structure to hold each state in the StateMachine
typedef struct
{
   int  Step;
   char StepDescription[50];

} ThermoScript;
#endif

// step constants
```

```
enum { ST_SETUP=0, ST_READ_STATUS, ST_READ_ALARM, ST_READ_HIST,
       ST_READ_LOG, ST_CLEAR_MEM, ST_CLEAR_VERIFY, ST_WRITE_TIME,
       ST_WRITE_CONTROL, ST_WRITE_RATE, ST_FINISH, ST_GET_SESSION,
       ST_FIND_THERMO, ST_REL_SESSION, ST_READ_PAGES, ST_WRITE_MEM,
       ST_CLEAR_SETUP };

// status contants
enum { STATUS_STEP_COMPLETE, STATUS_COMPLETE, STATUS_INPROGRESS,
       STATUS_ERROR_HALT, STATUS_ERROR_TRANSIENT };

// download steps
static ThermoScript Download[] =
    {{ ST_READ_STATUS,  "Setup to read the mission status"},
     { ST_READ_PAGES,   "Read the status page"},
     { ST_READ_ALARM,   "Setup to read alarm pages"},
     { ST_READ_PAGES,   "Read the alarm pages"},
     { ST_READ_HIST,    "Setup to read histogram pages"},
     { ST_READ_PAGES,   "Read the histogram pages"},
     { ST_READ_LOG,     "Setup to read log pages"},
     { ST_READ_PAGES,   "Read the log pages"},
     { ST_FINISH,       "Finished"}};

// read status only steps
static ThermoScript GetStatus[] =
    {{ ST_READ_STATUS,  "Setup to read the mission status"},
     { ST_READ_PAGES,   "Read the status page"},
     { ST_FINISH,       "Finished"}};

// mission steps (assume already did StatusThermo)
static ThermoScript Mission[] =
    {{ ST_CLEAR_SETUP,  "Setup clear memory"},
     { ST_WRITE_MEM,    "Write clear memory bit"},
     { ST_CLEAR_MEM,    "Clear the memory"},
     { ST_READ_STATUS,  "Setup to read the mission status"},
     { ST_READ_PAGES,   "Read the status page"},
     { ST_CLEAR_VERIFY, "Verify memory is clear"},
     { ST_WRITE_TIME,   "Setup to write the real time clock"},
     { ST_WRITE_MEM,    "Write the real time clock"},
     { ST_WRITE_CONTROL,"Setup to write the control"},
     { ST_WRITE_MEM,    "Write the control"},
     { ST_WRITE_RATE,   "Setup to write the sample rate to start mission"},
     { ST_WRITE_MEM,    "Write the sample rate"},
     { ST_READ_STATUS,  "Read the new mission status"},
     { ST_FINISH,       "Finished"}};

// Local Function Prototypes
int DownloadThermo(int,uchar *,ThermoStateType *,FILE *);
int ReadThermoStatus(int,uchar *,ThermoStateType *,FILE *);
int MissionThermo(int,uchar *,ThermoStateType *,FILE *);
static int RunThermoScript(int,ThermoStateType *,ThermoScript script[],FILE *fp);
void MissionStatusToString(MissionStatus *,int,char *);
void  SecondsToDate(timedate *, ulong);
ulong DateToSeconds(timedate *);
uchar BCDToBin(uchar);
void  InterpretStatus(MissionStatus *);
void  FormatMission(MissionStatus *);
void  InterpretHistogram(Histogram *);
void  HistogramToString(Histogram *, int, char *);
void  InterpretAlarms(TempAlarmEvents *, MissionStatus *);
void  AlarmsToString(TempAlarmEvents *, char *);
void  InterpretLog(Log *, MissionStatus *);
void  LogToString(Log *, int, char *);
void  DebugToString(MissionStatus *, TempAlarmEvents *, Histogram *, Log *, char *);
float TempToFloat(uchar, int);
float CToF(float);
uchar ToBCD(short);
```

FindType.h

```
//--------------------------------------------------------------------------
// Copyright (C) 2001 Dallas Semiconductor Corporation, All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a
```

```
// copy of this software and associated documentation files (the "Software"),
// to deal in the Software without restriction, including without limitation
// the rights to use, copy, modify, merge, publish, distribute, sublicense,
// and/or sell copies of the Software, and to permit persons to whom the
// Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included
// in all copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
// OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
// MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
// IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
// OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
// ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
// OTHER DEALINGS IN THE SOFTWARE.
//
// Except as contained in this notice, the name of Dallas Semiconductor
// shall not be used except as stated in the Dallas Semiconductor
// Branding Policy.
//---------------------------------------------------------------------------
//
//  findtype.c - Header for the module to find all devices of one type.
//
//  Version: 2.00
//
//---------------------------------------------------------------------

SMALLINT FindDevices(int,uchar FamilySN[][8],SMALLINT,int);
```

Main.c

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Main.h"
#include "About.h"
#include "RealTime.h"
#include "Stored.h"
//#include "HA7net.c"

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

TfrmOW *frmOW;
//---------------------------------------------------------------------------
__fastcall TfrmOW::TfrmOW(TComponent* Owner)
   : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TfrmOW::cmbPortNameChange(TObject *Sender)
{
   //ownet.setportname(cmbPortName->Text.c_str());
   //txtPort->Text = ownet.getportname();
}
//---------------------------------------------------------------------------

void __fastcall TfrmOW::btnAboutClick(TObject *Sender)
{
   AboutBox->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TfrmOW::btnRealClick(TObject *Sender)
{
    frmRealTime->ShowModal(); // Real
}
//---------------------------------------------------------------------------
```

```
void __fastcall TfrmOW::btnStoredClick(TObject *Sender)
{
   frmStored->ShowModal(); // Stored
}
//---------------------------------------------------------------------------

void __fastcall TfrmOW::btnExitClick(TObject *Sender)
{
   Close();
}
//---------------------------------------------------------------------------
```

RealTime.c

```
//===========================================================================
//    Temperature application DS1920/DS1820 - Version 1.00
//    The following is a test to excersize a DS1920/DS1820.
//    Temperature Find and Read from a DS1920/DS1820 (at least 1)
//===========================================================================

#include <vcl.h>
#pragma hdrstop

#include "RealTime.h"

#include <stdlib.h>
#include <stdio.h>
#include "multinet.cpp"
#include "temp10.h"
#include "findtype.h"

// defines
#define MAXDEVICES         20

// global serial numbers
uchar FamilySN[MAXDEVICES][8];

// variables
int family_code;
bool stop = false;
bool closed = true;

float current_temp;
int i = 0;
int NumDevices=0;
int portnum = 0;
//char aname[20]= "\\\\.\\DS2490-1";
char aname[20]= "{1,6}";

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "PERFGRAP"
#pragma resource "*.dfm"
TfrmRealTime *frmRealTime;
//---------------------------------------------------------------------------
__fastcall TfrmRealTime::TfrmRealTime(TComponent* Owner)
   : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TfrmRealTime::cmbPortNameChange(TObject *Sender)
{
   if(closed == true) {
      if(cmbPortName->Text=="COM1"){
       ;
      }
      if(cmbPortName->Text=="COM2"){
         ;
      }
      if(cmbPortName->Text=="COM3"){
         strcpy(aname,"{1,6}");
```

```
      }
      if(cmbPortName->Text=="HTTP"){
         ;
      }

      // attempt to acquire the 1-Wire Net
      if((portnum = owAcquireEx(aname)) < 0)
      {
         stb->SimplePanel = true;
         stb->SimpleText = "Device not present on Port " + cmbPortName->Text;
      }
      else {
         stb->SimplePanel = true;
         stb->SimpleText = "Port " + cmbPortName->Text + " open and ready to use";
         closed = false;
      }
   }
}
//---------------------------------------------------------------------------

void __fastcall TfrmRealTime::tmrTempTimer(TObject *Sender)
{
   // read the temperature and print serial number and temperature
   for (i = 0; i < NumDevices; i++){
      if (ReadTemperature(portnum, FamilySN[i],&current_temp)){
         lbxC->Items->Add( current_temp );
         lbxC->Refresh();
         lbxC->ItemIndex = lbxC->Items->Count - 1;

         lbxF->Items->Add( current_temp * 9 / 5 + 32 );
         lbxF->Refresh();
         lbxF->ItemIndex = lbxF->Items->Count - 1;

         Series1->Add( current_temp , lbxC->Items->Count     , clTeeColor );

         stb->SimpleText = "Reading temperature status OK!"; // + AnsiString(a);
      }
      else {
         stb->SimpleText = "Error reading temperature, verify device present";
      }
   }
}
//---------------------------------------------------------------------------


void __fastcall TfrmRealTime::btnStartClick(TObject *Sender)
{
   tmrTemp->Enabled = true;
}
//---------------------------------------------------------------------------

void __fastcall TfrmRealTime::btnStopClick(TObject *Sender)
{
   tmrTemp->Enabled = false;
   stb->SimpleText = "Reading Temperature aborted by user";
}
//---------------------------------------------------------------------------

void __fastcall TfrmRealTime::btnOpenClick(TObject *Sender)
{
   Series1->Clear();

   // Find the device(s)
   NumDevices = FindDevices(portnum, &FamilySN[0], 0x10, MAXDEVICES);
   if (NumDevices>0)
   {
      closed = false;
      for (i = 0; i < NumDevices; i++)
      {
         //PrintSerialNum(FamilySN[i]);
         //char a[30];
         // sprintf(a,"%x%x%x%x%x%x%x%x",
FamilySN[7],FamilySN[6],FamilySN[5],FamilySN[4],FamilySN[3],FamilySN[2],FamilySN[1],Fami
lySN[0]);
         stb->SimpleText = "Device DS1920/DS1820 found!";
      }
```

```
   }
   else {
      stb->SimpleText = "ERROR, device DS1920/DS1820 not found!";
      closed = true;
   }
}
//---------------------------------------------------------------------------

void __fastcall TfrmRealTime::btnCloseClick(TObject *Sender)
{
   // release the 1-Wire Net
   if(closed == false){
      owRelease(portnum);
      stb->SimpleText = "Device Closed by User!";
      closed = true;
      lbxC->Clear();
      lbxF->Clear();
      Series1->Clear();
   }
   else
      stb->SimpleText = "Device allready Closed!";
}
//---------------------------------------------------------------------------

void __fastcall TfrmRealTime::btnExitClick(TObject *Sender)
{
   if(closed == false)
      owRelease(portnum);
   Close();
}

void __fastcall TfrmRealTime::FormActivate(TObject *Sender)
{
   stb->SimpleText = "";
   lbxC->Clear();
   lbxF->Clear();
   Series1->Clear();
}
//---------------------------------------------------------------------------
```

Stored.c

```
//===========================================================================
//   Stored Temperature Reader application DS1921- Version 1.00
//    The following is a test to excersize a DS1921.
//    Temperature Archives Read from a DS1921 (at least 1)
//===========================================================================

#include <vcl.h>
#pragma hdrstop

#include "Stored.h"

#include <stdio.h>
#include <stdlib.h>
#include "multinet.cpp"
#include "thermo21.h"
#include "findtype.h"

// defines
#define MAXDEVICES   20

bool closed = true;
//char aname[20];
char aname[20]= "{1,6}";

int Fahrenheit=FALSE,filenum=0,num,i,j;
FILE *fp;
ThermoStateType ThermoState;
uchar ThermoSN[MAXDEVICES][8]; //the serial numbers for the devices
int portnum=0;

// local function prototypes
```

```
void PrintResults(ThermoStateType *,FILE *,int);
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmStored *frmStored;
//---------------------------------------------------------------------------
__fastcall TfrmStored::TfrmStored(TComponent* Owner)
   : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TfrmStored::cmbPortNameChange(TObject *Sender){
   if(closed == true) {
      if(cmbPortName->Text=="COM1"){
       ;
      }
      if(cmbPortName->Text=="COM2"){
         ;
      }
      if(cmbPortName->Text=="COM3"){
         strcpy(aname,"{1,6}");
      }
      if(cmbPortName->Text=="HTTP"){
         ;
      }

      // attempt to acquire the 1-Wire Net
      if((portnum = owAcquireEx(aname)) < 0)
      {
         stb->SimplePanel = true;
         stb->SimpleText = "Device not present on Port " + cmbPortName->Text;
         //strcpy(aname,"-1");
         closed=true;
      }
      else {
         stb->SimplePanel = true;
         stb->SimpleText = "Port " + cmbPortName->Text + " open and ready to use";
         closed = false;
         fp = fopen("output.txt","w+");
         if(fp == NULL){
             stb->SimpleText = "ERROR, Could not open output file!";
         }
      }
   }
}
//---------------------------------------------------------------------------

void __fastcall TfrmStored::btnOpenClick(TObject *Sender){
   if (closed==false){
      Series1->Clear();
      // get list of Thermochron's
        num = FindDevices(portnum, &ThermoSN[0],THERMO_FAM, MAXDEVICES);
      // check if not present or more than 1 present
      if (num == 1) {
         //for (i = 0; i < num; i++){
         stb->SimpleText = "Device DS1921 found!";
         for (j = 7; j >= 0; j--){
           ThermoState.MissStat.serial_num[j] = ThermoSN[0][j];
          }
      }
   }
   else{
      stb->SimpleText = "ERROR, Device DS1921 not aquired, find corresponding COM port";
   }
}
//---------------------------------------------------------------------------

void __fastcall TfrmStored::btnStartClick(TObject *Sender){
      // download the Thermochron found
      if (closed==false){
         stb->SimpleText = "Reading device, Please wait";
         if (DownloadThermo(portnum,&ThermoSN[0][0],&ThermoState,fp)){

            // interpret the results of the download
            InterpretStatus(&ThermoState.MissStat);
```

```
InterpretAlarms(&ThermoState.AlarmData, &ThermoState.MissStat);
InterpretHistogram(&ThermoState.HistData);
InterpretLog(&ThermoState.LogData, &ThermoState.MissStat);

// print the output
// PrintResults(&ThermoState,fp,Fahrenheit);
char *str;

// get big block to use as a buffer
str = (char *)malloc(80000);
if (str == NULL){
   //printf("Insufficient memory available to print!\n");
   //closed=true;
   return;
}

// mission status
MissionStatusToString(&ThermoState.MissStat, Fahrenheit, &str[0]);
fprintf(fp,"\n%s\n",str);
// managing the output to fit in to the drop-down lists
// following code replaces \n with \0
char tmp[200];
for(int c=0,t=0;str[c]!='\0';c++,t++){
   if(str[c]=='\n'){
      tmp[t]='\0';
      memMission->Lines->Add(tmp);
      t=-1;
   }
   else{
      tmp[t] = str[c];
   }
}

// alarm events
AlarmsToString(&ThermoState.AlarmData, &str[0]);
fprintf(fp,"%s\n",str);
// managing the output to fit in to the drop-down lists
// following code replaces \n with \0
for(int c=0,t=0;str[c]!='\0';c++,t++){
   if(str[c]=='\n'){
      tmp[t]='\0';
      memAlarms->Lines->Add(tmp);
      t=-1;
   }
   else{
      tmp[t] = str[c];
   }
}
// histogram
HistogramToString(&ThermoState.HistData, Fahrenheit, &str[0]);
fprintf(fp,"%s\n",str);
int sn=0;
float hx=0,hy=0;
for(int c=0,t=0;str[c]!='\0';c++,t++){
   if(str[c]=='\n'){
      tmp[t]='\0';
      sn++;
      if(sn>3){
         char num[10];
         int x,y;
         for(x=0;x<6;x++)
            num[x]=tmp[x];
         num[x]='\0';
         //lbxC->Items->Add(num);
         hx=atof(num);
         for(x=18,y=0;tmp[x]!='\0';x++,y++)
            num[y]=tmp[x];
         num[y]='\0';
         hy=atof(num);
         //bottom graph sowing temp histogram
         Series1->Add( hy , hx , clTeeColor );
      }
      t=-1;
   }
   else{
      tmp[t] = str[c];
```

```
            }
         }

         // log data
         LogToString(&ThermoState.LogData, Fahrenheit, &str[0]);
         fprintf(fp,"%s\n",str);
         sn=0;
         char shx[25];
         hy=0;
         for(int c=0,t=0;str[c]!='\0';c++,t++){
            if(str[c]=='\n'){
               tmp[t]='\0';
               sn++;
               if(sn>3){
                  char num[25];
                  int x,y;
                  for(x=0;x<17;x++)
                     num[x]=tmp[x];
                  num[x]='\0';
                  strcpy(shx,num);
                  for(x=19,y=0;tmp[x]!='\0';x++,y++)
                     num[y]=tmp[x];
                  num[y]='\0';
                  hy=atof(num);
                  //top graph sowing stored temperature
                  lbxC->Items->Add(AnsiString(shx)+ " -> " +AnsiString(num));
                  lbxC->Refresh();
                  lbxC->ItemIndex = lbxC->Items->Count - 1;
                  Series2->Add( hy , shx     , clTeeColor );
               }
               t=-1;
            }
            else{
               tmp[t] = str[c];
            }
         }
          // not using the raw data, the code is ready for use

         // debug raw data
         //   DebugToString(&ThermoState->MissStat, &ThermoState->AlarmData,
         //      &ThermoState->HistData, &ThermoState->LogData, &str[0]);
         //   fprintf(fp,"%s\n",str);

         // free the memory block used
         free(str);
         stb->SimpleText = "Finished reading device, System Ready";
         }
         else{
         stb->SimpleText = "Error reading, verify device present";
         fprintf(fp,"\nError downloading device: ");
         for (j = 0; j < 8; j++)
            fprintf(fp,"%02X",ThermoSN[i][j]);
            fprintf(fp,"\n");
         }
      }
      else
        stb->SimpleText = "Device Closed or Device not present, first choose COM port
and press Open Device" ;
}
//---------------------------------------------------------------------------

void __fastcall TfrmStored::btnCloseClick(TObject *Sender){
   // release the 1-Wire Net
   if(closed == false){
      owRelease(portnum);
      stb->SimpleText = "Device Closed by User!";
      lbxC->Clear();
      Series2->Clear();
      Series1->Clear();
      memAlarms->Clear();
      memMission->Clear();

      closed=true;
      // close opened file
      if (fp != NULL)
      {
```

```
         //printf("File '%s' closed.\n", argv[filenum]);
         fclose(fp);
      }
   }
   else
      stb->SimpleText = "Device already Closed!";
}
//---------------------------------------------------------------------------

void __fastcall TfrmStored::btnExitClick(TObject *Sender)
{
   if(closed == false){
      owRelease(portnum);
      closed=true;
      Close();
   }
   else
      Close();
}
//---------------------------------------------------------------------------

void __fastcall TfrmStored::FormActivate(TObject *Sender)
{
   stb->SimpleText = "";
   lbxC->Clear();

   Series1->Clear();
   Series2->Clear();

}
//---------------------------------------------------------------------------
```

TempHum.c

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "TempHum.h"
#include <stdlib.h>
#include <stdio.h>
#include "multinet.cpp"
#include "temp10.h"
#include "findtype.h"
#include "atod26.h"
// defines
#define MAXDEVICES         20
#define ONEKBITADD 0x89

// create a file to store data for DEMO APP
FILE *fpTemp , *fpHum, *fpHumTemp;

// global serial numbers
uchar FamilySN[MAXDEVICES][8];

// variables
int family_code;
bool stop = false;
bool closed = true;
float current_temp;
int i = 0;
int NumDevices=0;
int portnum = 0;
char aname[20]= "{1,6}";

   char msg[200];
//   int portnum = 0;
   float Vdd,Vad;
   double humid,temp;
//   int i;
   int numbat,cnt=0;
   uchar famvolt[MAXDEVICES][8];

//---------------------------------------------------------------------------
```

```
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmTempHum *frmTempHum;
//---------------------------------------------------------------------------
__fastcall TfrmTempHum::TfrmTempHum(TComponent* Owner)
   : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmTempHum::cmbPortNameChange(TObject *Sender)
{
   if(closed == true) {
      if(cmbPortName->Text=="COM1"){
       ;
      }
      if(cmbPortName->Text=="COM2"){
         ;
      }
      if(cmbPortName->Text=="COM3"){
         strcpy(aname,"{1,6}");
      }
      if(cmbPortName->Text=="HTTP"){
         ;
      }

      // attempt to acquire the 1-Wire Net
      if((portnum = owAcquireEx(aname)) < 0){
         stb->SimplePanel = true;
         stb->SimpleText = "Device(s) not present on Port " + cmbPortName->Text;
      }
      else {
         stb->SimplePanel = true;
         stb->SimpleText = "Port " + cmbPortName->Text + " open and ready to use";
         // store temps from DS1920 to a file named 1920Temp.txt
         fpTemp = fopen("1920Temp.txt","w+");
         if(fpTemp == NULL)
            stb->SimpleText = "ERROR, Could not open output file!";
         // open the 1920Temp.txt file and write the titles
         fprintf(fpTemp,"%s","  DATE       TIME     DS1920       DS2438");

         // store temps from DS2438 to a file named 2438Temp.txt
         fpHumTemp = fopen("2438Temp.txt","w+");
         if(fpHumTemp == NULL)
            stb->SimpleText = "ERROR, Could not open output file!";
         // open the 2438Temp.txt file and write the titles
         fprintf(fpHumTemp,"%s","  DATE       TIME     CELSIUS");

         // store humidity to a file named 2438hum.txt
         fpHum = fopen("2438hum.txt","w+");
         if(fpHum == NULL)
            stb->SimpleText = "ERROR, Could not open output file!";
         // open the 2438hum.txt file and write the titles
         fprintf(fpHum,"%s","  DATE       TIME     HUMIDTITY");

         closed = false;
      }
   }
}
//---------------------------------------------------------------------------
void __fastcall TfrmTempHum::btnOpenClick(TObject *Sender){
   numbat = FindDevices(portnum,&famvolt[0],SBATTERY_FAM,MAXDEVICES);
   Series1->Clear();
   // Find the device(s)
   NumDevices = FindDevices(portnum, &FamilySN[0], 0x10, MAXDEVICES);
   if (NumDevices>0){
      closed = false;
      for (i = 0; i < NumDevices; i++){
         stb->SimpleText = "Device DS1920 and/or Device DS2438 found!";
      }
   }
   else{
      stb->SimpleText = "ERROR, No device(s) DS1920 and/or DS2438, found!";
      closed = true;
   }

}
```

```
//---------------------------------------------------------------------------
void __fastcall TfrmTempHum::btnExitClick(TObject *Sender)
{
   if(closed == false)
      owRelease(portnum);
   Close();
}
//---------------------------------------------------------------------------
void __fastcall TfrmTempHum::FormActivate(TObject *Sender)
{
   //clear ALL the graphs from previous data
   stb->SimpleText = "";
   lbxC->Clear();
   lbxF->Clear();
   Series1->Clear();
   lbxHum->Clear();
   FastLineSeries1->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TfrmTempHum::btnStartClick(TObject *Sender)
{
   if (closed==false)
      tmrTemp->Enabled = true;
   else
      stb->SimpleText = "Please choose PORT and press 'Open DeviceS' button before";
}
//---------------------------------------------------------------------------
void __fastcall TfrmTempHum::btnStopClick(TObject *Sender)
{
   tmrTemp->Enabled = false;
   stb->SimpleText = "Reading Device(s) aborted by user";
}
//---------------------------------------------------------------------------
void __fastcall TfrmTempHum::tmrTempTimer(TObject *Sender)
{
   // read the temperature and print serial number and temperature
   for (i = 0; i < NumDevices; i++){
      if (ReadTemperature(portnum, FamilySN[i],&current_temp)){
         //storing current temp to the file named 1920temp.txt
         //storing also time and date for data indegrity
         fprintf(fpTemp,"\n%s ",DateToStr(Date()));
         fprintf(fpTemp,"%s ",TimeToStr(Time()));
         fprintf(fpTemp,"   %g",current_temp);
         // if there is also a DS2438 present, get the temp also
         fprintf(fpTemp,"   %g",temp);

         stb->SimpleText = "Reading temperature status OK!";
         lbxC->Items->Add(current_temp);
         lbxC->Refresh();
         lbxC->ItemIndex = lbxC->Items->Count - 1;
         lbxF->Items->Add( current_temp * 9 / 5 + 32 );
         lbxF->Refresh();
         lbxF->ItemIndex = lbxF->Items->Count - 1;
         Series1->Add( current_temp , lbxC->Items->Count     , clTeeColor );


         //this code is for DS2438 humid reader
               Vdd = ReadAtoD(portnum,TRUE,&famvolt[0][0]);
               if(Vdd > 5.8){
                  Vdd = (float)5.8;
               }
               else if(Vdd < 4.0){
                  Vdd = (float) 4.0;
               }
               Vad = ReadAtoD(portnum,FALSE,&famvolt[0][0]);
               temp = Get_Temperature(portnum,&famvolt[0][0]);
               //show temp on the same graph with the other button
               Series2->Add( temp , lbxC->Items->Count     , clTeeColor );
               //storing current temp to the file named 2438temp.txt
               //storing also time and date for data integrity
               fprintf(fpHumTemp,"\n%s ",DateToStr(Date()));
               fprintf(fpHumTemp,"%s ",TimeToStr(Time()));
               fprintf(fpHumTemp,"   %g",temp);

               humid = (((Vad/Vdd) - 0.16)/0.0062)/(1.0546 - 0.00216 * temp);
               Series3->Add( humid , lbxC->Items->Count     , clTeeColor );
```

```
                //storing current humid to the file named 2438hum.txt
                //storing also time and date for data integrity
                fprintf(fpHum,"\n%s ",DateToStr(Date()));
                fprintf(fpHum,"%s ",TimeToStr(Time()));
                fprintf(fpHum,"   %g",humid);

                if(humid > 100){
                   humid = 100;
                }
                else if(humid < 0){
                   humid = 0;
                }
/*              printf("\n");
                printf("The humidity is:  %4.4f\n", humid);
                printf("Given that the temp was:   %2.2f\n", temp);
                printf("and the volt supply was:   %2.2f\n", Vdd);
                printf("with the volt output was:  %2.2f\n", Vad);
                printf("\n");

*/          // to mayn digits in humid, cutting down afew
            char bbb[80];
            sprintf(bbb,"%4.4f",humid);
            lbxHum->Items->Add(AnsiString(bbb));
            stb->SimpleText = "Reading humidity status OK!";
            lbxHum->Refresh();
            lbxHum->ItemIndex = lbxHum->Items->Count - 1;

            //showing temp from humid sensor
            lbxhmdTemp->Items->Add(temp);
            lbxhmdTemp->Refresh();
            lbxhmdTemp->ItemIndex = lbxC->Items->Count - 1;
      }
      else {
         stb->SimpleText = "Error reading temperature and/or humidity, verify device(s)
present";
      }
   }

}
//---------------------------------------------------------------------------


void __fastcall TfrmTempHum::btnCloseClick(TObject *Sender)
{
   if (closed==false){
      if (fpHum != NULL){
         fclose(fpHum);
      }
      if (fpHumTemp != NULL){
         fclose(fpHumTemp);
      }
      if (fpHum != NULL){
         fclose(fpHum);
      }

      closed=true;
      owRelease(portnum);
      stb->SimpleText = "Device(s) released";
   }
   else{
      stb->SimpleText = "Button already pressed, or Devices were not aquired";
   }
}
//---------------------------------------------------------------------------
```

About.c

```
//---------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
```

```
#include "About.h"
//---------------------------------------------------------------------
#pragma resource "*.dfm"
TAboutBox *AboutBox;
//---------------------------------------------------------------------
__fastcall TAboutBox::TAboutBox(TComponent* AOwner)
        : TForm(AOwner)
{
}
//---------------------------------------------------------------------
```